UNIVERSITY OF BEIRA INTERIOR
DEPARTMENT OF COMPUTER SCIENCE

Study of the Impact of Intensive Attacks in the Self-Similarity Degree of the Network Traffic in Intra-Domain Aggregation Points

Pedro Ricardo Morais Inácio

(5-year Bachelor of Science)

Thesis submitted to the University of Beira Interior in candidature for the Degree of Doctor of Philosophy in Computer Science and Engineering

Tese submetida à Universidade da Beira Interior para obtenção do Grau de Doutor em Engenharia Informática

Covilhã, Portugal
2009

UNIVERSITY OF BEIRA INTERIOR
DEPARTMENT OF COMPUTER SCIENCE

Study of the Impact of Intensive Attacks in the Self-Similarity Degree of the Network Traffic in Intra-Domain Aggregation Points

Pedro Ricardo Morais Inácio

(5-year Bachelor of Science)

Thesis submitted to the University of Beira Interior in candidature for the Degree of Doctor of Philosophy in Computer Science and Engineering

Tese submetida à Universidade da Beira Interior para obtenção do Grau de Doutor em Engenharia Informática

Covilhã, Portugal
2009

Thesis written under supervision of Dr. Mário Marques Freire,
Full Professor of the Department of Computer Science of the
University of Beira Interior, Portugal, and
Performed in enterprise environment under the
Co-supervision of Dr. Paulo Miguel Nepomuceno Pereira Monteiro,
Head of the Research, Technology and Platforms group of
Nokia Siemens Networks Portugal S.A.

Tese realizada sob a orientação do Doutor Mário Marques Freire,
Professor Catedrático do Departamento de Informática da
Universidade da Beira Interior, Portugal, e
Com co-orientação em ambiente industrial do
Doutor Paulo Miguel Nepomuceno Pereira Monteiro,
Director do grupo de Investigação, Tecnologia e Plataformas da
Nokia Siemens Networks Portugal S.A.

My music and my poetry have been, among other things, Computer Science and Mathematics. This document is technical, and maybe too hard for you to understand: mother, father, grandparents, Mr. Leonel, Mrs. Ema, Cláudia, João and Patrícia. Either way, I would like to dedicate this work to you, as well as all the effort I have put on its term, as proof that I acknowledge that, without you, this music would not have lyrics nor instruments; and this poetry would not have rimes nor fundament.

Thank you very much.

A minha música e a minha poesia têm sido, entre outras, a Informática e a Matemática. O presente documento é técnico e talvez difícil de perceber para vocês: mãe, pai, avós, Sr. Leonel, Dona Ema, Cláudia, João e Patrícia. Contudo, gostaria de vo-lo dedicar, assim como todo o esforço que coloquei no seu termo, como prova de que reconheço que, sem vocês, esta música não teria letra nem instrumentos; e esta poesia não teria rimas ou fundamento.

A vocês, o meu muito bem-haja.

Acknowledgements

The first person I would like to acknowledge is my mother, that first gave me life, then taught me love, and guided me trough all major steps of life until now. If today I am an honest man of science, I owe the inspiration to that beautiful and wise woman, with whom I learnt to write even before going to school, who showed me simple math operations, smiled at me when she was sad and fought despite being tired.

The second person I would like to acknowledge is my father, for always treating me with the respect I think he deserves, stimulating the hierarchy of trust and respect I currently apply towards others, and that enables me to make friends so easily.

I want to thank my brother for being a model of honesty and loyalty, always ready to hear me or to lighten my day with a funny joke. The fact that he looks up to me motivates my spirit every moment of the day.

Next, I want to mention Patrícia, my dearest friend, to whom I am eternally grateful for listening to all the detailed and lengthy descriptions of theories and ideas she was not academically prepared to understand, for always helping me and encouraging me, for being my companion 24 hours a day, for accepting the trade-off between romanticism and the number of hours spent in front of a computer and, above all that, for accepting to marry a guy with nothing more than a lot of good intentions in one hand and a Ph.D. scholarship in the other.

I would also like to acknowledge my grandmother for always being worried with me and for praying for the success of my adventure. I am sure that if there is any kind of mystical subtract in this thesis, she is the one to blame. I am grateful to Mr. Leonel, Mrs. Ema and Miss Cláudia Nunes for all the encouragement and for the apparently unlimited trust they have in me.

One of the best outcomes of this journey concerns the people I had the pleasure to meet along the way. I want to thank João Miguel Santos, Branka Lakic, Przemek Lenkiewicz, Stephane Cauchie, Rui Meleiro, João Pedro, Silvia Pato, Lara Pellegrino, Ruben Luis, Daniel Fonseca, Daniel Martins, Carlos Santiago, Jorge Castro, Fernando Jesus, João Redol, Pan Jieke, José Pina, Ana Ferreira, Tiago Silveira, Marco Pinho, José Miguel Santos, Margarida Pereira, Prof. Manuela Pereira, Artur Arsénio, Marek Hajduczenia and Nuno Garcia for making this experience richer and unpredictable. I am especially grateful to Rui Morais and João Gomes for all the invaluable discussions about all kind of subjects and for putting up with me. I have appreciated all the conversations with no apparent objective, all the table tennis matches with Eurico Cabrita and all the dinners, followed by bowling sessions, with Luis Gonçalves.

This section would not be complete without acknowledging the merit of the two men that started this project: Prof. Mário Marques Freire and Prof. Paulo Monteiro. I thank them for always treating me as a friend, for the opportunity they gave me to work along their side and for the sympathy demonstrated since day one.

Last but not least, I would like to make a special mention concerning the two persons that are no longer with me physically (grandfather Morais and grandfather José), but that accompanied me every step of the way, including in my dreams. The knowledge they passed to me comes in handy in many different aspects of life and their fantastic sayings appear in my mind spontaneously when needed, reminding me of Plato theory.

Foreword

This thesis is the result of the Ph.D. research programme performed in the enterprise environment of Nokia Siemens Networks Portugal S.A. (NSN SA), made possible by the special collaboration protocol between former Siemens S.A. and the University of Beira Interior. This work was also partially supported by *Fundação para a Ciência e a Tecnologia, Portugal*, through the grant contract SFRH/BDE/15592/2006, and by TRAMANET Project contract PTDC/EIA/73072/2006.

During three years, the above mentioned environment has constantly changed, inspired by the evolution the telecommunications industry is suffering. In 2005, Information and Communications (IC) was one of the major business divisions of the multinational company Siemens AG, with headquarters in Germany and several centres of competence around the world, namely in Portugal. The contract that delimits the beginning of this Ph.D. was signed in September that year with the Research (R) group of the Research and Development 1 (RD1) department of the IC division at Portugal. After accepting the proposal for a fifty-fifty joint venture from the Finnish telecommunications company Nokia, the IC division of Siemens AG was separated from the mother company, originating the Siemens Networks Company. Nokia Siemens Networks (NSN) started operations in April 2007, as a result of a merger between the Networks Business Group of Nokia and Siemens Networks. During this process, the Portuguese IC RD1 R group demarcated its position inside the long term research frame of the company, denominated by Research, Technology and Platforms (RTP), by obtaining funding for an integrated project, by having a very active role in the creation of intellectual property inside NSN and by conducting research of recognised quality. Besides being a researcher in the RTP group, the author worked also as a *system architect* of the BroadBand Access Business Unit (BBA BU) during 6 months, in a project concerning *next generation optical access*.

The unique characteristics of the environment in which this work was conducted

translated themselves into advantages and disadvantages. The most obvious benefits of being part of a multinational company lies on the fact that one is constantly surrounded by different cultures and opinions, that ultimately results in a crash course on maturity and tolerance.

The telecommunications industry is mostly interested in practical solutions that can be quickly adapted to fit the consumer needs, while the academy is not so concerned with the immediate results. Being exactly in the middle of these two worlds gives one the discernment to conclude that, although most of the times the academy has the perfect solution to a problem, the small gap between the two impedes the first from adopting it in the short term. The difference lies with the *apparently different language* that is *spoken* in the two types of organization. However, it is interesting to notice that the academy is, most of the times, ahead of the needs of the moment.

The development of new solutions in the short-term is (obviously) more accentuated outside the research division of the company, and the work there is not oriented towards the formalisation and in-depth analysis of the problems. The urge for fixing a problem results typically from a well-defined need of a client, for which the developers devise often a *best effort* solution. It is of the opinion of the author that the collaboration between the university and the industry results particularly interesting not only in the design of future solutions, but also during the process of finding a better option for a problem. Nonetheless, as the bound between the academy and the student gets weaker, it is easier to be influenced by the business perspective of the company.

As a final remark, the author would like to add that the NSN portfolio did not include a stand-alone security product (as an Intrusion Detection System (IDS)) at the time this thesis was written. NSN considers *network security* as one of the key factors that have to be taken into account when devising network equipment, but that prerequisite is fulfilled by buying the solutions (or the service) to partners specialized in that field, or by adapting the standardized mechanisms to their products. For now, research on security matters is interesting in the scope of the collaboration with those partners and for the creation of intellectual property. In spite of the efforts made in the direction of experimenting some of the concepts described herein in a real network monitoring equipment, such attempt was not successful to the date this thesis was ended.

Abstract

The possibility to interconnect information networks in a global scale system contracted the World to a virtual village, where the distances are measured in *clicks*, information is compressed in keywords, people are communicating and business is being made. Unfortunately, the digital world is frequently menaced by malicious (or, sometimes, naive) intents, aiming to profit from the disturbance of its normal functioning. Due to the mass adoption of the networks as working tools or as content delivery platforms, the application of security policies is not only becoming more important, but also more difficult. The reinforcement of these measures in central network points has been gaining the interest of network operators and administrators, preoccupied with maintaining the levels of *availability* and of *Quality of Service (QoS)* as high as possible. As the amount of traffic handled by those nodes is significantly larger than the one arriving to, or transmitted by, terminal nodes, several techniques based on the fast characterisation of the flows have been tried out. Studies conducted for the traffic captured in central nodes show that the flows resulting from aggregation display a somewhat unusual characteristic, designated by *self-similarity*. This characteristic, which now defines partially the nature of the aggregated traffic, is the consequence of a combination of several factors, being the probability distribution of the sizes of the transmitted files one of the most important ones.

This thesis is focused on the statistic used to measure the degree of self-similarity, known as the *Hurst parameter*, and investigates how that metric is altered by a network attack. To accomplish that, the investigation draws first on the concept of self-similarity and on the several means to estimate its intensity. Each estimator is observed from the points of view of its mathematical formulation and practical implementation, since both perspectives are equally important for the achievement of the proposed objectives, and for the efficient application of these mathematical tools in high-speed network systems. The approach is directed towards the identification of the methods with the potential to be

altered so as to return an up-to-date value of the Hurst parameter per point submitted to analysis. The first type of methods herein presented enable the assessment of the referred parameter for an increasing set of values, and are designated by *point-by-point estimators*. After discussing the importance of studying the local scope self-similarity characteristics, and under the excuse that the sensibility of the incremental estimates to departures from normality in the data series decreases with the number of points under analysis, the set of alterations that enable the methods to operate on a fixed size, but iterative, window of values is described.

Next, the best way of testing the precision of the modified estimators is pursued, starting with a complete revision to the state of the art in terms of procedures for generation of series of values with the self-similarity structure, giving special emphasis on the ones that are able to assure the value of the Hurst parameter of the produced sequences. After reaching the conclusion that the complexity or operational model of the available procedures was not compatible with that of the modified estimators, a new generator of sequences exhibiting persistence and following the Gaussian distribution is described with detail and evaluated. The algorithm is exact for aggregation scales that are powers of 2, and configures the ideal solution for the problem at hand.

Inspired by the previous success, a second algorithm, based on a different reasoning that improves significantly the precision of the procedure without compromising its computational speed, is proposed and developed. Its advantages as a number generator are notorious, since it is, at the present time, one of the most efficient generators of its class of precision. Presenting a complexity of $O(n)$ and requiring extremely small memory resources, the procedure is particularly useful in the simulation of arbitrarily long processes, as for example synthetic network traffic traces.

To guarantee that all the simulation results are supported by a righteous source of events, a new algorithm for generation of pseudo random sequences, whose bedding lays on an intuitive and simple idea that confers it great quality (in terms of the randomness of the sequences produced), is also presented. The procedure is described with detail, and compared with one of the best pseudo random number generators available in the literature. Its quality is assessed by submitting the outputs of the algorithm to a well known and stringent battery of statistical tests of randomness. Because the above mentioned

generators are built on top of Gaussian variables, the explanation on how the uniformly distributed numbers can be efficiently transformed into normally distributed sequences is also included.

The theory described or developed along the thesis converges to the definition of the type of attack with potential of being detected by means of analysis of the Hurst parameter, and it is on the basis of the proposal of a traffic simulator with approximate self-similar structure. The proposal, which includes modelling both the packet sizes and the inter-arrival times processes, inherits the performance of the procedure for synthesis of self-similar sequences, built on top of the pseudo random number generator.

The analysis of the impact of the anomalies designated by *network intensive attacks* is initially performed by resorting to traces containing real intrusions, captured during a well known experiment of the Lincoln laboratory of the Massachusetts Institute of Technology, which are available for download from the Internet. The results are then consolidated via the application of the modified estimators to synthetic traffic traces injected with data units representative of an intrusion of the above mentioned type. It is demonstrated that the occurrence of such anomalies affects the constant component of the sequence under investigation, leading the windowed estimators to signal an increase of the (self-)similarity between relatively close values, even when the former culminate in an expressive loss of stationarity (and of the property of self-similarity). Starting from these observations, an intrusion detector based on the continuous and real-time analysis of the local scope self-similarity degree is presented. After enumerating the limitations of this traffic characterisation in the dark mechanism and discussing the weight of its main assumption, it is concluded that the usefulness of the approach is written in terms of a first line of defence, and that the new windowed and incremental perspectives are better defined as indicators of the network health status.

Resumo

A possibilidade de interligar as redes de informação num sistema à escala global transformou o Mundo numa aldeia virtual, onde as distâncias se medem em cliques, a informação está compressa em palavras-chave, as pessoas comunicam e milhares de negócios são feitos a todo o instante. Infelizmente, este mundo digital é constantemente ameaçado por intenções maliciosas (ou, por vezes, ingénuas) que visam lucrar, de alguma forma, com a perturbação do seu normal funcionamento. Devido à adopção em massa das redes de informação como ferramentas de trabalho ou como meio de distribuição privilegiado de conteúdos de entretenimento, a aplicação de políticas de segurança não só se está a tornar cada vez mais importante, como também mais difícil. O reforço destas medidas em pontos de rede centrais tem vindo a ganhar o interesse dos operadores e dos administradores de rede, preocupados em manter os níveis de *disponibilidade* e de *Qualidade de Serviço* o mais elevados possível. Como o volume de tráfego que passa nesses nós é significativamente maior do que aquele que é produzido em nós extremos, várias técnicas baseadas na rápida caracterização dos fluxos de dados têm sido tentadas. Estudos efectuados ao tráfego capturado nesses dispositivos centrais mostram que o fluxo resultante da agregação exibe uma propriedade algo invulgar, designada por *auto-semelhança*. Esta propriedade, que agora define parcialmente a natureza do tráfego agregado, resulta da combinação de vários factores, sendo que o mais importante deriva da distribuição de probabilidade dos tamanhos dos ficheiros transmitidos.

Esta tese foca-se na estatística conhecida como o *parâmetro de Hurst*, usada para medir o grau de auto-semelhança, e investiga de que forma é que a referida métrica é alterada por um ataque de rede. Para tal, começa-se por estudar o conceito de auto-semelhança e as várias maneiras de estimar a sua intensidade. Cada estimador é observado do ponto de vista da sua formulação matemática e da sua implementação prática, já que ambas as perspectivas são igualmente importantes para a concretização dos objectivos

propostos e para a aplicação eficaz destas ferramentas matemáticas em dispositivos de rede de alto débito. A abordagem é direccionada à identificação dos métodos passíveis de ser alterados de maneira a devolverem um valor actualizado do parâmetro de Hurst, por cada ponto do processo que lhes é submetido para análise. O primeiro tipo de método desenvolvido é designado por *estimador ponto-a-ponto*, e permite obter o valor do referido parâmetro para uma sequência crescente de pontos. Porque a sensibilidade das estimativas de carácter incremental decresce à medida que o número de pontos processados aumenta, e depois de se constatar a importância do estudo das características de auto-semelhança de âmbito local, investigam-se e descrevem-se as alterações que transformam os métodos em estimadores *móveis* eficientes, que operam sobre uma janela de tamanho fixo, mas ambulante.

De seguida, procura-se a melhor maneira de testar a precisão dos métodos modificados, fazendo uma revisão ao estado da arte em termos de procedimentos de geração de processos auto-semelhantes, recaindo o interesse sobre aqueles que são capazes de assegurar o valor do parâmetro de Hurst da sequência produzida. Depois de se chegar à conclusão de que a complexidade ou o modelo operacional dos procedimentos disponíveis não é compatível com a dos estimadores modificados, é detalhadamente descrito e avaliado um novo gerador de sequências com distribuição de Gauss e dependências de longo-alcance. O algoritmo é exacto para escalas de agregação que são potências de 2, e configura a solução ideal para o problema a ser momentaneamente resolvido.

Inspirado pelo sucesso anterior, é proposto e desenvolvido um segundo algoritmo, baseado numa teoria diferente que melhora significativamente a precisão do procedimento, sem com isso comprometer a velocidade computacional do mesmo. As suas vantagens como gerador são notórias, visto ser actualmente um dos mais eficazes geradores do seu nível de precisão. Ostentando uma complexidade computacional de $O(n)$ e escassos requisitos de memória, o procedimento revela-se particularmente útil na simulação de processos aproximadamente auto-semelhantes extremamente longos, como registos sintéticos de tráfego de rede.

Para garantir que todos os resultados de simulação são suportados por uma fonte idónea de eventos, é também apresentado um novo algoritmo de geração de números pseudo aleatórios, cujo fundamento assenta numa ideia intuitiva e simples, mas que lhe

confere uma grande qualidade em termos da aleatoriedade das sequências produzidas. O procedimento é descrito com afinco e comparado com um dos melhores geradores do mesmo género da actualidade. A sua qualidade é provada com recurso a uma bateria de testes estatísticos de aleatoriedade. É ainda incluída a explicação de como é que os números uniformemente distribuídos são transformados em números gaussianos, sendo estes últimos aqueles que alimentam os dois geradores acima mencionados.

Toda a teoria descrita ou desenvolvida ao longo da tese converge para a definição do tipo de ataque com potencial para ser detectado usando o parâmetro de Hurst, e é a base da proposta de um simulador de tráfego com estrutura aproximadamente auto-semelhante. A proposta inclui a modelização dos processos do tamanho dos pacotes e dos intervalos entre chegadas, e herda a performance do procedimento para a síntese de incidências de um processo auto-semelhante que, por sua vez, é construído sobre o gerador de números pseudo aleatórios.

A análise do impacto das anomalias aqui designadas por *ataques de rede intensos* é inicialmente conduzida recorrendo a registos contendo ataques reais, capturados durante uma conhecida experiência do laboratório Lincoln do Instituto de Tecnologia de Massachusetts, disponíveis para *download* da Internet. A consolidação das conclusões é posteriormente efectuada através da aplicação dos estimadores modificados a registos sintéticos, aos quais são injectadas unidades de dados representativas de uma intrusão do tipo antes indicado. Demonstra-se que tais anomalias afectam a componente constante do processo estudado, levando a que os novos estimadores móveis assinalem um aumento da (auto-)semelhança entre pontos relativamente próximos, mesmo quando estas ocorrências se traduzem numa expressiva perda de estacionariedade (e da propriedade da auto-semelhança). Partindo destes resultados, é apresentado um detector de intrusões baseado na análise contínua e em tempo-real do grau de auto-semelhança de âmbito local. Depois de se enumerarem as limitações aplicacionais deste método de caracterização de tráfego no escuro e o peso da sua principal assumpção, justifica-se a sua utilidade como primeira linha de defesa, e definem-se as estimativas móvel e incremental como potenciais indicadores da saúde da rede.

Keywords

Anti-Persistence, Correlated Random Walk, Flood Attack, fractional Brownian motion, fractional Gaussian noise, Intrusion Detection, Long Range Dependence, Modified Embedded Branching Process, Modified Variance Time, Monte-Carlo Simulation, Network Aggregation Point, Network Traffic Behavior, Network Intensive Attacks, Online Analysis, Persistence, Prime Remainder Revolution, Pseudo Random Number Generator, Real-Time Analysis, Self-Similarity, Self-Similar Sequences Generator, Sequential Generation Algorithm, Traffic Characterisation in the Dark Mechanism, Traffic Simulation, Windowed Hurst Parameter Estimator.

Palavras Chave

Algoritmo de Geração Sequencial, Análise em Linha, Análise em Tempo-Real, Anti-Persistência, Ataque de Inundação, Ataques de Rede Intensivos, Auto-Semelhança, Comportamento de Tráfego de Rede, Dependência de Longo Alcance, Detecção de Intrusões, Estimador Móvel do Parâmetro de Hurst, Gerador de Números Pseudo Aleatórios, Gerador de Sequências com Estrutura Auto-Semelhante, Mecanismo de Caracterização de Tráfego *no Escuro*, movimento Browniano fracionário, Passeio Aleatório Correlacionado, Persistência, Ponto de Agregação de Rede, Processo Ramificado Embutido Modificado, ruído Gaussiano fracionário, Revolução dos Restos de um Primo, Simulação de Tráfego de Rede, Simulação Monte-Carlo. Variância Tempo Modificado.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Acronyms

4SG	Simple Self-Similar Sequences Generator
AMT	Absolute Moments Time
AMT$_{n=1}$	Absolute Moments Time with $n = 1$
AV	Abry Veitch
AIDS	Application based Intrusion Detection System
BBA BU	BroadBand Access Business Unit
CERT	Computer Emergency Response Team
DDoS	Distributed Denial of Service
DRDoS	Distributed Reflected Denial of Service
DoS	Denial of Service
DPI	Deep Packet Inspection
DWT	Discrete Wavelet Transform
DFA	Detrended Fluctuation Analysis
EBP	Embedded Branching Process
FARIMA	Fractional Autoregressive Integrated Moving Average
fBm	fractional Brownian motion
fBm-SGA	fractional Brownian motion Sequential Generation Algorithm
FFT	Fast Fourier Transform

FTP	File Transfer Protocol
fGn	fractional Gaussian noise
GCD	Greatest Common Divisor (test)
GRNG	Gaussian Random Numbers Generator
HEAF	Hurst Exponent by Autocorrelation Function
HDTV	High Definition Television
HIDS	Host Based Intrusion Detection System
HTTP	Hypertext Transfer Protocol
IC	Information and Communications
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IPSec	Internet Protocol Security
IPTV	Internet Protocol Television
IPv6	Internet Protocol version 6
LAN	Local Area Network
LCG	Linear Congruential Generator
MAMT	Modified Absolute Moments Time
MEBP	Modified Embedded Branching Process
MIT/DARPA	Massachusetts Institute of Technology / Defense Advanced Research Projects Agency
MPLS	Multi Protocol Label Switching
MT	Mersenne Twister
MVT	Modified Variance Time
NLNAR	National Laboratory for Applied Network Research

NIDS	Network Intrusion Detection System
NSN	Nokia Siemens Networks
NSN SA	Nokia Siemens Networks Portugal S.A.
OS	Operating System
OSI	Open Systems Interconnect
P2P	Peer-to-Peer
P2PTV	Peer-to-Peer Television
PC	Personal Computer
POTS	Plain Old Telephone Service
PRR	Prime Remainder Revolution
PRR PRNG	Prime Remainder Revolution Pseudo Random Number Generator
PRNG	Pseudo Random Number Generator
OPSO	Overlapping-Pairs-Sparse-Occupancy
OQSO	Overlapping-Quadruples-Sparse-Occupancy
QoS	Quality of Service
R	Research
RAM	Random Access Memory
RD1	Research and Development 1
RFC	Request For Comments
RMD	Random Midpoint Displacement
RNG	Random Number Generator
RS	Rescaled Range Statistics
RTP	Research, Technology and Platforms
SP	Service Provider

TCD	Traffic Characterization in the Dark
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol over Internet Protocol
TMS	Traffic Monitoring System
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
VR	Variance of Residuals
VT	Variance Time
WMAMT	Windowed Modified Absolute Moments Time
WMEBP	Windowed Modified Embedded Branching Process
WMVT	Windowed Modified Variance Time
WWW	World Wide Web

Abbreviations

Please consider the meaning of the following abbreviations when you find them later in the text:

a.k.a.	also known as;
e.g.	for example;
i.e.	that is to say; in other words;
i.i.d.	independent and identically distributed;
KS test	Kolmogorov Smirnoff test;
Ph.D.	Doctor of Philosophy;
std. dev.	standard deviation;
SYN	Syncronization;
vs.	versus.

Extended Abstract in Portuguese

Introdução

Esta secção, escrita em língua Portuguesa, resume com algum detalhe o corpo da tese com o título "*Study of the Impact of Intensive Attacks in the Self-Similarity Degree of the Network Traffic in Intra-Domain Aggregation Points*". A sua estrutura é semelhante à do documento principal. O seu propósito é o de enunciar o problema a ser tratado, os principais objectivos dos trabalhos de doutoramento e suas contribuições para o avanço da ciência. Cada uma das contribuições é, então, melhor descrita durante o resumo das fases mais importantes da investigação. Por último, apresentam-se as principais conclusões desta tese e mencionam-se algumas linhas de investigação futura. A síntese não inclui quaisquer figuras e omite alguns dos pormenores mais técnicos ou justificações matemáticas, sendo tal facto complementado com apontamentos para o corpo da tese.

Âmbito e Motivação da Tese

A massificação do uso dos computadores como ferramentas de trabalho ou de lazer deve-se não só à sua capacidade de armazenar e processar enormes quantidades de dados rapidamente, como à possibilidade de se ligarem em redes de informação. As facilidades oferecidas e os serviços suportados por estes sistemas de informação tornam-nos não só ferramentas de apoio ao negócio por excelência, como também num íman de más intenções, que normalmente visam lucrar com o distúrbio do seu normal funcionamento. O tipo de ameaças que assombram estas redes vai desde a tentativa de roubo da informação, para fins lucrativos pessoais ou corporativos, até à intenção, por parte alheia, de provocar estragos nas comunicações, só pelo relevo social (louro) que daí possa provir, dentro de uma determinada comunidade de atacantes [5]. O combate a estes problemas motivou

a criação de várias equipas (como por exemplo as equipas conhecidas pela designação inglesa *Computer Emergency Response Team* - CERT [6]) e o desenvolvimento de vários sistemas especializados na área da segurança informática. Um importante grupo destes sistemas são os Sistemas de Detecção de Intrusões (tradução literal da expressão inglesa *Intrusion Detection Systems* - IDSs [7, 10]).

A maior parte dos IDSs são baseados na detecção de assinaturas de ataques conhecidos, cuja base de dados é actualizada regularmente. Ao nível de rede, a detecção com base em assinaturas acarreta, normalmente, o tratamento de enormes quantidades de informação, sendo a tarefa agravada em nós de reencaminhamento, onde passam unidades de dados vindos de, ou destinados a, *várias* máquinas da rede. Porque a sofisticação dos ataques está a aumentar, as máquinas verdadeiramente especializadas são desenvolvidas sobre o conceito de *inspecção do pacote* e, mais recentementente, de *inspecção profunda do pacote* (tradução literal da expressão inglesa *Deep Packet Inspection* - DPI), sendo capazes de analisar automaticamente o conteúdo integral (ou quase) das unidades de dados, e até de reconstruir e investigar os fluxos antes que estes cheguem ao seu destino final. A inspecção profunda de pacotes é uma das técnicas que mais tem vindo a ganhar interesse nas soluções de detecção de intrusões para redes, mas o seu modo de operação é uma tarefa computacionalmente intensiva, que obriga a uma troca notável entre a eficiência do sistema e o seu custo.

O segundo tipo de mecanismos de detecção de intrusões é normalmente denominado de *detecção baseada em anomalias*. A anomalia é entendida como um desvio ao comportamento esperado, dado pelo modelo de sistema adoptado. Tal modelo é programado na máquina de rede, ou aprendido por esta, que se encarrega de tomar as providências necessárias à sua comparação com o modelo de registo de tráfego que está a receber. Nestes casos, o processamento do tráfego restringe-se à análise dos cabeçalhos das unidades de dados ou a campos específicos do seu conteúdo. Quando levado ao limite inverso do modo de actuação da *inspecção profunda do pacote*, este modo de proceder recaí sobre a categoria de *classificação do tráfego no escuro* (tradução literal da expressão inglesa *Traffic Characterization in the Dark* - TCD [15]), sobretudo pela *tentativa* de filtrar tráfego baseado em pouca informação e pressupostos. Os mecanismos com este modelo operacional são computacionalmente mais leves, apresentando-se como uma solução atractiva

para nós de alto débito.

Apesar de alguns modelos de sistema poderem ser aprendidos automaticamente por via de algoritmos de inteligência artificial, há aqueles que dependem da observação empírica e análise crítica de resultados para que possam ser implementados de modo adequado. É precisamente no estudo de uma das mais emblemáticas características do tráfego de rede agregado que esta tese se enquadra. A referida característica é conhecida como *auto-semelhança* [16, 17].

O estudo da auto-semelhança com vista ao seu potencial uso na identificação de intrusões foi motivado por uma combinação de diferentes factores, que podem ser enumerados como se segue: (i) vários estudos provaram que a auto-semelhança é uma das características que definem a natureza do tráfego de rede em pontos de agregação [16, 17, 18, 19]; (ii) à data de início dos trabalhos de doutoramento (2005), existiam apenas dois estudos relativos à aplicação daquele modelo como meio de identificação de intrusões [20, 21], mas baseavam-se na perda da auto-semelhança, e não na análise da evolução do seu grau durante um ataque; (iii) apesar da noção intuitiva da auto-semelhança ser relativamente simples de obter, o seu tratamento empírico é por vezes rodeado de alguma ambiguidade que deriva, na opinião do autor, da própria formulação matemática e interpretação pessoal; (iv) os estudos existentes não se debruçam necessariamente sobre as ferramentas usadas para medir a auto-semelhança, contrariamente à abordagem aqui descrita; (v) o aumento sustentado da velocidade de transmissão das redes de informação justifica a investigação de potenciais mecanismos de detecção baseados em anomalias; e (vi) apesar das soluções de *inspecção profunda de pacotes* apresentarem vantagens indiscutíveis em termos do grau de certeza dos resultados, a tendência crescente para a aplicação de encriptação integral das comunicações resulta favorável ao desenvolvimento de técnicas de *caracterização de tráfego* [15] baseadas em modelos estatísticos.

Descrição do Problema e Objectivos

O problema a resolver nesta tese é o de perceber se a propriedade conhecida como *auto-semelhança* pode ser directamente usada na detecção de intrusões ao nível da rede. Um dos seus principais objectivos é, portanto, o de identificar os ataques que afectam a

referida característica estatística, e de que modo é que esse impacto se pode descrever. Para isso, é necessário entender o modelo do tráfego de rede auto-semelhante, bem como estudar os meios usados para medir a intensidade com que a mencionada propriedade se reflecte nele.

Uma das componentes mais importantes a ter em conta é a aplicabilidade das conclusões e procedimentos desenvolvidos ao longo do trabalho em equipamentos de rede. As análises efectuadas para registos pré-gravados de tráfego (a expressão inglesa *offline analysis* é usada para referir tal investigação), assim como as simulações de rede, devem imitar a análise e o funcionamento em linha, temendo sobretudo as limitações de processamento e memória nesses equipamentos. O funcionamento em linha presume a potencial análise e actuação do dispositivo em tempo útil, sobre conjuntos de valores que são constantemente actualizados.

A maior parte dos estimadores do grau de auto-semelhança disponíveis na literatura são usados de modo *retrospectivo*, acarretando uma ordem de complexidade que não é compatível com a sua implementação em linha. Um dos objectivos desta tese é identificar os motivos da onerosidade destes procedimentos, e procurar maneiras de contornar esse problema. A investigação é orientada de modo a apontar a melhor maneira de observar a evolução do grau de auto-semelhança em tempo real, e a maximizar a sua susceptibilidade a potenciais intrusões.

Porque o tema da auto-semelhança está por vezes exposto a alguma ambiguidade, torna-se mais vincada a necessidade de conciliar os resultados práticos com o argumento teórico. O estudo do impacto de ataques no grau de auto-semelhança segue ambas as vertentes, de modo a obter uma descrição mais exacta do comportamento em análise. A análise prática incorpora o processamento de registos de tráfego verdadeiros e simulados, sendo particularmente útil a modelização de tráfego em nós de agregação de elevado débito, visto ser nesses pontos que a auto-semelhança encontra maior enlevo.

Principais Contribuições para o Avanço do Conhecimento Científico

A primeira contribuição desta tese é a descrição e formulação matemática de estimadores *móveis* e *ponto-a-ponto* do parâmetro de Hurst. A modificação a que estes estimadores foram submetidos permite a sua implementação em máquinas de monitorização de tráfego de rede e o estudo da evolução do grau de auto-semelhança de uma série de dados em tempo real. Os referidos métodos foram construídos a partir de estimadores existentes na literatura, e todas as modificações que sofreram foram descritas com detalhe no capítulo 3 desta tese, e referidas nos artigos [29, 30].

A segunda contribuição é a proposta de dois geradores de processos auto-semelhantes. Os dois novos algoritmos apresentam um excelente rácio qualidade / eficiência, e são descritos como os mais eficientes da sua classe de precisão. Para além de modestos, em termos de requisitos de memória e processamento, estes algoritmos são capazes de gerar sequencialmente (ponto-a-ponto), sem que isso aflija a sua performance. A teoria sobre a qual os dois geradores foram construídos encontra-se detalhada no capítulo 4 desta tese e nos artigos [31, 32].

A terceira contribuição diz respeito à proposta de um novo gerador de números aleatórios chamado *Revolução dos Restos de um (Número) Primo* (tradução literal da designação inglesa *Prime Remainder Revolution - PRR*). A versão descrita no capítulo 4 desta tese é uma das possíveis implementações óptimas da família de geradores descrita e amplamente analisada em [33].

A quarta e última contribuição desta tese concretiza-se nos meios usados e nas conclusões do estudo do impacto de ataques intensivos no grau de auto-semelhança do tráfego de rede. Esta análise inclui a modelização e simulação de tráfego em pontos de agregação intra-domínio, bem como a simulação de ataques intensivos ao nível da rede. O simulador elabora numa teoria fácil de entender que, aliada aos geradores de sequências auto-semelhantes apresentados, permite a geração de registos de tráfego arbitrariamente longos, possuindo a propriedade da auto-semelhança. A referida aliança permite a criação dos registos pacote-a-pacote e a simulação de ligações de alto débito com poucos recursos computacionais. A conclusão desta tese delimita claramente a aplicabilidade da teoria da auto-semelhança na detecção de intrusões de rede. A simulação de tráfego em pontos de

agregação intra-domínio usando um gerador de processos auto-semelhantes sequencial foi discutida com detalhe em [31], e o estudo aos registos de tráfego, bem como as principais conclusões que daí foram obtidas, foram relatados em [29, 30]. Nesta tese, os dois relatos fazem parte do capítulo 5.

Auto-Semelhança e Métodos de Estimação do Parâmetro de Hurst

O início deste trabalho focou-se na investigação do conceito da auto-semelhança e da sua relação com o tráfego de rede. Dado a intensidade dessa característica se poder medir pela estatística conhecida como *parâmetro de Hurst*, redireccionaram-se depois os esforços para os meios usados na sua estimação. O capítulo 2 desta tese contém a definição matemática da auto-semelhança e discute o porquê da sua *presença* nos registos de tráfego agregado. A última parte do capítulo é dedicada à descrição dos métodos mais populares para determinar o valor do parâmetro de Hurst.

Auto-Semelhança e Parâmetro de Hurst

A auto-semelhança é uma propriedade estatística. A sua definição é normalmente feita recorrendo a uma das duas equações seguintes:

$$X(t) \stackrel{d}{=} a^{-H} X(at),\ a \in \mathbb{N}; \tag{1}$$

$$Y(t) \stackrel{d}{=} m^{1-H} Y^{(m)}(i), \text{ onde } Y^{(m)}(i) = m^{-1} \sum_{j=im}^{(i+1)m-1} Y(j),\ m \in \mathbb{N}. \tag{2}$$

A equação (1) aplica-se a processos estocásticos incrementais $\{X(t)\}_{t\in\mathbb{N}}$, como o movimento Browniano fraccionário (tradução da expressão inglesa *fractional Brownian motion* - fBm) (ver Figura 2.2), enquanto que a segunda é normalmente aplicada ao processo dos incrementos $\{Y(t)\}_{t\in\mathbb{N}}$, como o ruído Gaussiano fraccionário (tradução da expressão inglesa *fractional Gaussian noise* - fGn) (ver Figura2.3). O segundo tipo de processos deriva do primeiro (ou vice-versa) por intermédio de uma expressão equivalente

a $Y(t) = X(t) - X(t-1)$ (para $t > 0$). Repare-se que os processos estão definidos para um domínio temporal discreto (a definição pode estender-se a domínios contínuos, mas essa possibilidade é irrelevante no contexto deste resumo). O símbolo $\stackrel{d}{=}$ denota *igualdade em distribuição*, e afirma que os processos de um lado e do outro das expressões só precisam ser *iguais em termos probabilísticos*. As concretizações de m são por vezes denominadas por *blocos* ou *escalas de agregação*.

O processo $\{X(t)\}_{t\in\mathbb{N}}$ é dito auto-semelhante, com parâmetro de Hurst $0 < H < 1$, se e somente se, a equação (1) for verdadeira para todo o $a \in \mathbb{N}$ ou, alternativamente, se o processo das diferenças de primeira ordem $\{Y(t)\}_{t\in\mathbb{N}}$ respeitar a equação (2) para qualquer $m \in \mathbb{N}$. Por vezes também se diz que $\{Y(t)\}_{t\in\mathbb{N}}$ é auto-semelhante no sentido dado por (2) [44].

Se a autocorrelação de $\{Y(t)\}_{t\in\mathbb{N}}$ e de $\{Y^{(m)}(i)\}_{i\in\mathbb{N}}$ for a mesma para todo o $m \in \mathbb{N}$, então $\{Y(t)\}_{t\in\mathbb{N}}$ é dito ser *exactamente auto-semelhante de segunda ordem*. Se, por outro lado, a autocorrelação de $\{Y(t)\}_{t\in\mathbb{N}}$ e a de $\{Y^{(m)}(i)\}_{i\in\mathbb{N}}$ só coincidirem para $m \to \infty$, o processo é dito ser *assimptoticamente auto-semelhante de segunda ordem*. Quando o parâmetro de Hurst é superior a 0.5 e inferior a 1, $\{X(t)\}_{t\in\mathbb{N}}$ (ou $\{Y(t)\}_{t\in\mathbb{N}}$ no sentido dado por (2)) exibe a propriedade da *persistência* ou da *dependência de longo-alcance*, enquanto que para valores entre 0 e 0.5 (exclusivamente), os mesmos processos exibem a propriedade da *anti-persistência*, sendo sobretudo regidos por *dependências de curto-alcance*. Quando o parâmetro de Hurst é 0.5, diz-se que o processo estocástico não possui memória [40, 41].

Expressão da Auto-Semelhança no Tráfego de Rede

O estudo que chamou a atenção da comunidade científica para a relação entre auto-semelhança e o tráfego de rede foi levado a cabo por Leland *et al.* e reportado exaustivamente em [39], publicado em 1994. O artigo entitulado *On the Self-Similar Nature of Ethernet Traffic* apresenta os resultados do estudo conduzido para registos de tráfego de uma rede de área local implementada sobre *Ethernet*. Segundo a referida análise, a *dependência de longo-alcance* é fruto da agregação de processos com memória curta, mas cuja função de distribuição de probabilidades é uma curva com cauda alargada (como

exemplo, considere a distribuição de Pareto). Em [19, 39], cada fonte de tráfego (cada nó terminal) é modelada como uma variável independente que toma o valor 1 (=ON) sempre que o terminal está a transmitir, e o valor 0 (=OFF) quando está em silêncio. O número de bits por unidade de tempo que chegam a um determinado ponto de rede meeiro é então o resultado da agregação (soma) de diversos processos concorrentes, independentes e identicamente distribuídos, e é precisamente nesse aspecto do tráfego que a auto-semelhança se revela (considere visitar a secção 2.3.2. para obter uma figura matemática destas palavras). O processo do número de bits por unidade de tempo é auto-semelhante no sentido dado por (2) e a sua fisionomia é parecida com a do fGn (a chamada prova gráfica da auto-semelhaça no modelo de tráfego de rede é dada pela Figura 2.5). O sustento deste modelo recai sobre a distribuição dos tamanhos dos ficheiros ou mensagens transmitidas, e na própria forma como o tráfego é tratado (guardado na memória da placa de rede ou na da aplicação que o gerou) antes de ser enviado para a rede.

Os Estimadores do Parâmetro de Hurst

A maior parte dos métodos de estimação do parâmetro de Hurst elabora no carácter retrospectivo da definição de auto-semelhança, e propõem a repetida análise da mesma série de dados para diferentes níveis de agregação como meio de o obter. O seu modelo operacional traduz-se muitas vezes num custo computacional que inviabiliza a aplicação em tempo-real dos procedimentos, se implementados na sua forma original. Métodos como o das *Estatísticas Re-escaladas* (da designação, em inglês, *Rescaled Range Statistics* - RS), da *Variância Tempo* (*Variance Time* - VT), dos *Momentos Absolutos Tempo* (*Absolute Moments Time* - AMT), do *Processo Ramificado Embutido* (*Embedded Branching Process* - EBP), da *Análise da Flutuação não Inclinada* (adaptação da designação inglesa *Detrended Fluctuation Analysis* - DFA), de *Higutchi*, ou daquele baseado em *Onduletas*, elaboram em concretizações das fórmulas (1) ou (2) para diferentes operadores estatísticos conhecidos ou propositadamente concebidos para o efeito (ver por exemplo a medida definida por Higuchi [93]). A aplicação de uma medida estatística a ambos os lados de $\stackrel{d}{=}$ reduz aquele operador a uma simples igualdade, permitindo baixar o expoente H por via da função logaritmo. A forma final das transformações é linear e H é uma função do declive da recta, podendo ser obtido através de análise regressiva. Já o método baseado na função de

autocorrelação, o do *Periodograma* ou o de *Whittle* são construídos sobre as propriedades espectrais dos processos auto-semelhantes, e dependem da aproximação paramétrica das suas transformadas, ou do tratamento adequado da função autocorrelação.

A investigação inicial foi portanto direccionada à identificação dos métodos com potencial de serem usados como ferramentas de monitorização de sequências de dados crescentes. Da análise foram destacados os métodos VT, AMT e EBP, sendo evidenciadas ao longo da secção 2.4. as razões por não se considerarem os outros métodos. Apesar de serem aqui considerados separadamente, o AMT não é mais do que a generalização do VT, já que a variância é o momento absoluto de ordem 2. Por uma questão de coerência, a demonstração de como o AMT pode ser implementado no modo ponto-a-ponto e móvel para momentos absolutos de ordem par ou de ordem 1 é incluída no capítulo 3 desta tese. Contudo, este estimador não é usado na prática para momentos de ordem superior a 2, por não adicionar nada de novo às estimativas devolvidas pelo VT. A versão retrospectiva do AMT para momentos de ordem 1 foi usada na avaliação do segundo algoritmo de geração de sequências auto-semelhantes, descrito no capítulo 4.

O Método Variância Tempo

O VT elabora na particularização de (2) dada por $\mathbb{V}(Y) = m^{2-2H}\mathbb{V}(Y^{(m)})$, onde $\mathbb{V}(Y)$ denota a variância do processo a que está aplicada (neste caso $\{Y(t)\}_{t\in\mathbb{N}}$ e $\{Y^{(m)}(i)\}_{i\in\mathbb{N}}$). Se a sequência de pontos estudada possuir a estrutura da auto-semelhança, as coordenadas $\left(\log(m), \log\left(\mathbb{V}\left(Y^{(m)}\right)\right)\right)$ devem ser razoavelmente bem aproximadas por uma recta, para quaisquer concretizações de $m \in \mathbb{N}$ (por exemplo, para $m_k = 2^k, k = 1, 2, ..., K, K \in \mathbb{N}$). Do declive β dessa recta deduz-se uma estimativa de H usando a fórmula $H = 1 - 0.5\beta$. É o facto do estimador VT ser construído sobre uma estatística que pode ser calculada (recalculada) cumulativamente que permite a sua exploração no âmbito deste trabalho, e a sua promoção a estimador ponto-a-ponto e móvel.

O Método do Processo Ramificado Embutido

O proponente do EBP [69] baseou-se no facto da expansão espacial de qualquer processo auto-semelhante no sentido dado por (1) ser dependente do parâmetro de Hurst.

Em [69], esta dependência é explorada por via da definição de famílias de *linhas horizontais de cruzamento* $f(k) = \delta 2^k \mathbb{Z}$ (onde δ denota a distância mínima de cruzamento considerada e $k = 1, 2, ..., K$), que são *atravessadas* várias vezes pelo processo em análise durante a sua evolução temporal. Se N_k contar o número de vezes que cada família de linhas é cruzada pelo processo e μ for a média ponderada desses valores, então uma estimativa do parâmetro de Hurst pode ser obtida de $H = \log(2)/\log(\mu)$, ou do declive da recta que aproxima os pontos com coordenadas $\left(\log\left(\frac{N_{k-1}}{N_K}\right), \log(2^k)\right)$. O motivo pelo qual o autor se decidiu pela modificação deste método deve-se ao facto da contagem do número de cruzamentos poder ser feita de modo incremental, o que deixa adivinhar a sua predisposição para tal adaptação.

O EBP é especialmente útil na análise de processos persistentes, e apresenta problemas de convergência (em termos de significância estatística da estimativa) acentuados para processos anti-persistentes. O método requer que o processo das diferenças de primeira ordem esteja devidamente centrado na origem e, caso não esteja normalizado, que o valor de δ seja escolhido em conformidade com o desvio padrão do processo das diferenças absolutas.

Estimação Eficiente e Móvel do Parâmetro de Hurst

É no capítulo 3 da tese que se colocam as hipóteses de implementação *ponto-a-ponto* e *móvel* (adaptação da designação inglesa *windowed [estimator]*) como meio de obter uma perspectiva da evolução do parâmetro de Hurst contínua, compatível com a operação em tempo-real. A descrição inclui a formalização matemática das modificações aos métodos EBP, VT e AMT, embora neste resumo não se considere este último, por motivos já aqui mencionados.

Estimação Ponto-a-Ponto

O *estimador ponto-a-ponto* é um algoritmo de $O(n)$ capaz de devolver um valor do parâmetro de Hurst para *cada* ponto da série em análise. O seu modelo operacional é ilustrado pela Figura 3.1, onde se realça a necessidade de definir *variáveis auxiliares* (que

basicamente sumarizam o processamento já efectuado sobre pontos anteriores do sinal) e de formalizar um algoritmo recursivo que opera sobre estas variáveis.

Método do Processo Ramificado Embutido Modificado

O primeiro método para o qual se apresentam as modificações necessárias ao seu funcionamento em linha é o EBP, intitulando-o de *EBP modificado* ou, em inglês, *Modified Embedded Branching Process* (MEBP). A implementação deste método no modo ponto-a-ponto requer que a construção do processo ramificado embutido se faça segundo uma perspectiva de verticalidade, que obriga a que todas as famílias de cruzamento $f(k) = \delta 2^k \mathbb{Z}$ sejam visitadas sempre que um novo ponto se torna disponível. As *variáveis auxiliares* são os *pontos de cruzamento* do sinal até ao momento mais recente, denotados por $CL_k(t-1)$, e o *número total de cruzamentos* $N_k(t-1)$ para cada um dos níveis em consideração. Quando um novo ponto $X(t)$ chega, verifica-se se o sinal atravessou alguma(s) das rectas $f(k) = \delta 2^k \mathbb{Z}$ e, em caso afirmativo, conta-se o número de vezes que o fez através da fórmula $C_k(t-1) = \left\lfloor \frac{|X(t) - CL_k(t-1)|}{\delta 2^k} \right\rfloor$. O número total de cruzamentos pode então ser renovado usando $N_k(t) = N_k(t-1) + C_k(t-1)$, tal como o último ponto de cruzamento ($CL_k(t) = CL_k(t-1) + sinal\,(X(t) - CL_k(t-1)) \times \delta 2^k C_k(t-1)$, onde $\lfloor . \rfloor$ é a função chão, por vezes conhecida como função de truncagem). O parâmetro de Hurst pode ser obtido dos novos valores, já de acordo com o racional do EBP.

A normalização em tempo real da série empírica à entrada do MEBP é conseguida à custa do cálculo incremental da média e da variância, formalizado pelas expressões (3) e (4), respectivamente:

$$\mathbb{E}_t(Y) = \frac{(t-1) \times \mathbb{E}_{t-1}(Y) + Y(t)}{t}; \tag{3}$$

$$\mathbb{V}_t(Y) = \mathbb{E}_t(Y)^2 - (\mathbb{E}_t(Y))^2\,. \tag{4}$$

O valor $X(t)$ que deve ser usado na contagem de cruzamentos é obtido de $X(t) = X(t-1) + Y(t)$, onde $Y(t)$ é o resultado da normalização do valor empírico mais recente da série das diferenças de primeira ordem, simbolizado por $Y'(t)$ (i.e. $Y(t) = (Y'(t) - \mathbb{E}_t(Y'))/\sqrt{\mathbb{V}_t(Y')}$). Infelizmente, o viés introduzido por esta operação afecta a contagem

de cruzamentos do processo às várias famílias de rectas, pelo que se deve ter em conta que o MEBP pode estar a sobrestimar o parâmetro de Hurst no caso da série de valores empírica ter um desvio padrão maior que o esperado, ou a subestimá-lo no caso contrário.

Método Variância Tempo Modificado

O segundo método para o qual são apresentadas modificações é o VT, sendo a sua implementação ponto-a-ponto designada, em inglês, de *Modified Variance Time* (MVT). As *variáveis auxiliares* deste método foram identificadas como sendo as *variâncias do processo e das suas respectivas agregações*, que podem ser actualizadas de maneira recursiva usando fórmulas equivalentes a (3) e a (4). Contudo, neste caso, há que ter em conta que o bloco de agregação mais recente da escala m_k, $Y^{(m_k)}(T_k)$, pode não estar completo à chegada do valor mais recente de $\{Y(t)\}_{t\in\mathbb{N}}$ (note que $Y(t)$ contribui apenas com $1/m_k$ para o valor de $Y^{(m_k)}(T_k)$). Para colmatar este problema, é proposta a substituição de $Y^{(m_k)}(T_k)$ pela aproximação definida por $Y_{aux}^{(r_k)}(T_k) = r_k^{-1}\sum_{j=T_k m_k}^{T_k m_k + r_k} Y(j)$, que converge para $Y^{(m_k)}(T_k)$ à medida que o número de valores disponíveis para a construção do bloco $Y^{(m_k)}(T_k)$ tende para m_k, ou seja, à medida que o número inteiro r_k evolui para m_k (r_k denota o número de valores disponíveis para construção do bloco $Y^{(m_k)}(T_k)$). As várias variâncias vão sendo actualizadas com estas variáveis auxiliares, até serem definitivamente substituídas pelo seu valor correcto quando $Y_{aux}^{(r_k)}(T_k)$ iguala $Y^{(m_k)}(T_k)$.

O estratagema aplicado não respeita rigorosamente a definição do VT durante toda a execução ponto-a-ponto, mas a discussão contida na secção 3.2.2. deixa claro que os efeitos deste desvio são tangenciais (só a realização mais recente de $\{Y^{(m_k)}(i)\}_{i\in\mathbb{N}}$ é que é aproximada). Porque todos os valores aproximados utilizados na estimação estão constantemente a convergir para os valores exactos, independentemente da série de entrada estar normalizada ou não, este método não herda viés de nenhuma dessas causas, apresentando-se como uma escolha mais robusta que o MEBP.

Estimação Móvel

As ferramentas brevemente enunciadas anteriormente permitiram ao autor obter um gráfico da evolução do grau de auto-semelhança de uma sequência de pontos, mas cedo levou também à conclusão de que essas ferramentas não eram suficientes para alcançar os objectivos dos trabalhos de investigação. Os estimadores incrementais sofrem do efeito da *lei dos grandes números* pelo que, após a amostra da população ter atingido um certo tamanho, o estimador deixa de ser sensível a potenciais variações no sinal de entrada (a confirmação desta afirmação pode ser obtida, por exemplo, da observação da Figura 3.5 e da Figura 3.6). Esta constatação cativou a investigação dos *estimadores móveis*, cujo racional assenta na definição de uma *janela de observação deslizante*, retratada na Figura 3.2. A máquina conceptual de um estimador móvel é, por sua vez, ilustrada na Figura 3.3, onde é realçado o facto do estimador móvel continuar a produzir um valor do parâmetro de Hurst por cada ponto da série, através da incorporação do efeito de novos valores à medida que chegam e da eliminação do efeito dos pontos que saem da janela de observação. Note-se que a verdadeira novidade desta abordagem está, contudo, na eficiência e na operação ponto-a-ponto do algoritmo, e não na possibilidade de estimar o grau de auto-semelhança de âmbito local.

Tanto o EBP como o VT foram adaptados à filosofia móvel. As respectivas implementações foram apelidadas (em Língua Inglesa) de *Windowed Modified Embedded Branching Process* (WMEBP) e de *Windowed Modified Variance Time* (WMVT). O funcionamento do WMEBP ou do WMVT admite dois estados possíveis: (i) operação ponto-a-ponto, como antes discutido, empregue enquanto a janela de observação não está cheia; e (ii), operação *janela deslizante*, aplicada após número suficiente de pontos hajam sido processados. As *variáveis auxiliares* destes procedimentos são, portanto, parecidas àquelas definidas para os métodos ponto-a-ponto, tal como a sua actualização, apesar de serem em maior número. Para além de algumas variáveis auxiliares específicas, o funcionamento do estimador móvel presume o armazenamento de todos os valores da janela de observação. As variáveis auxiliares definidas para o ponto mais antigo da janela de observação são em tudo parecidas com aquelas usadas na adição do efeito de valores de entrada no estimador incremental, inclusive na sua actualização. A maior diferença é que em vez de esses valores serem somados ao estado do estimador, são subtraídos.

No caso do WMEBP e à chegada de um novo valor normalizado, soma-se o número de vezes que este cruza cada família de cruzamentos e subtraí-se o número de vezes que o ponto na última posição da janela de observação o faz, renovando a janela logo de seguida. A normalização do sinal deve, neste caso, ser feita com recurso à média móvel e à sua respectiva generalização para o cálculo do desvio padrão.

No caso do WMVT, a remoção do efeito do valor à saída da janela de observação não é assim tão directa, e de modo a evitar problemas de consistência nos resultados, foi necessário divergir um pouco das assumpções base da filosofia de iteração do estimador móvel. Mais uma vez, o problema prende-se com o facto dos blocos de agregação relativos ao ponto de saída, simbolizados por $Y_{aux}^{(e_k)}(E_k)$, só se encontrarem completos a cada m_k execuções. Em alguns casos, $Y_{aux}^{(e_k)}(E_k)$ pode estar a sobrestimar o real valor do bloco de saída, pelo que o seu uso associado ao sinal de subtracção em fórmulas como a (5) ou a (6), pode resultar em valores da variância negativos. A solução para este problema passa por forçar o WMVT a actuar como um estimador incremental enquanto os blocos de saída não estão completos. Para cada escala de agregação individual e durante esses períodos de tempo, a janela expande temporariamente, sendo a sua integridade restaurada logo que $Y_{aux}^{(e_k)}(E_k)$ iguale $Y^{(e_k)}(E_k)$, altura em que o seu valor é terminantemente removido do estado do método através das duas recursões seguintes:

$$\mathbb{E}_{T_k,w/m_k}(Y^{(m_k)}) = \mathbb{E}_{T_k-1}(Y^{(m_k)}) + \frac{Y_{aux}^{(r_k)}(T_k) - Y_{aux}^{(e_k)}(E_k)}{w/m_k}; \tag{5}$$

$$\mathbb{E}_{T_k,w/m_k}\left(Y^{(m_k)}\right)^2 = \mathbb{E}_{T_k-1}\left(Y^{(m_k)}\right)^2 + \frac{\left(Y_{aux}^{(r_k)}(T_k)\right)^2 - \left(Y_{aux}^{(e_k)}(E_k)\right)^2}{w/m_k}. \tag{6}$$

Análise Crítica e Comparação

Após apresentação formal dos estimadores modificados, discutem-se vários aspectos da sua utilização prática recorrendo a simulação computacional. O gráfico da Figura 3.4, por exemplo, ilustra a análise comparativa de performance efectuada para o WMVT. A comparação é feita em relação à versão retrospectiva do VT, para janelas de observação diferentes, e demonstra que o número de operações que o WMVT efectua não depende do tamanho daquela estrutura de dados. Por outro lado, a secção 3.4.2. chama a atenção

para o facto de que a operação de agregação diminui o número de amostras em análise, o que pode incorrer em situações em que a quantidade de dados é insuficiente para aferir correctamente algumas estatísticas. É argumentado que a escolha do tamanho máximo dos blocos deve ser feita de modo a que as estatísticas sejam calculadas para um número razoável de amostras, e que neste trabalho se considerou o 32 como limite inferior a esse número.

A análise crítica prossegue com a comparação entre estimadores ponto-a-ponto e retrospectivos, e com a avaliação da precisão do MEBP e do MVT. A perspectiva da evolução do grau de auto-semelhança construída a partir destes métodos é ilustrada nos gráficos na Figura 3.5 e na Figura 3.6, respectivamente, e a Tabela 3.1 é composta pelos valores resultantes da compilação estatística dos resultados de simulações intensivas à precisão dos estimadores ponto-a-ponto. Nela se dispõem os valores esperados do parâmetro de Hurst contra a média (e desvio padrão) das estimativas dadas pelos dois métodos incrementais para 1000 sequências aproximadamente auto-semelhantes, e dela se pode concluir que nenhuma das modificações afecta irreversivelmente a exactidão dos métodos.

A diferença entre as estimativas devolvidas por estimadores móveis e incrementais é estudada na secção 3.4.4.. Os gráficos da Figura 3.7 e da Figura 3.8 são o mais perfeito exemplo da vantagem de calcular o grau de auto-semelhança para uma janela deslizante de valores, visto demonstrarem que a referida propriedade toma intensidades diferentes para diferentes períodos de tempo. O sumário da análise à precisão dos estimadores móveis pode ser encontrado na Tabela 3.2, onde os valores esperados do parâmetro de Hurst para séries sintéticas são colocados lado a lado com a média (e desvio padrão) dos valores devolvidos pelo WMEBP e WMVT.

A última comparação do capítulo 3 da tese tem como objectivo princípal demonstrar que as modificações infligidas aos estimadores não afectam de modo irreversível a sua precisão. Para tal, compararam-se os valores devolvidos pelos novos estimadores com os devolvidos pelas respectivas versões retrospectivas. A excelente performance do WMEBP e do WMVT é realçada nos gráficos da Figura 3.9 e da Figura 3.10, respectivamente, onde as curvas de evolução dos dois métodos coincidem quase perfeitamente. Contudo, é na Tabela 3.3 que as declarações de exactidão encontram suporte empírico, por aí se fazer notar que a média do erro absoluto entre as estimativas devolvidas pelas implementações

retrospectivas e as devolvidas pelos estimadores móveis nunca é superior a 1.78E-02.

Geração Eficiente de Sequências Auto-Semelhantes

No capítulo 4, o foco da investigação deixa de estar sobre a estimação do parâmetro de Hurst para se centrar na síntese computacional de sequências com estrutura fractal. Os dois motivos que fomentaram o estudo ao estado da arte nesta área e, mais tarde, o desenvolvimento de dois novos geradores de sequências auto-semelhantes foram: (i) a necessidade de testar os estimadores ponto-a-ponto e móveis e (ii), a necessidade de construir um simulador de tráfego que requer, por sua vez, a utilização de um mecanismo computacional capaz de criar sequências de números que exibam dependência de longo-alcance.

Revisão do Estado da Arte Sobre Geradores de Processos Auto-Semelhantes

A primeira grande secção do capítulo resume os procedimentos que podem ser usados para *gerar auto-semelhança* mais populares. Estes procedimentos são normalmente divididos em *geradores exactos* e em *geradores aproximados* [38], dependendo a classificação do fundamento teórico e do modo como são implementados. Os geradores exactos são normalmente construídos sobre a matriz de covariância do processo estocástico, e elaboram na definição altamente retrospectiva da auto-semelhança para produzir processos com a estrutura fractal. Cada vez que uma ocorrência do processo é produzida, o número de relações que têm de ser tomada em conta pelo procedimento desde a última iteração aumenta, pelo que a sua complexidade *nunca* pode ser inferior a $O(n \log(n))$. Já os geradores pertencentes à segunda categoria são construídos sobre simplificações às assumpções iniciais da auto-semelhança e compreendem uma troca entre a eficiência do algoritmo e a precisão das sequências produzidas, apesar de poderem atingir complexidades de $O(n)$. Em termos de requisitos de memória, os geradores da primera classe são naturalmente mais ambiciosos que os da segunda, já que aqueles presumem o armazenamento integral da série de pontos sintetizada ou de uma representação igualmente exigente.

Da discussão a 3 geradores exactos e a 9 geradores aproximados concluíram-se algumas das propriedades mais desejadas deste tipo de algoritmos: (i) a sua qualidade, em termos do valor esperado para o parâmetro de Hurst; (ii) a eficência computacional, quer a nível de processamento, quer de memória; (iii) a sua capacidade de produzir resultados ponto-a-ponto e à medida que são necessários; e (iv) a sua capacidade para produzir sequências das quais não é conhecida a exacta extensão à partida. De acordo com a linha de investigação tomada no capítulo 3, assumiu-se que qualquer algoritmo que ocupasse não mais que um pequeno múltiplo de $\log(n)$ em memória seria adequado.

O Algoritmo de Geração Sequencial do Movimento Browniano Fraccionário - fBm-SGA

A explicação matemática do primeiro gerador de sequências auto-semelhantes, batizado de *Algoritmo de Geração Sequencial do movimento Browniano fraccionário* (ou, em inglês, *fractional Brownian motion Sequential Generation Algorithm* - fBm-SGA), pode ser encontrada na secção 4.3., dividida em duas partes principais. Começa-se por se descrever um modo de gerar passeios aleatórios persistentes, passando depois para a generalização que permite a aplicação directa das (aqui denominadas de) *probabilidades de persistência* na geração de valores com distribuição de Gauss.

Por construção, o algoritmo é apenas útil na forja de processos com dependência de longo-alcance, em parte porque a persistência é definida como a probabilidade de determinado ponto ser igual a um homólogo mais ou menos longínquo e, por outro, porque essas probabilidades só se encontram definidas para valores do parâmetro de Hurst entre 0.5 e 1. Para esses valores, as referidas probabilidades tomam também valores entre 0.5 e 1, respectivamente. O esquema de dependências definido pelo fBm-SGA encontra-se ilustrado na Figura 4.3 e na Figura 4.4.

Na primeira das duas subsecções antes mencionadas, o raciocínio por detrás do fBm-SGA é apresentado com detalhe para passeios aleatórios com incremento unitário $\{S(t)\}_{t\in\mathbb{N}}$ (com passos iguais a 1 ou -1). O foco é colocado na dedução da fórmula explícita para o cálculo das *probabilidades de persistência* $p_{m_k} = P\left(S(t) = S(t + m_k/2), t \in m_k\mathbb{N}\right)$, definidas para o intervalo $[0.5, 1[$ e para $m_k = 2^k$, com $k = 1, 2, ...$ A segunda parte

da exposição elabora no modo de descrever o algoritmo em termos de ruído Gaussiano correlacionado.A explicação parte do facto da soma de vários processos incrementais unitários poder ser entendida como o número de sucessos ($= 1$) e insucessos ($= -1$) de uma variável com distribuição Binomial com probabilidade de sucesso p_{m_k}, de resto ilustrado na Figura 4.2. Ocorrências de um fGn com estrutura aproximadamente auto-semelhante $\{Y_N(t)\}_{t>0}$ podem então ser geradas por uma codificação da expressão recursiva (7) partindo de $Y(0) = G(0,1)$, onde $G(\mu, \sigma^2)$ simboliza uma incidência particular da variável com distribuição gaussiana com média μ e variância σ^2:

$$Y_N(t) = G((2p_{m_k} - 1) \times Y_N(t - m_k/2), 4p_{m_k}(1 - p_{m_k})). \tag{7}$$

A última parte da discussão matemática do fBm-SGA diz respeito ao procedimento para obtenção do valor de m_k para cada instante t (o alcance da dependência), culminando na apresentação das expressões (4.30) e (4.31). A completa descrição do algoritmo concretiza-se então no último conjunto de condições da secção 4.3.1..

O fBm-SGA foi usado para testar os estimadores modificados apresentados no capítulo 3 desta tese. Por isso, a avaliação da precisão do algoritmo foi feita recorrendo a implementações *retrospectivas* de vários estimadores do parâmetro de Hurst (o EBP, seguindo a sugestão em [69], e em concordância com o que foi escrito no capítulo 2 da tese; o VT, como descrito em e.g. [38, 82]; o RS, como descrito em [38]; e o DFA, como descrito e.g. em [85, 86]). A avaliação é levada a cabo através da instanciação do gerador para valores do parâmetro de Hurst que vão desde 0.5 a 0.99 (com incrementos de 0.01), seguida da síntese e análise de várias sequências com supostas propriedades fractais. O parâmetro de Hurst estimado é então comparado com o esperado. A maior parte dos resultados dos testes de precisão foi compilada estatisticamente e incluída na Tabela 4.1, na Tabela 4.2, e representados nos gráficos da Figura 4.10 ou da Figura 4.11. Todas as experiências foram feitas para escalas do tipo $2 \times 2^k, k \in \mathbb{N}$ (para as quais o método é teoricamente exacto) e do tipo $3 \times 2^k, k \in \mathbb{N}$. Os resultados contidos na Tabela 4.1 e na Figura 4.10 defendem a natureza exacta do método para o primeiro tipo de escalas de agregação, enquanto que os resultados da Tabela 4.2 e da Figura 4.11 mostram que para escalas que não são potências de 2, as sequências geradas parecem ser levemente atraídas para a aleatoriedade e que, portanto, o grau de auto-semelhança tende a ser um pouco

menor que o esperado para tais escalas (a diferença máxima de 4 centésimas entre o valor esperado e o estimado foi obtida para o valor do parâmetro de Hurst de 0.95 usando o RS).

O Gerador Simples de Sequências Auto-Semelhantes - 4SG

A explicação matemática e análise do algoritmo de geração de sequências auto-semelhantes intitulado, nesta tese, de *Gerador Simples de Sequências Auto-Semelhantes* (da designação inglesa *Simple Self-Similar Sequences Generator* - 4SG) podem ser encontradas na secção 4.4.. O 4SG é resultado da preocupação do autor em colmatar algumas das falhas do fBm-SGA, nomeadamente a que remonta à incapacidade de assegurar melhores relações entre pontos e blocos de agregação para escalas diferentes de potências de 2. A ideia sobre a qual se construiu o 4SG é a de que cada ocorrência de um processo auto-semelhante é a soma de várias componentes que podem ou não mudar ao longo da evolução temporal do processo. O desafio consistiu na definição dessas partes, e na discriminação do peso que têm na referida soma.

O primeiro ponto que o 4SG devolve é $Y(0) = \sum_{k=1}^{N_p}(w_{2^k} d_{2^k}(0))$, onde $d_{2^k}(0)$ é uma incidência de uma variável gaussiana, para cada $k = 0, 1, ..., N_p$. A afectação das várias componentes $d_{2^k}(0)$ é feita por $w_{2^k} = \sqrt{(p_2)^{k-1}(1-p_2)}$, para $k < N_p$, ou por $w_{2^k} = \sqrt{(p_2)^{N_p-1}}$, se $k = N_p$, onde $p_2 = 2^{2H-2}$. O número inteiro positivo N_p é, neste caso, o número *finito* de componentes suportado, determinado durante a inicialização do algoritmo. A demonstração de que estes pesos são dados, como se antevia, em função do parâmetro de Hurst está implicitamente contida na explicação, mas interessa realçar a forma simples de p_2, que resulta em parte do facto de se continuar a lidar com blocos de agregação com tamanho 2^k, onde $k \in \mathbb{N}$ e que, em última análise, inspirou a designação dada ao algoritmo.

Cada iteração seguinte requer o cálculo do número de componentes K que mudam no instante t, e a actualização de algumas delas de acordo com a expressão recursiva $d_{2^k}(t) = G\left((2p_2 - 1)d_{2^k}(t-1), 4\sqrt{p_2(1-p_2)}\right)$. É sabido, na parte final da secção 4.4.1., que K é o menor número inteiro a verificar a condição $t \bmod 2^K = 2^{K-1}$, facto que é utilizado na sua resolução. $Y(t)$ é escrito como uma repercussão de $Y(t-1)$, ao qual

são retirados os efeitos de todas as componentes que mudam o seu valor ou sinal, e somadas as novas contribuições ou espelhamentos. Contrariamente ao que acontece com outros geradores aproximados que procuram a redução da complexidade computacional pela truncagem do impacto do passado na sequência produzida, o 4SG atinge o mesmo objectivo pela decomposição da sequência de valores nas componentes que guardam a parte imutável por um período de tempo virtualmente infinito, concentrando-se *apenas* na manipulação das partes variáveis.

De modo análogo ao que foi feito para o seu congénere, a precisão do 4SG foi avaliada por via da simulação de diversos processos auto-semelhantes e da sua subsequente submissão a estimadores do parâmetro de Hurst. Contudo, e desta feita, o intervalo de variação do parâmetro de Hurst ía de 0.01 a 0.99. Os estimadores escolhidos para arbitrar a exactidão do 4SG foram: o MEBP, implementado como descrito na secção 3.2.1.; o MVT, implementado como descrito na secção 3.2.2.; o DFA, codificado como descrito em [67, 86]; o RS, implementado como descrito em [38]; o $\text{AMT}_{n=1}$, codificado como descrito em [38] para a métrica do erro absoluto ($n = 1$); e a versão retrospectiva do estimador AV, implementado como dito em [38, 91].

Na tese, a análise do 4SG é feita após se conhecerem algumas das vicissitudes de um método de geração de fGn *exacto* mas altamente retrospectivo (o método utilizado foi o de Hosking - ver secção 4.4.2.) e de outro *aproximado* com com uma complexidade computacional de $O(n \log(n))$ (um gerador baseado em Onduletas - ver secção 4.4.2.). Da cuidada observação dos gráficos contidos na Figura 4.16 e dos resultados na Tabela 4.5 conclui-se que os estimadores que melhor defendem a precisão do gerador são o AV, o MVT e o $\text{AMT}_{n=1}$. Apesar de serem também estes os métodos que mais abonam em favor do gerador baseado em Onduletas, avaliado antes do 4SG, as estimativas são ligeiramente *melhores* do que as anteriormente dispostas. Quanto aos restantes estimadores, sobressai um comportamento em tudo semelhante ao demonstrado para o gerador baseado em Onduletas e para o de Hosking, com alguns dos métodos a sobrestimarem sobejamente o parâmetro de Hurst quando este se aproxima de 0. A evolução do parâmetro de Hurst de 0 para 1 é acompanhada por uma aproximação gradual das estimativas devolvidas por qualquer um dos métodos utilizados aos valores esperados, revelando claramente a especial apetência do gerador para garantir dependências de longo-alcance. Todos os

resultados conduzem manifestamente à conclusão de que, em termos de qualidade, o 4SG é pelo menos tão bom como o método baseado em Onduletas, mas obviamente não tão bom como um método altamente retrospectivo.

A complexidade computacional do fBm-SGA e do 4SG foi inicialmente testada através de simulações cronometradas, e depois teoricamente comprovada. Os gráficos da Figura 4.12, na secção 4.3.3., bem como os valores da Tabela 4.6 e os gráficos da Figura 4.17, ambos contidos na secção 4.4.3., são demonstrativos do cariz linear da complexidade dos algoritmos, assim como da sua soberba performance em relação a outros geradores usados nas experiências. A discussão teórica clarifica que existe um limite superior ao número de operações necessárias para gerar cada ponto, e que esse limite não depende nem do índice de iteração dos algoritmos nem do número de níveis de precisão ou componentes tomados em consideração. O assunto dos requisitos computacionais dos dois algoritmos é encerrado com o argumento de que a sua execução não requer mais do que o espaço para armazenar mais que um pequeno múltiplo de $\lfloor \log_2(n) \rfloor$ variáveis, onde n é o tamanho da sequência a ser sintetizada.

A possibilidade de sintetizar incidências de um fGn por um período indeterminado de tempo, e de um modo sequencial, permitiu criar os registos de tráfego pacote-a-pacote, e imitar na perfeição o procedimento de captura de unidades de dados em linha. A análise dos registos é também feita dessa forma, visando um dos objectivos desta tese. A elevada performance dos algoritmos permitiu, adicionalmente, que a carga de tráfego simulada fosse na ordem dos Gbps, sem que isso implicasse a exaustão dos recursos de simulação.

A Fonte de Aleatoriedade: a Revolução dos Restos de um Primo

A correcta implementação dos dois algoritmos apresentados depende da capacidade de obtenção de sequências de valores com distribuição de Gauss não correlacionados. O autor desta tese debruçou-se sobre este assunto na secção 4.5., onde é descrito um novo gerador de sequências pseudo aleatórias com distribuição uniforme (com designação, em inglês, de *Pseudo Random Number Generator* - PRNG). A última secção do capítulo 4 discute precisamente a transformação dessas sequências em ocorrências de uma variável gaussiana, através daquele que é conhecido como o *método Polar* [142, 143], e remata o

assunto da complexidade computacional dos algoritmos propostos nesse capítulo.

O PRR é uma das possíveis concretizações da família de geradores descrita em [33] e a figura que melhor o descreve é a Figura 4.21. O conjunto de expressões matemáticas que formalizam a implementação usada no âmbito deste trabalho vão desde aquela referenciada por (4.60), até à que exibe o rótulo (4.66). De acordo com o racional por elas indiciado, o algoritmo produz sequências finitas de números inteiros $[R_n]_{n=0,1,...,N}$ entre 0 e $M-1$, em última análise determinados por $R_n = (F(X_{n+1})W_{n+1} + X_{n+1}^2) \bmod M$, onde X_{n+1} e W_{n+1} são as $n+1$-ésimas ocorrências de duas sequências lineares congruenciais diferentes, isto é, $X_{n+1} = (A_1 \times X_n + 1) \bmod M_1$ e $W_{n+1} = (A_2 \times W_n + 1) \bmod M_2$. $F(X_{n+1})$ denota uma transformação dos números inteiros $0, 1, ..., M-1$ em valores deste mesmo conjunto, sendo a sua representação computacional reflectida do armazenamento integral um vector de inteiros, preenchido durante o processo de inicialização do algoritmo, com recurso a uma fonte de dados aleatória ou pseudo aleatória. Em conjunção com a expressão (4.60), esta estrutura de dados comprime uma tabela de números aqui denominada de *tabela dos restos*, por causa da operação modular.

A explicação contida na secção 4.5. só é válida para o caso particular em que M é primo, apesar do procedimento poder ser generalizado (dentro de certas circunstâncias) para outros valores de $\mathbb{N}$. A descrição contempla os vários pré-requisitos que devem ser preenchidos para a correcta codificação e inicialização do PRR, bem como algumas das suas propriedades. Inclui ainda a proposta de mecanismos de actualização frequente para os parâmetros A_1 e A_2, que não só aumentam o período do gerador, como também melhoram o grau de aparente aleatoriedade.

A análise qualitativa do PRR foi feita recorrendo a uma bateria de testes de aleatoriedade denominada, em inglês, por *The Diehard Battery of Stringent Tests of Randomness* [137, 141]. A Tabela 4.7, contida na secção 4.5.2., contém os resultados dos testes de aleatoriedade conduzidos para o PRR e para outros dois geradores conhecidos da literatura (o *Mersenne Twister*-MT [119, 129] e o PRNG nativo da linguagem Java [130, 131]), que basicamente servem o propósito de fornecer substrato comparativo à análise. A tabela antes mencionada abona a favor da qualidade do PRR e do MT (que passaram *todos* os testes a que foram submetidos), mas também expõe várias lacunas de aleatoriedade no gerador nativo do Java. Destas experiências foi ainda concluído que a largura da tabela

de números teria de ser maior que 512 unidades, ou alguns dos testes não seriam ultrapassados com sucesso. A implementação usada durante os trabalhos de investigação utiliza um total de 1024 *factores de inicialização*, por questões de eficiência de processamento e armazenamento.

É feita também uma análise comparativa à velocidade computacional da implementação em Java do algoritmo proposto. Da observação do gráfico que resume essa análise (ver Figura 4.22), pode-se concluir que a velocidade do PRR se equipara à do procedimento da classe Java Random, e que estes últimos são cerca de 2,5 vezes mais lentos que a implementação optimizada do MT, usada no âmbito desta crítica. O PRR corre a uma velocidade aproximada de 13×10^6 pontos produzidos por segundo.

Simulação de Tráfego e Análise do Impacto de Ataques Intensivos de Rede

O capítulo 5 contém a análise detalhada à evolução do grau de auto-semelhança durante intrusões com expressão significativa ao nível da largura de banda. Os resultados ali contidos foram obtidos maioritariamente por simulação de registos de tráfego. A possível perda da auto-semelhança, durante os referidos ataques, é testada recorrendo a duas abordagens estatísticas diferentes, e todos os resultados são analisados do ponto de vista teórico. O capítulo está, portanto, organizado de modo a contextualizar o assunto da detecção de intrusões, a formalizar a geração de tráfego com estrutura auto-semelhante embutida e a discriminação dos ataques com relevância no trabalho.

Âmbito Aplicacional de um Método de Detecção de Intrusões Baseado no Parâmetro de Hurst

A primeira parte do capítulo 5 discute o âmbito aplicacional de um possível método de detecção de intrusões baseado na análise da auto-semelhança. Para tal, são descritas as formas mais comuns de classificar IDSs, procurando convergir para as categorias que melhor enquadram a referida abordagem.

Um IDS é um dispositivo ou um módulo de software especializado na detecção de incidentes de segurança, partindo da análise de ficheiros, logs, tráfego ou actividade de rede [7, 8, 9, 25, 10]. A sua classificação é usualmente feita nos seguintes quadrantes: o da *fonte da informação* ou *localização do sistema*, que os subdivide em sistemas localizados no anfitrião ou em nós de interligação (em inglês, são usadas as designações *Network Intrusion Detection System* - NIDS ou *Host Based Intrusion Detection System* - HIDS, respectivamente); o do tipo de *abordagem analítica*, que os categoriza em sistemas baseados em assinaturas ou em anomalias; o do tipo de *resposta*, que engloba os sistemas que aplicam medidas de segurança após detecção de uma intrusão e aqueles que apenas emitem alarmes; e, finalmente, o da *celeridade da análise*, que os determina em termos da velocidade de resposta.

Dado a auto-semelhança ser uma propriedade estatística do tráfego de rede, e atingir a sua máxima expressão em pontos de elevada agregação de fluxos, a sua análise com vista à aplicação em sistemas de segurança incorpora um potencial mecanismo para NIDS com funcionalidade de detecção baseada em anomalias. Os métodos de estimação do parâmetro de Hurst desenvolvidos no âmbito deste trabalho permitem a actuação em tempo-real, mas por se tratar de uma tentativa de caracterizar o tráfego no escuro, a resposta não pode tomar contornos maiores que o despoletar de outros procedimentos de investigação e alarmes.

Após o enquadramento da ferramenta, foram descritas várias soluções de segurança comerciais ou de código aberto. Tanto quanto foi possível apurar, a análise da auto-semelhança não faz parte de nenhuma delas, provavelmente porque a construção da perspectiva contínua da evolução do parâmetro de Hurst de âmbito local é nova, ou porque o conceito sempre esteve mais associado a estudos *offline* de gestão de filas de tráfego.

Trabalhos Relacionados

Porque a auto-semelhança é uma das características que distinguem o tráfego em pontos de agregação, alguns estudos [20, 21, 25, 26] encontraram nessa propriedade uma oportunidade para categorizar o tráfego, e para detectar anomalias relacionadas com intrusões. A maior parte desses estudos [20, 21, 26] partem do pressuposto de que a

auto-semelhança é perdida durante um ataque de rede intenso.

No início do artigo de Ming [25], a análise parece direccionada ao entendimento do comportamento do parâmetro de Hurst durante um ataque, mas acaba por concretizar um trabalho confuso, cujas conclusões colidem com as dos outros, e mesmo com as desta tese. A conclusão de que o parâmetro de Hurst *decresce* durante uma intrusão é fruto de uma análise ao *tamanho das unidades de dados*, e não de um estudo do *processo da quantidade de informação por unidade de tempo*.

O trabalho relatado em [21] é mais vocacionado para a descoberta do melhor tamanho da amostra de tráfego, que para a análise da auto-semelhança. O artigo é construído à volta do que Idris, Abdullah and Maarof chamaram de *método de optimização*, cujo racional se resume à análise sucessiva do mesmo registo de tráfego com ataques para tamanhos amostrais cada vez maiores, e à identificação do volume de pontos da melhor taxa de detecção. Após escassa discussão, e sem apresentarem razões teóricas para o facto, o tamanho amostral de 1400 s é indicado como aquele para o qual a taxa de detecção de intrusões com duração superior a 500 s é melhor.

O trabalho descrito em [26] faz referência ao estudo de [25], parecendo corroborar as suas conclusões mas, contrariamente ao esperado, no seu conteúdo é demonstrado que os valores do parâmetro de Hurst *aumentam* durante as anomalias investigadas. No artigo são usadas janelas de observação de 30 minutos, e os registos de tráfego são agregados para unidades de tempo que variam entre os 10 e os 1000 ms. A perda de auto-semelhança é sinalizada por desvios médios superiores a 10^{-3} entre a função de autocorrelação empírica e teórica, mas nada é dito acerca da intensidade ou duração das anomalias que podem provocar esses desvios.

O tipo de ataques que Allen *et al.* se propõem detectar em [20] corresponde ao tipo de ataques abrangidos pelo presente estudo. O tamanho das quantidades amostrais varia entre os 10 e os 30 minutos, dependendo do tamanho dos registos disponíveis e da carga de tráfego, e o valor do parâmetro de Hurst é calculado de 5 em 5 minutos. Um *ataque de exploração de tráfego* (designação usada na referência) é sinalizado quando o valor do parâmetro de Hurst, devolvido pelo método do Periodograma *ou* pelo método de Wittle, é superior a 1, ou inferior a 0.5. O artigo não contém um estudo à evolução do parâmetro

de Hurst, nem refere a possibilidade de existirem *ataques de exploração de tráfego* que não resultem na perda da auto-semelhança.

Em nenhum dos artigos mencionados é mostrada uma evolução contínua dos valores do parâmetro de Hurst, sendo esse um dos principais factores de diferenciação do estudo aqui descrito. As simulações levadas a cabo durante este trabalho de investigação permitiram verificar uma panóplia mais abrangente de cenários de anomalia, e tirar conclusões daí. De igual modo, a implementação dos estimadores aqui proposta permitiu estudar o comportamento do grau de auto-semelhança para janelas temporais mais pequenas (na ordem dos 8 segundos) que em qualquer outra contribuição científica. A interpretação teórica dos resultados não só explica os valores observados, como permite generalizar as conclusões para qualquer cenário de rede que se coadune com a condição do tráfego ser auto-semelhante.

Simulação de Tráfego de Rede

A maior parte dos resultados expressos no capítulo 5 foram obtidos através de simulação computacional. Dado a especificidade do problema não requerer mais do que emulação do tráfego ao nível inferior da camada de ligação de dados, todos os registos de tráfego foram modelados como sequências de *tamanhos de pacotes* e *intervalos entre chegadas*. Convencionou-se que os tamanhos de pacotes eram incidências de uma variável aleatória com distribuição empírica conhecida (por exemplo de [173]) e que, por conseguinte, o seu valor esperado $\mathbb{E}(PS)$ era sabido *a priori*. $\{PS(t)\}_{t\in\mathbb{N}}$ designa a sequência dos tamanhos dos pacotes (do inglês *Packet Sizes* - $PS(t)$), ordenada pela variável natural t. A simulação de uma carga de rede efectiva L (*Load*), dada em relação a uma largura de banda total de BW (*BandWidth*), é conseguida através da geração de $n_{packets}$ *tamanhos de pacotes* e $n_{packets}$ *intervalos entre chegadas* por unidade de tempo, onde $n_{packets} = \frac{BW \times L}{\mathbb{E}(PS)}$. Nestas circunstâncias, a média dos tempos entre chegadas $\mathbb{E}(IA)$ é necessariamente descrita por $\mathbb{E}(IA) = \frac{BW \times (1-L)}{n_{packets}}$ e, dado a sequência ordenada de intervalos entre chegadas $\{IA(t)\}_{t\in\mathbb{N}}$ ser inferiormente limitada por um valor mínimo positivo IA_{min}, decidiu-se que o seu intervalo de variação se devia confinar a $[IA_{min}, 2 \times (\mathbb{E}(IA) - IA_{min})]$.

A impressão da auto-semelhança no *processo da quantidade de informação por*

unidade de tempo foi conseguida através da modelização dos tempos entre chegadas como sendo incidências de um processo fGn, de acordo com $IA(t) = G_H(t) \times \sqrt{\mathbb{V}(IA)} + \mathbb{E}(IA)$, onde $G_H(t)$ denota a ocorrência t de um fGn com parâmetro de Hurst H. Note-se que o desvio padrão do processo pode ser escolhido de modo a que 95.4% ou 99.7% das vezes, o valor sintetizado desta maneira esteja contido no intervalo de variação definido. Para isso, e de acordo com a definição de variável Gaussiana, é apenas necessário decidir se $\sqrt{\mathbb{V}(IA)} = (\mathbb{E}(IA) - IA_{min})/2$ ou se $\sqrt{\mathbb{V}(IA)} = (\mathbb{E}(IA) - IA_{min})/3$.

O gerador de tráfego usado neste trabalho de investigação trunca automaticamente todos os valores que estejam fora do referido intervalo, para evitar inconsistências. A simulação das incidências do fGn é feita através da utilização do algoritmo 4SG e, de modo a minimizar o efeito da truncagem, $\sqrt{\mathbb{V}(IA)}$ é sempre inicializada com o valor $(\mathbb{E}(IA) - IA_{min})/3$.

Definição e Simulação de Ataques de Rede Intensos

Antes de prosseguir com a descrição do modo de como os ataques foram simulados, é feito o comentário ao tipo de anomalias que um método baseado em auto-semelhança tem possibilidades de detectar. Dado a principal estatística em análise estar dependente de uma quantidade amostral necessariamente grande (o parâmetro de Hurst reflecte *dependências de longo-alcance*), o interesse incide naqueles ataques que, a determinada altura da sua investida, ocupam uma quantidade de largura de banda não negligenciável. São estes os ataques que aqui são denominados de *ataques de rede intensos*.

A simulação dos ataques de rede intensos foi feita recorrendo à especificação do parâmetro de Intensidade (I), que determina a quantidade de pacotes que chega ao nó de agregação fictício, por unidade de tempo e em função da largura de banda disponível. Depois de se escolher o tamanho do pacote malicioso (por exemplo, o tamanho de um pacote SYN referente ao protocolo *Transmission Control Protocol* (TCP)), calcula-se a média do ritmo de geração dos pacotes maliciosos. A injecção do tráfego relativo ao ataque é conseguida pela implementação directa do procedimento ilustrado na Figura 5.4. As unidades de dados do tráfego malicioso são simplesmente inseridas no tráfego legítimo por ordem de chegada, podendo isso incorrer no atraso de ambos os tipos de tráfego.

Análise do Impacto de um Ataque no Grau de Auto-Semelhança do Tráfego de Rede

A primeira tentativa de observar o comportamento dos valores devolvidos pelos (recentemente implementados) WMVT e WMEBP perante anomalias materializou-se na aplicação dos dois métodos a um conjunto de registos de tráfego contendo ataques, capturados nos laboratórios MIT/DARPA e disponíveis para *download* em [145]. Alguns dos gráficos escolhidos para representar este primeiro conjunto de experiências podem ser encontrados na Figura 5.5 e na Figura 5.6, onde as manifestações dos ataques de rede intensos designados por *SATAN*, *SYN Flood* e *MailBomb* [178] são ilustradas. Em todos eles é flagrante o aumento das estimativas devolvidas pelo WMVT ou pelo WMEBP durante o ataque, apesar da fraca garantia que estes registos oferecem em termos de estrutura fractal [23] e do possível viés introduzido por *scripts* de emulação de tráfego de fundo.

Foram os dois motivos apontados em último, e a necessidade de controlar melhor a carga de rede e o grau de auto-semelhança do registo analisado, que levaram ao desenvolvimento do simulador de tráfego e ao estudo retratado nas secções 5.5.2., 5.5.3. e 5.5.6.. É aí que se discutem os dois cenários principais que um estimador móvel do parâmetro de Hurst pode enfrentar, e os três possíveis resultados da confrontação.

Um dos primeiros cenários examinados é aquele em que a duração do ataque é menor do que o tamanho da janela de observação. A representação gráfica na Figura 5.7 mostra um dos cerca de 150 histogramas construídos durante esta parte do trabalho, e refere-se mais concretamente à simulação de um ponto de agregação capaz de operar a 1 Gbps durante 30 s, mas cuja carga útil é de 10%. O parâmetro de Hurst do gerador de tráfego legítimo foi inicializado a 0.75, e um ataque com intensidade de 10% e duração de 4 s foi injectado aos 10 s da experiência. Para além da *quantidade de bits por milissegundo* e da respectiva média móvel (calculada para uma janela de observação de 8192 ms), são também apresentadas no gráfico as curvas de evolução do parâmetro de Hurst, calculado usando o WMVT e o MVT. Os três pontos chave da análise são enfatizados na Figura 5.8, onde a janela de observação está também representada (à escala). Ambos mostram que para as circunstâncias em análise, e no caso em que intensidade do ataque não incorre numa perda expressiva da estacionariedade, o valor do parâmetro de Hurst de âmbito

local aumenta enquanto o ataque perdura, retornando ao valor normal apenas quando o tráfego relativo à intrusão sai completamente da janela de observação.

A encenação seguinte diz respeito à situação onde a duração do ataque supera o tamanho da janela de observação. Uma possível ilustração é incluída na Figura 5.8, conseguida quando o gerador de tráfego foi instruído a simular uma carga de rede de 70% e um ataque com 10% de intensidade aos 8 s. O parâmetro de Hurst do tráfego legítimo foi ajustado para 0.80 e o tamanho da janela de observação valia metade da duração da anomalia. Desta feita, o valor das estimativas aumenta apenas durante a fase transitória em que o tráfego relativo ao ataque está a entrar ou a sair da janela de observação. Durante a fase em que o estimador móvel está a observar a parte do registo constituída pela soma do tráfego legítimo e ilegítimo, o valor do parâmetro de Hurst diminui para aquele que foi estimado antes do ataque. Isto acontece porque assim que o ataque é completamente *absorvido* pela janela de observação, os estimadores actualizam totalmente o conjunto das estatísticas para as do processo deslocado, que são aproximadamente iguais às do processo inicial. Nessa situação específica, os estimadores móveis não conseguem distinguir a parte translada do processo em análise, daquele que é considerado legítimo. Assim que o ataque termina, e o tráfego que o concretiza começa a sair da janela de observação, regista-se novo acréscimo significativo nas estimativas antes do seu regresso ao valor considerado legítimo.

Partindo das constatações antes mencionadas, é proposto um detector de anomalias que elabora na perspectiva contínua da evolução do grau de auto-semelhança, o qual foi testado para cerca de 324 combinações do par (L, I). De entre várias estatísticas, o procedimento devolve a diferença máxima entre valores do parâmetro de Hurst local, antes e durante o ataque, o momento apontado como sendo o início do ataque e a duração do mesmo. O início do ataque era sinalizado por estimativas locais do parâmetro de Hurst superiores ao valor esperado em cerca de 0.01, por períodos de tempo superiores a 100 ms; o fim do ataque correspondia ao *regresso* do parâmetro de Hurst a valores próximos daqueles observados antes da intrusão ocorrer. Para facilitar a sua análise crítica, as diferenças máximas entre valores do parâmetro de Hurst foram normalizadas (divididas pelo supremo de todos os valores calculados) e representadas nos gráficos da Figura 5.12, em função de L e de I. As diferenças atingem máxima expressão para aproximadamente

metade das combinações simuladas, tornando-se mais notáveis à medida que a intensidade do ataque aumenta. Já a capacidade do detector em apontar o inicio e o fim dos ataques pode ser extrapolada dos gráficos da Figura 5.13. Qualquer ataque com expressão superior a 30% (em termos de largura de banda total) produz efeito suficiente para que ao menos o início do ataque seja apontado com alguma exactidão.

O tema da potencial destruição da estrutura da auto-semelhança durante um ataque de rede intenso é abordado na secção 5.5.6., onde são discutidos dois testes estatísticos implementados propositadamente para o efeito. O teste baseado no de Kolmogorov Smirnov (*teste K-S*) foi usado para apurar se a distribuição de um registo de tráfego contendo um ataque é ou não *semelhante* à distribuição de várias agregações da série de dados em análise. O segundo teste avalia a qualidade da estimativa devolvida pelo WMVT (para mais detalhes, ver secção 2.4.2.), através do estudo da estatística conhecida como o *coeficiente de determinação*, normalmente simbolizada por R^2 [80]. A análise foi efectuada para o período temporal em que o início do ataque está situado no primeiro quarto da janela de observação.

Os resultados obtidos para o *teste K-S* mostram que a resiliência da propriedade a desfasamentos aplicados à sequência de valores em análise depende do grau de auto-semelhança. Apesar da dependência não aparentar ter uma fórmula explícita que a explique, depreende-se da leitura do gráfico do lado esquerdo da Figura 5.14 que, à medida que o parâmetro de Hurst aumenta, maior é o desfasamento que o processo suporta, antes da destruição da auto-semelhança. Para valores do parâmetro de Hurst entre 0.75 e 0.85, o processo analisado parece ser especialmente resistente às transformações a que foi sujeito, sendo capaz de suportar ataques com intensidades próximas de 30% (igual ao desvio padrão do processo). Já a superfície exposta à esquerda na Figura 5.15 sugere que, se a perda da propriedade estudada fosse definida em função de R^2, só após uma intensidade considerável é que se podia concluir acerca do falhanço da regressão linear da última fase do VT. O valor de R^2 mantém-se elevado mesmo na presença de ataques com intensidade média, pelo que nessas condições não seria possível descartar a possibilidade de existir uma relação exponencial (fractal) entre o processo e as suas agregações com base nessa estatística.

Dos resultados antes discutidos, conclui-se que: (i) apesar da inserção de tráfego ma-

licioso resultar sempre numa perda de estacionariedade, essa perda pode não resultar na destruição da auto-semelhança; e que (ii) o parâmetro de Hurst aumenta *sempre* que um fluxo constante de tráfego é injectado na rede. É sabido, por exemplo de [38], que a estacionariedade dos *processos com dependências de longo alcance* é difícil de avaliar, já que a própria natureza dos processos fractais dita o deslocamento das propriedades estatísticas a nível local. Estes deslocamentos podem, numa primeira análise, ser confundidos com falta de estacionariedade, mas na verdade são apenas produto das autocorrelações. A introdução de modestos (em relação à carga de rede) fluxos de tráfego malicioso pode nalguns casos resultar num deslocamento local e pequeno da largura de banda ocupada, que aumenta momentaneamente a componente constante da série em análise. Alheios à qualidade exacta da série de valores estudada, tudo o que os estimadores utilizados são capazes de observar é, basicamente, a transformação de um processo variável em um mais estável, para o qual a estimativa do parâmetro de Hurst nunca pode ser inferior.

Conclusões Finais

Os quatro capítulos intermédios da tese constituem um dos estudos mais sistemáticos e completos no contexto da análise da auto-semelhança em tráfego de rede. No decurso do trabalho de investigação, foram propostas e testadas várias modificações a estimadores do grau de auto-semelhança. As versões incrementais e móveis do VT e do EBP são algoritmos pouco exigentes em termos de requisitos de memória e apresentam uma complexidade computacional de $O(n)$. Os métodos modificados podem ser vistos como excelentes ferramentas de monitorização em tempo real do tráfego da rede, embora o seu âmbito aplicacional não se confine a essa área. A transformação dos estimadores do parâmetro de Hurst em métodos de cálculo de estatísticas móveis é acompanhada por algumas desvantagens, sobretudo relacionadas com as instabilidades temporais causadas pela construção incremental dos blocos de agregação, mas também por algumas vantagens. Os estimadores móveis permitem o ajuste da relação inversamente proporcional entre a sensibilidade a modificações efémeras do grau de auto-semelhança e a significância estatística das estimativas. Por outro lado, os métodos modificados são também mais tolerantes a perdas de estacionariedade momentâneas, facto que favorece a sua aplicação na análise em linha de séries empíricas.

Foram também propostos dois novos métodos para aproximar processos fBm com estrutura auto-semelhante. Um dos dois (4SG) é apresentado como um dos métodos mais eficientes da sua classe de precisão e, na opinião do autor desta tese, concretiza um dos melhores resultados do trabalho de investigação. Uma média de quatro somas, duas multiplicações e duas execuções de um GRNG por cada ponto gerado fazem da complexidade do algoritmo um marco difícil de superar. Num processador Pentium IV 1.61 GHz, o 4SG debita mais de 1300 pontos por milissegundo e não precisa de mais do que 1 KB de memória para sintetizar uma série com $2^{128} \approx 3.4 \times 10^{38}$ pontos. O algoritmo representa uma solução particularmente atractiva para simulações que pretendam reproduzir situações em que a análise é feita ao mesmo tempo que os pontos se tornam disponíveis, ou para a geração rápida de sequências de valores arbitrariamente grandes, sendo por isso perfeito para a geração de tráfego com estrutura auto-semelhante.

Incluída na tese está também a proposta de um novo procedimento de geração de números pseudo aleatórios com distribuição uniforme. O conceito base do PRR é simples de entender, e o algoritmo é portátil, em termos da sua codificação em linguagem máquina. Na lista das suas melhores propriedades inclui-se a sua eficiência computacional e a possibilidade de melhorar a aparente aleatoriedade das sequências produzidas à custa de recursos de memória, mas não à custa da degradação da sua performance. A implementação do PRNG usada no âmbito deste trabalho passou todos os testes de aleatoriedade a que foi submetido, e o seu período é de 2^{60}. O PRR é capaz de produzir uma sequência com aproximadamente 11×10^6 números pseudo aleatórios em menos de 1 segundo num processador Pentium IV 2.4 GHz, usando apenas 4140 B de memória.

O gerador de tráfego usado no âmbito dos trabalhos de investigação reportados nesta tese foi desenhado para sintetizar os registos pacote-a-pacote, através da devolução ordenada do tamanho das unidades de dados e do intervalo de tempo que as separa. O simulador é apresentado como uma aplicação directa do fBm-SGA ou do 4SG, dos quais herda a capacidade de geração sequencial e o excelente desempenho computacional. Uma das suas maiores vantagens reside no facto de poder ser utilizado na simulação de registos de tráfego extremamente longos, exibindo a propriedade da persistência, sem que isso se reflicta num peso de computação proibitivo para a máquina onde a simulação é posta a correr: a implementação em linguagem Java do simulador é capaz de simular o tráfego de

rede a uma taxa de transferência de dados superior a 8 Gbps, e produz um registo com 100 GB com menos de 340 B de memória.

O completo desenvolvimento de um simulador de tráfego de rede com estrutura aproximadamente auto-semelhante foi um dos maiores desafios deste trabalho, mas dele resultou não só uma excelente ferramenta de experimentação, como também uma nova perspectiva sobre a teoria da auto-semelhança. O esquema por detrás do algoritmo 4SG atingiu, neste caso, particular relevância, já que foi usado na construção da justificação teórica dos resultados obtidos. De acordo com o referido esquema, uma sequência de valores auto-semelhante pode ser aproximada através da soma ponderada de diversos componentes com duração e pesos estatísticos fixos, mas dependentes do valor do parâmetro de Hurst. A ocorrência de alguns tipos de ataques traduz-se na injecção de uma nova componente constante, que pode apenas reforçar a propriedade da persistência, ou destruir-la, mas nunca diminuir o grau de auto-semelhança. Ambos casos foram estudados com detalhe nesta tese.

Com base nos resultados obtidos, conclui-se que a auto-semelhança pode ser usada para detectar anomalias cuja natureza *pode* estar relacionada com ataques de rede intensos. Tal pode ser conseguido utilizando uma técnica que elabore na monitorização contínua dos valores do parâmetro de Hurst, detectando variações abruptas das estimativas, durante períodos de tempo predeterminados. Usando o tamanho da janela de observação como uma variável de entrada é possível apontar o princípio e duração aproximada de qualquer fluxo de tráfego particularmente intenso e longo. Da discussão contida no capítulo 3, pode-se inclusivamente afirmar a possibilidade de aplicar os estimadores ao processo da quantidade de informação por *décima de milissegundo* usando um processador a 2.8 GHz e menos de 4 KB de memória. Devido às métricas em que este tipo de análise se baseia, qualquer método que elabore na estimação do grau de auto-semelhança pode ser colocado imediatamente após os mais simples colectores de informação de tráfego, e usado para levantar alertas úteis acerca de fluxos de dados potencialmente perigosos ou para despoletar outros procedimentos de investigação.

Independentemente dos resultados mais promissores, o tipo de análise aqui investigada não pode ser vista como uma solução isolada ou extraordinária para o problema da detecção de intrusões. O âmbito de aplicação de qualquer método baseado na análise

da auto-semelhança confina-se à intersecção das áreas de acção de NIDSs e dos mecanismos de *caracterização de tráfego no escuro*. Mais do que com qualquer outra estatística comum, a estimação do parâmetro de Hurst requer necessariamente um grande número de amostras, pelo que as únicas intrusões com potencial para afectar a auto-semelhança são aquelas cujo modo de operação recaí na categoria de *ataques de rede intensos*, como é o caso de ataques DoSs (e DDoSs). Para além disso, e no que se refere às estimativas do parâmetro de Hurst, fluxos de dados particularmente intensos produzem o mesmo tipo impacto que os referidos ataques, sendo portanto impossível decidir a legitimidade do tráfego baseado apenas nessa métrica. Por último, há que ter em conta que as assumpções por detrás de uma detecção deste tipo são especialmente ambiciosas, dado ser necessário presumir que o tráfego sob análise exibe (e continuará a exibir), a propriedade da auto-semelhança.

Face a tudo o que foi dito, o autor termina a secção 6.1. com o argumento de que o grau de auto-semelhança do tráfego de rede é melhor definido se compreendido como uma das medidas da sua integridade num dado ponto de agregação. Nesses termos, os estimadores incrementais e móveis não são mais do que as ferramentas que possibilitam a observação da evolução dessa métrica, permitindo que a gestão da rede seja feita também com esse factor em conta.

Possíveis Direcções Para Trabalho Futuro

As principais orientações de trabalho futuro dizem respeito à possibilidade de implementar alguns dos algoritmos e mecanismos desenvolvidos ao longo do doutoramento em máquinas de monitorização de rede reais, e às possíveis repercussões de estar a estimar o grau de auto-semelhança para uma janela de observação deslizante. A análise de registos reais (e em tempo-real) é uma das linhas de investigação futura mais interessantes.

No que se refere aos geradores de sequências pseudo aleatórias e auto-semelhantes, o autor ambiciona investigar a possibilidade de gerar tráfego de rede em tempo real, sob a perspectiva de usar os registos simulados (em paralelo) na antecipação do estado da rede nos momentos que sucedem a análise e geração. A modelização e incorporação de diferentes aspectos do tráfego de rede no simulador apresentado no capítulo 5, como

por exemplo o dinamismo de protocolos das camadas superiores do modelo OSI ou os conteúdos produzidos por aplicações telemáticas, apresentam-se igualmente interessantes. A descrição formal das alterações que fortificam as dependências entre os pontos devolvidos pelo fBm-SGA são também um tópico de trabalho futuro, apesar de já se encontrarem devidamente implementadas e testadas. A explicação de como o 4SG pode ser utilizado para a síntese de processos multi-fractais está também agendada para breve, bem como a descrição do uso do algoritmo para a síntese de ocorrências de uma variável com distribuição gaussiana, partindo de sequências com distribuição uniforme. É ainda insinuado o facto do valor do PRR como primitiva criptográfica não ter sido determinado, e sugerida a análise do comportamento do algoritmo perante outras baterias de testes de aleatoriedade.

A participação num projecto cujo objectivo seja o de criar (e tornar disponíveis) registos de tráfego contendo ataques, capturados em ambientes laboratoriais controlados, faz parte dos planos de investigação futura, já que a *detecção de intrusões e ameaças à rede* é um tópico com bastante interesse na comunidade académica e na indústria, embora seja difícil encontrar registos de tráfego com tais características. Outro tópico a que o autor gostaria de dedicar especial atenção é o dos mecanismos DPI e sua interacção com os estimadores móveis, ou com outros mecanismos de classificação de tráfego no escuro. Os ataques baseados em *ligações aparentemente legítimas*, que perduram como normais durante períodos indeterminados de tempo até atingirem um ponto crítico de ataque, constituem também um bom tópico de investigação, estando o aumento de ataques do tipo DDoS vindos de endereços legítimos na base da motivação para o seu estudo.

Chapter 1

Introduction

1.1. Thesis Focus and Scope

History is the best witness of the popularity that the computational devices have gained in the last few years. Their success is not only due to their ability to quickly perform enormous amounts of operations over equally large sets of data, but also to the possibility to interconnect and organise them in information networks, where the space is said to be *virtual* and mankind communicates freely. The most popular and largest information system interconnects several information networks at the planetary scale, and is known as *Internet*. It allows computational devices to interchange data through a standardized communications suite, known as the Transmission Control Protocol over Internet Protocol (TCP/IP) suite [1, 2], and it organises its constituents in a globally unique address space based in the Internet Protocol (IP) [3, 4]. Along with the telematic applications and devices comprising their interface, these information networks embody nowadays valuable business tools, constituting the origin or the destiny of significant amounts of data. The ease of usage and the diversity of services supported by the Internet or by other peripheral or private networks make them the perfect means to transport important, and often confidential, information (money transaction orders, industrial secrets, etc.). Unfortunately, the reasons that make the technology attractive, are also the reasons that make it a potential target for malicious users, which normally aim to profit from the disturbances of its normal functioning.

New forms of attacking the networks or their constituent components are developed every day. This fact is easily explained if one takes into account that security breaches

arise from a wide combination of factors, as the growth of the Internet or the software applications running on its nodes. The motives and purposes of an attack are numerous: some aim for the retrieval or exposure of private, or confidential, information; others for blackmailing remote entities or to simply erase important data; others yet to distribute publicity to the end users; etc. Some attacks are perpetrated with the malicious intent of gaining something in return; others are simple bad taste jokes, conducted by irresponsible users, just for the pleasure of doing them and receiving the credits for it [5] (within a community of attackers, a successful attempt may be used as a measure of the skill of the attacker). It soon became evident that it was necessary to create specialised communities (e.g. Computer Emergency Response Teams (CERTs) [6]) and systems (or modules of software) whose main function is to *try* to prevent the network from being affected by malicious intents. The systems or modules specialised in the detection of network intrusions are commonly known by Network Intrusion Detection Systems (NIDSs) [7, 8, 9, 10].

It is common to categorise Intrusion Detection Systems (IDSs) according to the analysis technique they employ. *Signature-based IDSs* (a.k.a. *Misuse-based IDSs*) are designed on top of fast pattern matching procedures, whose main function is to compare the sequence of bits under investigation with an *entire* database with signatures of known attacks. At the network level, misuse-based detection requires the device to be capable of handling huge amounts of data in real-time, being that task particularly aggravated in popular aggregation nodes, since they receive a tribute from numerous devices. As the attacks became more sophisticated, the scope of action of the technology behind the NIDSs had to be extended to the payload of the protocol data units, leading to the development of packet inspection techniques.

Within the scope of action of packet inspection techniques, the concept of Deep Packet Inspection (DPI) is one of the features that most sells NIDSs, for it is seen as the best way of assuring total control over the traffic [11]. Nevertheless, the repetitive comparison of sequences of bits with long lists of signatures is computationally expensive, and it results in a remarkable trade-off between the efficiency of the system (in terms of detection rate) and its cost. Handling the traffic flows in high-debit nodes is only possible by resorting to parallel processing, and scales at the expense of the development and installation of dedicated processing cards [12]. This may become a serious limitation with

the advent of higher transmission technologies, as for example the 40 Gbps or the 100 Gbps Ethernet technologies [13, 14], which are currently in the standardisation process.

The second type of IDSs is often termed *Anomaly-based IDSs* (a.k.a. *Behaviour-based IDSs*). The underlying mechanism of this kind of IDSs relies on the description of the correct behaviour of the system (if the scope of action of the IDS is the device it is monitoring) or of the traffic (in the case of a traffic monitoring equipment), and on the means to detect possible deviations from normal behaviour. Contrarily to Signature-based IDSs, Anomaly-based IDSs rely on the definition of normality, instead of defining and storing all the possible undesired occurrences, resulting in a computational benefit for the IDS. The construction of the model or set of premisses that define normal behaviour may either be done by a person skilled in the art of security, or automatically learnt via artificial intelligence algorithms, which are programmed in the device and ran during an initial learning phase. The comparison between what is expected and what is observed is commonly done resorting to statistics, reason by which it is also common to describe the normal behaviour under probabilistic terms. DPI techniques may be employed prior to the application of the anomaly detection function, but that is not so common in this type of IDSs. Due to its nature, DPI is more suitable for Signature-based NIDSs.

The inspection module of an Anomaly-based NIDSs restricts itself to the retrieval of header information, as for example addressing information, transport layer ports, etc. or to specific fields of the contents of the protocol data units. When the reduction of the scope of action of DPI is taken to the opposite limit, it is said that the system is performing (or trying to perform) Traffic Characterization in the Dark (TCD) [15]. The last mentioned operational model requires a small amount of information and may be used to suspect about a malicious intent in situations where the analysis of the contents of the protocol data units is meaningless (e.g. the payload of the protocol data unit follows encrypted) or the attack is unknown. This functional model is inherently associated with a certain degree of uncertainty, which may be reflected in the issuance of false alarms.

The scope of this thesis falls within the area of traffic monitoring and analysis, and it is focused on the investigation of the statistical properties of the network traffic, under the perspective of using them to detect anomalies. The investigation is focused around the property referred to as *self-similarity*, which has been shown to be present in the process

of the bit count per time unit of the traffic in network aggregation points [16, 17, 18, 19], as further detailed in chapter 2 of this thesis.

1.2. Problem Definition and Objectives

The problem this thesis aims to investigate is the one of understanding to what extent the self-similarity property of aggregated traffic is affected by network level intrusions, and the feasibility of using the analysis of the referred property as an intrusion indicator. The motivation for the study of self-similarity under the perspective of its application as a network intrusion mechanism resulted from the combination of different factors, which may be enumerated as follows:

1. The self-similar nature of the network aggregated traffic was already proved by several prominent studies [16, 17, 18, 19]. Thus, it was logical to expect that some intrusion related anomalies would produce some kind of impact in the aforementioned characteristic.

2. At the beginning of this Ph.D. research programme (2005) there were only two studies concerning the application of the aforementioned mathematical model as a means of intrusion detection [20, 21], but they draw on the loss of self-similarity during an attack, and not on the analysis of the evolution of the self-similarity degree. Other papers available at the time [22, 23, 24] establish a relation between the two subjects (*self-similarity* and *intrusion detection*), but they are mostly focused on the importance of assuring the self-similar properties of the traffic when conducting tests to an intrusion detection proposal than on the usage of the metric as an indicator of the intrusion. The pertinence of the subject has motivated the studies in [25, 26], published in the last few years. Despite some of these works present some common points with the one presented herein, they did not alter significantly the initial line of research, being the reasons for that explained in chapter 5 of this thesis.

3. Even thought the intuitive notion of self-similarity is relatively easy to grasp, its application in real-life events is sometimes surrounded by *ambiguity* [27]. In the

opinion of the author of this thesis, this fact is mostly due to the ambivalence of the mathematical formulation of the concept, and to erroneous (sometimes forced) interpretations of its consequences or origins. A large part of this thesis is thus dedicated to the consolidation of the theoretical framework in which the conclusions of the practical analysis are drawn.

4. Previously published studies concerning self-similar network traffic analysis are mostly preoccupied with the conclusions taken from such analysis, rather than with the possible online application of the tools used to measure the self-similarity degree. Most of these tools are not optimized for online monitoring. Herein, these tools are also subject to examination.

5. The increase of the speed of transmission and the proliferation of bandwidth demanding services (High Definition Television (HDTV), Internet Protocol Television (IPTV), Interactive Games, Voice over Internet Protocol (VoIP), Peer-to-Peer (P2P) file sharing applications, Peer-to-Peer Television (P2PTV), web-based multimedia services as *Youtube*, etc.) justifies the investigation of potential means to classify the traffic with fewer resources. In spite of being unsuited for the identification of specific threats flowing in the contents of the protocol data units (malicious code), mechanisms based on statistical models (like the one studied herein) may help to rapidly isolate some menaces, before the application of heavier DPI techniques.

6. Last but not least, the tendency towards the adoption of end-to-end encryption and (protocol) evasion mechanisms results in favour of the development of anomaly-based techniques for traffic classification and intrusion detection [15, 28].

One of the main objectives of this thesis concerns the identification of the attacks whose operational model may produce a measurable effect in the fractal structure of the network traffic. It was critical to study and understand the self-similar nature of the aggregated traffic, as well as the means used to measure the intensity of the referred property (the self-similarity degree).

The applicability of the findings and procedures developed along the investigation on real network equipment was one of the most important objectives of this work. The analysis taken over pre-collected traffic traces (*off-line* analysis) was carried out as if it

was being performed online, while taking into consideration the potential processing and memory limitations of the network devices. Network simulations were also conducted so as to imitate online operation.

As reported in chapter 2, at the time this thesis was written, most of the estimators of the self-similarity degree available in the literature were used in a *retrospective* manner, presenting an order of complexity that was not compatible with an online implementation. One of the objectives of this thesis was to identify the reasons behind their *poor* performance, and seek ways to circumvent that issue. The investigation was focused on finding the best means to observe the evolution of the self-similarity degree in real-time, and on the identification of the method or calculation philosophy with greater susceptibility to potential anomalies.

Because the subject of self-similarity is at times surrounded by some ambiguity, it was important to reconcile any practical result with a theoretical argument. The study of the impact of network attacks in the self-similarity degree was made in both planes (theoretical and observational), so as to obtain a more precise and coherent description of the observed behaviour. The practical analysis included the examination of both real and synthetic traffic traces, being particularly useful the modelling of traffic in high-debit aggregation nodes, since those are the ones where self-similarity finds its best empirical approximation. The goal was to draw conclusions based on the theoretical explanation of the facts, rather than seeking those on experiments only. The accomplishment of this objective was dependent from the correct modelling and simulation of the network attacks as well.

1.3. Thesis Organisation

The **body** of the thesis is constituted by four main chapters, preceded and succeeded by the *Introduction* and *Final Conclusions and Future Work* chapters, respectively. The compilation of the bibliographic references used along the work is included after chapter 6. The contents of each one of the chapters composing this manuscript can be summarised as follows.

Chapter 1 provides the context for the subject on which this thesis is going to elaborate on, identifying the main objectives and the problem to be solved. The synopsis of this manuscript is also included in this first chapter, along with the enumeration and brief description of its main contributions for the advance of Science.

Chapter 2 introduces the mathematical concepts of self-similarity and of Hurst parameter, and contains the definition of one of the most well known self-similar processes: the so-called *fractional Brownian motion*. The important notion of *fractional Gaussian noise* is also formalised there. The mathematical introduction is followed by an explanation on how that concepts relate to the networking area. The chapter ends with the description of a fairly big list of techniques to estimate the self-similarity degree of a data series, discussing their advantages and real-time applicability.

Chapter 3 draws on some of the methods for the estimation of the Hurst parameter, presented in chapter 2, and elaborates on the modifications they require in order to be applicable as real-time estimators. The chapter is composed by three major sections. The first section includes the discussion and mathematical formalisation of the set of modifications that enable the implementation of the methods as point-by-point estimators. The second section discusses the importance of assessing the self-similarity degree inside a limited context observation window, and includes the formal description of the adaptations the methods have suffered to comply with that requirement. The comparison between the newly proposed estimators and the legacy (retrospective) ones is included in the third main section, along with a critical analysis of the results.

Chapter 4 commences by presenting a survey of the most common means to simulate self-similar processes, observing them from the perspective of the computational complexity, and of the quality of the sequences produced, in terms of expected / estimated Hurst parameter values. From that study, the exposition evolves to the detailed description and evaluation of two completely new self-similar sequences generators, devised for the purpose of simulating network traffic, and for testing the previously mentioned windowed and point-by-point estimators. As the algorithms are build on top of a Pseudo Random Number Generator (PRNG), and because the quality of the sequences of numbers that feed the procedures is critical, a completely new PRNG is proposed and formally described. A brief analysis to the algorithm is also included, along with the results of

its submission to a third party battery of tests of randomness. Finally, the means to transform the uniformly distributed numbers into occurrences of a Gaussian variate are briefly discussed, while paying special attention to their efficient implementation for the sake of the purposes of this thesis.

Chapter 5 unifies most of the previously discussed concepts in favour of the objectives of this thesis. It starts by providing a context for the subject of intrusion detection, aiming for the preliminary identification of the type of system that could benefit from the usage of an Hurst parameter estimator. It then evolves to the description of a network traffic model and simulator, which is succeeded by the discussion on the type of the attacks that may impact the self-similarity degree of the network traffic. The effect that an attack may produce in the Hurst parameter estimates, and the potential loss of the self-similarity structure, are then assessed via the analysis of synthetic and real traces, using the windowed estimators and two other well known statistical tests. The findings of this analysis are on the basis of a proposal for an intrusion detection procedure.

Chapter 6 wraps up the most important conclusions of this thesis, and discusses briefly some research topics for future work.

Please notice that most of the chapters contain their own overview of the state-of-the-art in an introductory section, for it was thought to be the most appropriate way to explain each topic.

1.4. Main Contributions for the Advance of the Scientific Knowledge

This section presents the main contributions of this thesis for the advance of the Scientific Knowledge, in accordance with the opinion of the author.

The first contribution of this thesis concerns the description and mathematical formulation of the windowed and point-by-point Hurst parameter estimators. The modifications to what these estimators were subdued to favour their implementation in traffic monitoring and analysis systems, and the study of the evolution of the self-similarity de-

gree of an empirical data series in real-time. The definition of the windowed estimator allows the analysis of a self-similar sequence of values (e.g. the bit count per time unit of the network traffic in aggregation points) in a continuous manner and inside a window of observation, through iterative processing means. This type of analysis (*non-retrospective* plus *windowed* plus *point-by-point*) is novel, since most of the methods are used for offline studies only (where the necessity to process the data in a real-time manner does not exist), and the windowed analysis is mostly performed by means of parallel or repetitive processing of data blocks. The above mentioned methods derive from estimators known from the literature, and all the modifications that they have suffered are described in detail in chapter 3 of this thesis. They were also on the basis of most of the results included in the papers entitled *"Analysis of the Impact of Intensive Attacks on the Self-Similarity Degree of the Network Traffic"* [29] and *"A Evolução do Parâmetro de Hurst e a Destruição da Auto-Semelhança Durante um Ataque de Rede Intenso"* [30]. The former was presented in *The Second International Conference on Emerging Security Information, Systems and Technologies (SECUREWARE 2008)*, which accepted 78 of a total number of 267 papers (an acceptance ratio of 29%), and the latter has won the *best paper award* of the Portuguese conference entitled *Segurança Informática nas Organizações 2008 (SINO'08)*.

The second contribution of this thesis is the proposal of *two* generators for the synthesis of series of values with approximate self-similar structure. The first generator was the result of the necessity to test the modified Hurst parameter estimators for extremely long processes, whereas the second one came as a natural development of the acquired knowledge, partially inspired by the necessity of simulating network traffic traces (exhibiting the self-similar property) in high-speed connections, in an efficient way. The two new algorithms present an excellent trade-off between efficiency and quality, and are described as being amongst the most proficient procedures of their class of precision. Besides being modest, in terms of memory and processing prerequisites, these algorithms are able to produce series of values sequentially (i.e. in a point-by-point manner) and on an on-demand basis, at no cost to their performance. The theory on which the two generators draw on is detailed in chapter 4 of this thesis and in the papers entitled *"Fast Synthesis of Persistent fractional Brownian motion"* [31] and *"The Design and Evaluation of the Simple Self-Similar Sequences Generator"* [32]. The first paper has been accepted for publication with minor changes in the *ACM Transactions on Modelling and Computer*

Simulation, while the second has been accepted for publication with minor changes in the *Elsevier Information Sciences International Journal*, which has an impact factor of 2.147 according to the Journal Citation Reports 2008, published by Thomson Reuters.

The third contribution of this thesis is the proposal of a new PRNG. The algorithm resulted from the necessity of assuring the quality and celerity of the source of randomness of the two previously mentioned generators. It performs as fast as a series of three linear congruential generators, and presents a trade-off between the quality of the outputted values and the quantity of worn-out memory. The version described in chapter 4 of this thesis is one of the optimized implementations of the family of generators thoroughly analysed in the paper entitled *Remainders Revolution Pseudo Random Number Generator* [33], submitted to an international journal.

The fourth and last contribution of this thesis stems from its main goal, and gets concrete in the means and in the conclusions of the study of the impact of intensive attacks in the self-similarity degree of network traffic. The research work includes the modelling and simulation of traffic in aggregation points, as well as the emulation of intensive attacks at the network level. All the concepts described before the aforementioned analysis are used to build the arguments for the results obtained during the experiments conducted to real and synthetic traffic. The simulator elaborates on theory easy to understand that, allied to the aforementioned self-similar sequences generators, allows the creation of arbitrary long traces of traffic with an embedded fractal structure. The referred alliance allows the creation of traffic traces in a packet-by-packet manner and the simulation of high-speed connections with minor computational resources. Moreover, the applicability of the self-similarity theory in the detection of network intrusions is clearly delimited by the main conclusion of this thesis. The simulation of traffic in aggregation points through the usage of a sequential self-similar sequences generator was discussed in detail in the paper entitled *"Fast Synthesis of Persistent fractional Brownian motion"* [31], and the study to several traffic traces containing attacks, as well as the main conclusions that were drawn from there, were reported in [29, 30]. In this thesis, both papers are part of chapter 5.

Chapter 2

Self-Similarity and Hurst Parameter Estimation

2.1. Introduction

The initial stage of the research work was directed towards the investigation of the concept of self-similarity and of its relation with the network traffic. After understanding the concept, and after noticing that the intensity of the fractal character of the network traffic was given by the so-called Hurst parameter, the efforts were redirected towards the means to estimate its value.

The main objective of this chapter is to introduce some of the most important concepts of the self-similarity theory. It starts from the formalisation of the properties that define a self-similar stochastic process, and ends with the non-exhaustive list of methods used to measure the Hurst parameter, interleaved by a small section that establishes the connection between the presented concepts and the networking area. Notice that the formalisation is not intended to be too extensive, and that only the concepts considered useful for the complete comprehension of this manuscript are to be presented. The references included refer the reader to more detailed descriptions of each topic. As one may notice, the definitions contained in this chapter are used in other sections of the thesis. Nonetheless, this fact does not precludes the introduction of other notations and definitions within the context in which they are useful.

2.2. Self-Similarity, Hurst Parameter, Walks, Motions and Noise

The first concepts that are going to be defined are the ones of *self-similarity* and *Hurst parameter*. After that, the formalisation of *random walks* is used to introduce one of the most popular families of self-similar stochastic processes, the so called *fractional Brownian motion*, and its respective incremental process, known as *fractional Gaussian noise*. The author would like to clarify that the aforementioned stochastic processes do not represent the sole class that falls into the denomination of *self-similar processes*. He decided to focus on the referred processes after noticing that their notion was being easily assimilated, and because they are supported by a well developed theory. This choice, however, should not be understood as a statement of ineptitude or dislike towards other commonly adopted models as e.g. Fractional Autoregressive Integrated Moving Average (FARIMA) [34, 35].

2.2.1. Self-similarity and Hurst parameter

Let $\{X(t)\}_{t\in\mathbb{R}_{\geq 0}}$ be a continuous-time stochastic process defined for $t \in \mathbb{R}_{\geq 0}$, and H in (2.1) be the Hurst parameter. The process is said to be self-similar with Hurst parameter H [36] if the (finite-dimensional) distribution of $\{X(at)\}_{t\in\mathbb{R}_{\geq 0}}$ is equal to the (finite-dimensional) distribution of $\{a^H X(t)\}_{t\in\mathbb{R}_{\geq 0}}$ (equation (2.1)), for any real number $a > 0$. In other terms, $\{X(t)\}_{t\in\mathbb{R}_{\geq 0}}$ is said to be self-similar if its statistical description does not change when scaling simultaneously its amplitude by a^{-H} and the time axis by a. Notice that the symbol $\stackrel{d}{=}$, in the following expression, denotes *equality in all finite-dimensional distributions*:

$$X(t) \stackrel{d}{=} a^{-H} X(at). \tag{2.1}$$

Self-similar processes follow a *scaling law* that influences statistically the future state of the process based on its past and actual status. This dependency is given by the Hurst parameter, sometimes referred to as the *Hurst exponent* (see e.g. [37]). Values of the Hurst parameter between 0 and 1, and different from 0.5, characterize *processes with*

memory. When the value of the Hurst parameter is bigger than 0.5 (exclusively) and smaller than 1 (exclusively), it is said that the process exhibits *persistent* behaviour or that it is *long-range dependent* [37, 38, 39]. The process is said to exhibit *anti-persistent* behaviour, or that it is negatively correlated, if the Hurst parameter takes values between 0 and 0.5 (exclusively). The process is said *memoryless*, if the Hurst parameter is equal to 0.5 [40, 41].

As one can notice, the dependence from the past exists whenever $H \neq 0.5$. This means that for values smaller than 0.5, each point of the self-similar process is also dependent from its past realisations, but the range of that dependence decreases as the Hurst parameter tends to 0.

2.2.2. First Order Differences Process

The previous definition applies to the most general case (continuous-time stochastic processes) but, most of the times, it is sufficient and actually more useful to define self-similarity for the class of *discrete-time* stochastic processes with stationary increments (analysis and generation of self-similar series is done for finite sets of values). In such cases, self-similarity can also be described in terms of the so-called *first order differences*, as follows.

Consider that $\{X(t)\}_{t\in\mathbb{N}}$ is a discrete-time stochastic process with stationary increments and that $\{Y(t)\}_{t\in\mathbb{N}}$, given by (2.2), is its first order differences process. Consider also the formulation of the respective aggregated series of $\{Y(t)\}_{t\in\mathbb{N}}$ given by (2.3):

$$Y(t) = X(t+1) - X(t); \tag{2.2}$$

$$Y^{(m)}(i) = \frac{Y(mi) + Y(mi+1) + ... + Y((m+1)i-1)}{m},\ m \in \mathbb{N}. \tag{2.3}$$

$\{X(t)\}_{t\in\mathbb{N}}$ is said to be self-similar, with Hurst parameter H, if the finite-dimensional distributions of $\{Y(t)\}_{t\in\mathbb{N}}$ and $\{m^{1-H}Y^{(m)}(i)\}_{i\in\mathbb{N}}$ are equal for all $m \in \mathbb{N}$. This definition is formalised by the following expression, where $\overset{d}{=}$ denotes *equality in all finite-dimensional*

distributions:

$$Y(t) \stackrel{d}{=} m^{1-H} Y^{(m)}(i),\ m \in \mathbb{N}. \tag{2.4}$$

Dedicate special attention to the positive integer number m, that defines a particular *aggregation scale* each time it is instantiated, and that will come in handy in the subsequent exposition. Notice also that, sometimes, the expressions *aggregation block* or *block size* are used indistinctively to refer to the same notion. The first order differences process is the sequence of individual steps of the stochastic process $\{X(t)\}_{t\in\mathbb{N}}$. Thus, a given *aggregation scale* m_k defines the series of the length of the steps between the two points of $\{X(t)\}_{t\in\mathbb{N}}$, separated by $m_k - 1$ values in the time domain. Each occurrence $Y^{(m_k)}(i)$ is the average step size between incidences of $\{X(t)\}_{t\in\mathbb{N}}$, taken from a sample pool of m_k values.

The autocorrelation function of the first order differences process, denoted by $\gamma(k)$ and defined as usual (expression (2.5)), is typically used to specify a stationary process as being *exactly second order self-similar* or *asymptotically second order self-similar*, depending on whether the autocorrelation of the aggregated series $\{m^{1-H}Y^{(m)}(i)\}_{i\in\mathbb{N}}$ coincides with the one of $\{Y(t)\}_{t\in\mathbb{N}}$ for all $m \in \mathbb{N}$, or only as $m \to \infty$:

$$\gamma(k) = \frac{\mathbb{E}\left(Y(t) \times Y(t+k)\right) - \left(\mathbb{E}(Y)\right)^2}{\mathbb{E}(Y(t) - \mathbb{E}(Y))^2}. \tag{2.5}$$

The same notions are commonly used to define *long-range dependence* [37, 39] as the property of the processes for which the autocorrelation function decays hyperbolically, rather than exponentially. This property suggests a (positive) dependence from the past occurrences of the process, and implicitly justifies the relation in (2.6), where $\beta = 2 - 2H$ and c_γ is a positive constant number. Notice that $\mathbb{E}(Y)$, in (2.5), denotes the *expected value* of the process $\{Y(t)\}_{t\in\mathbb{N}}$ (sometimes also referred to as *expectation*):

$$\lim_{k\to\infty} \gamma(k) = c_\gamma \times k^{-\beta}. \tag{2.6}$$

2.2.3. Random Walk

One of the concepts that most come in handy when studying the subject of self-similarity is the one of *Random Walk*. A *Random Walk* is a discrete-time stochastic process $\{R(t)\}_{t\in\mathbb{N}}$, for which the first order differences process $S(t) = R(t+1) - R(t)$ takes only fixed values exhibiting no correlation at all [42]:

$$\{S(t)\}_{t\in\mathbb{N}} \text{ is a random and discrete variable defined for } t \in \mathbb{N}, \tag{2.7}$$

$$R(t) = 0, \text{ for } t = 0, \text{ and} \tag{2.8}$$

$$R(t) = R(t-1) + S(t), \text{ for } t > 0. \tag{2.9}$$

These processes are specially useful because they may be utilised in the construction of a simple mental structure for the more complicated notions that are going to be introduced next. The simplest Random Walk one can imagine can be created by tossing a coin in the air, and by writing down 1 or -1 as a function of the side it shows after being stopped. If the resulting sequence is iteratively summed and plotted against the order of the occurrences, the result will be something similar to what is shown in Figure 2.1.

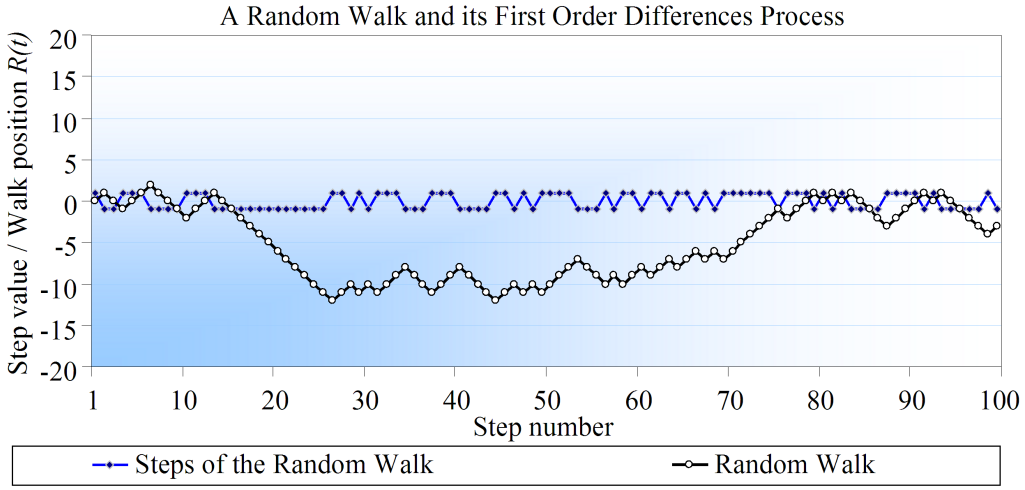

Figure 2.1: Graphical representation of a random walk with increments equal to 1 or -1.

As $S(t)$ is a random variable, the Random Walk is a self-similar process with Hurst parameter equal to 0.5 (meaning that it possesses no memory at all). Nevertheless, it

is possible to create walks with persistent or anti-persistent characteristics, as further discussed in chapter 4.

2.2.4. Brownian Motion

The *Brownian motion* is a widely known model for the movement of particles, named after the person that proposed it first, the botanist Robert Brown. It is sometimes referred to as the *Wiener process*, named in honour of Norbert Wiener for its many significant contributions to the subject. A Brownian motion $\{B(t)\}_{t\in\mathbb{R}_{\geq 0}}$ is the Gaussian process that respects conditions (2.10), (2.11) and (2.12) , where $t, s \in \mathbb{R}_{\geq 0}$, $s < t$ and $G\left(0, (t-s)^2\right)$ denotes a Gaussian variable with mean 0 and variance equal to $(t-s)^2$ [43]:

$$B(t) = 0 \text{ if } t = 0, \tag{2.10}$$

$$B(t) \text{ is almost surely continuous,} \tag{2.11}$$

$$B(t) \text{ has independent increments with distribution } B(t) - B(s) \sim G\left(0, (t-s)^2\right). \tag{2.12}$$

As the increments of the Brownian motion are independent, the said process shows no signs of persistence nor of anti-persistence (the Hurst parameter of such processes is, once again, 0.5). The visual (discrete-time) representation of this stochastic process may be obtained by orderly summing a series of realisations of a Gaussian variable, and by plotting the several sums against the iteration number. Figure 2.2 depicts the first 1000 points of 3 discrete-time Brownian motions, emphasising the fact that a motion will almost surely cover a different path each time it is instantiated. The Brownian motion can also be understood as the natural limit of the sum and normalisation of a series of the previously defined Random Walks $\{R_i(t)\}_{i\in\mathbb{N}}$, which can be formalised by

$$B(t) = \lim_{n\to\infty} \frac{\sum_{i=0}^{n} R_i(t)}{n}. \tag{2.13}$$

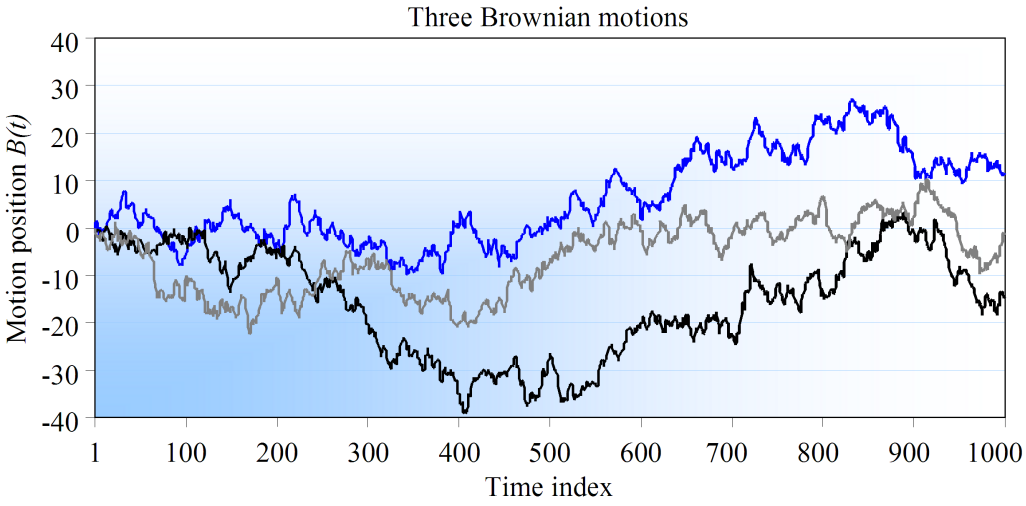

Figure 2.2: Graphical representation of several Brownian motions.

2.2.5. Fractional Brownian Motion

The *fractional Brownian motion (fBm)* generalises the notion of Brownian motion by formalising persistence and anti-persistence for that type of processes. Its definition, according to Mandelbrot and Van Ness [43], is based on the *stochastic representation* given by expression (2.14), which presents an fBm as an integral with respect to a random variable which, in this case, is the Brownian motion:

$$B_H(t) = \frac{1}{\Gamma(H+1/2)} \left(\int_{-\infty}^{0} [(t-s)^{H-1/2} - (-s)^{H-1/2}] dB(s) + \int_{0}^{t} [(t-s)^{H-1/2}] dB(s) \right). \tag{2.14}$$

Notice that $\Gamma(.)$, in expression (2.14), denotes the Gamma function $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx$, and that $0 < H < 1$ is the Hurst parameter. (A brief discussion about the stochastic representation on which this definition is based on can be found in [38].) One may notice that the previous definitions of Brownian motion and fBm are valid for the continuous-time case. However, for the sake of this explanation and in the remaining part of this thesis, consider the restriction by which both stochastic processes are defined for the discrete-time case only, i.e. their domain is a countable set.

2.2.6. Fractional Gaussian Noise

The first order differences process of a time-discrete fBm, denominated by *fractional Gaussian noise (fGn)*, symbolised by $G_H(t)$ (where $0 < H < 1$ is the Hurst parameter) and defined by (2.15), is of critical importance to the work presented, as it is commonly used to model the number of bits arriving to a network aggregation point per time unit (refer to section 2.3.1. for a discussion on *network aggregation point* and to section 2.3.3. for a more detailed discussion on this topic):

$$G_H(t) = B_H(t+1) - B_H(t),\ t \in \mathbb{N}. \tag{2.15}$$

The fGn was also adopted as the model for the simulation of network traffic in this thesis. For further details, consider referring to section 5.4..

Because of its definition and designation, the process resulting from the integration of an fGn is said to be self-similar if $G_H(t)$ respects condition (2.4). Sometimes, it is also said that the fGn is self-similar in the sense given by (2.4) [44]. A visual example of an fGn can be found in Figure 2.3, where 1000 points of a normally distributed variable are plotted against their index of occurrence.

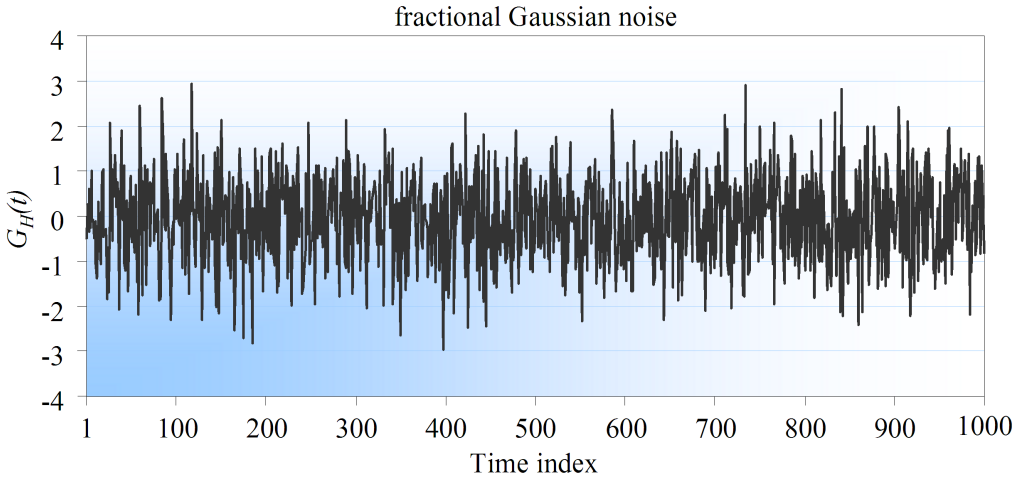

Figure 2.3: Example of fractional Gaussian noise for $H = 0.5$. Because of its flat power spectral density, this process is often denominated by *white noise*.

2.2.7. Normalised Fractional Brownian Motion

An fBm can be defined by a set of conditions that are simpler to understand, if some assumptions are made. The formal definition of the so-called *normalised fBm* describes it as a function $B_H(t)$, with domain $t \in \mathbb{R}_{\geq 0}$, which verifies the following five conditions:

$$B_H(t) \text{ is a Gaussian variable for } t \geq 0; \tag{2.16}$$

$$B_H(t) \text{ has stationary increments;} \tag{2.17}$$

$$B_H(0) = 0; \tag{2.18}$$

$$\mathbb{E}(B_H) = 0, \text{ for } t \geq 0; \tag{2.19}$$

$$\mathbb{V}(B_H) = t^{2H}, \text{ for } t \geq 0. \tag{2.20}$$

$\mathbb{E}(B_H)$ and $\mathbb{V}(B_H)$, in the previous expressions, denote respectively the expected value and the variance of the normalised fBm $\{B_H(t)\}_{t \in \mathbb{R}_{\geq 0}}$. Any fGn associated with a particularisation of a normalised fBm to the discrete-time domain case is, by construction, a Gaussian variable with mean equal to 0 and variance equal to 1, but there is generally no independence between the increments $G_H(t)$, when $H \neq 0.5$.

In the next section, it will be shown that a snapshot of the amount of information that passes in a network aggregation point per time unit resembles a shifted and scaled version of a normalised fGn. Therefore, this simplified definition serves well the purposes of this thesis, offering some room for abstraction of the more complex concepts. Any required transformations will be conveniently introduced if such is considered to be pertinent. Also because of the importance of fGns in the scope of this work, the means to accurately generate incidences this particular family of Gaussian variables are going to be explored in chapter 4, under the curious look that may enable the extraction of the interesting features of these stochastic processes. For now, the focus is to be placed in

other aspects of the self-similarity subject.

2.3. Self-Similarity in Network Traffic

Until this section, the relation between the subjects of self-similarity and networking has been hinted by brief statements that say that, when analysed in a network aggregation point, traffic exhibits long-range dependence. In the next paragraphs, the *origin*, the *meaning* and the *consequences* of this statement are going to be discussed with more detail.

2.3.1. Network Aggregation Point

Regarding the subject of *self-similarity in network traffic*, one of the most important expressions is the one of *network aggregation point*. While passing through a network, a protocol data unit visits several intermediate systems (nodes) that redirect it towards a next hop or final destiny, according to the implemented forwarding policies (e.g. best effort, Quality of Service (QoS) with differentiated services assured or expedited forwarding, etc.). Some of that nodes are visited by more than one flow of information at a time, being obliged to manage them under constraints of *resources availability* and QoS requirements. As the flows arrive concurrently to different interfaces of the equipment (or already mixed up in a single one), the later is forced to organise some in buffers, while processing the others. These nodes are *network aggregation points*. Figure 2.4 may help to understand what the referred expression means. In there, several data flows, represented by arrows and originated by different computational devices (the *sources* of information), make their way through Local Area Networks (LANs), and possibly through the Access, Metro and Core networks before reaching their final destination. Herein, the absolute source or destination of those data flows are sometimes referred to as *terminal nodes of the network*. In the middle of the network, many of that flows pass through the same routers, switches or other forwarding devices (represented by black circles) where they are lined in waiting queues (multiplexed), processed and forwarded to another intermediate destination. Though self-similarity is a consequence of aggregation, the explanation for its origin lies at the sources of information.

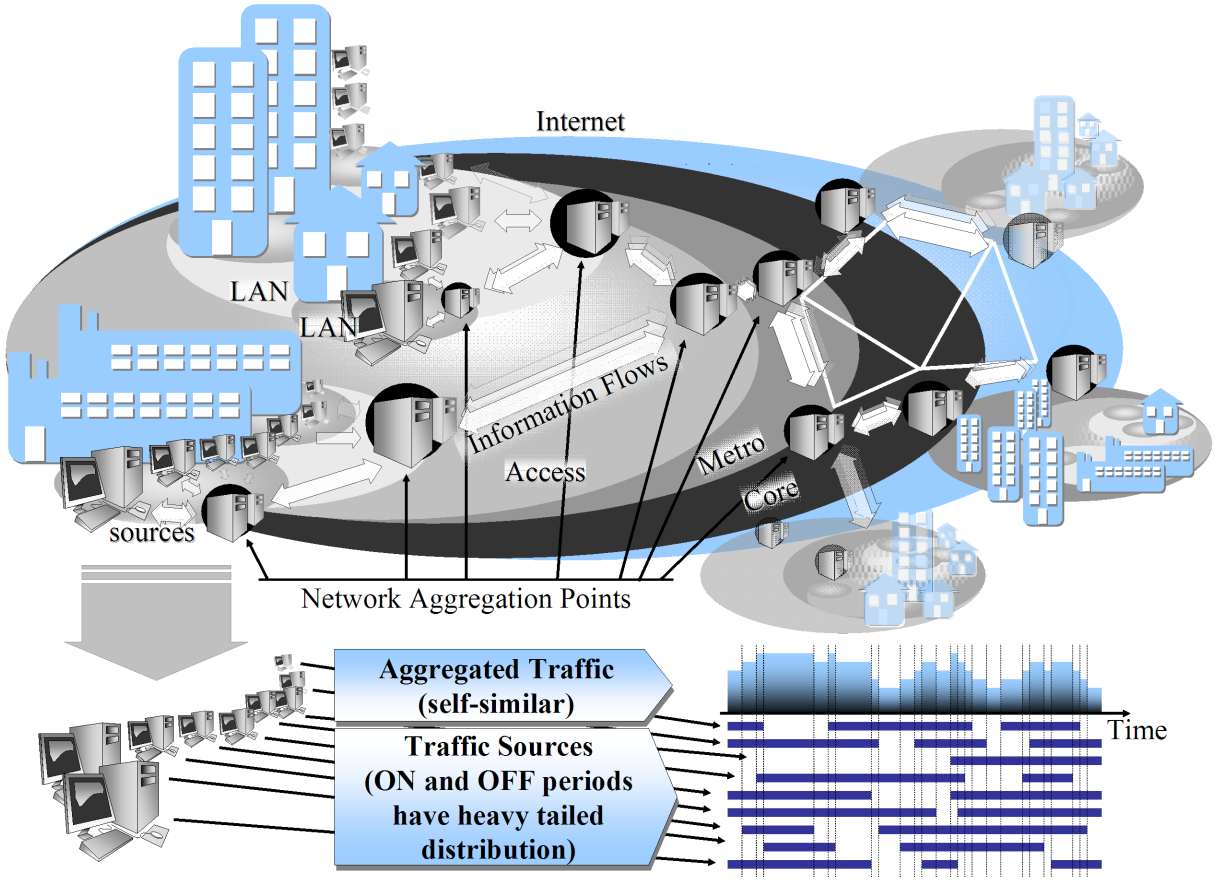

Figure 2.4: Conceptual representation of the Internet and of its constituent computer networks. Also depicted by black circles and (bi)directional arrows are network aggregation points and communication flows, respectively. A graphical explanation of how the several *renewal* processes contribute to the self-similarity of the network traffic is included at the bottom.

2.3.2. The Origin of Self-Similarity

In the early nineties, Will Leland, Murad Taqqu, Walter Willinger, and Daniel Wilson wrote one of the most influential papers of the history of the Internet. *On the Self-Similar Nature of Ethernet Traffic* [17] brought the beauty of fractals into the world of the information networks, unveiling an unexpected, and until then unknown, facet of the way traffic is multiplexed during its path to a destination. It has to be emphasised that it took as much imagination as rigour and perseverance for these men to get to such conclusion. In [17], and later in [39], the process describing the amount of information per time unit was proven to be long-range dependent for Ethernet LANs, and the tools used to prove that were rigorously reported, as well as the possible explanation for the inability to define the said process as e.g. a purely random Poisson variable [45]. Actually,

it would seem more - lets say - logical to expect that the protocol data units arriving to a given transitory (but popular) point of its path, would follow the exact same statistical laws that rule the way the origin of the information sent them to the network. It would be even simpler to accept that those laws could change within the network, but that that fact would not result in a behaviour described by *self-similarity*. From a naive perspective, one could even ask if all the theory developed for the several aspects of the communications was wrong or if it has been lost somewhere in time, as a consequence of the evolution of the networks. The answer to that question would be definitely *no*.

It is true that networks changed in a lot of aspects. The number of connected users has grown exponentially [46, 47], the architectures have changed to a more dynamic and complex organization, the technology and the physical infrastructure that transports, switches, routes and processes the packets that travel from a node to another is completely different from the one used to support the Plain Old Telephone Service (POTS). All these changes have obviously something to do with the self-similar nature of the traffic, but they are not the primary actors of that act.

Self-similarity finds its origins precisely in the way the several sources of the traffic send the data to the network. When transmitting, each terminal node sends variable length bursts (i.e. a variable number of packets within a small amount of time) into the network, which are normally followed by relatively bigger inactivity periods. According to [18, 19, 39], both processes can be modelled by random variables with an heavy-tailed distribution, defined by the power law in (2.21), where $\mathbb{P}(x > X)$ denotes the probability of x being bigger than X, for any real number $X \in \mathbb{R}$, and c is the minimum value x can take:

$$\mathbb{P}(x > X) \sim cX^{-\alpha}, \text{ as } X \to \infty, 1< \alpha < 2. \tag{2.21}$$

The often called *renewal process* $\{W(l)\}_{l\in\mathbb{N}}$ results from the combination of that two processes into a single one. It describes the activity of the transmitter at time l as a 1, if the transmitter is *ON*, or as a 0, if the transmitter is *OFF*. It can be proven [48] that the superimposition (and aggregation) of a *considerable amount* of independent renewal processes approximates an fGn exhibiting persistence behaviour. Furthermore, the Hurst

parameter of the resulting process depends of the value of α, being the relation between them given by $H = \frac{3-\alpha}{2}$, $1 < \alpha < 2$. A tentative to graphically explain this phenomenon can be found at the bottom of Figure 2.4, and the mathematical formalisation is given by (2.22), where B is the number of independent and identically distributed renewal processes (sources), $\{W_b(l)\}_{l\in\mathbb{N}}$ is the renewal process of source b, H is the Hurst parameter, m is the size of the aggregation block and $\sigma(\alpha)$ is a constant depending on α. Notice that the convergence expressed in (2.22) should be understood as a *convergence in law*, since the concepts it refers to and its assumptions are also described in those terms:

$$\lim_{m\to\infty}\lim_{B\to\infty}\sum_{l=mj+1}^{m(j+1)}\sum_{b=1}^{B}W_b(l) = \frac{1}{2}mB + m^H B^{1/2}\sigma(\alpha)G_H(j). \tag{2.22}$$

Basically, expression (2.22) provides an extremely sound justification for the observation of long-range dependence in aggregation points, without refuting the hypothesis of the traffic generated at the source level being only weakly autocorrelated or possessing no memory at all. The approximation to a self-similar process depends positively of the two factors under the mathematical symbol of *limit*: the *number of sources*, and the *size of the aggregation scale*. This dependency from *high variability* is emphasised in [19].

In a more informal way, these facts may be interpreted in the following manner. At the source, the *ON-periods* are uncorrelated or short-range dependent, meaning that the memory they possess is limited, or non-existent. However, when interacting with different renewal processes, each renewal process assures that, for the time it is ON, the memory of the aggregated process is assured by it and that, when it is OFF, another process will almost surely take such responsibility. The *memory* of the resulting process is therefore being fed by several small tributes, resulting in a long-range dependent series. As it will be mentioned in chapter 4, for each fixed interval of time, a self-similar process can be seen as the sum of two components: a constant and a variable one. In this case, the extension of the tail of the OFF and ON sequences defines exactly the fraction of the aggregated process that changes in the aforementioned time interval, and how much remains constant. The bigger the tail is, the longer the constant parts are, and the more self-similar the aggregated process is. Under the given assumptions, the processes of the *bit count* or of the *number of protocol data units per time unit* are, as previously referred,

resemble a shifted and scaled fGn.

2.3.3. The Face of Self-Similarity

It is possible to get an idea on how self-similarity manifests itself visually, if a series of values possessing such property is plotted for different aggregation factors of itself. The procedure to obtain such graphical representation is simple. After plotting the empirical sequence of the first order differences process $\{Y(t)\}_{t\in\mathbb{N}}$ against the time index t, aggregate it several times, and plot the resulting series against the respective indexes as well. Using this procedure, one obtains k different charts. The degree of self-similarity of the associated process determines its *resistance to change its initial aspect*, as the aggregation scales increase. To get a practical idea of what these and the proceeding words mean, observe Figure 2.5, where the results of the application of the previously described procedure to a persistent and to a completely random process are depicted and sorted out vertically. For a start, it is safe to conclude that a superficial observation of the charts is sufficient to identify in which of the sides of the figure the self-similar process is represented: the left side. The sequence displayed on the right side is thus characteristic of memoryless processes, which suffer more rapidly from the effect of the *law of large numbers* than the ones exhibiting positive and long-term autocorrelations. Such effect manifests itself in a faster convergence to a multiple of the average value of the series being aggregated. On the other hand, the self-similar process tends to maintain its statistical properties (e.g. variance) more effectively for the several aggregation scales, avoiding the mentioned smoothing effect during more time. Nowadays, the sequence of charts on the left side of the figure is known as the *classical pictorial proof of self-similarity in network traffic* [49], as a consecration of the designation used by Leland *et al.* [39].

Both sets of charts in Figure 2.5 were produced using computational means, for exemplification purposes. While the left side of the figure was produced using one of the self-similar sequences generators presented afterwards, the other concerns the values returned by a Pareto source simulator. In the first, the Hurst parameter was set to 0.85, while the second had the α set to 1.3 (resulting from the substitution of H by 0.85, in the formula presented in the previous section: $\alpha = 3-2H$). Both were scaled so as to produce

a representation that would look like with the bit count per time unit of a network device performing at 1 Gbps, with an used bandwidth ratio of 0.5.

Regardless of the synthetic nature of the series on the left, a direct analogy between the latter and the observations in [18, 39] can be done. It was probably a *picture* like the one in Figure 2.5 that catch the curiosity of the researchers: no matter from how far the traffic was being observed, it *appeared* to be always the same. On the other hand, the amount of information per time unit was not stabilising to the average value, which would somehow be expected if the process was following a Poisson or exponential law without any additional assumption.

2.3.4. The Consequences of Self-Similarity

The most critical secondary effect of having the amount of information per time unit exhibiting the property of self-similarity is that it results in the so-called *burstiness* of the traffic, which basically means that the data comes in bursts rather than randomly spaced in time. This artefact unbalances the amount of data arriving to the aggregation point, resulting in bit rates above the average that may last longer than expected, followed by (statistically) equally large periods of time where the bit count is below the expected value. Such phenomenon is sometimes referred to as *the Joseph Effect* [19], as a metaphor to the biblical story of the "*seven fat years and seven lean years*". Such behaviour impacts the way the traffic is handled at the aggregation points and one of the most immediate conclusions that could be taken from these remarks, is that a wrong assumption about the traffic model could lead to poorly planned networks, or to misconceived devices, which would lack memory for storing an unusual amount of data, during congestion periods [45, 50]. This problem was first brought up in [50] (and later backed up by many other studies, as for example [16, 51, 52, 53, 54]), where it is stressed that a linear increase of the size of the buffers of routers or switches does not necessarily result in the linear decrease of the losses during congestion periods. The identification of this issue preceded the discovery of the self-similar structure of the traffic, emphasising the fact that, unlike POTS, the information networks could not be properly scaled (in terms of congestion-avoidance mechanisms) based on past assumptions. The *Joseph Effect* applies

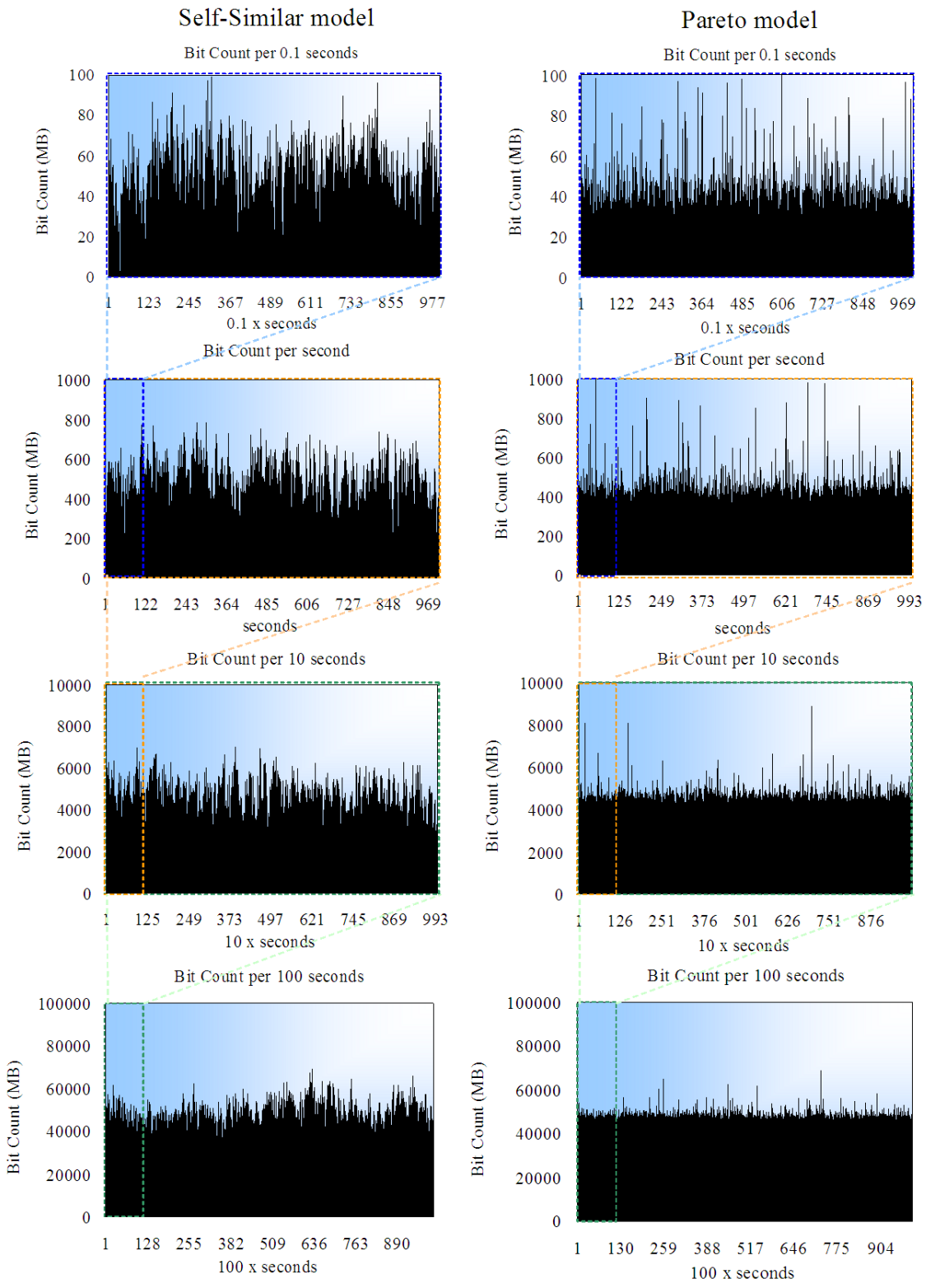

Figure 2.5: Self-similar vs. non self-similar trace. The classical pictorial representation of self-similarity (persistence) in the bit count per time unit process of a network trace is depicted on the left side of the figure; the typical behaviour of the effect of aggregation on a random variable is shown on the right side.

to the losses also, since after the devices have entered the *loss condition*, the *persistence* of unpredictably long busy periods may consequently result in persistent losses. Notice that a *loss condition* occurs when the network device is obligated to discard protocol data units, whether because it has no available resources to process them, or because the validity of the said units of information has already expired (due to the delays they were imposed to).

While short-range dependent or memoryless models, as for example the Poisson or Exponential models, offer some *predictability* (short-range dependent models converge faster to their expected value, as the aggregation scale increases - *law of large numbers*), the same does not happen with series with long-range dependence. Many of the studies conclude that the real solution to the congestion problem would be more on the increase of the *link capacity* or on the congestion algorithms [16, 51], rather than in the increase of the size of the buffers, as the latter would only induce bigger delays.

Problems like the one described above, allied to the technological advance the Internet (and all of its inherent technologies) has met in the last few years, plus the *mysticism* that, somehow, surrounds the subject of self-similarity, culminated in a vast list of scientific contributions that interconnect the subjects of *networking* and *long-range dependence*. It is only fair to classify this spurt without precedents as a consequence of its importance. Most of the studies argue in favour of the self-similar nature of the traffic [16, 17, 18, 19, 51, 52, 53, 54]; a few argue against it [55] (for this matter, consider also the reference [27]); some elaborate on a consistent theory that justifies self-similarity at the data link layer of the Open Systems Interconnect (OSI) model [19]; others look for the same justification in higher layers of TCP/IP architecture, namely on the *dynamics* of the Transmission Control Protocol (TCP) [56, 57]; some focus on the means to analyse the self-similarity degree [58, 59, 60, 61]; others yet on the models for the bit count or inter arrival processes [16, 19], or on the algorithms to generate long-range dependent series [38, 44, 62]. More recent studies, as e.g. [37, 63], elaborated on a conjecture that unites the two models (long-range dependence and Poisson) and state that, at the core, depending on the aggregation scale from which the traffic is observed, the underlying processes can be either random and Poisson (at small scales) or long-range dependent (when observed from higher scales). However, this is still not a consummated fact. The author

would like to refer the reader to [64], where an impressive (but not up to date) list of bibliographical references on the subjects of *self-similarity* and on *performance modelling for modern high-speed networks* can be found. The list of references was elaborated in 1996, and may be considered as a *colourful* picture of this paragraph.

Because self-similarity seems to partially define the traffic arriving at a given aggregation point, some works [20, 21, 25, 26] have also focused on how to use it to detect anomalies (which, in this case, include network level intrusions), by assuming that the presence of an attack in the network traffic incurs in an overall deviation to the long-range dependence model. These works are not unfounded, as it is going to be proven by some of the results of this thesis. However, these references should be considered with caution, as some of them start from rather strong assumptions or develop their approach in a confusing manner [21, 25] (for example, sometimes it is difficult to distinguish exactly to which concepts they are referring to). Moreover, when considered individually, the references within the papers are presented as if the results are consistent for all the studies, while the overall picture reveals contradicting results (e.g. [25] states that the Hurst parameter decreases in the presence of an attack, while in [26] the same parameter shows higher values for the same situation). These references will be discussed with more detail in chapter 5.

The findings of Leland *et al.* imply an elementary change of perspective, which motivated a lot of research around the subject of long-range dependence. The major advantage of all this interest is that the self-similarity theory evolved and is now largely disseminated in the literature. The disadvantages are that, given the specificity of the subject at hands, it becomes often difficult to distinguish good from bad contributions; and that it is hard to handle the constant flow of publications in this area. In [27] one can find an interesting *talk* given by Willinger, where he tries to make such distinction.

2.4. Hurst Parameter Estimation

As so well concluded by Norros in [16], "*(...) finding the correct value for the self-similarity parameter H is crucial (...)*". The parameter that Norros identified as one of the values to which the queue size distribution was more sensitive to, is nowadays known

as the value that measures the degree of self-similarity of a long-range dependent process in general, and the degree of burstiness of the traffic in particular. The *enviable statute* it has acquired, justifies the research and dissemination of the several means to estimate it, and places the latter in the same priority level of the other means to analyse the traffic or to actually model it. The appearance of tools [65, 66], surveys and comparative analysis between the several methods [38, 58, 59, 60, 61] is therefore justifiable, as it is normal the continuous search for new and reliable means to assess the Hurst parameter [67, 68, 69, 70]. However, while the Hurst parameter is perfectly well defined mathematically, assessing its value can be rather problematic [58].

The definition of Hurst parameter is made under probabilistic terms and for arbitrarily long (*infinite*) time series. The term *estimation*, however, implies that (i) the exact value of the Hurst exponent is still unknown and that one only has information about the time series and (ii), that the assessment is being made for a *finite* set of points that one believes to possess the property of self-similarity. The Hurst parameter is normally estimated by analysing the same series for several aggregations of itself, or by quantifying the laws that describe the spectral domain of the associated stochastic process. Most of the times, such procedure is computationally complex and time consuming. As the data series under analysis (e.g. network traffic trace) are not a *perfect* instantiation of a mathematical law, it possesses *imperfections* - read this as *small deviations* from the model - that may impact differently the referred estimators. As outlined in [58], some estimators are vulnerable to trends, others to noise or to periodicity, etc. Dealing with these problems requires often the same series to be visited several times, so as to be differently processed, imposing again an additional computational effort. When a given procedure for obtaining a single value requires the same sample pool to be analysed more than once, it is termed *retrospective*. Thus, the computational complexity of an estimator requiring the signal to be processed in a retrospective manner is never smaller than $O(n \times o(n))$, where $o(n)$ is the computational complexity of the (at least) *second* analysis of the series of values, being that $o(n)$ is strictly dependent on the length of the series to be processed.

The next sections present a compilation of the most popular Hurst parameter estimation methods. Note that some of them will be explained in more detail than others. The reasons for such to happen is related with the fact that not all the methods were

useful for this research work, and with the fact that some were noticeably not suitable for its purposes. The justification was nevertheless included in the respective subsections.

All the methods described below make use of the logarithmic function at some stage of their rationale. The base for which the logarithms are defined for is a completely irrelevant detail in most of the cases. Nonetheless, for the sake of this explanation and if nothing is said on the contrary, just assume that $\log(x)$ denotes the base 10 logarithm of x, i.e. that $\log(x) = \log_{10}(x)$. Notice also that, below, H is interchangeably used to denote the Hurst parameter or a specific estimate of its value provided by a given method. This notation is somehow abusive, but it was adopted for the sake of simplicity of the explanation.

2.4.1. Rescaled Range Statistics

The first estimation method that is going to be described was proposed and used by the British hydrologist Harold Edwin Hurst (1880-1978) [71] himself, during his experiments to understand the vicissitudes of the flows of the Nile river [72, 73]. Its historical component is actually the reason for which the author decided to introduce it first, in spite of some of the accuracy problems it presents (see below).

Consider that the aggregated process in (2.2) is defined for k aggregation scales with size $m_k \in \mathbb{N}$, with $m_k > 1$, and let the series of the readjusted sums $\{S_k^i(j)\}_{j\in\mathbb{N}}$ be defined as in

$$S_k^i(j) = \sum_{l=im_k}^{im_k+j} (Y(l) - l \times Y^{(m_k)}(i)), m_k \in \mathbb{N}. \tag{2.23}$$

For each aggregation scale m_k, take the maximum (2.24) and the minimum (2.25) of that sums, and define the range of the adjusted process as the difference between them (2.26). Notice that it is assured by construction that $\{Range_k(i)\}_{i\in\mathbb{N}}$ takes only positive values:

$$Max_k(i) = \max(0, S_k^i(1), S_k^i(2), ..., S_k^i(m_k - 1)), \tag{2.24}$$

$$Min_k(i) = \min(0, S_k^i(1), S_k^i(2), ..., S_k^i(m_k - 1)), \tag{2.25}$$

$$Range_k(i) = Max_k(i) - Min_k(i). \tag{2.26}$$

The rescaled range $RS_k(i)$ is then obtained from (2.26) by dividing the $Range_k(i)$ by the standard deviation ($\sigma(i)$) of the m_k values of $\{Y(t)\}_{t\in\mathbb{N}}$ that contribute for the construction of the particular incidence $Y^{(m)}(i)$, as formalised by

$$RS_k(i) = \frac{Range_k(i)}{\sigma(i)}. \tag{2.27}$$

Hurst [72] noticed that for self-similar processes, the average value of $\{RS_k(i)\}_{i\in\mathbb{N}}$, given by $\mathbb{E}(RS_k)$, could be described in terms of the values of m_k and of a scaling exponent, later known as the Hurst parameter, as shown in equation (2.28). In the referred expression, C denotes a positive constant number, proven by the following reasoning uninteresting for the matter of estimating the Hurst parameter:

$$\mathbb{E}(RS_k) = C \times (m_k)^H. \tag{2.28}$$

The Rescaled Range Statistics (RS) analysis is normally performed by defining a definite set of values for m_k and by calculating the respective $\mathbb{E}(RS_k)$ for each k. If the process is self-similar, the obtained values *should almost surely* be fit by the implicit curve of (2.28) or, equivalently, by the line defined by (2.29), where the Hurst parameter appears as its slope:

$$\log\left(\mathbb{E}(RS_k)\right) = \log(C) + H \times \log\left(m_k\right). \tag{2.29}$$

The graphical representation of the coordinates $(\log(m_k), \log\left(\mathbb{E}\left(RS_k\right)\right))$ along with the line that best fits them (using regression) in a chart, is often called the *pox-plot* representation. The estimation of the Hurst parameter is therefore reduced to the calculation of the slope of that line. The chart in Figure 2.6 is an example of a pox-plot, constructed during the RS analysis made to sequences realising two fBms (with $H = 0.70$ and with

$H = 0.30$) and a Brownian motion ($H = 0.5$). As can be seen in the figure, this method is susceptible to short-range dependence relations (for the smallest scales, the values of $\mathbb{E}(RS_k)$ tend to be underestimated), reason by which some authors have already described possible modifications to the procedure [74]. Nevertheless, the method is still capable of producing reliable results, if handled carefully. In [75, 76], for example, it is proven that, for $H = 0.5$, $\mathbb{E}(RS_k)$ really converges to $C \times \sqrt{m_k}$, as $m_k \to +\infty$. This fact is also noticeable in the aforementioned chart.

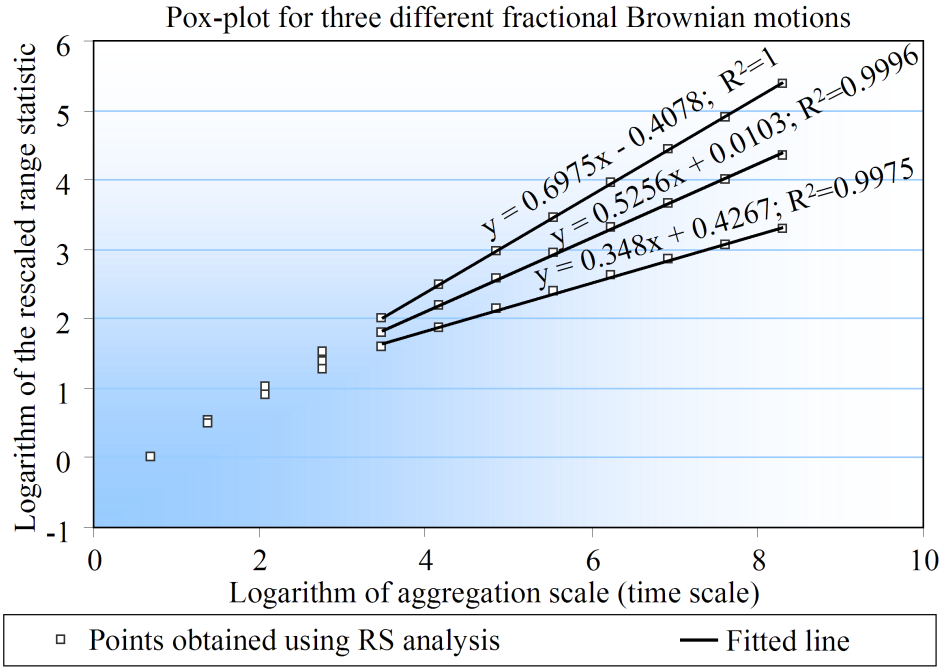

Figure 2.6: Pox-plot for the RS analysis of two fBms ($H = 0.3$ and $H = 0.7$) and of Brownian motion ($H = 0.5$). The aggregation scales were all powers of 2, starting at 1, and the estimated Hurst parameter values were approximately 0.35 and 0.70 for the fBms, and 0.53 for the Brownian motion.

The author has decided to implement and use (see chapter 4) the RS method, mostly because of its historical value. However, this implementation has merely served the purposes of testing the self-similar sequences generators proposed below and of providing a baseline comparison for the remaining methods. This justification also explains why this method was so thoroughly described herein.

Observing (2.23) critically and with an eye in the objectives of this work, it is easy to conclude that the RS method is not suitable to be used in a point-by-point or windowed manner, because the rescaling procedure requires that all the points of a

given aggregation scale to be available *a priori*, which is not desired for an iterative estimator. The actualisation of the rescaled variable $S_k^i(j)$ would require an average of $\frac{m_k}{2}$ operations, turning infeasible the application of the method as a real-time estimator. This should become more obvious as the modifications for some of the following estimators are presented, in chapter 3.

2.4.2. Variance Time

Equation (2.4) can have many different interpretations, being those a reflex of the properties of self-similar processes. The multifaceted character of such mathematical expression is due to the abstract notation $\stackrel{d}{=}$, that defines that all the possible statistics of the two processes it applies to are equal. The particularisation of (2.4) that best fits the purpose of explaining the estimator known as the Variance Time (VT) method [77, 78] is given by (2.30), where $\mathbb{V}(Y)$ and $\mathbb{V}(m^{1-H}Y^{(m)})$ denote the variance of $\{Y(t)\}_{t\in\mathbb{N}}$ and of $\{m^{1-H}Y^{(m)}(i)\}_{t\in\mathbb{N}}$, respectively:

$$\mathbb{V}(Y) = V\left(m^{1-H}Y^{(m)}\right). \tag{2.30}$$

Thanks to the properties of the variance [79], (2.30) can be rewritten into form (2.31) and, if the logarithm is applied to both sides of the equality, it is possible to write equation (2.33), which is the most direct outcome of (2.32). Once again, the Hurst parameter H defines the slope of the line with *y-intercept* equal to $-\log(\mathbb{V}(Y))$ and that passes through the point with coordinates $\left(\log(m), \log\left(\mathbb{V}\left(Y^{(m)}\right)\right)\right)$:

$$\mathbb{V}(Y) = m^{2-2H}\mathbb{V}(Y^{(m)}); \tag{2.31}$$

$$\log(\mathbb{V}(Y)) = \log(m^{2-2H} \times \mathbb{V}(Y^{(m)})) \tag{2.32}$$

$$\Leftrightarrow \log(\mathbb{V}(Y)) = (2-2H) \times \log(m) + \log(\mathbb{V}(Y^{(m)})). \tag{2.33}$$

Equation (2.33) may be solved in order to H, as shown by the following expression,

but it is worth nothing if m is not determined somehow:

$$H = \frac{\log(\mathbb{V}(Y^{(m)})) - \log(\mathbb{V}(Y))}{2\log(m)} + 1. \tag{2.34}$$

This small problem may be overcome by defining a finite set of aggregation scales m_k (exactly as in the previously described method). The several variances of the aggregated processes can then be calculated and used indiscriminately in formula (2.34) to retrieve Hurst parameter estimates. However, due to the statistical nature of this method, such estimates will most certainly not be unique for all the considered scales. A unique value for the estimation is usually obtained by fitting a line to the coordinates given by $\left(\log(m_k), \log\left(\mathbb{V}\left(Y^{(m_k)}\right)\right)\right)$ and by using expression (2.35) to get the Hurst parameter, where β represents the slope of the referred line:

$$H = 1 + \frac{\beta}{2}. \tag{2.35}$$

Figure 2.7 shows how the VT method should be tackled from the graphical analysis point-of-view. The chart in this figure plots the values of the transformation of the variances against the respective logarithm of the aggregation scales. The latter are commonly understood as *time scales*, which led to the denomination of the plots as the *Variance/Time plots*. Needless to say that the method owes its name to this designation. As can be seen in Figure 2.7, the slope of the line of the Variance/Time plot is negative, conversely to what happens with RS (and others), which basically means that the variances of the aggregated processes decay exponentially, as the time scales increase.

Focus on the R^2 values included next to the equations of the fitted lines, in the chart of Figure 2.7 (and in other identical charts along this thesis). The R^2 statistic, often denominated by *the coefficient of determination*, measures how well the points are fit by a line and, in the simplest case of linear regression, its value is the fraction between the variances of the y-coordinates of the fitted and of the modelled points [80]. The closer the referred value is to 1, the more *tight* (and in this case *linear*) the relation, between the coordinates subdued to regression, *seems* to be. Although R^2 should not be directly understood as a *goodness-of-fit metric*, it surely provides one with an idea

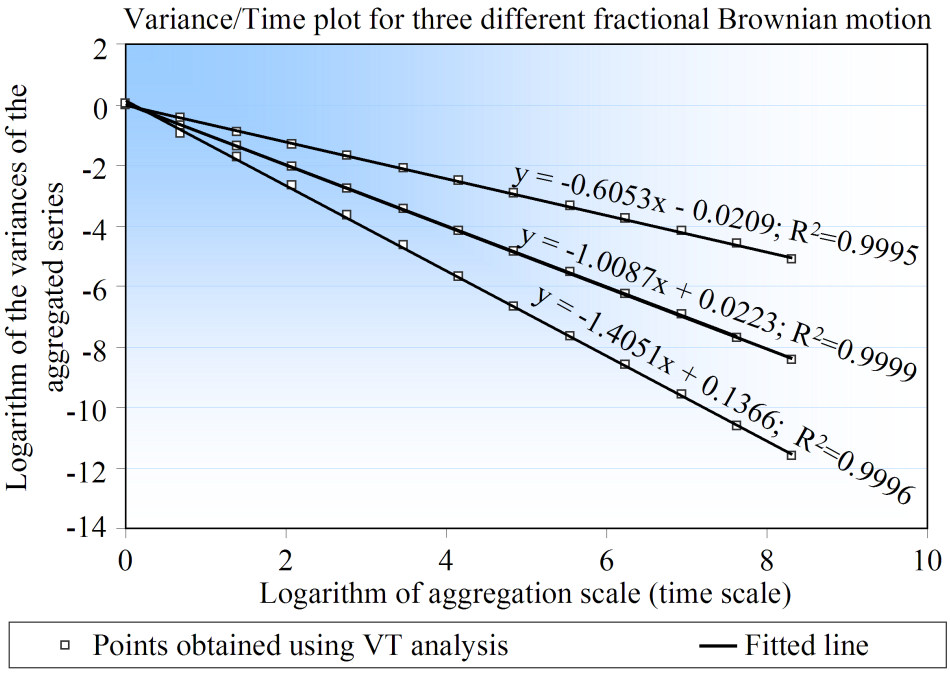

Figure 2.7: The Variance/Time plot for two fBms ($H = 0.3$ and $H = 0.7$) and for Brownian motion ($H = 0.5$). The aggregation scales were all powers of 2, starting at 1. The estimated Hurst parameter values were approximately 0.30 and 0.70 for the fBms and 0.5 for the Brownian motion.

on whether the conducted analysis makes sense or not. If required, a *goodness-of-fit test* can be applied resorting to the fact that F, obtained from the formula (2.36), follows an *F-distribution* with 1 and $K - 2$ degrees of freedom (K denotes the number of points subdued to regression):

$$F = \frac{R^2}{(1 - R^2)/(K - 2)}. \tag{2.36}$$

If F is larger than the *critical value* for a pre-selected level of significance, the null hypothesis stating that *the linear model is not useful at all* may be rejected [81], supporting its alternative.

While the value of R^2 alone may not be enough to prove the goodness of fit of the regression, a significative change of its value for several estimations constitutes an indicator that something has affected the correlation between the fitted points. In the case of self-similarity analysis, a drop of the R^2 value means that the property of self-similarity is not so strongly embedded in the empirical data, or that the referred property

has actually been lost. Within the enunciated limitations, the metric is going to prove its usefulness in chapter 5, where the effects of loss of stationarity are going to be studied in more detail.

The most interesting remark that can be made about VT is that its operational model depends only on statistics that can be updated in a point-by-point manner (the average and the variance), but the correct opportunity to discuss that is going to find its moment in the following chapter. Still to be mentioned is the fact that, in this case, it is empirically *visible* on the chart of Figure 2.7 that the method is not biased for low aggregation scales (short-range dependence).

2.4.3. Absolute Moments Time

The Hurst parameter estimation method entitled Absolute Moments Time (AMT) is a not so commonly used generalisation of VT [82] and derives (as its predecessor did) directly from (2.4). Consider the definition of *second order self-similar process* and of *first order differences process*. The same way (2.4) determines the equality of the variances of the processes in both sides of the $\stackrel{d}{=}$ symbol, it also states the equality of their order n absolute moments, denoted herein by $M_n(Y)$, and generically defined by an expression equivalent to

$$M_n(Y) = \mathbb{E}|Y(t) - \mathbb{E}(Y)|^n, \text{ where } n \text{ is a positive integer number.} \tag{2.37}$$

Therefore, the main equality on which this method is based on is given by a simplification of (2.38), herein formalised in (2.39):

$$M_n(Y) = M_n(m^{1-H}Y^{(m)}) \tag{2.38}$$

$$\Leftrightarrow M_n(Y) = m^{n(1-H)}M_n(Y^{(m)}). \tag{2.39}$$

The same reasoning and notation used in the description of VT can now be applied to deduce expression (2.41) from the transformation in (2.40), where the Hurst parameter is already isolated:

$$\log(M_n(Y)) = n(1-H) \times \log(m) + \log(M_n(Y^{(m)})), \text{ where } n \in \mathbb{N}; \tag{2.40}$$

$$H = 1 + \frac{\beta}{n}, \text{ where } \beta = \frac{\log(M_n(Y)) - \log(M_n(Y^{(m)}))}{\log(m)}. \tag{2.41}$$

For each n, the estimation of β presumes the definition of e.g. K aggregation scales $(m_k, k = 1, ..., K)$ and the statistical treatment of the potentially different values of β. As (2.40) implies a linear relation between the logarithms of the absolute moments and the considered size of the block, plotting these values in a chart results in a straight line with *y-intercept* equal to $\log(M_n(Y))$ and *slope* equal to β. Therefore, the least squares method can be used to retrieve the value that best summarises all the K estimations of β (linear regression is the *de facto* manner of obtaining the estimate).

As previously stressed out, VT is a particular instantiation of this method, whose core formula can be recovered by substituting n with 2. The comments made to VT apply naturally to this one, namely the remark concerning the applicability of the estimator in real-time. However, this has to be carefully considered, as an higher complexity is implicitly hidden behind the choice of the order of the absolute moment.

2.4.4. Embedded Branching Process

The following mathematical explanation concerns the method entitled Embedded Branching Process (EBP), proposed by Jones and Shen in [69]. EBP presumes the stationarity of the increments of the process $\{X(t)\}_{t\in\mathbb{N}}$ to which the analysis is to be conducted (the fBm falls into this category), to then construct a procedure around the so-called *crossing tree structure* which, according to [69, 83], is inherently embedded in the referred process. This method is the one that most directly elaborates on the first definition of self-similarity, written in terms of incremental processes as the fBm.

Consider that $\{X(t)\}_{t\in\mathbb{N}}$ fulfils the previous conditions and, without loss of generality, suppose that its occurrences are happening in the space domain scaled by the constant factor 2, as depicted by the equation in (2.42). Note that the relation illustrated there is always possible when the process is self-similar in the sense given by (2.1), as it is always possible to find a μ satisfying the equality:

$$\exists \mu \in \mathbb{R}_{\geq 0} \text{ such that } 2 \times X(t) \stackrel{d}{=} X(\mu \times t). \tag{2.42}$$

By knowing that $X(t) \stackrel{d}{=} \mu^{-H} X(\mu t)$, the previous equation leads one to the conclusion that $2^{-1} = \mu^{-H}$, for some $\mu \in \mathbb{R}_{\geq 0}$. Getting an estimate of the Hurst parameter is confined to getting an estimate for μ, as proven by the following reasoning:

$$2^{-1} = \mu^{-H} \tag{2.43}$$

$$\Leftrightarrow H = \log_{\mu} 2 \tag{2.44}$$

$$\Leftrightarrow H = \frac{\log 2}{\log \mu}. \tag{2.45}$$

It is interesting to note that the value of μ is a measure of how much time the process spends until it doubles its statistical properties, as it is the variable that multiplies the time index t. In the case of discrete-time processes, this value should preferably be an integer but, in this case, the analysis is being made as if both space and time were continuous. At the end, μ represents an average value of the aforementioned time periods, which continues to be consistent with the definition of both types of processes.

Under the assumption of self-similarity, it is possible to further generalise the previous formulas by considering different (but convenient) space scaling factors. Focus on expression (2.46) for the mathematical formalisation of these words. Consider that K, which is a fixed positive integer number, denotes the maximum number of crossing levels taken into account in the analysis. For each $k = 1, ..., K$, it is possible to obtain an estimate for the Hurst parameter by assessing the time the process takes to double, quadruplicate, octuplicate, etc., its statistical properties. As stressed out by equation (2.47),

the relation between $\log(\mu)$ and $\log(2^k)$ may then be described by a line:

$$\text{For each } k \in \mathbb{N}, \exists \mu_k \in \mathbb{R}_{>0} \text{ such that } 2^k \times X(t) \overset{d}{=} X(\mu_k t) \tag{2.46}$$

$$\Leftrightarrow \log(2^k) = H \times \log(\mu_k). \tag{2.47}$$

Thus, the problem resides in the calculation of the several μ_k. For that, consider the fixed positive number δ and the family of parallel lines defined by $f(k) = \delta 2^k \mathbb{Z}$, designated by *crossing lines* (consider complementing this explanation with the observation of Figure 2.8). With those structures is possible to define the *initial crossing times* between incidences of $\{X(t)\}_{t\in\mathbb{N}}$ and the several crossing lines using, for example, an expression similar to the following one:

$$T_k(i+1) = inf\{t > T_k(i) : X(t) \in \delta 2^k \mathbb{Z}, X(t) \neq X(T_k(i))\}. \tag{2.48}$$

The right member of (2.48) identifies the first time index t for which $\{X(t)\}_{t\in\mathbb{N}}$ crosses a line of the family $f(k)$, while assuring that this crossing did not happen immediately after that same line has been crossed (hence $X(t) \neq X(T_k(i))$). In other words, for each k, $T_k(i+1)$ gets the time of the crossing of $\{X(t)\}_{t\in\mathbb{N}}$ after $T_k(i)$, unless the line that is being crossed is the same that was transposed at time $T_k(i)$. The number of crossings N_k is then given by the *cardinality* of the set comprising the crossing times $T_k(i)$ of each *level* k (expressions (2.49) and (2.50)), herein denoted by the symbol # in expression (2.50):

$$T_k = \{T_k(i)\}_{i=0,\ldots,T}; \tag{2.49}$$

$$N_k = \#T_k. \tag{2.50}$$

From here, one can choose two distinct ways to get the estimate of the Hurst parameter. In [69], each one of the μ_k are obtained by dividing the N_k by N_{k+1} (i.e. $\mu_k = N_k/N_{k+1}$), which are then combined using formula (2.51) to retrieve - lets say - a *global* estimate for μ, which is a *weighted* average of the several μ_k values. The Hurst

parameter is then obtained by means of formula (2.45):

$$\mu = \frac{N_0 \times \mu_1 + ... + N_{K-1} \times \mu_K}{N_1 + ... + N_K} \text{ , where } K \text{ is the maximum considered level.} \quad (2.51)$$

The second way of getting an estimate of the Hurst parameter comprises the usage of linear regression, as suggested by equation (2.47). However, this approach requires the several $\dot{\mu}_k$ (notice that a different notation is being used) to be acquired differently. Notice that, generally speaking, the fraction $\frac{N_k}{N_{k+1}}$ returns the average number of occurrences the process $X(2^k t)$ took to double its statistical properties. This means that each one of the μ_k values retuned by that division respects condition (2.42), but not necessarily condition (2.47). If, instead of N_{k+1}, N_{k+2} is used in the same expression, then the respective fraction would reflect the average number of steps that the process $X(4^k t)$ took to quadruplicate the said probabilistic characteristics, and so on. Because the second approach to obtain the Hurst parameter estimate requires the several μ_k to reflect the amount of time $\{X(t)\}_{t \in \mathbb{N}}$ takes to multiply its statistical properties by the increasing factor given by 2^k, for $k = 1, ..., K$, the different notation $\dot{\mu}_k$ was adopted. $\dot{\mu}_k$ is defined as follows:

$$\dot{\mu_k} = \frac{N_k}{N_K} \text{ , where } k < K, \text{ and } K \text{ is the maximum considered level.} \quad (2.52)$$

After the application of the linear least squares method to the coordinates $(\log(\dot{\mu}_k), \log(2^k))$, the designation of *Hurst parameter estimate* can be directly assigned to the slope of the fitted line.

The embedded crossing tree of a simulated Brownian motion is partially depicted in Figure 2.8. The mentioned structure can be constructed by representing the calculated crossing times of each level as *leafs*, and by connecting them through lines, under the constraint that each leaf can only connect to a single leaf with higher rank than its own. From the figure, it is easy to see that, as k decreases, the cardinality of T_k increases (the number of crossings increases because the lines $f(k)$ become closer to each other).

According to [69], EBP behaves reasonably well for self-similar processes *moved by persistence*. For *anti-persistent* series, the method takes longer to retrieve reliable estima-

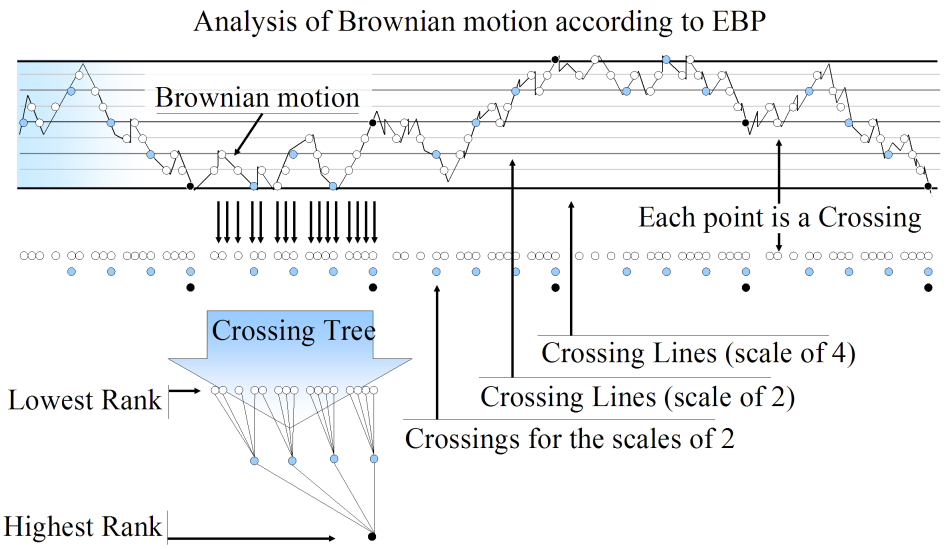

Figure 2.8: Graphical representation of the rationale *behind* the EBP method.

tions, as the spatial span that the realisations of $\{X(t)\}_{t\in\mathbb{N}}$ cover for $H < 0.5$ is severely reduced as $H \to 0$. For long-range dependent processes, $\{X(t)\}_{t\in\mathbb{N}}$ diverges more than in the previously mentioned case, which basically results in a bigger number of crossings and, therefore, in better estimates. However, one has to be careful with the adjustment of δ, since a bad choice of the value of this parameter may lead to overestimated number of crossings for the lowest levels. Recall that δ specifies the spacing between the lines of $f(k)$: the smaller this value is, the more times these lines will be transposed. In [84], it is said that a good choice for δ may be the standard deviation of the absolute values of the first order differences process of $\{X(t)\}_{t\in\mathbb{N}}$. However, empirical observations [69] have shown that a bigger value should be chosen, when possible, providing that such choice does not reduces drastically the number of crossings. Additionally, as EBP analyses the cumulative process itself, it is forced to presume that the first order differences series is centred at the origin, at the cost of returning only $H = 1$.

EBP has been used to assess the self-similarity degree of network traffic but, in [83], that was done in a retrospective manner, as the traces were analysed offline and the calculations concerning the number of crossings were made on a per level basis. For now, just consider this method as a potential candidate for a real-time estimator, postponing a more detailed discussion on how to use it in a point-by-point manner to chapter 3.

2.4.5. Detrended Fluctuation Analysis

The method presented in this section may be understood as an improved version of RS. RS makes use of the rescaled sums, defined in (2.23), which are the differences between the partial sums $\{S_k^i(j)\}_{j\in\mathbb{N}}$ and the line originating at 0 and ending at $Y^{(m_k)}(i)$ (for each i and each aggregation scale m_k considered). The Detrended Fluctuation Analysis (DFA) method, proposed by Peng *et al.* [67, 85, 86] with the intuit of analysing biological data, basis its operation in a similar rationale, but the differences are taken with respect to the line obtained by linear regression. This basically means that, for each block m_k and for each $i \in \mathbb{N}$, the partial sums $P_k^i(d)$, given by (2.53), are first approximated using the linear least squares method and then measured against the newly fitted line $y = \alpha_k^i + \beta_k^i \times x$, following the suggestion in (2.54). This procedure quantifies the *dispersion* of the values inside each aggregation scale:

$$P_k^i(d) = \sum_{j=im}^{im+d} Y(j); \tag{2.53}$$

$$F_k(i) = \sqrt{\frac{1}{N}(P_k^i(d) - \alpha_k^i - \beta_k^i \times d)^2}. \tag{2.54}$$

According to [85], the expected value of $F_k(i)$ is directly proportional to $C \times m_k^H$, where C is a positive real number (irrelevant in the scope of this explanation) and H is, as before, the Hurst parameter:

$$\mathbb{E}(F_k) = C \times m_k^H. \tag{2.55}$$

An estimate of H is normally obtained by plotting the values of the logarithm of m_k and of $\mathbb{E}(F_k)$ and by getting the slope of the line that best fits those values, as exemplified for several fBms in Figure 2.9. The slope of the line is an estimate of the Hurst parameter.

Even though this method is not overly popular, it has some good properties that should be emphasised here, as they justify why this estimator has been chosen to be part of some of the experiments reported in chapter 3, along with RS. The fact is that DFA is not susceptible to short-range dependency nor to small trends that may appear in the

data under analysis.

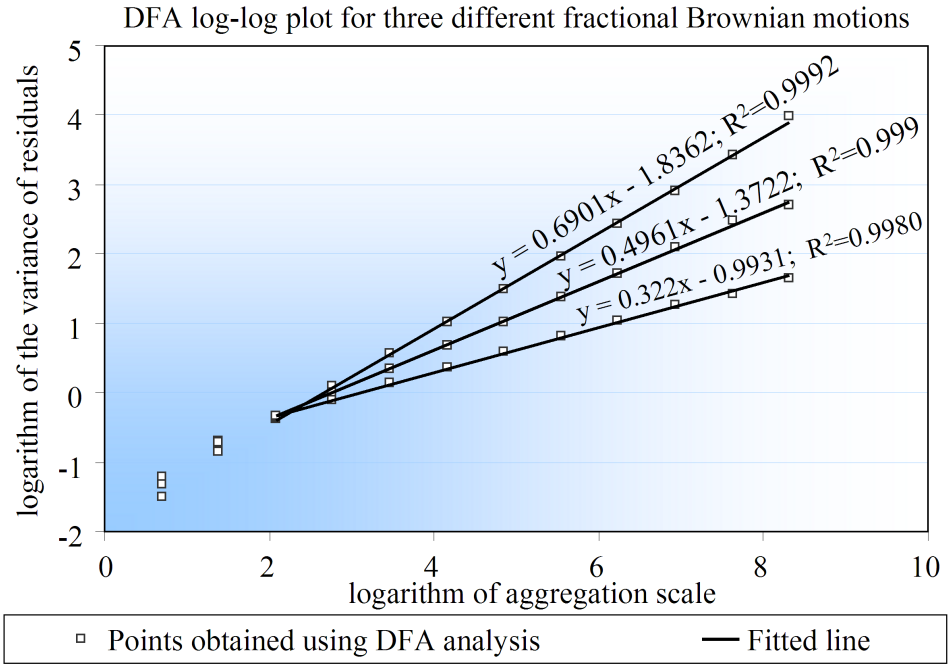

Figure 2.9: The DFA log-log plot for two fBms ($H = 0.3$ and $H = 0.7$) and for Brownian motion ($H = 0.5$). The aggregation scales were all powers of 2, starting at 1. The estimated Hurst parameter values were approximately 0.32 and 0.69 for the fBms and 0.5 for the Brownian motion.

The major drawback of DFA is its retrospective character, which cannot be disassociated from its nature. The calculation of $F_k(i)$ relies on the prior knowledge of *all* the points of a given aggregation block with size m_k, rendering its computational complexity as being of $O(n^2)$. As so, this method was merely chosen to be an impartial evaluator of the algorithms described afterwards, and it will not be the subject of further investigation.

In many references (see, for instance, [38, 61, 82]), this method is denominated as the Variance of Residuals (VR) method, inspired by the fact the initial definition of $F_k(i)$ (in [67]) has been made in terms of the variances of the *residuals* given by $P_k^i(d) - \alpha_k^i - \beta_k^i \times d$, and not in terms of their standard deviation. Because of that, when used as initially defined, the slope of the fitted line has to be divided by 2 before being considered an estimate of the Hurst parameter value.

2.4.6. Periodogram

The Periodogram method is based on the analysis of the spectral domain of the self-similar process, and more particularly on the fact that, for stochastic processes, the spectrum $f(\lambda)$ can be taken out of the autocorrelation function $\gamma(k)$ [38], as indicated by (2.56) where $e^{\dot{\imath}x}$ expands according to the formula of Euler for complex numbers:

$$f(\lambda) = \sum_{j=-\infty}^{+\infty} \gamma(j) \times e^{\dot{\imath}j\lambda}. \tag{2.56}$$

Thanks to the definition of $\gamma(k)$ (see section 2.2.2.), $f(\lambda)$ may be approximated by $c_f \times |\lambda|^{1-2H}$ when $\lambda \to 0$ (expression (2.57)), where H is the Hurst parameter and λ is the frequency:

$$f(\lambda) \sim c_f \times |\lambda|^{1-2H}, \text{ when } \lambda \to 0. \tag{2.57}$$

The Periodogram proposes to approximate $f(\lambda)$ by $I(\lambda)$, defined in (2.58), where N represents a predefined number of samples that ultimately delimits the calculations to a finite and computationally *supportable* number, and $\hat{\gamma}(j)$ is a statistic similar to the autocorrelation function of N samples, defined as in (2.59):

$$I(\lambda) = \frac{1}{N} \sum_{j=-(N-1)}^{N-1} \hat{\gamma}(j) \times e^{\dot{\imath}j\lambda}, \text{ where} \tag{2.58}$$

$$\hat{\gamma}(j) = \frac{i}{N} \sum_{k=0}^{N-|j|-1} (Y(k) - \mathbb{E}(Y))(Y(k+|j|) - \mathbb{E}(Y)). \tag{2.59}$$

It can be shown [38, 87] that the Periodogram $I(\lambda)$ can be further simplified into form (2.60), which embodies the most comfortable way of obtaining its values:

$$I(\lambda) = \frac{1}{N} \left| \sum_{j=0}^{N-1} (Y(j) - \mathbb{E}(Y)) \times e^{\dot{\imath}j\lambda} \right|^2. \tag{2.60}$$

The Hurst parameter estimation is obtained by calculating the Periodogram for several frequencies λ_k bigger than 0, and by fitting the logarithms of $I(\lambda_k)$ against the ones of λ_k. For the fitting procedure, it is common to consider only the values of λ_k that are *closer to the origin* (e.g. in [61], the authors used only 10% of the 10000 points for which $I(\lambda_k)$ was calculated). As hinted by (2.57), the Hurst parameter can then be retrieved from the formula $H = (1 - \beta)/2$, where β is the slope of the fitted line.

If it was not for the fact that $I(\lambda)$ constitutes a bad estimator of the spectral function [58], and for the fact that the values of the said density function oscillate a lot when calculated for realisations of the stochastic process (see Figure 2.10 and pay special attention to the indicated R^2 value), this estimator would be suitable for point-by-point estimation, as (2.59) could be easily adapted to retrieve a value of $I(\lambda)$ for each realisation of $\{Y(t)\}_{t\in\mathbb{N}}$, without requiring the recalculation for all the previous values of the series. Nonetheless, the estimations returned by the point-by-point implementation of this method would suffer from the bias introduced by the incremental calculation of the average $\mathbb{E}(Y)$.

Figure 2.10 contains the graphical representation of the Periodogram method. As can be seen, contrarily to the previously described methods, it is difficult to identify the linear relation in the log-log plot, even for small values of $\log(\lambda)$. The estimator was fed with a series simulating an fGn containing 131072 points, which had an expected Hurst parameter of 0.85. The estimated scaling exponent, however, was far from the expected value, as the application of the previously described procedure to 10% of the 1000 points of the chart (the ones most to the left), returned an Hurst parameter value of 0.79.

The previous statements do not render the method worthless for the assessment of the self-similarity degree of a given empirical series. If handled with caution and if implemented in the right way, the Periodogram can be used to retrieve good estimations of the Hurst parameter. For instance, the λ_k can be constructed in such a way (e.g. $\lambda_k = \lambda_{k-1} \times F_{actor}$) their respective logarithms are equally distributed in the x-axis of the log-log plot [61], resulting in a *more balanced* analysis of the fitted points. Nevertheless, the graphical analysis should never be disregarded, as the estimation depends critically on the number and set of points chosen to be fitted. For example, when repeating the previous experience (Figure 2.10) using exponentially spaced frequencies, and after observing the

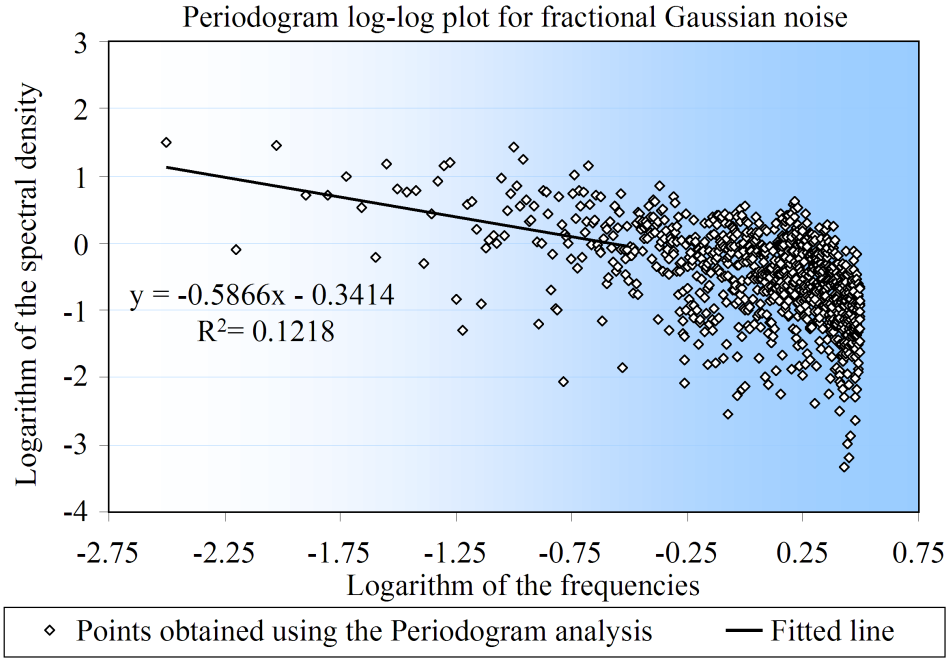

Figure 2.10: The Periodogram log-log plot for an fGn with expected Hurst parameter equal to 0.85. The estimated Hurst parameter value (obtained using 10% of the points most to the left in the chart) was 0.79.

plot carefully, 950 coordinates (out of 1000) were chosen to be fitted, giving rise to an estimate of the Hurst parameter of 0.86 (recall that the expected value is of 0.85). Because of these specific requirements, this method does not pose a particularly attractive solution for online applications, and it was decided not to consider it further.

2.4.7. Whittle Estimator

The Whittle Method (see e.g. [38, 58, 59, 61]) is based on the Maximum Likelihood Estimation of some of the parameters of long-range dependence, namely of the Hurst parameter. Analogously to the Periodogram, this method draws also on the spectral density of the self-similar stochastic process, and aims for the discovery of the parameters that minimise a *goodness of fit* function $Q(H)$, after assuming a predefined model for the spectral density. The function that one needs to minimise is usually written as in (2.61), where $f(\lambda)$ is as previously defined, and $f_H(\lambda)$ is the function describing the spectral

density of the presumed model, which in this case, depends on the Hurst parameter H:

$$Q(H) = \int_{-\pi}^{\pi} \frac{f(\lambda)}{f_H(\lambda)} d\lambda \tag{2.61}$$

If the functional form of the spectral density is only assumed for the frequencies near 0, the method is called *local Whittle estimator* (more on that subject can be found on [88]).

As the calculation of the spectral density is computationally intensive (the definition of $f(\lambda)$ may be found in section 2.4.6.), the approximation given by the Periodogram $I(\lambda)$ is commonly used as a substitute for the true spectral density function (equation (2.62)):

$$\hat{Q}(H) = \int_{-\pi}^{\pi} \frac{I(\lambda)}{f_H(\lambda)} d\lambda. \tag{2.62}$$

To make the calculations feasible, the integral in (2.62) has to be approximated by a Riemann sum (2.63), defined for $2K$ different λ_k values:

$$\hat{Q}(H) = \sum_{k=0}^{2K} \frac{I(\lambda_k)}{f_H(\lambda_k)}, \text{ where } \lambda_k = \pi k/K - \pi. \tag{2.63}$$

If the fact that the spectrum is symmetric at the origin is taken into consideration, the minimisation of (2.63) resumes itself to the minimisation of (2.64), which is the *most computationally efficient* form of $Q(H)$:

$$\hat{Q}(H) = \sum_{k=1}^{K} \frac{I(\lambda_k)}{f_H(\lambda_k)}, \text{ where } \lambda_k = \pi k/K. \tag{2.64}$$

The confidence interval of the output value is logically dependent of the number of points K considered in the Riemann sum, and can be formalised by (2.65), where $\hat{H}$ is the estimate of the Hurst parameter:

$$\sqrt{K}(\hat{H} - H) \underrightarrow{}_{N\to\infty} G(0,1) \sqrt{2 \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} \left(\frac{d}{dH} \log f_H(\lambda) \right)^2 d\lambda \right)^{-1}}. \tag{2.65}$$

The Whittle Estimator is principally known by three major facts: (i) it is asymptotically correct [38], (ii) it explicitly provides the estimation with the confidence intervals (see expression (2.65)), and (iii) it does not produce any kind of graphical outputs. The last mentioned fact is presented as a curiosity of this method, as it is common for this type of methods to provide some kind of a graphical perspective of self-similarity.

Despite of the first two presented advantages, the method has two major drawbacks. First of all, it needs *at least* twice the computations of the Periodogram method, and second of all, its operation lies on a strong assumption about the model of the sequence under analysis (the method presumes the functional form of the autocorrelation), which may not be known at the beginning of the procedure or may not exactly hold for empirical traces, which may lead to erroneous estimations of the Hurst exponent [38, 59, 58]. It is also true that this method can be fooled by *periodicity* [37], which may lead the estimator to categorise a periodic signal as being self-similar, and by trends (for which it produces erroneous results for the Hurst parameter). These disadvantages should not be perceived as prohibited for the usage of this method, but they are certainly not desirable for an online estimator, being this the main reason why this method is not going to receive any further consideration.

2.4.8. Wavelets-Based Estimator

In 1998, inspired by the apparent urge to efficiently estimate the parameters of long-range dependence, Darryl Veitch and Patrice Abry proposed Wavelets [68] as a new means to analyse self-similar processes, and to obtain estimates of the said parameters. The method that is nowadays known as the Abry Veitch (AV) estimator [89], is based on the recursive decomposition of the signal under investigation into a set of *detail* $\{d_j(k)\}_{k\in\mathbb{Z}}$ and *approximation* coefficients, and on the suitable analysis of that coefficients. For an exhaustive description of these concepts and of how to calculate the detail coefficients, the reader is referred to section III. A. of [90].

In [91] it is stated that, if the family of wavelets is defined by $\{\psi_{j,k}(t) = 2^{-j/2}\psi_0(2^{-j}t - k)\}_{j=1,\ldots,J}$, where $J, K \in \mathbb{Z}$, the estimation of the Hurst parameter is achieved by fitting a line to the points defined by $(j, \log_2(\mu_j))$, where j is as previously defined (also referred

to as *the octave*) and μ_j is the *variance* of the detail coefficients, defined by

$$\mu_j = \frac{1}{n_j} \sum_{k=1}^{n_j} d_j^2(k), \text{ where } n_j \text{ is the number of detail coefficients for octave } j. \quad (2.66)$$

The last presented metric (μ_j) may be understood as a *measure of the amount of energy* of the transformed signal at octave j and, under the long-range dependence assumption, its logarithm should depend linearly of the latter, being the slope β of the line written in terms of the Hurst parameter: $\beta = 2H - 1$.

The AV estimator is characterized by its many advantages. The method is fast, robust to trends and small deviations from stationarity, and may be implemented as a real-time estimator (all these advantages are duly explored and justified in [68, 89, 91]). Being its proposers perfectly aware of all these properties, specially of the last one, the procedure that handles the process as an incoming stream of points, in a computationally efficient manner, is now protected by the international patent application number PCT/AU1999/000077, entitled *Real-Time Estimation of Long Range Dependent Parameters* [92]. The authors of this patent are thus, pioneers in the identification (and modification) of the methods that can be used to estimate the Hurst parameter in a point-by-point manner, though it has been mostly applied to data series with an increasing number of points, rather than to sliding windows of values. In chapter 3, it is argued that the procedures with the operational model referred in last are actually more useful within the scope of this thesis.

Because the AV method is protected by a patent application, and taking into consideration the competitive industrial environment in which this work was conducted in, the wavelet method was not implemented in its point-by-point form. A retrospective version of this estimator, with a computational complexity of $O(n \log(n))$, was used to arbitrate the precision of one of the self-similar sequences generators presented in chapter 4.

2.4.9. Higuchi Method

The estimator entitled *Higuchi Method*, named after its creator [93], proposes the analysis of the quantity $L(m)$, designed in (2.67), as a means to estimate the scaling exponent of a self-similar process:

$$L(m) = \frac{N-1}{m^3} \sum_{i=1}^{m} \left\lfloor \frac{m}{N-i} \right\rfloor \sum_{l=1}^{\left\lfloor \frac{N-i}{m} \right\rfloor} \left| \sum_{j=i+(l-1)m+1}^{i+lm} X(j) \right|. \tag{2.67}$$

In (2.67), $\lfloor . \rfloor$ denotes the *truncation function*, sometimes referred to as the *floor function*, which is the transformation that returns the biggest integer smaller than x, and N denotes the total number of samples of the empirical sequence submitted to investigation. Once again, the referred analysis involves fitting a line to the logarithms of the values of $L(m)$ and of m, particularised for the several aggregation scales m_k. An estimate of the Hurst parameter can then be obtained by using the formula $H = \beta + 2$, where β is the slope of the fitted line.

As can be concluded from careful observation of (2.67), the computational complexity of this method is extremely high, mostly because $L(m)$ is calculated for *overlapping* blocks, instead of doing so for non-intersecting windows of values. The philosophy in which it draws on is similar to the one of the AMT for $n = 1$, but its exaggerated level of complexity does not result into a proportional gain in terms of precision. Furthermore, as stressed out by Dieker [38], the smallest aggregation scales should not be taken into account during the fitting procedure, suggesting that the method may be biased for that scales. As so, the Higuchi method will not be object of further exploitation in this thesis.

2.4.10. Hurst Exponent by Autocorrelation Function

As a clear demonstration that the interest around the subject of assessing the self-similarity degree is actual, in 2006, Gubner and Kettani of [70] proposed a *new* method for estimation of the Hurst parameter, and termed it of Hurst Exponent by Autocorrelation Function (HEAF). The basis of their proposal lies on the autocorrelation function $\gamma(k)$ of the self-similar process (in the sense described by (2.4)), which can be simplified into

the form depicted by

$$\gamma(k) = \frac{1}{2}\left(|k+1|^{2H} - 2|k|^{2H} + |k-1|^{2H}\right). \tag{2.68}$$

If the autocorrelation is calculated for $k = 1$, the previous formula degenerates into (2.69), which can then be solved for H, providing for an immediate estimate for the Hurst parameter (expression (2.70)). The authors go even further when stating that this *tool* can also provide confidence intervals, superiorly and inferiorly limited by the values of $H \pm 5/\sqrt{N}$, where N is the total number of occurrences of the exactly second order self-similar process:

$$\gamma(1) = 2^{2H-1} - 1 \tag{2.69}$$

$$\Leftrightarrow H = \frac{1}{2}\log_2(\gamma(1)+1) + \frac{1}{2}. \tag{2.70}$$

As defined, the method seems rather obvious and presents itself as a low computational complexity procedure. However, the author of this thesis is of the opinion that the method does not fully capture the self-similar properties of a given sequence of values, because it only explores the *relations* between the points that are distanced by a lag of 1 (recall that $\gamma(1)$ is the autocorrelation between $Y(t+1)$ and $Y(t)$, for $t \in \mathbb{N}$). It is a fact that exactly second order self-similar processes should respect (2.68) in general, and (2.69) in particular, but they are not the only type of stochastic processes that fulfil the condition mentioned in last.

After understanding the method proposed in [70] and after having developed two self-similar sequences generators, the author of this thesis has thought about a construction that would drive the procedure in [70] into the conclusion that the process under observation is long-range dependent when, actually, it is not. Such sequences can be obtained using the procedure presented by expressions (2.71) and (2.72), where $p_2 = 2^{2H-2}$ (refer to chapter 4 for an explanation about this particular construction):

$$S_{rd}(0) = G(0,1); \tag{2.71}$$

$$S_{rd}(t) = G\left((2p_2 - 1) \times S_{rd}(t-1), 4p_2(p_2 - 1)\right), \text{ for } t > 0. \tag{2.72}$$

The process $\{S_{rd}(t)\}_{t\in\mathbb{N}}$, generated according to the previous expressions is *short-range dependent* since, as can be concluded from careful observation of (2.72), each point depends *only* on the previous one. Nevertheless, it fulfils condition (2.69), as can be confirmed by the reasoning for the calculation of the autocorrelation for $k = 1$, represented by expressions (2.73) and (2.74):

$$\gamma_{S_{rd}(t)}(1) = \frac{\mathbb{E}\left((S_{rd}(t-1) - 0)((2p_2 - 1)S_{rd}(t-1) - 0)\right)}{\mathbb{E}(S_{rd}(t) - 0)^2} = \tag{2.73}$$

$$= (2p_2 - 1)\frac{\mathbb{E}(S_{rd})^2}{\mathbb{E}(S_{rd})^2} = 2^{2H-1} - 1. \tag{2.74}$$

This reasoning does not invalidate the method, but reinforces the idea that the Hurst parameter estimation has to be made for several aggregation scales and not for one only. The method works well when the properties of the series are known (except for the value of the Hurst parameter), being otherwise only useful to obtain the scaling exponent of points that are close to each other. Extending the rationale of the estimator to higher aggregation blocks would not be difficult, as for each aggregation scale, the process defined in (2.3) behaves precisely as the non aggregated one. However, this would increase the computation complexity to a level comparable with any other Hurst parameter estimation method. This is the main reason why the comparative analysis in [70] or elsewhere [60, 94] do not seem to be fair.

If HEAF was available by the time the first overview to the state of the art was conducted, the author would have consider it as a potential candidate to a real-time estimator, because it draws on a statistic that may be calculated in an incremental manner. As the method was published during the problem investigation phase, it was only discovered in a final stage of the Ph.D. research programme, during one of the last revisions to the state of the art. Nonetheless, the author thought it would be beneficial to include his personal analysis of the procedure, while emphasising that it will not be used in the scope of this research project. The pertinence of this analysis is justified during the discussion of some of the recent works in the area, namely of [25, 26], included in chapter 5, because some of those works are closely related with HEAF.

2.5. Conclusion

The chapter has introduced some of the most important concepts related with *self-similarity*. It contains the definition of *self-similarity*, *Hurst parameter*, *exactly second order self-similar process* and *asymptotically second order self-similar process*. It presents the notions of *persistence* and *anti-persistence*, and it specifies how they relate to the ones of *long-range dependence* and *short-range dependence*. The stochastic processes known by the acronyms fBm and fGn were described with detail, for they play an important role in the modelling of network aggregated traffic. Special attention was given to the definition of *Random Walk* and of its *first order differences process*, mostly because they are going to be key concepts on the subject of chapter 4.

The bridge between the theory and its application in the networking area is built in the intermediary section of the chapter. The description was made resorting to several prominent studies that can be found in the literature, aiming for the demystification of the subject at hands. Self-similarity is known to be embedded in the amount of information per time unit of the traffic, resulting from the aggregation of several flows coming from different terminal nodes of the network (e.g. a LAN). A possible explanation for that fact was provided by Taqqu et al. [48], who proved that the superimposition of several independent *ON/OFF processes* with heavy-tailed distribution functions approximates a long-range dependent series. A scaled and shifted fGn is often proposed as the model for the associated process.

The presence of long-range dependence in the network traffic impacts the way the protocol data units arrive to aggregation nodes, and explains why they may arrive in bursts. The necessity to understand how burstiness is reflected in queueing and congestion avoidance has been fuelling the interest of researchers and explains the numerous scientific contributions in this area of knowledge.

After describing and analysing several methods for the assessment of the self-similarity degree of an empirical sequence of values, it became clear that the bound between some of the estimators and the retrospective manner in which they are usually utilised, cannot be broken without irremediably loosing the accuracy of the calculations. Some of the estimators were said suitable for offline analysis, but not robust enough to handle an

incoming stream of data, either because they require the estimator parameters to be carefully adjusted or because they make assumptions that may lead to erroneous estimation of the Hurst parameter. Others yet presume human intervention before the production of meaningful estimates. From all the methods, three were pointed out as potential candidates for online analysis, because their retrospective character may be momentarily or definitely circumvented without prejudicing their precision.

It was said that, to complement the analysis made to the self-similar sequences generators described in chapter 4, two additional retrospective estimators were selected for implementation: the RS method, for historical reasons; and the DFA estimator, for its robustness. The method based on the wavelets has already been implemented as a real-time estimator, being that particular instantiation protected by a patent application [92]. Because of that, its implementation as a point-by-point estimator will not be tempted during the remaining part of this work. The $O(n \log(n))$ complexity procedure is, nonetheless, going to be used as an impartial evaluator of the quality of one of the two above mentioned algorithms for the simulation of approximate fGn.

Chapter 3

Fast and Windowed Estimation of the Hurst Parameter

3.1. Introduction

In the previous chapter, some of the most important concepts of the self-similarity theory were used to introduce the methods to estimate the Hurst parameter, being the latter designated as the *de facto* measure of the self-similarity degree of a given time series. Most of the methods are said *retrospective*, because they require the series under analysis to be entirely processed more than once for an unique assessment of the referred exponent. Their operational model is thus highly expensive in terms of computational resources.

Up until now, the analysis of network traffic traces has been mostly conducted offline, in which case the computational complexity of the estimators does not pose a direct problem. Only more recently, the curiosity about possible real-time applications of this type of assessment was considered, fomented by the proliferation of new Internet services, that increased the *dynamics and complexity* of the Internet and of all its subservient networks. The estimator based on wavelets [89] was the first being proposed as a real-time point-by-point estimator.

This chapter describes how three (previously identified) Hurst parameter estimators can be formally adapted to produce results in a continuous and fast manner. After presenting the philosophy of a windowed estimator and the reasons that led to their

development in the scope of this thesis, the same procedures are further modified, so as to comply with the newly described requirements. A brief discussion about the trade-off between *precision* and *real-time compliance* is included, prior to the section that compares the point-by-point and the windowed-modified estimators. The last section wraps up the most important remarks of the chapter.

The set of modifications that enable the VT method to operate as a windowed estimator were briefly enumerated in paper [29]. Herein, the formalisation of such alteration is postponed to the second section of the chapter. The windowed estimators were on the origin of most of the results discussed in [29, 30].

3.2. Point-by-Point Estimators

Before discussing the above mentioned modifications, it is pertinent to elaborate on the notions of *real-time* and *point-by-point* estimation. *Real-time* is the designation given to a system or to a procedure that provides results in a relatively short (depending on the context) and predictable time frame, as a response to an external event. In this work, the real-time Hurst parameter estimator is the (computational) procedure that is capable of completing the analysis of a data series and returning an accurate estimate of the referred parameter, in a period of time upon which the processing device can act in. Because the estimation of the self-similarity degree is done for an arbitrarily long series of points, real-time compatibility forces the procedure to be independent of the number of points to be analysed. In other words, the only way of *forcing* an Hurst parameter estimator to be real-time compliant, is to make it analyse the series in a point-by-point manner. Herein, the *point-by-point* estimator is the algorithm capable of returning a value of the Hurst parameter for each point of the series under analysis. Additionally, such procedure has to perform those computations in an efficient manner, avoiding recalculation over the entire series each time it is fed with a new point of the self-similar series.

Conceptually, such objective can be accomplished by assessing the possibility to construct a recursive algorithm that operates over a (finite) set of convenient *auxiliary variables* ($AV_1, AV_2, ..., AV_l$, where l is a fixed positive integer number) that summarise the calculations made so far. These variables are used to simply update (instead of

completely recalculate) the next estimate of the parameter, each time a new point of the process becomes available. The operational model of a point-by-point estimator is depicted in Figure 3.1.

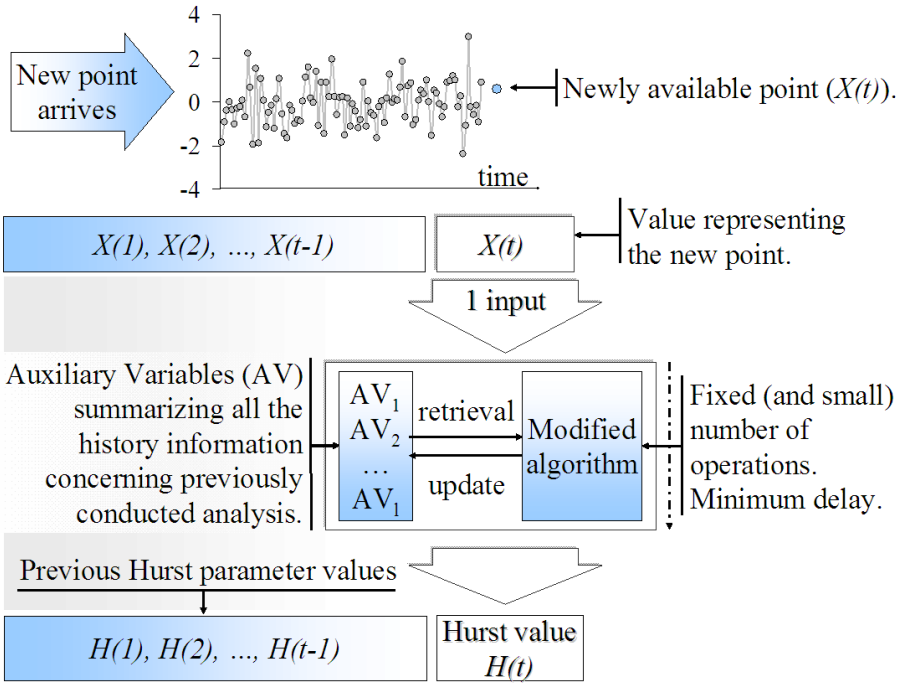

Figure 3.1: Point-by-point Hurst parameter estimation using a modified method: every time a new point of the series is available, it is fed to the estimator as its only input, originating a new estimate for the Hurst parameter in return.

The main purposes of a point-by-point estimator is to (i) provide an historical and continuous perspective of the self-similarity degree of the time series under analysis, while (ii) opening the possibility to act upon the outcomes of that view in a real-time manner. These two features *seemed* to perfectly fit the objective of studying the impact of a network level intrusion in the self-similarity degree of the traffic. The term *seemed* is emphasised because, as it will be explained in section 3.3., the point-by-point estimator does not embody the most adequate tool for such investigation.

3.2.1. Modified Embedded Branching Process

The modification of an estimator constitutes an effort to describe it as an $O(n)$ complexity algorithm, while bearing in mind such endeavour may encompass small deviations to its original definition. This requires the identification of the previously mentioned

set of *auxiliary variables* that, within the context of a specific method, can summarise the analysis conducted for $t-1$ points, in such a way that they can be used to rapidly update the Hurst parameter when the point with time index t becomes available. The first method modified during this research work was EBP and, during the remaining part of this document, the designation Modified Embedded Branching Process (MEBP) will be used to refer its *adapted* point-by-point implementation. Notice that the following explanation makes use of the same notation and definitions introduced in section 2.4.4..

Recall that EBP makes use of the so-called crossing tree structure that, according to [69], is embedded in any self-similar process. The number of crossings N_k that define the tree, for each level $k = 1, ..., K$, is normally calculated following an horizontal perspective, meaning that for each level k, the process $T_k(i)$ is entirely constructed before moving to the next level. Therefore, using that estimator to get an historical assessment of the Hurst parameter requires that perspective to be changed from *horizontal* to *vertical*. This new calculation philosophy requires that, for a given moment in time t, the previous number of crossings for each level is known.

Consider that $\{X(t)\}_{t\in\mathbb{N}}$ has met $t-1$ occurrences and that, until that moment, the number of times the process *crossed* the family of horizontal lines defined by $f(k) = \delta 2^k\mathbb{Z}$ is given by $N_k(t-1)$. Consider also that the line that was last crossed at level k is given by $y = CL_k(t-1)$. When a new point of the process $\{X(t)\}_{t\in\mathbb{N}}$ *arrives*, these *auxiliary variables* ($N_k(t-1)$ and $CL_k(t-1)$) can be actualised using formulas (3.1) and (3.2), where $\lfloor . \rfloor$ denotes the *floor function*, $C_k(t-1)$ is the number of crossings for level k between occurrences $X(t-1)$ and $X(t)$, and $sign(x)$ is defined as $sign(x) = x/\lfloor x \rfloor$:

$$N_k(t) = N_k(t-1) + C_k(t-1), \text{ where } C_k(t-1) = \left\lfloor \frac{|X(t) - CL_k(t-1)|}{\delta 2^k} \right\rfloor; \tag{3.1}$$

$$CL_k(t) = CL_k(t-1) + sign\left(X(t) - CL_k(t-1)\right) \times C_k(t-1) \times \delta 2^k. \tag{3.2}$$

The updated values of $N_k(t)$ can then be used to get the values of μ_k (equation (2.52)), and by means of a linear regression, the Hurst exponent can be retrieved.

The two first formulas of this section show that the actualisation of the critical

variables for estimation of the Hurst parameter, using the EBP method, can be made by means of a small number of operations. They also show that the complexity of the computations does not depend on the index of the point to be processed by any means, only from the number of crossing levels taken into consideration. Every time a point of the process arrives, the number of crossings is calculated using the *distance* from the current point and the previous crossing line. This number, divided by the scale factor $\delta 2^k$ expresses the total number of lines it crossed. Finally, the crossing line (given by $CL_k(t)$) is updated so as to reflect the last line of the family $f(k)$ to be crossed, for future assessments.

The last thing one must take into consideration are the characteristics of the empirical sequences fed to this modified estimator. During an online analysis, one may not know many of the characteristics of the data series as, for example, its average or variance. However, in section 2.4.4., it was emphasised that one of the requirements of EBP is that the input series first order differences has to be centred at the origin. It was also mentioned that the δ value has to be set accordingly to the standard deviation of that process. Thus, the application of MEBP, as defined, still requires the empirical realisations of $\{X(t)\}_{t\in\mathbb{N}}$ (or of its first order differences process, to be more precise) to be normalised on-the-fly, before being processed.

Normalisation requires the average and the variance to be calculated in a point-by-point manner, which may be done via the utilisation of the well-known formulas (3.3) and (3.4), where $\mathbb{E}_t(Y)$ and $\mathbb{V}_t(Y)$ denote respectively the average and the variance of exactly t samples of $\{Y(t)\}_{t\in\mathbb{N}}$:

$$\mathbb{E}_t(Y) = \frac{(t-1)\times\mathbb{E}_{t-1}(Y)+Y(t)}{t}; \tag{3.3}$$

$$\mathbb{V}_t(Y) = \mathbb{E}_t\left(Y^2\right) - \left(\mathbb{E}_t(Y)\right)^2. \tag{3.4}$$

The value $X(t)$ that should be used for counting the number of crossings to the several families of lines $f(k)$ is retrieved from $X(t) = X(t-1)+Y(t)$, where $Y(t)$ is the result of normalising the most recent empirical value of the first order differences series (the latter is herein denoted by $Y'(t)$). Equation (3.5) describes the transformation suffered by the

non-normalised input series $\{Y'(t)\}_{t\in\mathbb{N}}$, which is applied before each point of the series enters MEBP:

$$Y(t) = \frac{Y'(t) - \mathbb{E}_t(Y')}{\sqrt{\mathbb{V}_t(Y')}}. \tag{3.5}$$

However, this procedure is not totally transparent and it is unavoidably associated with the problem of performing normalisation in an incremental manner. During an initial period (which may be more or less long, depending on how shifted and variable the empirical series is), the incoming points are differently affected (scaled) by the momentary values of the average and of the variance, which may not be equal to the exact values to which that statistics will eventually converge. This fact introduces some bias into the calculations, as the number of crossings depends drastically on the normalisation prerequisite. Eliminating the bias would only be possible using retrospection, which would ultimately render the method useless for real-time estimation. Thus, when using MEBP, one must take into account that the method may be overestimating the Hurst parameter if the standard deviation is smaller than expected and the sequence of values is shifted (i.e. not centred at 0).

3.2.2. Modified Variance Time

The second method selected to be modified is the VT estimator. In accordance with the nomenclature used in the previous section, the version of VT capable of estimating the Hurst parameter in a point-by-point manner for an increasing set of values is termed herein Modified Variance Time (MVT).

As explained in section 2.4.2., VT assesses the degree of self-similarity of a given stochastic process by aggregating the first order differences process several times, by calculating the variances of each one of the aggregated series, and by exploring the equality given by $\mathbb{V}(Y)) = m_k^{2-2H}\mathbb{V}\left(Y^{(m_k)}\right)$. Thus, the *auxiliary variables* of this estimator are the variances of the aggregated process for each m_k considered, that is to say $\mathbb{V}\left(Y^{(m_k)}\right)$, and the variance of the process itself, $\mathbb{V}(Y)$. But while the variance can be easily calculated for a *normal* series of values in a point-by-point manner, the same cannot be affirmed about the variances of the aggregated series. In fact, the value of $\mathbb{V}(Y)$ can be directly obtained

using formulas (3.3) and (3.4), while the precise values of $\mathbb{V}\left(Y^{(m_k)}\right)$ can only be obtained when the last occurrence of $\{Y^{(m_k)}(i)\}_{i\in\mathbb{N}}$, say $Y^{(m_k)}(T_k)$, is correctly constructed (T_k is the index of the k aggregated block to which $Y(t)$ belongs).

Consider that $\{Y(t)\}_{t\in\mathbb{N}}$ has met $t-1$ occurrences and that, until that moment, the previously identified k *auxiliary variables* where already calculated in a point-by-point or retrospective manner. When point $Y(t)$ becomes available, $\mathbb{V}(Y)$ can be immediately updated using (3.3) and (3.4).

In order to actualise $\mathbb{V}\left(Y^{(m_k)}\right)$, one has to first construct the *most recent point* of the aggregated series $\{Y^{(m_k)}(i)\}_{i\in\mathbb{N}}$ which, unfortunately, requires m_k points. In other words, the newly available realisation $Y(t)$ can only be seen as a small contribution of $Y^{(m_k)}(T_k)$ because, by definition, the latter is given by (3.6):

$$Y^{(m_k)}(T_k) = \frac{Y(T_k m_k) + ... + Y(t) + ... + Y((T_k+1)m_k - 1)}{m_k}. \tag{3.6}$$

Notice that, in the previous equation, the point that is *entering* the estimator is strategically placed within the dividend of the fraction, precisely to emphasise that its statistical weight in the construction of $Y^{(m_k)}(T_k)$ is given by the factor $1/m_k$.

To overcome this minor setback, consider the approximation of $Y^{(m_k)}(T_k)$ given by

$$Y_{aux}^{(r_k)}(T_k) = \frac{\sum_{j=T_k m_k}^{T_k m_k + r_k} Y(j)}{r_k}, \text{ where } r_k = 1, ..., m_k. \tag{3.7}$$

Allow the incoming point $Y(t)$ to be the r_k^{th} point of the most recent block of $\{Y^{(m_k)}(i)\}_{i\in\mathbb{N}}$, previously denoted by $Y^{(m_k)}(T_k)$. Notice that the new *auxiliary variable* $Y_{axu}^{(r_k)}(T_k)$, defined in (3.7), is the average of the r_k most recent (and available) values of $\{Y(t)\}_{t\in\mathbb{N}}$ belonging to the aggregation block under consideration (i.e. it is the only available until that moment). Note also that the value of the referred variable tends to the one of $Y^{(m_k)}(T_k)$, as the number of points available to construct the aggregation block increases, as shown by expression (3.8):

$$Y_{aux}^{(r_k)}(T_k) \longrightarrow Y^{(m_k)}(T_k), \text{ as } r_k \to m_k. \tag{3.8}$$

It is obvious that, when the incoming point is the first contribute to $Y^{(m_k)}(T_k)$, the auxiliary variable takes its sole value (expression (3.9)), and that once m_k values become available, the said variable is equal to $Y^{(m_k)}(T_k)$ (expression (3.10)):

$$Y_{aux}^{(r_k)}(T_k) = Y(t+1), \text{ if } r_k = 1; \tag{3.9}$$

$$Y_{aux}^{(r_k)}(T_k) = Y^{(m_k)}(T_k), \text{ if } r_k = m_k. \tag{3.10}$$

While a given aggregation block is not fulfilled yet, the value of $Y_{aux}^{(r_k)}(T_k)$ can be recursively updated each time a new point enters MVT, by means of the algorithm formalised by (3.11) and (3.12):

$$Y_{aux}^{(r_k+1)}(T_k) = \frac{Y^{(r_k)}(T_k) \times r_k + Y(t)}{r_k + 1}; \tag{3.11}$$

$$r_k \longleftarrow r_k + 1, \text{ if } r_k < m_k. \tag{3.12}$$

The values of $\mathbb{V}\left(Y^{(m_k)}\right)$ can then be approximated using the ephemeral values of $Y_{aux}^{(r_k)}(T_k)$, as described by (3.13) and (3.14):

$$\mathbb{E}_{T_k}^{aux}\left(Y^{(m_k)}\right) = \frac{(T_k - 1) \times \mathbb{E}_{T_k-1}\left(Y^{(m_k)}\right) + Y_{aux}^{(r_k+1)}(T_k)}{T_k}; \tag{3.13}$$

$$\mathbb{V}(Y^{(m_k)}) = \mathbb{E}_{T_k}^{aux}\left(Y^{(m_k)}\right)^2 - \left(\mathbb{E}_{T_k}^{aux}\left(Y^{(m_k)}\right)\right)^2. \tag{3.14}$$

The iterative actualisation of $\mathbb{E}_{T_k}^{aux}\left(Y^{(m_k)}\right)^2$, required for the calculation of the variance in (3.14), is not included, because it is also a direct application of (3.13), only for different variables. Once the aggregation block is complete, the average of the aggregated series (and of its square values) can be definitely substituted by the correct value, as formalised in the next expression:

$$\mathbb{E}_{T_k}\left(Y^{(m_k)}\right) = \mathbb{E}_{T_k}^{aux}\left(Y^{(m_k)}\right), \text{ if } r_k = m_k. \tag{3.15}$$

The reasoning is actually simpler than it looks. The algorithm proposes updating all the variances in a point-by-point manner, as previously discussed. As for each aggregated series, the last block may not be completed, MVT suggests using the aggregation of the $r_k < m_k$ values available, until the required amount of points is sufficient to fill up that aggregation block. This procedure implies disrespecting the definition of the original method for *some moments* but, as that definition is restored once enough points are available, the foundations and the precision of the method are maintained. Since for a fixed and finite number of block sizes k, the number of variables needed for estimation of the Hurst parameter is also fixed and finite (it does not increase with the number of samples to be observed), and since all the variables can be upgraded using recursive formulas, MVT can be used to estimate the Hurst parameter in a point-by-point manner. Unlike MEBP, MVT does not presume that the input process is normalised. MVT inherits the *robustness* of the variance metric which, by construction, is independent of the average of the stochastic process under analysis.

3.2.3. Modified Absolute Moments Time

The point-by-point adaptation of VT was made by first describing the *auxiliary variables* (the variances) in term of expected values. As the expected value of an arbitrary process can be updated in an iterative manner, the reasoning was reduced on how to account for the incomplete aggregation blocks. The adaptation of AMT to a point-by-point estimator (termed herein Modified Absolute Moments Time (MAMT)) can be made by following the same line of thought applied to MVT and, given that the means to handle the incomplete aggregation blocks and the formulas to update the averages were already disclosed, such task is confined to the demonstration that, under certain restrictions, the absolute moment of a stochastic process can be defined in terms of expected values.

For clarity reasons, the definition of absolute moment of order n of a stochastic process $\{Y(t)\}_{t\in\mathbb{N}}$ was transposed here (expression (3.16)):

$$M_n(Y) = E|Y(t) - \mathbb{E}(Y)|^n, \text{ where } n \in \mathbb{N}. \tag{3.16}$$

For odd values of n, expression (3.16) cannot be simplified in a trivial way. However, if

n is even, the symbol for the absolute value can be replaced by parenthesis (depicted in expression (3.17)) and the Newton formula can be applied to the term on the right side of the equality (3.17):

$$M_n(Y) = \mathbb{E}(Y(t) - \mathbb{E}(Y))^n, \text{ if } n \in 2\mathbb{N}. \tag{3.17}$$

The expansion of (3.17) according to Newton is given by (3.18), where $\binom{n}{i}$ denotes the combinations of n elements in subsets of size i, $\mathbb{E}(Y)^i$ is the expected value of $\{Y^i(t)\}_{t\in\mathbb{N}}$, and $(E(Y))^{n-i}$ is the power of $n-i$ of the expected value of $\{Y(t)\}_{t\in\mathbb{N}}$:

$$M_n(Y) = \sum_{i=0}^{n} (-1)^i \binom{n}{i} \mathbb{E}(Y)^i (\mathbb{E}(Y))^{n-i}, \text{ if } n \in 2\mathbb{N}. \tag{3.18}$$

$\mathbb{E}(Y)^i$ can be efficiently calculated, for any $i = 1, ..., n$, using formula (3.3). As for the expected values of the aggregated series, the same *auxiliary variables* $Y_{aux}^{(r_k)}(T_k)$ that were used for MVT (see section 3.2.2.), can be used to momentarily update the referred averages, and to retrieve an Hurst parameter estimate in a point-by-point manner, after feeding the logarithms of the aggregation scales m_k and of the k absolute moments to a least squares method (see section 2.4.3.). The only condition AMT must comply with, is that the order n of the absolute moment *has to be an even number*. If the number of aggregation scales is finite, the number of necessary variables and operations for upgrading them is also finite.

The actual implementation of AMT for the sake of this investigation is not critical, because its use does not bring any gain to the investigation, when compared to MVT. MVT is simpler and appropriate for the task. The previous description was included here for the sake of completeness, since AMT was identified as being a suitable candidate for the point-by-point adaptation, after VT. For these reasons, this estimator will not receive further consideration, apart from the brief discussion about its potential implementation as a windowed estimator, in section 3.3.3..

3.3. Windowed Estimators

It soon came to the attention of the author that the values returned by the point-by-point methods suffer from the same effect any other statistical measure suffers: the *law of large numbers*, i.e., when calculated for an increasing set of points, the Hurst parameter converges to a stable value (the average) and, once a sufficiently large number of samples has been processed, the estimated values are no longer sensitive to variations of the incoming values. Consider visiting section 3.4.3. for a graphical perspective of these words.

The aforementioned drawback redirected the investigation towards the means to get *local scope* (instead of *global*) Hurst parameter estimates. The main goal was still to do so in an efficient and point-by-point manner. The most logical way of proceeding was to define a *sliding observation window* of values, for which the Hurst parameter was to be estimated.

The sliding window philosophy that was considered in this thesis is schematised in Figure 3.2, where it is emphasised the fact that the fixed set of points under observation is being updated in a point-by-point manner: when a new realisation of the process under analysis becomes available, the observation window is *shifted* in one position, meaning that the *oldest* point of the set is discarded, and the new one is included. From now on, to avoid ambiguities, the expression *step-by-step* will be used to refer the point-by-point movement of the windowed estimators, while the previous designation will remain valid for the incremental methods. It may be noticed that, sometimes, the designations *local scope Hurst parameter estimates* or *estimation* are respectively used to refer to the values returned by the windowed estimators, or to the procedure itself. The designation *global scope Hurst parameter value* is used to refer to the estimate that reflects the self-similarity degree of the entire series. *Incremental estimator* may also be used interchangeably with *point-by-point estimator*, in contexts where the historical perspective of these procedures is to be emphasised.

The application of a retrospective method to the observation window does not embody an attractive choice for an efficient method, specially after having described point-by-point estimators. Again, the interest falls upon the possibility to, somehow, restrict

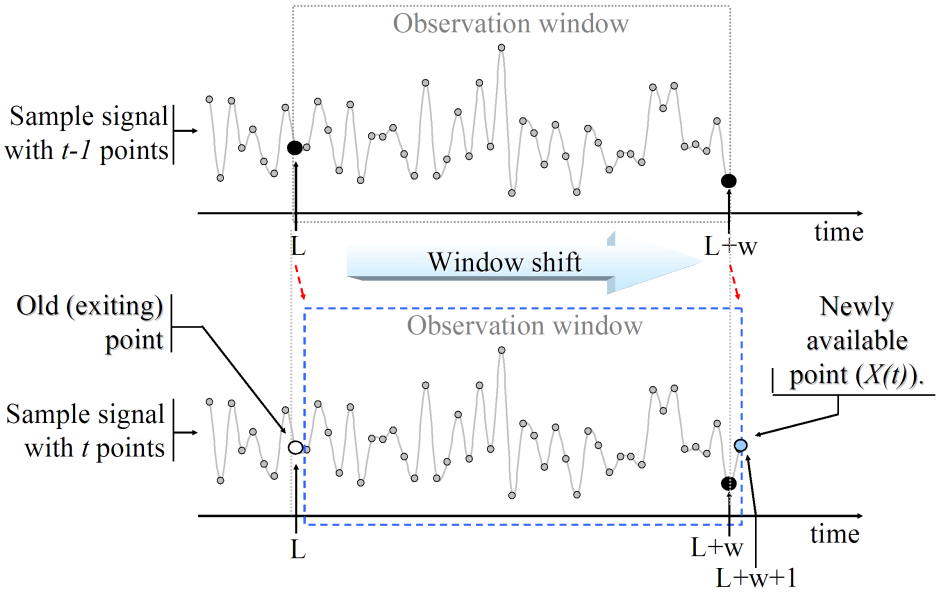

Figure 3.2: The sliding window philosophy: each time a new realisation of the process becomes available, the oldest point within the observation window is discarded and the window is shifted to the right by one position, so as to include the new value.

the number of operations to the processing of the incoming (as in point-by-point estimators) and of the outgoing points only. Figure 3.3 schematises the way of functioning of the procedure where the window shift implies the actualisation of the auxiliary variables, so they reflect the newly arrived point (say $X(L+w+1)$, where w is the size of the window) and, simultaneously, discard the information concerning the value that exits from the observation window ($X(L)$). This new type of estimator requires the values inside the observation window to be stored against their index, not to be repeatedly processed, but for the estimator to be capable of eliminating their effect in the *auxiliary variables* when they are about to leave. This novel memory structure was not depicted in the conceptual presentation of the point-by-point estimators.

It should be mentioned that the concept of sliding observation window, or the way that that concept is to be used herein differs from the ones in e.g. [95, 96]. For example, while Stoev, Park, Taqqu, Michailidis and Marron [95] have the same notion of observation window, the concept is not applied in the same way. This last mentioned paper describes an *animation tool*, that shows the evolution of the Hurst parameter during a windowed analysis of a given trace. The authors of the tool presented the *curse of the non-stationarity* of a traffic trace as main driver for their work, emphasising the fact that

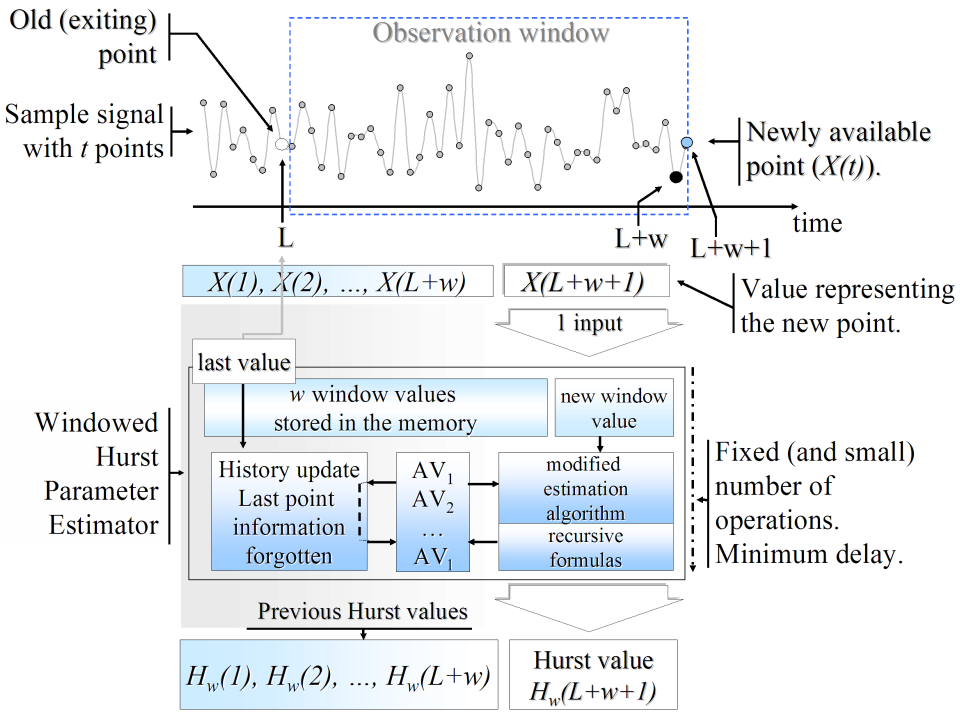

Figure 3.3: Step-by-step Hurst parameter estimation using a windowed-modified method. Every time a new point of the series becomes available, it is fed to the estimator as its only input. The modified algorithm *erases* the impact of the exiting point from the *auxiliary variables*, and *adds* the information concerning the new one.

with this type of analysis, such problem may be solved. As the Hurst parameter values are calculated in a retrospective manner, and to keep the computational complexity low, the movement of the observation window is defined by *jumps* and not on a point-by-point manner. The jumps are of the same size of the observation window. The work in [95] favours the approach taken herein, providing an additional motivation for the development of a tool to efficiently assess the local long-range dependence structure of a time series.

When compared with the tool in [95], the main advantage of the windowed estimator proposed here (see Figure 3.3) is that it enables the construction of a *continuous* historical perspective of the local scope Hurst parameter values, instead of providing only the values obtained for several non-intersecting parts of the series under analysis. The term *continuous* is, in this context, utilised to refer to the perspective provided by the histograms that may be built using the consecutive estimates of the Hurst parameter, which may appear to be continuous lines for the human eye. The notion of window in

[96] (which dates from 2007) is also equal to the one used herein, and the analysis conducted over a stock market related time series was used to produce a (almost) continuous illustration of the evolution of the local Hurst parameter values. Nonetheless, each one of those values was calculated in a retrospective manner, by means of DFA.

In the following sections, the methods that were previously modified are going to be revisited once more, so as to discuss their application as windowed estimators. The descriptions apply mostly to the cases where the number of points available are already sufficient to fill up the observation window (i.e. $t \geq w$), since for the other occasions, the windowed estimator behaves like the point-by-point estimators.

3.3.1. Windowed Modified Embedded Branching Process

In section 3.2.1., it was explained how the method designated by MEBP could be used to assess the self-similarity degree in a point-by-point manner. In order for the estimator to work, the variables that need to be kept from its retrospective version are the number of crossings ($N_k(t)$) and the constant that defines the last crossed line ($CL_k(t)$), for each level $k = 1, ..., K$. Each time the stochastic process meets a concretisation, and once the newly available point is normalised, these two sets of variables are updated using the recursive formulas (3.1) and (3.2). Using very simple terms, one could say that (3.1) and (3.2) are only *adding* the effect of the newly arrived point to the *auxiliary variables*.

In a windowed estimator, besides having one point entering the observation window, there is also one leaving. Allow for the leaving point to be denoted by $X(L)$, and the entering point by $X(L + w + 1)$, where w is the size of the observation window. Consider that all the points inside the sliding window are stored in the memory, as suggested in Figure 3.3, and notice that, as w is a fixed and finite number, the amount of memory is also fixed and limited for any instantiation of such estimator.

Adding the number of crossings of $X(L + w + 1)$ to $N_k(L + w)$ can be made as previously described, but subtracting the effect of $X(L)$ requires further consideration. The ideal situation would be to subtract the exact number of crossings that $X(L)$ inflicted to $N_k(L - 1)$ when it entered the observation window, w steps in the past. This can be

achieved in two different ways: (i) the first comprises storing all the number of crossings (for each level) made by each one of the points inside the window ($C_k(i)_{i=L,\ldots,L+w}$); and (ii) the second implies keeping a third *auxiliary variable*, denoted herein by $EL_k(L-1)$, that represents the last line that $X(L-1)$ has crossed for level k.

The first procedure requires the storage of $k \times w$ values, but its application is straightforward. Each time a point $X(L)$ is leaving the set under observation, the procedure fetches the corresponding number of crossings $C_k(L-1)$ from the memory, updates $N_k(L+w+1)$, according to equation (3.19), and discards the said value, because it will not be needed any more. The value of $C_k(L+w)$ can then take the place of the disposed one. The value of $C_k(L+w)$, in (3.19), is calculated the same way $C_k(t-1)$ is assessed in (3.1). Notice that $C_k(t-1)$ and $C_k(L+w)$ are essentially the same, but are written using different notations:

$$N_k(t) = N_k(t-1) - C_k(L-1) + C_k(L+w). \tag{3.19}$$

The second procedure presumes the calculation of $C_k(L-1)$ on an online basis, which means that, when a point is leaving, the number of crossings is calculated as if it was entering a (global context) point-by-point estimator. The value of $C_k(L-1)$ is then subtracted (instead of added) to the value of $N_k(t-1)$, as depicted by (3.20) and (3.21):

$$N_k(t) = N_k(t-1) - C_k(L-1) + C_k(L+w), \text{ where} \tag{3.20}$$

$$C_k(L-1) = \left\lfloor \frac{|X(L) - EL_k(L-1)|}{\delta 2^k} \right\rfloor. \tag{3.21}$$

The auxiliary variable $EL_k(L)$ is updated (meaning its previous value may be discarded) by means of (3.22):

$$EL_k(L) = EL_k(L-1) + sign\left(X(L) - EL_k(L-1)\right) \times C_k(L-1) \times \delta 2^k. \tag{3.22}$$

As can be seen, none of the recursive formulas depends on the size of the window. This means that, under certain restrictions, EBP can be used as a windowed estima-

tor, being that particular version herein designated by Windowed Modified Embedded Branching Process (WMEBP).

The next concern is unavoidably related with the restrictions that the input data series must meet, in order to be compliant with the EBP method. Once more, the data series has to be normalised before entering the estimator and, this time, such normalisation should be of local scope as well. This means that the average and the variance of the empirical input series have to be calculated on-the-fly, for a moving window of values. This requirement is fully addressed by the notion of *moving average*, commonly formalised as in

$$\mathbb{E}_{t,w}(Y) = \mathbb{E}_{t-1,w}(Y) + \frac{Y(L+w+1) - Y(L)}{w}. \tag{3.23}$$

The variance and the readjusted version of the series can once again be obtained using (2.45) and (3.5).

The implementation of WMEBP used in the scope of this work is in accordance with the second approach described in this section. Additionally, the codifications of the modified versions of this method draw on the least squares method to obtain the Hurst parameter, according to what was suggested in section 2.4.4., and perform the normalisation of the series in an online basis, if such is requested in the initialization procedure.

3.3.2. Windowed Modified Variance Time

The last comments of the previous section suggest that VT can also be modified to behave like a step-by-step windowed estimator. Such procedure is herein termed of Windowed Modified Variance Time (WMVT) method.

The means to calculate the variance in a point-by-point manner were enumerated in section 3.2.2., where it was also described how the last aggregation block of each one of the $\{Y^{(m_k)}(i)\}_{i\in\mathbb{N}}$ series could be taken into account, even for the situations where the required number of points to fulfil that block was not sufficient. The sliding window estimator can be constructed in an analogous manner, by defining a novel *auxiliary variable* for the

point or blocks that are leaving the observation window. Once the blocks are fulfilled, their effect on the *auxiliary variables* is definitively removed from the latter.

Consider $\mathbb{E}_{T_k,w}(Y^{(m_k)})$ and $\mathbb{E}_{T_k,w}\left(Y^{(m_k)}\right)^2$ as being the main *auxiliary variables* of WMVT. Note that, according to the previously introduced notation (see (3.23)), they denote the moving averages of $\{Y^{(m_k)}(i)\}_{i\in\mathbb{N}}$ and $\{\left(Y^{(m_k)}(i)\right)^2\}_{i\in\mathbb{N}}$, but this time for observation windows with w/m_k values. The definition of moving average for aggregated series requires one to take into account that the total number of samples w is divided by m_k because, after the aggregation, the number of available samples is reduced by that factor (hence the value of the divisor in (3.27) and in (3.28)). For the sake of this explanation, assume that w is chosen in such a way that w/m_k are positive integer numbers for each one of the considered aggregation scales. Consider also that $Y_{aux}^{(r_k)}(T_k)$ and $Y_{aux}^{(e_k)}(E_k)$ provide for the approximations of the blocks whose points are entering and leaving the observation window, respectively. At certain moments of time, these blocks may be simultaneously inside and outside of the observation window. The formal definition of the first of these two variables ($Y_{aux}^{(r_k)}(T_k)$) was presented in (3.7), and the second is defined in a similar manner in (3.24), where E_k denotes the index of the aggregated block of size m_k to which the exiting point $Y(L)$ belongs:

$$Y_{aux}^{(e_k)}(E_k) = \frac{\sum_{j=E_k m_k}^{E_k m_k + e_k} Y(j)}{e_k}, \text{ where } e_k = 1, ..., m_k. \tag{3.24}$$

When the window shifts, variable $Y_{aux}^{(e_k)}(E_k)$ is updated the same way $Y_{aux}^{(r_k)}(T_k)$ is, but using different values. The first is updated by means of (3.25) and (3.26), using the value of the oldest point of the set under observation ($Y(L)$), while the second is actualised for the incoming value, as defined for the point-by-point estimator:

$$Y_{aux}^{(e_k+1)}(E_k) = \frac{Y^{(e_k)}(E_k) \times e_k + Y(L)}{e_k + 1}; \tag{3.25}$$

$$e_k \longleftarrow e_k + 1, \text{ if } e_k < m_k. \tag{3.26}$$

The values of $\mathbb{E}_{T_k,w}(Y^{(m_k)})$ and $\mathbb{E}_{T_k,w}\left(Y^{(m_k)}\right)^2$ can then be momentarily actualised

(or substituted) by drawing on the notion of moving average, as described by equations (3.27) and (3.28):

$$\mathbb{E}^{aux}_{T_k,w}\left(Y^{(m_k)}\right) = \mathbb{E}_{T_k-1,w}\left(Y^{(m_k)}\right) + \frac{Y^{(r_k)}_{aux}(T_k) - Y^{(e_k)}_{aux}(E_k)}{w/m_k}; \tag{3.27}$$

$$\mathbb{E}^{aux}_{T_k,w}\left(Y^{(m_k)}\right)^2 = \mathbb{E}_{T_k-1,w}\left(Y^{(m_k)}\right)^2 + \frac{\left(Y^{(r_k)}_{aux}(T_k)\right)^2 - \left(Y^{(e_k)}_{aux}(E_k)\right)^2}{w/m_k}. \tag{3.28}$$

$\mathbb{E}^{aux}_{T_k,w}(Y^{(m_k)})$ and $\mathbb{E}^{aux}_{T_k,w}\left(Y^{(m_k)}\right)^2$ are approximations of $\mathbb{E}_{T_k,w}(Y^{(m_k)})$ and $\mathbb{E}_{T_k,w}\left(Y^{(m_k)}\right)^2$, defined for the purpose of preserving the exact values of the latter during the phase where the aggregation blocks are not completely filled in yet. $\mathbb{E}_{T_k,w}(Y^{(m_k)})$ and $\mathbb{E}_{T_k,w}\left(Y^{(m_k)}\right)^2$ can be definitely updated once enough points have entered or left the observation window. The following expressions represent the case referred in last, i.e. $e_k = m_k$:

$$\mathbb{E}_{T_k,w}(Y^{(m_k)}) = \mathbb{E}^{aux}_{T_k,w}(Y^{(m_k)}), \text{ if } e_k = m_k; \tag{3.29}$$

$$\mathbb{E}_{T_k,w}\left(Y^{(m_k)}\right)^2 = \mathbb{E}^{aux}_{T_k,w}\left(Y^{(m_k)}\right)^2, \text{ if } e_k = m_k. \tag{3.30}$$

The only problem that may rise from this solution, is that the use of formula (3.28) may inadvertently lead to erroneous (negative) estimations of $\mathbb{E}^{aux}_{T_k}\left(Y^{(m_k)}\right)^2$. This problem is mostly due to the values that are being used to update this estimate, as the auxiliary variable $\left(Y^{(e_k)}_{aux}(E_k)\right)^2$ may be overestimating the real value of $\left(Y^{(m_k)}(E_k)\right)^2$ when $e_k < m_k$ (notice that $\left(Y^{(e_k)}(E_k)\right)^2$ is subtracted to $\mathbb{E}_{T_k}\left(Y^{(m_k)}\right)^2$). An overestimated value results in subtracting more from $\mathbb{E}_{T_k}\left(Y^{(m_k)}\right)^2$ than it was actually added to it, w steps in the past. Fortunately, this issue may be circumvented.

The solution comprises updating $\mathbb{E}^{aux}_{T_k}(Y^{(m_k)})$ and $\mathbb{E}^{aux}_{T_k}\left(Y^{(m_k)}\right)^2$ using only the incoming block (expressions (3.13)), while the outgoing one is not ready to be removed (i.e. while the block that is leaving is not completely filled in). Once $Y^{(e_k)}_{aux}(E_k)$ becomes equal to $Y^{(m_k)}(E_k)$, the last four presented expressions guarantee that the estimator gets *right on track* again. This fact introduces temporary instabilities in the calculations, and

deviates for *brief moments* from the original definition of VT and from the windowed philosophy, since only *half* of the impact can be measured in a point-by-point manner. Nevertheless, as this is a temporary situation, it is tolerable and does not depend on the number of points available nor on the observation window size.

Since for a fixed and finite number of blocks, the number of variables needed for estimation of the Hurst parameter is also fixed and finite (it does not increase with the number of samples to be observed nor with the observation window size) and all the variables can be updated using recursive formulas (that depend only on their previous values and on the current entering/exiting values), the VT method can be used as a windowed Hurst parameter estimator. The reasoning demonstrates that the foundations of the estimator are not altered by the remarks in this section. Some possible errors can occur during transitory stages, but their reach is limited. The *sanity* of the method is maintained via the usage of an artifice that avoids trivial erroneous states, while keeping the overall precision of the estimator intact.

Interestingly, Hagiwara, Doi, Tode and Ikeda [97] have probably designed one of the first drafts of a true windowed estimator, based precisely on VT. In [97], they describe how limiting the amount of inputs of VT, and how calculating the several variances in *concurrent* processors, would result in a *high-speed* estimator. However, such implementation is still retrospective, as the estimator has still to process all the values of the specified time frame. They focused on the gain obtained by processing less data, and not on how the variances could be updated using recursive formulas (as it is done herein). As it is going to be further elaborated below, the efficiency of such estimator depends on the size of the considered time frame. The estimator may be practical for small observation windows, but may present degraded performance when analysing larger sets of data.

3.3.3. Windowed Modified Absolute Moments Time

After the discussion on how to build WMVT, the adaptation of AMT to a windowed estimator is straightforward, and does not require too many considerations. In section 3.2.3., it was demonstrated that, for even values of n, the order n absolute moment of $\{Y(t)\}_{t\in\mathbb{N}}$ (or of its aggregated processes) can be written in terms of expected

values of the powers of the said process. That, and the description made for WMVT, are enough to construct the windowed version of this estimator, termed herein Windowed Modified Absolute Moments Time (WMAMT).

In summary, WMAMT makes use of the previously defined variables ($Y_{aux}^{(r_k)}(T_k)$ and $Y_{aux}^{(e_k)}(E_k)$), and of the recursive formulas in sections 3.2.2. and 3.3.2., to estimate the expected values of $\{Y(t)\}_{t\in\mathbb{N}}, ..., \{(Y(t))^n\}_{t\in\mathbb{N}}$ and $\{Y^{(m_k)}(i)\}_{i\in\mathbb{N}}, ..., \{\left(Y^{(m_k)}(i)\right)^n\}_{i\in\mathbb{N}}$ in a *point-by-point* manner. Once the blocks leaving the observation window $Y_{aux}^{(m_k)}(E_k)$ are duly constructed, their effect in the history of the calculations is definitively erased.

Analogously to what happens with WMVT, the algorithm described in the previous paragraph does not follow the windowed philosophy precisely, because the effect of the leaving point is not erased from the calculations while the *longest* aggregation block is not completely filled in. In the case of AMT, that problem could only be circumvented recurring to a retrospective technique, which would make its real-time application impractical. Despite that, this small deviation from the initial philosophy is not that critical, and an estimator defined like so can still fulfil the two main requirements it was proposed to: (i) returning step-by-step estimations of the Hurst parameter, and (ii) eliminate the impact of the points outside the observation window.

As said at the end of section 3.2.3., MAMT and WMAMT were not implemented for orders higher than 2 (MVT and WMVT are the order 2 implementation of MAMT and of WMAMT, respectively). The implementation of MVT, WMVT, MEBP and WMEBP presume the codification of a procedure for linear regression. Its point-by-point or step-by-step application does not impose a prohibitive burden to the overall efficiency of the methods, for its order of complexity is of $O(K)$ per point processed, where K is the maximum number of aggregation scales (in the case of MVT, WMVT) or crossing levels (in the case of MEBP and WMEBP) taken into consideration (K is typically a finite value defined by the practitioner).

3.4. Critical Analysis and Comparison

In the previous sections, the set of modifications that enable EBP and VT to operate in a point-by-point and windowed manner were presented and briefly discussed. In this section, the referred estimation procedures are going to be analysed from a more practical perspective, with focus on the differences between the several methods and calculation philosophies, and on their computational requirements.

3.4.1. Computational Complexity and Memory Requirements

When used in a retrospective manner, most of the estimators present a computational complexity of (at least) order $O(n^2 \times K)$, where n is the number of points to be analysed and K is the number of aggregation scales (or equivalent concept) considered. At the expense of a *momentary* loss of precision, the modified estimators presented herein reduce that computational complexity to $O(n \times K)$, for both windowed and point-by-point estimators.

Figure 3.4 depicts the results of an experiment to quantify the performance of the WMVT method. To provide the analysis with a baseline comparison, VT was implemented in its retrospective form and submitted to the same tests WMVT was submitted to. The values in the chart were obtained by measuring the time that the two different implementations took to assess the Hurst parameter of a data series with 10^5 points. The values were stabilised by repeating this procedure 100 times and by taking the average of all of the occurrences. The variable for which the impact was to be studied was the length of the observation window. As such, this variable was initialised with the value of 1024 and incremented 98 times, with steps of 512 units. The simulations were performed in a non dedicated machine with the following specifications: Pentium IV 2.8 GHz processor, Windows XP Operating System (OS) with Service Pack 2 and 504 MB of Random Access Memory (RAM). During the simulations, no other applications were running though.

As can be concluded from the observation of the chart in Figure 3.4, the performance of the modified method is superior to the retrospective one, and it does not depend on the size of the observation window. While the retrospective method has to deal with the

increasing amount of values in the context window, the windowed estimator only has to process the incoming and the outgoing values.

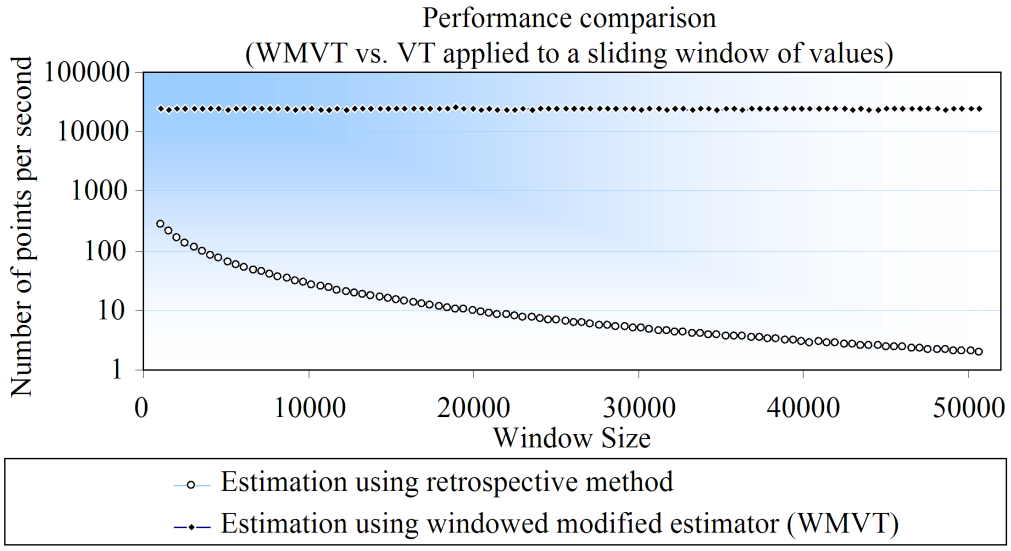

Figure 3.4: Performance of two different implementations of a windowed Hurst parameter estimator. The values in the chart concern the number of points that WMVT and VT (windowed but retrospective) can process per second. The y-axis is in logarithmic scale.

The modified estimators are also very efficient in terms of memory requirements. They do not presume that the entire series of values to be analysed has to be stored in the memory, or even known, prior to the analysis, conversely to their retrospective alike. As the data series do not need to be processed several times for a single estimate, there is no need to keep the entire series in a permanent or volatile block of memory. To function properly, the modified estimators require only the *auxiliary variables* (different for each method) to be stored in the memory. Just to give an idea of the values involved and taking as granted the values in [98], the actual Java implementation of MVT would not take more than 900 B of memory to process an arbitrarily long series of points, if 20 levels of aggregation are taken into account. These savings in terms of storage resources result from the compromise made with the fact that the length of the series is not known *a priori*, and may not necessary pose an advantage of the method. For example, as the entire sequence is not available prior to its analysis, it may not be preprocessed to reduce the effects of trends, periodicity or lacks of stationarity.

Windowed estimators are not as modest as the point-by-point ones, because their normal functioning presumes that the values inside the observation window are kept in the

memory. Nevertheless, the space occupied by a windowed estimator is not comparable to the total amount of memory required by a (global context Hurst parameter) retrospective estimator. Their objective is not comparable either, meaning that a local context Hurst parameter estimator will always require the storage of the occurrences inside the sliding window, independently of being a modified (step-by-step) or a retrospective estimator.

Nonetheless, this critical analysis must be understood within the context in which it was performed, specially the part that concerns the memory requirements. It is true that the point-by-point or windowed estimators do not require the entire series to be stored in the memory but, in practice, such only happens when the data series is being generated on-the-fly by some external event. The network environment is a perfect example of where such situation may occur. Based on the current specifications of network devices, the author would say that all the modified methods described in this section could more easily be integrated in the referred devices than the ones described in the literature. If such need arises, the rationale applied to these estimators is the one that leads to higher gains with respect to resources optimization.

3.4.2. Data (In)sufficiency

Setting an estimator requires one to know the basics of the theory on which they draw on. As the several techniques are often based on the aggregated processes, and as all of them try to quantify the relation between several occurrences of the same series, which may be separated by *large periods of time*, the Hurst parameter estimate results necessarily from the analysis of a significant amount of samples. It does not make sense to search for the scaling properties in manifestly small amounts of data.

When dealing with estimators based on aggregation, one has also to take into account that the number of samples decreases each time the series are aggregated. For example, if the sequence of values is being aggregated for scales of type $m_k = 2^k$, the number of samples decreases to half each type they are multiplexed according to the definition of $\{Y^{(m_k)}(i)\}_{i\in\mathbb{N}}$. This may lead to situations where the data is insufficient to construct a single block, or to accurately assess its statistical properties.

Dealing with the *data (in)sufficiency* problem requires one to meditate about the data set that is about to be analysed. If the length of the data series is known, the biggest aggregation block size should be chosen in such a way that the number of blocks (for that scale) is considered to be sufficient to estimate the statistics the method is based on. For example, if it is known that the data series has 2^{16} samples, the biggest block size can be set to 2^{10}, as it still gives origin to 2^6 data samples, which in most of the cases, are *sufficient* to get reasonably good estimates of their statistics. If the length is unknown, a maximum aggregation block (or crossing level, in the case of MEBP or WMEBP) should be set, but the statistics taken into account during the final phase of the Hurst parameter estimation should exhibit statistical significance, meaning that they should only be taken into account after a given number of samples has been reached (e.g. after the aggregated process has met 32 incidences).

The subject of data insufficiency is of particular relevance for windowed estimators. The very selection of the size of the observation window defines immediately the maximum amount of data samples the estimator is going to handle. Because of that, the referred value should also influence the maximum block size, but no straight recommendation can be done, though. If an unusual level of precision is required, one may ask the estimator to take into account several aggregation scales, and only then choose the ones that fulfil a given fitness criterion (e.g. the standard deviation of the target statistics to be less than a predefined value). Herein, the estimators were always set to have, at least, 32 samples in the highest aggregation scale, since during the empirical tests, the estimated values seemed to converge just fine.

3.4.3. Comparison Between Modified and Retrospective Hurst Parameter Estimators

The implementation of an Hurst parameter estimator in a point-by-point compliant manner constitutes a change in the way self-similarity assessment is performed. Instead of calculating a single Hurst parameter value for an entire data series, a point-by-point estimator returns a value for each sample, opening the possibility to follow the evolution of the self-similar properties of a given data series throughout an *evolution curve*.

Figure 3.5 and Figure 3.6 depict two evolution curves obtained through the usage of the point-by-point estimators developed along this research work. The first concerns the analysis conducted over a self-similar sequence with a known Hurst parameter of 0.8 using MEBP, and the second concerns the same type of analysis, but using MVT. The charts are obtained by plotting the Hurst parameter estimates against the index for which they were estimated. Even though the expected value for the estimations was the same for the two charts, these analysis were conducted for different instantiations of the generator described in section 4.3.. As it will be better explained in chapter 4, the referred generator was explicitly constructed to test these estimators. The total amount of points in both charts is 2048, since that length serves the purpose of this explanation well.

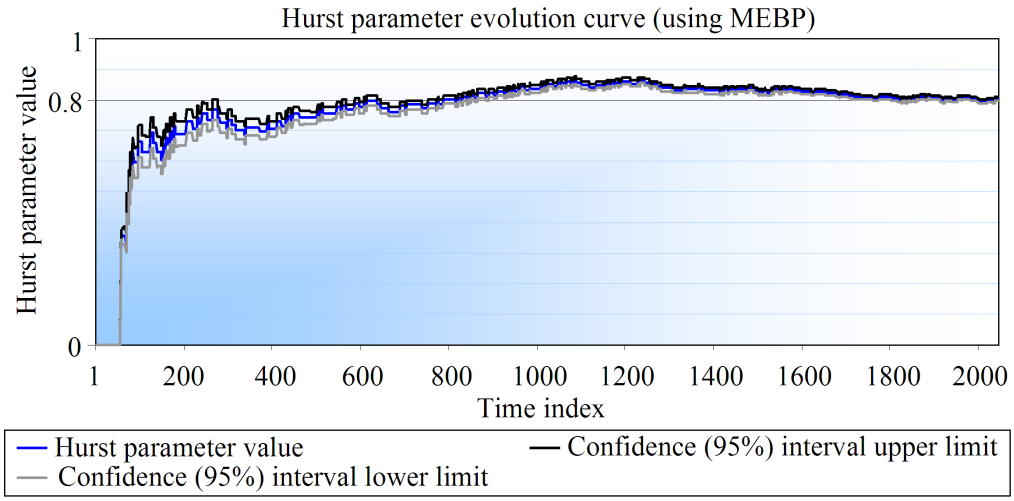

Figure 3.5: The evolution curve of the Hurst parameter estimated using the point-by-point version of EBP. This curve was obtained for a 2048 points long sequence with an expected self-similarity degree of 0.80, generated using the algorithm described in section 4.3.. The last value returned by MEBP was approximately 0.80.

As can be seen, the curves representing the Hurst parameter evolution converge quickly to the predefined value, after an initial period of instability. This initial instability is simply due to data insufficiency. The convergence to the expected value of the Hurst parameter provides for a visual demonstration that the modifications did not prejudice the precision of the estimators.

While this comparison favours the approach taken herein, clearly demonstrating that an histogram of the self-similarity degree can be made, it also proves that such perspective is still not suitable for the purposes of this thesis. Unfortunately, the estimators *abandon*

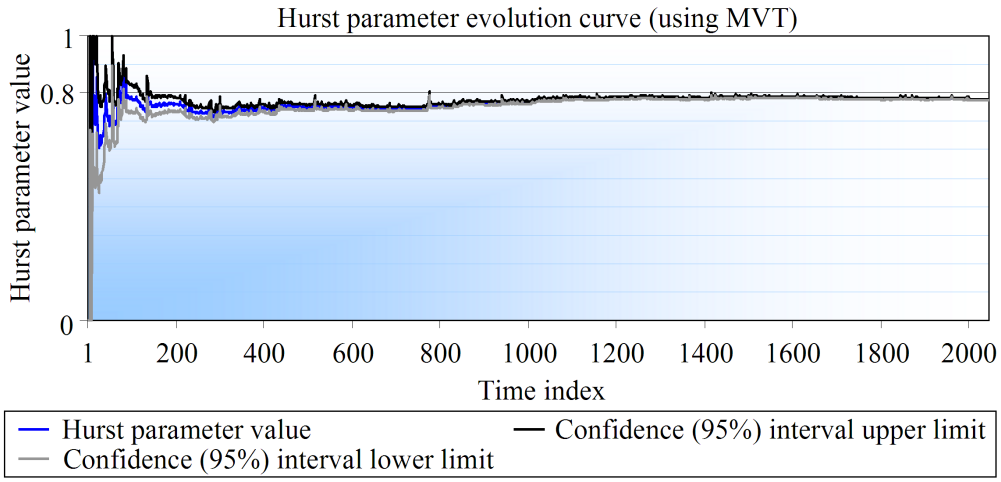

Figure 3.6: The evolution curve of the Hurst parameter estimated using MVT. This curve reflects the analysis of a sequence with 2048 values and an expected self-similarity degree of 0.80, generated using the algorithm described in section 4.3.. The last value returned by MVT was approximately 0.78.

their sensitivity after a considerable number of samples has been analysed. The *weight* of the history of the empirical series makes the variables of MEBP and MVT tend to constant values, and prevents them from reacting to ephemeral changes in the input series. This is also visible in the figures, since the confidence interval of the estimated values decreases when the time index increases. In the middle, however, it is possible to observe some interesting fluctuations of the values, unfolding the transitory states through which the self-similar process passes by. The analysis of the same data series using a retrospective method could only have one of two possible results: (i) the histogram would be constituted by a straight line with equation $y = 0.8$ (normally, the retrospective techniques are used to obtain a single value for the whole process), or (ii) the incremental perspective would be produced in a computationally costly manner, which would rapidly render the method useless, as the procedure would be recurrently and entirely processing an increasing set of values.

To test the implementations of MEBP and MVT (these procedures were implemented in the object oriented programming language Java), a self-similar sequences generator (which is known to be exact for the aggregation scales of type $m_k = 2^k$) was developed and put into action. MVT was set to explore the self-similar properties of the input series by taking into account the scales of type $m_k = 2^k$, starting at 1. MEBP was

Table 3.1: Statistical compilation of the precision tests made to MEBP and to MVT.

Hurst parameter	MEBP average(std. dev.)	MVT average(std. dev.)
0.50	0.52(9.49E-03)	0.50(3.98E-03)
0.55	0.56(1.14E-02)	0.55(3.74E-03)
0.60	0.61(1.10E-02)	0.60(4.27E-03)
0.65	0.65(1.35E-02)	0.65(4.22E-03)
0.70	0.70(1.83E-02)	0.70(5.22E-03)
0.75	0.75(2.18E-02)	0.75(5.83E-03)
0.80	0.80(2.86E-02)	0.80(6.40E-03)
0.85	0.85(3.24E-02)	0.85(7.88E-03)
0.90	0.90(4.10E-02)	0.89(9.20E-03)
0.95	0.94(4.18E-02)	0.94(8.41E-03)

initialised with the parameter δ equal to 2 and no normalisation was applied to the series, as they were already produced as a Gaussian variate with mean equal to 0 and variance equal to 1. The simulation was performed for a total amount of 1000 self-similar data series, which were the input of the two referred estimators. Table 3.1 summarises the results obtained by presenting the average and standard deviation (std. dev.) of the *last* incremental estimate of the Hurst parameter returned by the two methods. (Notice that all the values were rounded off to the hundredth, and that each data series was 131072 points long.)

With basis on the results, it is safe to conclude that the estimators were up to the expectation for the type of sequences submitted to analysis. Apart from a slight bias for the values of the Hurst parameter 0.5 and 0.60, demonstrated by MEBP, and for the value of 0.90, demonstrated by MVT (which are typical of these estimators), the values coincide perfectly. The difference between these values and the ones shown in Table 4.1 and in Table 4.2, namely for the ones concerning EBP, are a reflex of doing normalisation in an incremental manner (on chapter 4, the estimators are used in their retrospective form, and do not perform normalisation).

3.4.4. Comparison Between Locally and Globally Estimated Hurst Parameter Values

In the previous sections, it was emphasised that the perspective of the evolution of a global estimate of the Hurst parameter was not sufficient to detect small fluctuations in the self-similarity degree. This happens because the effect of small variations gets *statistically diluted* in the presence of large sample pools. The major difference between local and global scope statistical measures lies in the different amount of samples they represent. A windowed analysis is normally conducted over constant sized windows, reflecting only the characteristics of a subset of the population, while the global scope measure represents the entire data series. Because of that, the latter contain less information about each individual.

As the observation window is typically smaller than the sample size, the local scope values are often calculated for different windows until the entire population is covered by the analysis. The details of that analysis (length of window, step size) are normally dependent of the purposes of the investigation. In [95] for example, Stoev *et al.* have brought the focus into the importance of observing the local long-range dependence structure of a given data trace, for different moments in time, through the calculation of the Hurst parameter for N/w *non intersecting* blocks, where N denotes the sample size and w is the size of the observation window. Because they were using retrospective means to assess the said statistic, the graphical representation was formed by N/w estimations, which were not sufficient to construct a *dense* evolution curve, but were enough to prove that the trace passes through differently states of self-similarity.

In order to obtain an idea of what the tools described in this chapter are able to do, the generator in section 4.3. was instantiated for several values of the Hurst parameter, and the resulting data series were directly fed to the estimators in a point-by-point manner. The estimated values were plotted, as before, against their index, in *Hurst parameter vs. time* plots. From the set of charts produced (approximately 50), two were randomly selected and included in this section as figures.

Figure 3.7 and Figure 3.8 depict the difference between the locally and the globally estimated Hurst parameter. As expected, the limited context values oscillate around

the global (incremental) value (blue line), indicating clearly the transitory states of self-similarity that were not visible by mere point-by-point analysis, and that, hopefully, will allow for better examination of changes on the self-similarity degree.

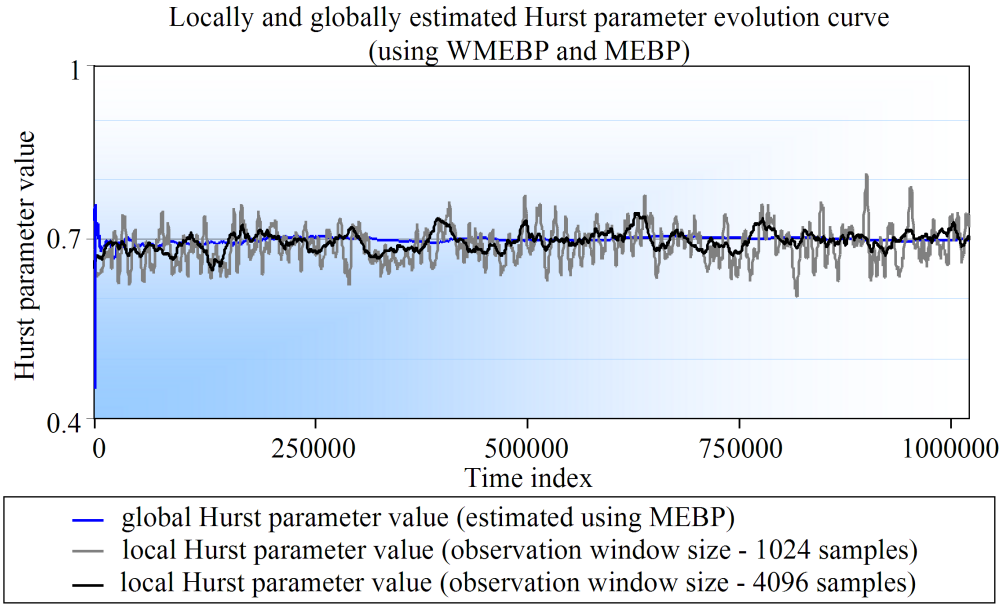

Figure 3.7: Local and global context Hurst parameter evolution curves. The local scope Hurst parameter values where calculated using WMEBP for two different observation windows: 1024 (gray line) and 4096 (black line). The global scope (blue line) Hurst parameter value was calculated by MEBP.

The relation between window size and sensitivity is also emphasised in the charts, since the estimations concerning smaller observation windows (1024 samples - grey lines), fluctuate more than the ones with a larger scope (4094 samples - black lines). It should be noticed that the size of the smallest observation window is manifestly small and that, for that amount of samples, one can only rely on the relatively short relations between the points. Their values were nevertheless included, to show that the estimators are capable to accurately capture self-similarity using a small number of aggregation scales. However, this type of analysis should be considered with caution before being applied. In this particular case, the input process was known to possess self-similar properties, but that may not be always true in other situations.

For a window of 1024 samples, WMVT was using only 6 aggregation scales ($m_k = 2^k$, with $k = 0, ..., 5$), while for 4096 points, two additional scales were considered ($k = 6, 7$). WMEBP was initialised with $\delta = 1$ (the standard deviation of the absolute values of

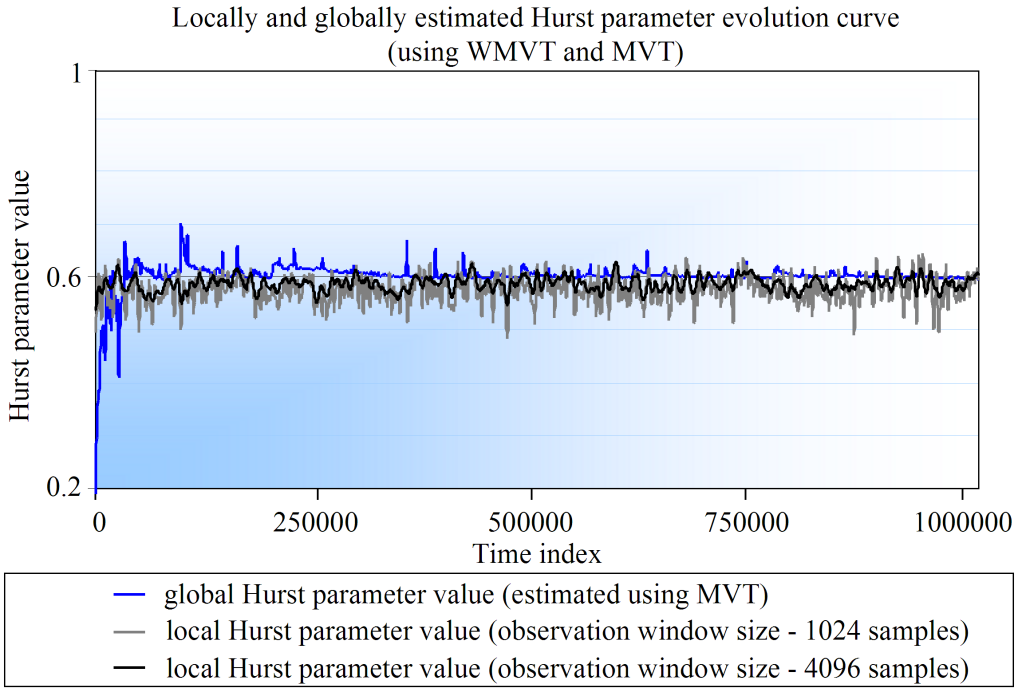

Figure 3.8: Local and global context Hurst parameter evolution curves. In this chart, the Hurst parameter values calculated using MVT (blue line) were plotted along with the values calculated by WMVT. The windowed estimator was instantiated twice for different observation window sizes: 1024 (grey line) and 4096 (black line).

the first differences process was $\sqrt{2/\pi}$) and the number of levels it was using depends strictly of the space span the process was *occupying* inside the observation window. For this particular case ($H = 0.7$, and $w = 1024$ or $w = 4096$), 4 or 5 levels were being used, depending on the position of the window.

To assess the precision of this version of the estimators, the experiment described in section 3.4.3. was repeated under almost the same conditions as before, but the statistics were differently calculated. WMVT was set to aggregate until the scale of $m_6 = 2^6$, starting at 1, WMEBP was initialised with δ equal to 2, and the size of the observation window was adjusted to $2^{13} = 8192$ for both of them. As the author was also interested in evaluating the fluctuations of the estimates along the evolution curve, the average and standard deviation were taken from *all* the Hurst parameter values returned by the methods. This means that the values in the second and third column of Table 3.2 are the average and the standard deviation (in brackets) of a total amount of 13107200 values.

This time, the results do not seem as good as for the previous experiment. However,

Table 3.2: Statistical compilation of the (*precision*) tests made to WMEBP and to WMVT.

Hurst parameter	WMEBP average(std. dev.)	WMVT average(std. dev.)
0.50	0.54(7.99E-02)	0.50(1.58E-02)
0.55	0.57(7.57E-02)	0.55(1.56E-02)
0.60	0.61(7.41E-02)	0.59(1.62E-02)
0.65	0.66(7.30E-02)	0.64(1.72E-02)
0.70	0.70(7.06E-02)	0.69(1.79E-02)
0.75	0.75(7.20E-02)	0.74(1.94E-02)
0.80	0.79(7.46E-02)	0.78(2.12E-02)
0.85	0.84(7.49E-02)	0.83(2.37E-02)
0.90	0.88(7.80E-02)	0.87(2.66E-02)
0.95	0.92(7.89E-02)	0.91(3.34E-02)

this particular behaviour was expected and can be better understood if some remarks are made on how the long-range dependent sequences generator works. As it will be explained in more detail in chapter 4, the referred generator is not particularly good at assuring the self-similarity degree for all aggregation scales, namely for the ones that are not powers of 2. As the windowed estimators *slide* through the sequence of values, they capture moments were the sequence is *less* self-similar than in others. As the Hurst parameter increases, that tendency is aggravated by a constant *willingness* of the process to *get back to an uncorrelated state*, to the state of *less energy*. This impacts both the average and the fluctuation of the estimates, as can be seen in the table. Under these terms, the author would say that the estimators were actually performing really good, as they captured the described phenomenon. (Please notice that, because of what was said in this last paragraph, the word *precision* was left between brackets in the caption of the table.)

A small remark can be made at this point. As can be concluded from careful observation of Table 3.1 and Table 3.2, EBP seems to suffer from a small bias when the Hurst parameter is close to 0.5. This fact is related with the *spatial expansion* of the fBm process, which decreases (statistically) with the value of the Hurst parameter. Unfortunately, this has a direct impact in the number of *crossings* in which EBP relies

on. The accuracy of the estimator can be improved by increasing the value of δ, and the length of the self-similar series (because the estimator is asymptotic). Unfortunately, the parameter that was last mentioned is not that often controllable.

3.4.5. Comparison Between Windowed-Modified and Windowed-Retrospective Estimation

The last comparison that has yet to be made concerns the potential differences between estimates from modified estimators and the ones from retrospective estimators. The reasons for these *potential* deviations to occur are related with the adaptations the methods have suffered, and with some of the assumptions that had to be made to make them compliant with the step-by-step calculation philosophy. To quantify this particular *issue*, a retrospective version of the windowed estimators was constructed, and applied alongside with the latter ones.

The four estimators (VT applied to the observation window, WMVT, EBP applied to the observation window and WMEBP) were fed with the data series generated by the self-similar sequences generator, and the output values were plotted against the input index, as in Figure 3.9 and Figure 3.10. To get the *exact* values of the Hurst parameter inside the observation window, the retrospective estimators (EBP and VT) were re-initialised and ran for the specific sample pool, for every shift of the window, resulting in a rather slow simulation procedure. Finally, for each $t \geq 0$, the absolute value of the difference between the two estimates (absolute error between the outputs of the windowed-modified and the retrospective method applied to the observation window) was taken and statistically compiled.

The charts in Figure 3.9 and Figure 3.10 were chosen to represent the analysis described in this section. The values in Figure 3.9 refer to the estimation of the local scope self-similar properties of a series with expected Hurst parameter equal to 0.7, obtained during one of the experiments with the estimators based on the EBP method; the values in Figure 3.10 were retrieved by the estimators based on the VT method, for a self-similar process with an expected Hurst exponent of 0.6. The size of the observation window was set to 2048 for the two simulated scenarios. For convenience, the values of the absolute

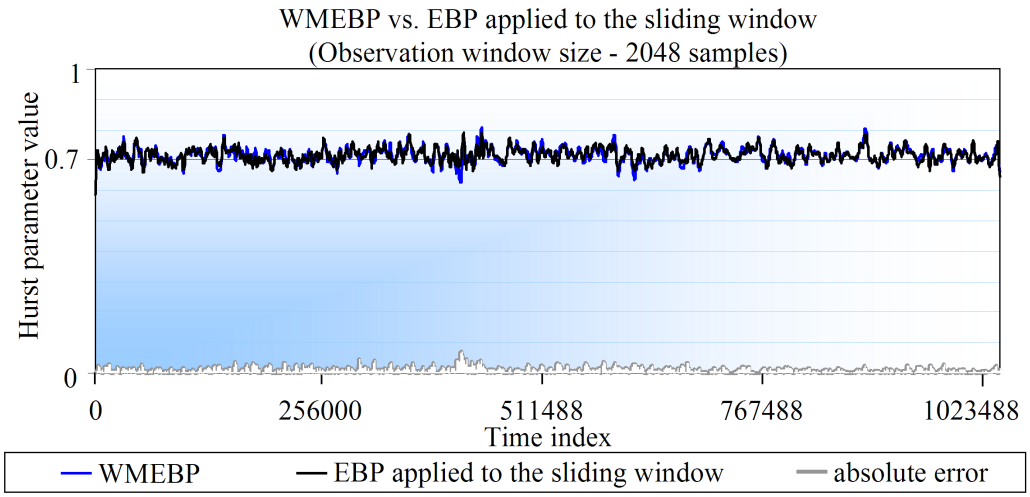

Figure 3.9: Comparison between different implementations of a windowed estimator based on EBP. The chart plots three different time series: (i) the values returned by the windowed-modified estimator (WMEBP), as a blue line; (ii) the values returned by the retrospective estimator (EBP) applied to the observation window, as a black line; and (iii), the absolute differences between the values outputted by the windowed-modified and the retrospective estimator, in grey.

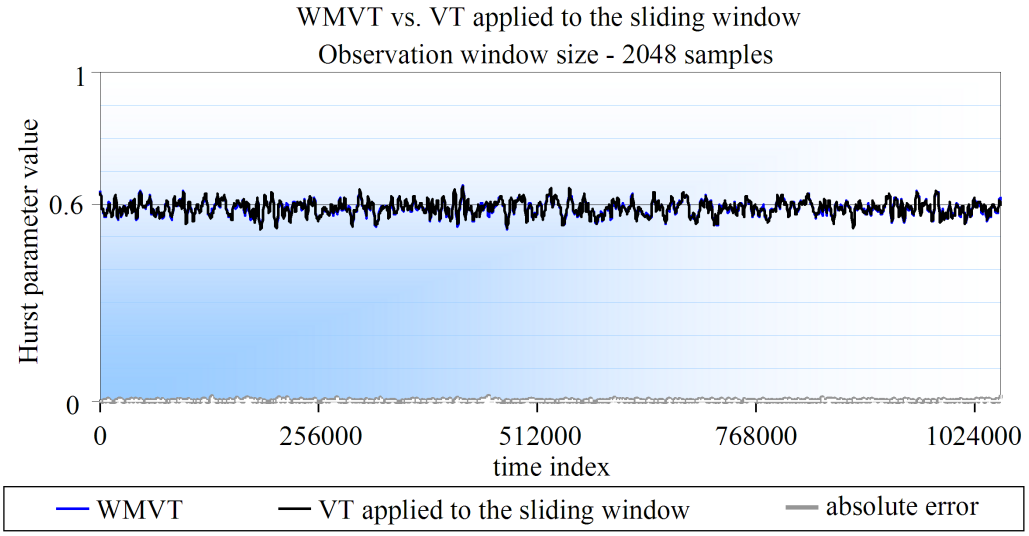

Figure 3.10: Comparison between different implementations of a windowed estimator based on VT. The chart plots three different time series: (i) the values returned by the windowed-modified estimator (WMVT), as a blue line; (ii) the values returned by the retrospective estimator (VT) applied to the observation window, as a black line; and (iii), the absolute differences between the values outputted by the windowed-modified and the retrospective estimator, in grey.

error for each time index, and each pair of estimators, were also represented in the bottom of the charts (grey lines).

The low relevance that the lines representing the absolute errors achieve in both charts constitutes a sign of how small the error introduced by the assumptions taken herein is. That, and the harmony between the two evolution curves of each chart (along the entire time span for which they are plotted), corroborates empirically that the error is limited and does not increase with the number of data samples.

The validity of these conclusions was confirmed using computer-based simulation, for which the results were statistically compiled in Table 3.3. The table contains the average and standard deviation (in brackets) of the absolute error between estimates, calculated for each pair of estimators (EBP and WMEBP, VT and WMVT), as previously described. Both statistics were taken from a total amount of 13107200 values, for each Hurst parameter value in the first column of the table, and each pair of estimators. The results are clear, and prove that the windowed-modified estimators are as reliable as the retrospective ones.

Table 3.3: The average and the standard deviation (in brackets) of the absolute error between the windowed-modified and the retrospective estimators.

Hurst parameter	EBP vs. WMEBP average(std. dev.)	VT vs. WMVT average(std. dev.)
0.50	1.78E-02(8.76E-02)	2.17E-04(1.16E-03)
0.55	1.79E-02(8.43E-02)	2.23E-04(1.18E-03)
0.60	1.74E-02(7.91E-02)	2.38E-04(1.23E-03)
0.65	1.57E-02(7.06E-02)	2.53E-04(1.30E-03)
0.70	1.42E-02(6.25E-02)	2.79E-04(1.41E-03)
0.75	1.28E-02(5.51E-02)	2.84E-04(1.43E-03)
0.80	1.08E-02(4.63E-02)	3.30E-04(1.65E-03)
0.85	8.54E-03(3.75E-02)	3.12E-04(1.69E-03)
0.90	6.13E-03(2.86E-02)	4.36E-04(2.30E-03)
0.95	3.76E-03(2.03E-02)	5.77E-04(3.54E-03)

3.5. Conclusion

In chapter 2, it is suggested that some of the estimators of the Hurst parameter may potentially be implemented in a real-time compliant manner. Up until now, most studies use these mathematical tools to assess the Hurst parameter of the entire data series, not leaving much room for the investigation of the local characteristics of the self-similar sequence. In this chapter, the three previously identified candidates (EBP, VT and AMT) were formally introduced to the set of modifications that enable them to produce results in a point-by-point manner (instead of returning a single value for the whole data series), with low computational requirements.

After having explained why the point-by-point estimators are not suitable enough to fulfil the objectives of this thesis, they were further modified so as to follow a sliding window philosophy, that enables them to dispose of the heavy weight of the history of the analysed series, improving their overall sensitivity to local changes. What is important to retain is that the windowed estimators provide one with the tools to construct a graphical perspective of the evolution of the self-similarity degree of a given data series and that, by adjusting the size of the window, one can control how much attention the estimator gives to small details (the smaller the observation window is, the more sensitive and unreliable the estimations are).

Two of the three modified estimators were chosen to be implemented and used for the sake of the objectives of this thesis. It was decided not to implement MAMT and WMAMT for orders higher than 2, because they aim to capture self-similarity the same way MVT and WMVT do, while presenting an higher degree of complexity than the latter (for $n > 2$). From this moment on, only MVT, WMVT, MEBP and WMEBP will be considered. The proposed point-by-point and windowed estimators may be seen as turnaround solutions to the real-time implementation of the Wavelets estimator [92], for they fulfil the same purposes through different means.

It was stressed out that, in spite of the possibility of using EBP as a point-by-point or windowed estimator, their applicability is constrained by the two following factors: (i) the series of values fed to the estimator has to be normalised prior to the analysis *per se*, and (ii) its accuracy depends noticeably of the number of points analysed and of the

Hurst parameter itself. These are the two drawbacks associated with the online utilisation of this procedure. If the input series are known to be normalized *a priori*, the modified versions would produce the exact same results of the original one, since the change in the calculation philosophy does not redefine the variables in which this method is based on.

Because the variance does not depend of the average of the empirical series, WMVT or MVT are presented as being a more *robust* choice than WMEBP or MEBP, as they are not susceptible to the normalisation procedure. Actually, VT does not require the data series to be processed prior to the analysis, in cases where it is not normalised. Because windowed estimators react better to non-stationarity in the input sequence, WMVT comprises a particularly interesting choice for real-time analysis. When sliding through the sequence (and depending on the size of the observation window) the operational model of the windowed estimator enables the latter to adapt the values on which it is based on to the ones of the local scope statistics during runtime.

The experiments conducted for implementations of MVT, MEBP, WMVT and WMEBP corroborate the statements concerning the precision of the algorithms made along the chapter. Though some of the modifications inflicted to the original methods imply smaller deviations to their true way of functioning, these deviations are not reflected in the estimates in the long-term, or they are limited. It was also said that, when dealing with windowed estimators, it is necessary to choose the size of the observation window, and the number and type of precision levels or aggregation scales, in such a way that a suitable number of samples is available for the calculation of the statistics on which they rely on. Herein, a minimum 32 samples are considered to be enough for the calculation of the variance and number of crossing levels.

Chapter 4

Efficient Generation of Self-Similar Sequences

4.1. Introduction

The previous chapter was focused on the means to assess the self-similarity degree of a given time series, which provenance was irrelevant or, at least, not asked. This chapter focuses on the reverse of the medal, and tries to answer the question of how to forge realisations of self-similar processes with predefined Hurst parameter values. The motivation to switch the perspective lies on two main reasons: first, the modified estimators presented in chapter 3 had to be tested for different degrees of self-similarity; and second, the construction of a self-similar traffic simulator presumes the availability of pseudo random sequences with the referred characteristic, or of the means to generate them.

The interest in methods for simulation of self-similar processes has increased over the last few years, specifically in the ones capable of generating long-range dependent sequences. This interest was enhanced not only by the discovery of the self-similar nature of the network traffic [17], but also by the importance that self-similarity analysis gained in many other areas of research, as biomedical sciences [99] or economics [43]. Actually, it seems that the self-similarity phenomenon is embedded in many natural [72, 73, 100, 101, 102, 103] or artificial processes [16, 17] but, surprisingly, their replication is rather complex, mainly due to the nature of their definition: an hyperbolically decaying autocorrelation function implies the existence of dependencies between *all* occurrences of a self-similar process. The generator of sequences with the self-similarity

structure requires the produced points to be orderly stored in the memory, and processed before further points are created. The procedure faces the dilemma of seeing its performance being degraded each time a point is created, or improved at the cost of losing accuracy.

Experiments involving self-similarity require often long sequences of values, being thus useful to handle them with computers. Because the referred property deals with dependencies, it is of utmost importance to assure that the bias of the data series is intended, and not caused by any other artefact. Apart from the application to the area of network traffic analysis and simulation, the reproduction of this type of stochastic processes in a controlled environment is proven to be useful for the simulation of weather conditions (for prediction purposes) [103], for the generation of landscapes (self-similarity has been noticed in country coast limits, in the line of the horizon or terrain stratification lines [102]), or for studying the evolution of stock market prices [43].

In this chapter, some of the currently available methods for simulation of self-similar processes, namely of fBms and fGns, are going to be briefly discussed, along with their main advantages and disadvantages. This analysis constitutes the baseline for the explanation of two new methods, designed to gather the preferred features of the already existing ones. The specific reasons that led to the proposal and development of each one of the algorithms is included near their explanation. The chapter does not end without the detailed description of a new procedure to create pseudo random sequences of numbers, which guarantees the quality of the output of the two self-similar sequences generators.

The following three sections of this chapter are mostly based on the papers [31, 32]. The fourth section elaborates on a realisation of the family of pseudo random number generators proposed in [33].

4.2. Overview to the State of the Art in Terms of fractional Brownian motion Generators

The main idea of this section is to briefly discuss some of the most important or popular methods for simulation of self-similar processes (some of the generators are only

capable of generating series exhibiting persistent behaviour), namely of fBms or fGns. The presentation is structured according to the classification of the methods, which can be *exact* or *approximate*. The description is not intended to be extensive, and it is mainly focused on the important features of each one of the methods.

4.2.1. Exact methods

The three following methods are used for the simulation of fGn. They are said to be *exact*, since it is theoretically assured that the generated series has the autocorrelation structure of a self-similar process, with a predefined Hurst parameter value. All the procedures reported in this section elaborate on the same mathematical structure: the covariance matrix of the process. The subsequent exposition is mostly based on [38, 44, 77] but, if pertinent, other references will be opportunely indicated.

The Hosking Method

In [104, 105], Hosking described the means to produce exact incidences of a general stationary Gaussian process, which is *asymptotically second order self-similar*. In [38], Dieker adapts the procedure in order to reproduce fGn, elaborating on the *covariance matrix* of this process. Besides being exact, this method allows on-demand generation of data series, since the number of points to be generated is not a variable of the algorithm. Each point is obtained recursively, as a function of *all* the past points of the process, and even though part of the calculations can be reduced by re-usage of some already computed values, the computation complexity of the method is still of $O(n^2)$, turning it a bad choice when it comes to real-time or long simulations. In order to turn the method applicable to non-stationary processes, a so-called *innovations algorithm* has been proposed as an extension to it.

The Hosking method is going to be used in one of the subsections below as a quality benchmark for some of the estimators used herein. For a more practical description on how this retrospective technique works, refer to [106]. The implementation of the procedure used in the scope of this research is the same Dieker made available in [107].

The Cholesky Method

The Cholesky method is as precise as the Hosking method and can be applied to generate non-stationary Gaussian processes. However, this method is slower than the aforementioned one, because the decomposition applied to the covariance matrix is different (normally referred to as *Cholesky decomposition*), and comparatively to the decomposition used in the Hosking method, it requires a larger number of intermediate operations to generate a single point of the process.

The Davies and Hart Method

The difference between the Davies and Hart method and the previous ones resides, again, in the means by which the covariance matrix is processed. This method states that, under certain restrictions, a *circulant matrix*, containing the covariance matrix, can be defined and factorised. As the factorization procedure can be carried out efficiently by means of the Fast Fourier Transform (FFT), the computational complexity of the method can be reduced to $O(n \log(n))$, rendering it as the fastest among the three exact methods.

The efficiency of the method can be further improved (doubled) if approximated values of the coefficients of the *circulant matrix* are used, instead of the exact ones. Because of this approximation, such method can no longer be classified as an exact method (but it tends to the exact motion, as the number of samples generated tends to infinity) and the expression *approximate circulant method* is normally used to refer it.

4.2.2. Approximate methods

The next presented methods can be used to produce sequences of values with properties that resemble the ones of fBm or fGn, though their outputs are not sampled realisations of these stochastic processes. The precision of the approximate methods is normally dependent on a criterion of convergence, and often counterbalances the computational complexity of the underlying algorithm.

The Method of Aggregation of Processes

The method of Aggregation of Processes is the one inspired in the operational model of LANs and on the result proven by Taqqu *et al.* [48], discussed in section 2.3.2. of this thesis. Recall that, according to that result, the aggregation and normalisation of renewal processes with power law distributions approximates an fGn. Each one of the computers connected to the LAN is seen as one of those *ON/OFF* sources. The aggregated process tends to the precise solution when the number of sources and the aggregation scale from which the processes are observed tend to infinity.

Because each one of the renewal processes has to be simulated separately, the computational complexity of this method is of $O(B \times m \times n)$, where B represents the number of sources and m is the aggregation scale. In this case, the precision of the generator is inversely proportional to its performance. Nevertheless, the method is used in many network traffic simulators (for example, it is used for the simulation of traffic in studies concerning Ethernet Passive Optical Networks [108]), mainly due to its conceptual nature, which is perfect to be implemented in an object oriented language. In the performance evaluation reported in [109], it has been shown that a generator based on the multiplexing of several heavy-tailed sources performs better than the ones based on chaotic maps, when more than 100 sources are considered. Chaotic maps have been proposed as alternative models for source traffic [110], since their aggregation may also *approximate* a self-similar sequence.

Random Midpoint Displacement Method

The Random Midpoint Displacement (RMD) method is used to produce approximate realisations of fBm in a recursive manner, by iteratively synthesising the points of higher aggregation scales before generating the points of the lower ones. Once all the points of an aggregation scale (sometimes referred to as *resolution*) are generated, the finer resolution ones can be created, while taking into account that the sum, mean and variance of their values must fulfil certain criterion (conditioning), dependent of the Hurst parameter and of the already generated *higher resolution* points.

The RMD method requires the time horizon to be known in advance and, therefore, it is not suitable for generation of arbitrarily long self-similar processes. Additionally, in [44], Paxson concluded that the method is biased, producing approximations with an estimated Hurst that is bigger than the expected one for $0.5 \leq H < 0.75$ and smaller than the latter for $0.75 < H < 1$. In [111], Norros *et al.* described the means that enable the method to partially overcome such limitation and produce higher quality outputs (in terms of long-range dependence). The new algorithm was named Conditionalized RMD. The idea behind this algorithm is to create the fBm in the same order the RMD does, but using more points in the conditioning (l points to the left and k points to the right). In the same reference, the authors also prove that the generation of the samples can be performed in a point-by-point manner, since with the new method it ceases to be critical to generate each resolution completely, before moving to a finer one. Nonetheless, there is a trade-off between memory requirements and the incremental usage of the algorithm, which they estimate to be approximately of $l \log_2(2n/\delta) + k \log_2(n/\delta)$ (where δ is the resolution). Its computational complexity is of $O(n)$, which classifies it as a good choice when it comes to simulations where the number of samples and required resolution are known parameters.

Stochastic Representation Method

Thanks to Mandelbrot and Van Ness [43], fBms can be analytically defined as a stochastic integral with respect to ordinary Brownian motion (see section 2.2.5.). The Stochastic Representation method is based on an approximation of that function through Riemann sums, since the stochastic integrals do not allow for deduction of an explicit generalised formula. The approximation is preceded by a truncation of one of the (infinite) integration limits, in order to turn it computationally acquiescent. Obviously, these two steps decrease the precision of the method, which can be improved by increasing the density of the grid of the Riemann sums, or by choosing other values for the truncation parameter. An increase of the precision of the method leads to an increase of its computational complexity, estimated as being of $O(n^2)$. As so, the stochastic representation method is not currently accepted as a good real-time generator of self-similar processes and, as emphasised in [77], it is only interesting from an historical point of view.

The Paxson Method

The idea behind spectral simulation is to generate a process in the frequency domain and to transpose it into the time domain. In the particular case, these processes are either fBm or fGn processes. The Paxson method [44] is a special case of spectral simulation were a set of coefficients describing the frequency information of a stationary discrete-time Gaussian process, are approximated and inversely transformed to the time domain. The importance of the FFT (or of the inverse FFT) is, therefore, easy to explain in spectral simulation methods. The spectral density of the stationary discrete-time Gaussian process, written in terms of sine and cosines, is supported by the so-called *spectral theorem*.

This method is similar to the Davies and Hart method, except that the first is approximate, instead of exact, and faster than the latter. Regardless of that, its computational complexity is still of $O(n \log(n))$, inherited from the inverse FFT procedure.

Wavelets-Based Method

The Wavelets-based generation procedure is again related with the generation of a process in a domain different than the time domain, whose transformation approximates an fGn. Only that this time, the conversion is performed by means of wavelets structures, which have the desirable property of retaining more time information for high frequencies than for low frequencies. The Wavelets method computational complexity is estimated as being of $O(n \log(n))$. Nevertheless, if the transformation is made accordingly to [112], which suggest using the inverse Discrete Wavelet Transform (DWT), the complexity can be reduced to $O(\psi \times n)$, where ψ is the length of the Wavelet *filter*.

Notice that a Wavelets-based generator was implemented in the Java programming language and used as a comparison baseline for one of the new methods presented in this chapter. This particular implementation makes use of the FFT to convert from the *Wavelets domain* to the *time domain* and, as such, its computational complexity is of $O(n \log(n))$.

An Accurate Fractional Brownian Motion Generator

Rambaldi and Pinazza [113] draw on one of the definitions of fBm to present a new approximate method. In the referred solution, the *problem* of having highly retrospective relations between the points of the process is overcome by defining a *composite contribution* of the most distant points of the past of the signal. This contribution is (iteratively) constructed in runtime, and used along with a restricted set of past values, to generate the next point of the motion. This procedure is thus based on a *truncation / approximation* mechanism, and presents a computational complexity of $O(n)$. However, according to [44], the assessment of the accuracy of the algorithm was made in a terse manner and can be a motive for discussion. By construction, the algorithm is not capable of accurately assuring the correlation structure between distant points.

Simple Construction of the Fractional Brownian Motion

The approach where the fBm is obtained from the application of the *Central Limit theorem* to a series of independent random walks may be found in [114]. In the referred work, Enriquez has focused on creating *correlated random walks*, elaborating on the concept of *persistence probability*. For self-similar processes with $0.5 \leq H < 1$, the *persistence probability* is directly applied to create positively correlated random walks. For self-similar processes with $0 < H < 0.5$, an alternative approach, designated by *alternated random walk with persistence* p, is introduced.

The computational complexity of the proposed algorithm is said to be of $O(n^o)$ (see section 5 of [114]), where o is a real number between 1 and 2. The convergence of the correlated walks to an fBm is only assured in the *presence* of an infinite number of walks, but that particular prerequisite does not pose a computational burden at the end of the proposal, since the author generalized the main idea behind the generation procedure to Gaussian variables also. Nonetheless, the quality of the self-similar structure created by these means depends of the aggregation factor applied to the (initially) short range dependent sequences, which counter-balances the proficiency of the solution. This happens because the *persistence probability* reflects the relation between adjacent points of the generated series only, and not longer dependencies.

As it will be emphasised afterwards, the work in [114] has some resemblances (and also some differences) with the generator presented in section 4.3.. As an example, it can be said that the main basic structures behind both algorithms are similar (correlated random walks), but while the proposal of Enriquez [114] is focused on the definition of short-range dependencies only, the one described afterwards is concerned with the deduction of a (specific) structure that takes some long-range dependencies into account also. Other resemblances and differences of the two methods are going to be opportunely indicated.

Simulation of Fractional Brownian Motion with Micropulses

The method for simulation of fBm using Micropulses was described by Caglar, in [115]. According to its proponent, this generator is somewhat similar to the one of Wavelets (only more general, since it enables the selection of different and smaller *waves* - hence Micropulses), and to the one of the *aggregation of processes*. The complexity of the method is said to be more precisely given by $O((n+2)\epsilon^{-2})$, where ϵ is an *accuracy parameter* in the interval $]0,1[$, which defines the quality of the approximation to a Gaussian variable (the smaller the value of ϵ is, the better the approximation is). The method reduces the weight of the computations by *truncating the past* of the signal to some extent, and by only assuring the relations inside a limited scope. The mathematical description in [115] seems accurate (as far as the author of this thesis could assess), but the empirical quality evaluation only uses the AV estimator to compute the Hurst parameter. As the Micropulses based generator has some resemblances with the one based on Wavelets, the said experiments could have been conducted using other estimators, for the sake of *impartiality*.

The author would like to cease the opportunity to mention that, even though the said work dates from the year 2000, and in spite of the several searches made on the topic during three years, this particular method to generate fBms was not known to him until the time the final draft of this thesis was being elaborated. This fact came in favour of the work described below, as it will be theoretically proven that the algorithms proposed herein are faster than this one, though with the same order of complexity.

Synthesis of Accurate Fractional Gaussian Noise by Filtering

In 2006, Ostry [116] proposed a method for the synthesis of fGn resorting to filtering. The algorithm is suitable for the generation of both persistent and anti-persistent sequences, and presents an order of complexity superior to $O(n)$, but inferior to $O(n\log(n))$ (the analysis conducted by the proponent of the method show that the computational complexity of the algorithm may be reduced to $O(n^{1.04})$, for a Hurst parameter value of 0.55, or to $O(n^{1.10})$, for a Hurst parameter value of 0.95, if a number of optimizations techniques is applied). The method is said to be exact for lags smaller than the selected filter length, and once the filter is duly constructed, it may be used for forging arbitrarily long sequences of values. Hence, the construction of the filter is of critical importance in the referred solution, because it defines the self-similar properties of the generated series. The dexterity of the underlying algorithm is somehow dependent of the one of the Fourier transform, since the filter is constructed in the spectral domain prior to be *excited* by Gaussian noise to produce fGn. The work shows that the research and development of new procedures for simulation of self-similar series is pertinent and actual, and that the motivation for that research lies on the necessity to create arbitrarily long sequences in an efficient manner.

Other Methods

Examples of methods that did not accomplish their purpose successfully can also be found in the literature. Strecker [62], for instance, introduced the so-called *fracture stretch algorithm* that, according to him, was capable of generating processes with fractal properties, but it was incapable of assuring that the synthesised series had a predefined Hurst parameter.

4.2.3. Desirable Features of Self-Similar Sequences Generators

From the previous description, it is possible to identify some of the features that the scientists were aiming to obtain with the proposal of the algorithms. First of all, the necessity to reduce the computational costs of these types of algorithms is flagrant. Most

of the researchers are willing to lose some of the precision of the procedure in favour of an algorithm that produces results faster. Some solutions are only designed for *long-range dependent* processes, but most of them tackle both *persistent* and *anti-persistent* series. Most of the generators are capable of returning points in an on-demand basis, being that left as an implementation detail, but others require the number of points to be generated to be known in advance. Unfortunately, the definition of self-similarity turns infeasible any tentative to construct an algorithm that fully respects all those prerequisites.

Even though it was not mentioned in the previous sections, the memory requirements of the generators is also of great importance, at least when the time to forge long data series comes. It is worth to be mentioned that it is usual to have a strict relation between computational complexity and storage capability. Algorithms of $O(n^2)$ usually require the entire past of a given series to be available in order to be able to generate the next point of the process. $O(n \log(n))$ algorithms normally take an amount of memory that is a multiple of $\log(n)$, where n is the size of the process to be generated, and $O(n)$ mechanisms often embody the most cheapest solution, demanding space to store only a fixed number of variables. In terms of memory, procedures requiring a small multiple of $\log(n)$ can be considered a moderately good choice. One of the objectives of this part of the research was to develop algorithms capable of producing long series of data without requiring the storage of large amounts of data nor of the generated values, so as to mimic the functioning of the estimators described in chapter 3.

4.3. Fast and Sequential Generation of Persistent Fractional Brownian Motion

After the overview of the state of the art, the author concluded that none of the currently available algorithms was fulfilling the prerequisites he was looking for. Some are too slow, others do not offer warranty of the self-similar properties, others yet are not capable of generating points on-demand, or in a real-time manner. Most of them are severely affected by the number of points to be generated, and have their performance diminished each time a point is requested. As one of the applications of such algorithms was the simulation of traffic traces at high data rates, which would potentially require

generation of data series with an unknown number of samples, he decided on the tentative to develop a new algorithm. From that endeavour resulted *two* new generators.

The author would like to admit that the inspiration behind the first algorithm he is about to present came originally from the need to test MVT because, at the beginning of this research work, the means to get several series of points with different Hurst parameter values were not that obvious. The doubt on how to make the variances of the aggregated processes to decrease according to the self-similarity rules was on the foundations of the procedure described and tested in this section, being the latter, one of the most concrete outcomes of the sedimentation of the mathematical concepts described in chapters 2 and 3.

4.3.1. Fractional Brownian Motion Sequential Generation Algorithm

The main purpose of this section is to introduce a new algorithm to approximate Gaussian processes with self-similar properties, namely fGns and, consequently, fBms. For that, a progressive explanation approach is going to be adopted: the section starts with the mathematical description of the herein called *secondary algorithm* to simulate random walks exhibiting persistent behaviour (i.e. with a predefined Hurst parameter between 0.5 and 1) and evolves to the description of the means that make use of those walks to approximate persistent fBms in an efficient manner (subsection 4.3.1.).

An Algorithm to Generate Persistent Random Walks

The processes for which the reasoning contained in this subsection applies to are not fGns yet, but correlated random walks with constant size steps (the definition of random walk can be found in section 2.2.3.). These processes will prove themselves fundamental for the reasoning in the following subsection because, as it is going to be pointed out afterwards, the sum and normalisation of several independent instances of such processes will result in an approximation of an fBm, as previously defined. This sentence is justified by the *central limit theorem* and by some of the properties of self-similar processes.

The values that the *secondary algorithm* directly outputs are realisations of the first

order differences process $\{S(t)\}_{t\in\mathbb{N}}$ of a random walk $\{R(t)\}_{t\in\mathbb{N}}$, being herein presumed that the former can only take the integer values 1 or -1 (see condition (4.2)). The most general definition of $\{S(t)\}_{t\in\mathbb{N}}$ may be expressed in terms of conditions (4.1) and (4.2), where (4.1) defines the starting point of the process:

$$S(0) = -1 \text{ or } S(0) = 1 \text{ with probability } 0.5; \tag{4.1}$$

$$\forall_{t\in\mathbb{N}}, S(t) = -1 \text{ or } S(t) = 1. \tag{4.2}$$

At any time moment t, the position of the walk denoted by $R(t)$ can be recovered by taking the sum of all the values of the first order differences process until that moment, as formalised by

$$R(t) = \sum_{i=0}^{t} S(i). \tag{4.3}$$

The correlations between points of the process will be given in the form of what has been called of *persistence probabilities*, in resemblance to what was done in [114]. Assume that a persistence probability exists for each aggregation scale of type $m_k = 2^k$, with $k = 1, 2, ...$, and that those probabilities influence the progression of the walk in the sense given by (4.4). Herein, the persistence probabilities are denoted by p_{m_k}. Condition (4.5) assures that a given point *of the future* can only be directly and positively affected by a single point *of the past*, while avoiding the trivial situation where the signal is constant (i.e. $p_{m_k} = 1$, $\forall k = 1, 2, ...$). If $p_{m_k} = 0.5$, $\forall k = 1, 2, ...$, the process is uncorrelated, and $\{R(t)\}_{t\in\mathbb{N}}$ assumes the denomination of Random Walk:

$$p_{m_k} = P\left(S\left(t + \frac{m_k}{2}\right) = S(t), t \in m_k\mathbb{N}\right), \text{ where } m_k = 2^k, k \in \mathbb{N} \text{ and} \tag{4.4}$$

$$0.5 \leq p_{m_k} < 1. \tag{4.5}$$

The algorithm will only be responsible for maintaining exact self-similarity properties for aggregation scales m which are powers of 2. For other scales, self-similarity cannot

be assured theoretically, but simulation results will demonstrate that the properties are preserved, at least approximately. Additionally, consider that the maximum number of aggregation scales the procedure is concerned with is represented by N_p, sometimes referred to as *the number of precision scales*. The expression *scales of type* $x \times y^{N_p}$ stands for all of the aggregation scales starting at x and ending at $x \times y^{N_p-1}$.

For the reasoning below, consider the second definition of self-similarity (equation (2.4)). From that definition, equation (4.6) can be derived:

$$\mathbb{V}(Y) = m^{2-2H}\mathbb{V}\left(Y^{(m)}\right). \tag{4.6}$$

This equation is the same defining the VT estimator, and it was transposed here for clarity reasons. Let the process $\{S(t)\}_{t\in\mathbb{N}}$ fulfil equation (4.6) for the aggregation scales m_k, as depicted by (4.7). Notice that $\{S^{(m_k)}(i)\}_{i\in\mathbb{N}}$ denotes the aggregated process of $\{S(t)\}_{t\in\mathbb{N}}$, for scale m_k:

$$\mathbb{V}(S) = m_k^{2-2H}\mathbb{V}\left(S^{(m_k)}\right). \tag{4.7}$$

Since (4.2), (4.4) and (4.5) hold, it is easy to conclude that $\mathbb{V}(S) = 1$ and that, therefore, ${m_k}^{2H-2} = \mathbb{V}\left(S^{(m_k)}\right)$. Under these circumstances, the first condition $\{S(t)\}_{t\in\mathbb{N}}$ must met is $2^{2H-2} = \mathbb{V}\left(S^{(2)}\right)$. Due to the same reasons, it can also be said that the expected value of $\{S(t)\}_{t\in\mathbb{N}}$ tends to 0, as the number of points to be generated tends to infinity (i.e., in the long-term, the number of incidences of $S(t) = 1$ equals the one of $S(t) = -1$, though this might not necessarily hold true in the short-term). The following equalities are thus valid:

$$2^{2H-2} = \mathbb{V}\left(S^{(2)}\right) = \mathbb{E}\left(S^{(2)}\right)^2 - (\mathbb{E}(S))^2 = \mathbb{E}\left(S^{(2)}\right)^2. \tag{4.8}$$

According to the definition of the *persistence probabilities* (4.4), p_2 is the probability of $S(t+1)$ being equal to its predecessor $(S(t))$, for realisations of $\{S(t)\}_{t\in\mathbb{N}}$ where the

time index t is a multiple of 2:

$$p_2 = P(S(t) = S(t+1), t \in 2\mathbb{N}). \tag{4.9}$$

Given the definition of $\{S^{(2)}(i)\}_{i\in\mathbb{N}}$, the condition in (4.2) and equation (4.8), the reasoning for the deduction of p_2, depicted by the subsequent expressions, is easy to follow:

$$2^{2H-2} = \mathbb{E}\left(S^{(2)}\right)^2 = p_2 \times 1 + (1-p_2) \times 0 \Leftrightarrow \tag{4.10}$$

$$\Leftrightarrow p_2 = 2^{2H-2}. \tag{4.11}$$

Probability p_2 assures that the variance of the (non-intersecting) aggregated series for $m_k = 2$ converges to 2^{2H-2}, as the number of points generated with property (4.9) tends to infinity (*law of large numbers*). In other words, the convergence of the Hurst parameter to the predefined value is guaranteed by this construction, at least for the aggregation scale of 2.

If the scale of 4 is now taken into consideration, one will notice that, at each 4 points, there is only one point for which no relation was presented until here (e.g. $S(1)$ depends of $S(0)$ and $S(3)$ depends of $S(2)$, but the dependence of $S(2)$ is still not defined). Consider that this point depends of the *first* of the *two* preceding ones, as depicted in Figure 4.3 and formalised by (4.12):

$$p_4 = P(S(t) = S(t+2), t \in 4\mathbb{N}), \tag{4.12}$$

To get the persistence probability p_4, one can follow the reasoning depicted by (4.13) and (4.14) (notice that a change of scale is performed in (4.13) and that, for that reason, the granularity of the indexes is doubled, i.e. $i = 2i'$):

$$4^{2H-2} = \mathbb{V}\left(S^{(4)}(i')\right) \overset{i=2i'}{=} \mathbb{V}\left(\frac{2}{4}S^{(2)}(i) + \frac{2}{4}S^{(2)}(i+1)\right) = \tag{4.13}$$
$$= \frac{1}{4}\mathbb{V}\left(S^{(2)}(i) + S^{(2)}(i+1)\right) = \frac{1}{2}\mathbb{V}\left(S^{(2)}\right) + \frac{1}{2}C\left(S^{(2)}(i), S^{(2)}(i+1)\right) \Leftrightarrow$$

$$\Leftrightarrow 4^{2H-2} = \frac{1}{2}2^{2H-2} + \frac{1}{2}C\left(S^{(2)}(i), S^{(2)}(i+1)\right). \tag{4.14}$$

Notice that $C\left(S^{(2)}(i), S^{(2)}(i+1)\right)$, in the previous expressions, denotes the covariance between $S^{(2)}(i)$ and $S^{(2)}(i+1)$ which, in this case, can be expressed in terms of p_4 as indicated by (4.15), where $\{S_+^{(m_k)}(i)\}_{i=1,2,\ldots}$ denotes the series obtained from $\{S^{(m_k)}(i)\}_{i=1,2,\ldots}$ via the multiplication of the value of each aggregated block by the sign of its first constituent addend, i.e. $S_+^{(m_k)}(i) = sign(S(i)) \times S^{(m_k)}(i)$:

$$\begin{aligned} C\left(S^{(2)}(i), S^{(2)}(i+1)\right) &= \mathbb{E}\left(S^{(2)}(i) \times S^{(2)}(i+1)\right) = \\ &= p_4 \times \mathbb{E}\left(S_+^{(2)}(i)\right) \times \mathbb{E}\left(S_+^{(2)}(i)\right) - (1-p_4) \times \mathbb{E}\left(S_+^{(2)}(i)\right) \times \mathbb{E}\left(S_+^{(2)}(i)\right) \Leftrightarrow \\ &\Leftrightarrow C\left(S^{(2)}(i), S^{(2)}(i+1)\right) = (2p_4 - 1) \times \mathbb{E}\left(S_+^{(2)}\right) \times \mathbb{E}\left(S_+^{(2)}\right). \end{aligned} \tag{4.15}$$

Equation (4.14) can now be written into form (4.16) or, equivalently, into form (4.17):

$$4^{2H-2} = \frac{1}{2}2^{2H-2} + \frac{1}{2}(2p_4 - 1) \times \mathbb{E}\left(S_+^{(2)}\right) \times \mathbb{E}\left(S_+^{(2)}\right) \Leftrightarrow \tag{4.16}$$

$$\Leftrightarrow p_4 = \frac{2 \times 4^{2H-2} - 2^{2H-2}}{2 \times \left(2^{2H-2}\right)^2} + \frac{1}{2}. \tag{4.17}$$

If an analogous reasoning is applied to all the other scales of type 2×2^{N_p}, a closed form expression (formula (4.18)) can be derived to all *persistence probabilities* defined in (4.4). Consider observing Figure 4.3 for a conceptual representation of the relations between points of the walk:

$$p_{m_k} = \frac{2 \times m_k^{2H-2} - (m_k/2)^{2H-2}}{2 \times \mathbb{E}\left(S_+^{(m_k/2)}\right) \times \mathbb{E}\left(S_+^{(m_k/2)}\right)} + \frac{1}{2}. \tag{4.18}$$

The unique parameter in equation (4.18) for which lacks information is $\mathbb{E}\left(S_+^{(m_k/2)}\right)$. By definition, $\mathbb{E}\left(S_+^{(1)}\right) = 1$, and the value of $\mathbb{E}\left(S_+^{(2)}\right)$, used to assess p_4, can still be drawn from a reasoning similar to the one in (4.19). For larger values of m_k, the successive values

of $\mathbb{E}\left(S_+^{(m_k/2)}\right)$ can be obtained using the recursive formula (4.20):

$$\mathbb{E}\left(S_+^{(m_k/2)}\right) = p_2 \times \left(\frac{1}{2} \times \frac{(1+1)}{2} - \frac{1}{2} \times \frac{(-1-1)}{2}\right) = 2^{2H-2} \text{ for } m_k = 4. \tag{4.19}$$

$$\mathbb{E}\left(S_+^{(m_k/2)}\right) = p_{m_k/2} \times \mathbb{E}\left(S_+^{(m_k/4)}\right) \ ,\forall m_k > 4. \tag{4.20}$$

The result presented in last is actually valid for $m_k \geq 4$, and it may be justified using expressions (4.21) and (4.22):

$$\mathbb{E}\left(S_+^{(m_k/2)}(i')\right) \overset{i=2i'}{=} \frac{\frac{m_k}{4}\mathbb{E}\left(S_+^{(m_k/4)}(i)\right) + \frac{m_k}{4}\mathbb{E}\left(S_+^{(m_k/4)}(i+1)\right)}{\frac{m_k}{2}} = \tag{4.21}$$

$$= \frac{1}{2}\left(\mathbb{E}\left(S_+^{(m_k/4)}(i)\right) + (2p_{m_k/2} - 1) \times \mathbb{E}\left(S_+^{(m_k/4)}(i)\right)\right) = p_{m_k/2} \times \mathbb{E}\left(S_+^{(m_k/4)}\right). \tag{4.22}$$

One of the main differences between the approach described in this thesis and the work of Enriquez, in [114], is the length of the dependence between points. From a naive perspective, one can simply say that p_{m_k} is the probability of an aggregated block (for a given aggregation scale m_k) to maintain the sign of its predecessor. The way these *persistence probabilities* are defined herein (expression (4.18)) makes clear that the dependence from distant points increases with the aggregation scale considered (also noticeable in Figure 4.1), while in [114], each step depends exclusively on the previous one. Just to give an idea of the values these persistence probabilities may take for different Hurst parameter values and aggregation scales, the chart in Figure 4.1 was included.

One of the biggest drawbacks of this proposal is that it is not possible to generate correlated random walks exhibiting anti-persistence behaviour because, depending on the aggregation scale and on the Hurst parameter, it is not possible to express the persistence probabilities as numbers in the interval $[0, 1]$. This happens because the dependencies scheme favours long-range dependencies, and these tend to fade to short-range as the Hurst parameter tends to 0.

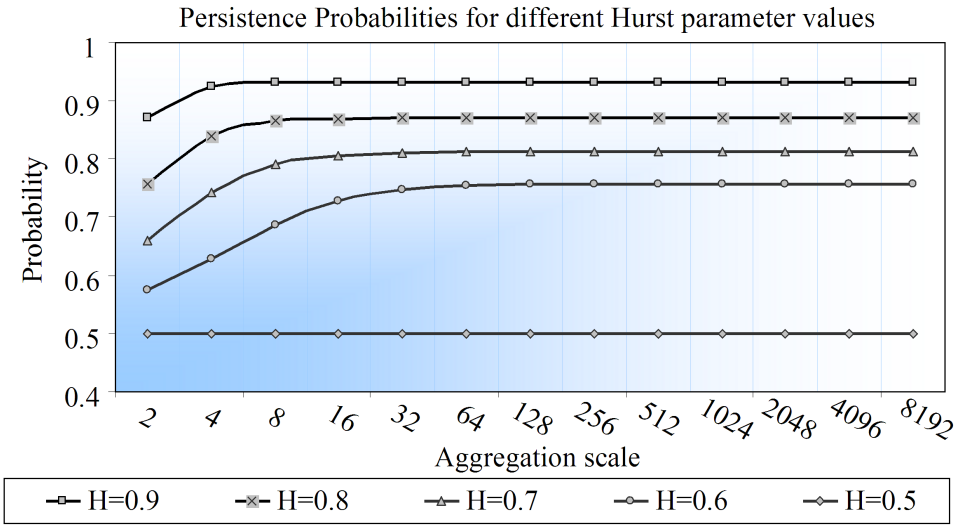

Figure 4.1: The values of the persistence probabilities for different Hurst parameter values. This values were obtained using the formulas described herein.

Approximating Fractional Brownian Motion

Now that the persistent random walks and the *persistence probabilities* have been defined, the procedure to approximate fBm processes can be more easily described. Consider that for each $i \in \mathbb{N}$, $\{S_i(t)\}_{t\in\mathbb{N}}$ is a correlated random walk as defined in the previous subsection with a fixed Hurst parameter H. Based on the *Central Limit theorem* and given the construction of the process $\{Y_N(t)\}_{t\in\mathbb{N}}$ in (4.23), one can write(4.24), where $G(\mu, \sigma^2)$ denotes a normally distributed variable with mean μ and variance σ^2. N denotes the number of independent and identically distributed walks $\{S_i(t)\}_{t\in\mathbb{N}}$, previously defined:

$$Y_N(t) = \frac{\sum_{i=0}^{N}(S_i(t) - 0)}{\sqrt{N}}; \tag{4.23}$$

$$Y(t) = \lim_{N\to\infty} Y_N(t) = G(0, 1). \tag{4.24}$$

It is a simple exercise to prove that the normalised process inherits the self-similar properties of the individual correlated walks (under the condition that the Hurst parameter is the same for all processes $\{S_i(t)\}_{t\in\mathbb{N}}$). Since the *central limit theorem* assures the convergence of $\{Y_N(t)\}_{t\in\mathbb{N}}$ to a Gaussian process for a large N, one can say that $Y_N(t)$ approximates an fGn and, therefore, that $B^*_H(t)$ (see (4.25)) approximates an fBm process

with predefined Hurst parameter H:

$$B_H^*(t) = \sum_{i=0}^{t} Y_N(i). \tag{4.25}$$

Condition (4.23) immediately provides one with a way to produce approximate realisations of fBm processes using the correlated walks. However, the approximation depends on the number of different and independent instances of random walks: the larger the number N is, the more accurate the approximation is. This fact raises an obvious problem in terms of the computation complexity of the procedure: makes it dependent from N the same way precision depends of this variable. At a first glance, this problem seems hard to bypass but, as it is going to be shown afterwards, it will not be impossible.

From a rough perspective, $Y_N(t)$ is merely the normalised sum of $N^+(t)$ 1s and $N^-(t)$ -1s, where $N^+(t) + N^-(t) = N$. Having this in mind, focus in the transition from $t = 0$ to $t = 1$ and notice that, again because of the *central limit theorem*, it can be said that

$$N^+(1) \approx G(p_2 \times N^+(0), p_2(1-p_2)N^+(0)) + N - N^-(1), \tag{4.26}$$

$$N^-(1) \approx G(p_2 \times N^-(0), p_2(1-p_2)N^-(0)) + N - N^+(1). \tag{4.27}$$

The two previous expressions state under formal terms that the number of 1s or -1s at the time moment $t = 1$ depend of the number of 1s and -1s at time $t = 0$, being that dependence defined by a binomial distribution (that converges to a Gaussian distribution as N tends to infinity). After some simple calculations, (4.26) and (4.27) support conclusion (4.28), which can then be generalized to (4.29):

$$Y_N(1) = \frac{N^+(1) - N^-(1)}{N} \approx G\left((2p_2 - 1) \times Y_N(0), 4p_2(1-p_2)\right); \tag{4.28}$$

$$Y(t) \approx G\left((2p_{m_k} - 1) \times Y(t - m_k/2), 4p_{m_k}(1-p_{m_k})\right). \tag{4.29}$$

Figure 4.2 represents an effort to describe the previous sentences and equations

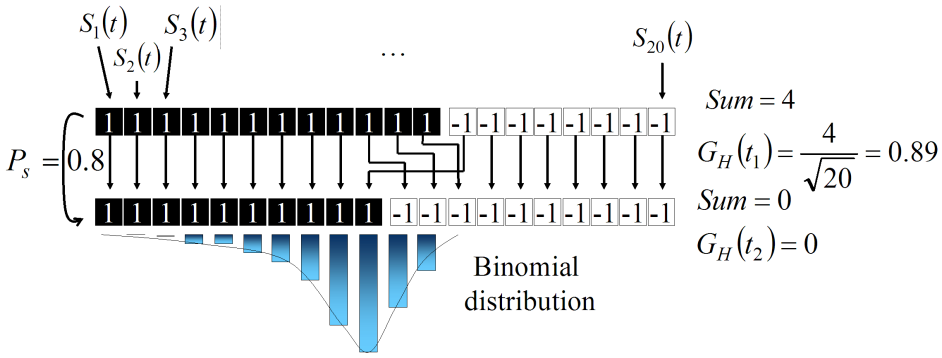

Figure 4.2: Using the sum and normalisation of independent correlated random walks to approximate an fGn. Example of the multiple transitions of 20 processes.

graphically, substantiating the statements with a numerical example. It illustrates the procedure to obtain a point of the series with approximate structure of an fGn, and the *theoretical result* that can be used to bypass the complexity problem of using a large number of correlated random walks to generate a single point of the aforementioned process. In the figure, the transition of 20 correlated random walks is presented, considering the persistence probability of 0.8 (on average, according to this probability, 4 of the 20 random walks change their sign during a transition). The transition from 0.89 to 0, as time changes from t_1 to t_2 is also exemplified. The blue bars on the bottom represent the distribution of the number of processes $\{S_i(t)\}_{t\in\mathbb{N}}$ that remain equal to 1 when time changes to t_2, providing that $S(t_1) = 1$. As previously discussed, this distribution is a binomial distribution with *success probability* equal to 0.8 (in this case).

Since there are algorithms with a computational complexity of $O(n)$ capable of generating Gaussian variables (see section 4.5.3.), it is possible to emulate the sum of a large number of independent processes $\{S_i(t)\}_{t\in\mathbb{N}}, i \in \mathbb{N}$ without proportionally increasing the computational complexity of the method. In other words, it is possible to virtually achieve an infinite degree of precision for the scales of type 2×2^k, without adding complexity to the calculations. This statement is corroborated by the simulation results discussed below.

The only thing left to be discussed is how to get the dependency (and the respective *persistence probability*) of the point $Y(t)$ the algorithm is about to generate. Notice that while the formalisation is being made as if the series were defined for (any) $t \in \mathbb{N}$, any practical implementation of the method will only be capable of generating a finite number

of samples $Y(0), ..., Y(\tau)$, where τ is a fixed and positive integer number. To find that relation, one just needs to find the first positive integer number k that satisfies (4.31), where t is the index of the point that is about to be produced. The *persistence probability* is then given by p_{2^K} and the extent of the dependence is given by $\frac{2^K}{2}$ (i.e. the actual step $Y(t)$ depends of step $Y(t - \frac{2^K}{2})$):

$$\text{Start with } K = 1, \text{ increment } K \in \mathbb{N}, \text{ until} \tag{4.30}$$

$$t \bmod 2^K = \frac{2^K}{2}. \tag{4.31}$$

The algorithm the author has called *fractional Brownian motion Sequential Generation Algorithm (fBm-SGA)* is, at this point, completely defined by the following set of conditions:

For $t = 0$, $Y(t) = G(0, 1)$.

For $t > 0$,

$$Y(t) = G\left((2p_{m_k} - 1) \times Y(t - m_k/2), 4p_{m_k}(1 - p_{m_k})\right).$$

Where,

$$m_k = 2^k,$$

$$p_{m_k} = \frac{2 \times m_k^{2H-2} - (m_k/2)^{2H-2}}{2 \times \mathbb{E}\left(S_+^{(m_k/2)}\right) \times \mathbb{E}\left(S_+^{(m_k/2)}\right)} + \frac{1}{2},$$

$$\mathbb{E}\left(S_+^{(m_k/2)}\right) = 1, \text{ if } m_k = 2.$$

$$\mathbb{E}\left(S_+^{(m_k/2)}\right) = p_{m_k/2} \times \mathbb{E}\left(S_+^{(m_k/4)}\right), \text{ if } m_k > 2,$$

To get K, start with $K = 1$, increment $K \in \mathbb{N}$, until $t \bmod 2^K = \frac{2^K}{2}$, for each $t \in \mathbb{N}$, $t > 0$.

Figure 4.3 and Figure 4.4 illustrate graphically the relations between the points generated using the fBm-SGA. In Figure 4.3, each point of the process is represented by a geometric figure and the relations are represented by arrows. As can be seen, the points that explicitly depend on their preceding one are symbolised with grey triangles;

the circles represent points that explicitly depend of the past point whose relative distance is of 2 unities, etc.

In Figure 4.4 some of the relations illustrated in Figure 4.3 are intentionally omitted, to emphasise that the difference between the two scenarios is the number of precision scales supported, which is larger (potentially infinite) in the first depicted case. In the second case, the biggest aggregation scale supported is 16, and the process behaves like a normal random walk for scales bigger than that. In other words, $X(16)$ holds no relation with any of the points preceding it.

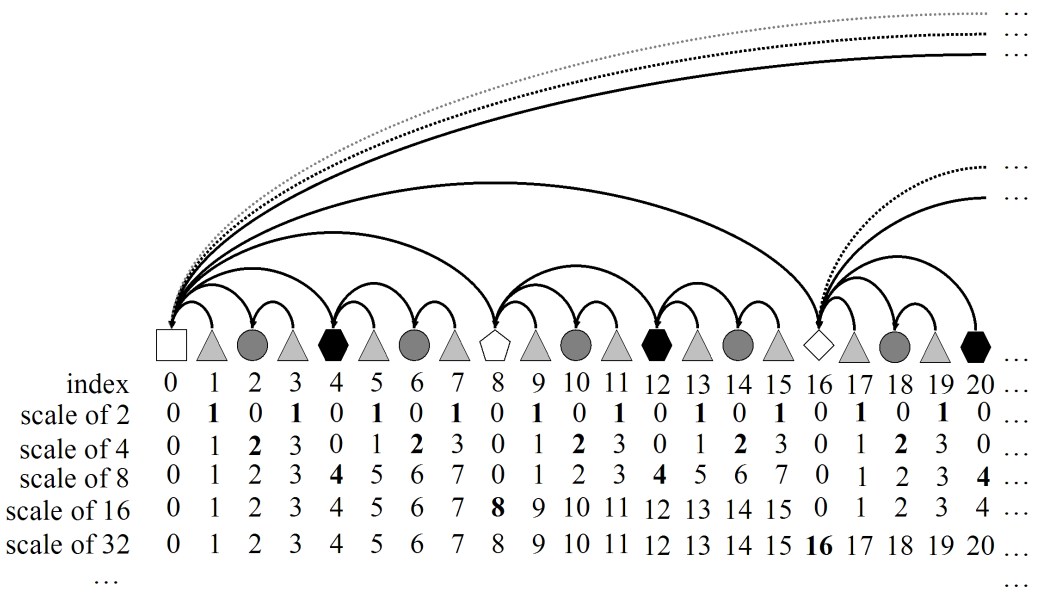

Figure 4.3: Direct and indirect relations (dependencies) between realisations of a correlated random walk or of an approximate fBm generated using the fBm-SGA. Long-range dependence is assured for all scales of type 2×2^{N_P}.

One interesting remark that can be made at this point is that the application of the current scheme guarantees that each point of the generated sequence depends, one way or another, from the value that was first produced. Ideally, if an infinite number of points was to be generated and an infinite number of precision scales was to be supported, the initial point of the process would have to be stored indefinitely since it would be *eventually* used to generate *distant* points. It is not very difficult to conclude that, at some point of the generation procedure, the generated values have to be memorised and labelled as *past points*. In the worst case scenario, the procedure will have to store one of these points for each *supported* aggregation scale of type 2×2^{N_P}.

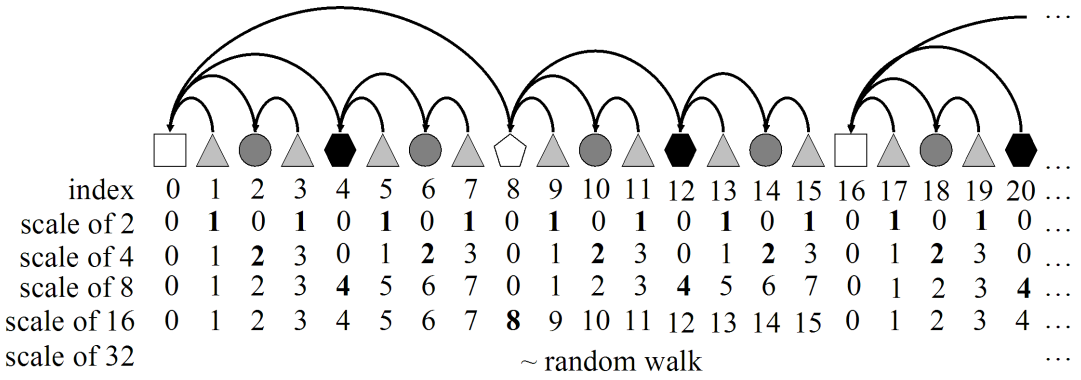

Figure 4.4: Direct and indirect relations (dependencies) between realisations of a correlated random walk or of an approximate fBm generated using the fBm-SGA. The process is long-range dependent for the scales of 2, 4, 8 and 16. When aggregated for scales larger than 16, the process behaves like a (memoryless) Random Walk.

4.3.2. Quality Assessment via Hurst Parameter Estimation

The evaluation of an fGn simulation procedure has to take into account two main conditions: (i) the generated series should follow a Gaussian distribution; and (ii), they should exhibit the typical scaling phenomena of a self-similar process. While, in some situations, it is possible to guarantee both conditions from the theoretical plane, in the case of having an *approximate* algorithm (like the ones presented herein), some of the prerequisites have to be demonstrated via empirical observation. The algorithms that are said to be *exact* assure that both conditions are fulfilled theoretically. On the other hand, the *approximate* procedures may only provide assurance for the convergence of the generated series to the exact one, or guarantee them to a certain extent. In such cases, *gaussianity* may be proven by some *goodness-of-fit* test (e.g. Jarque-Bera test [117]), and self-similarity is often evaluated by estimating the Hurst parameter of the series created by several instantiations of the generator [58, 61].

The foundations of the algorithms proposed herein assure that the generated series are Gaussian. They are also said to be *exact* for the scales of type 2×2^{N_P}, and *approximate* for the remaining ones. Because of this, the respective quality evaluation sections of the two self-similar sequences generator include the results obtained via the generation of series and consecutive estimation of the Hurst parameter.

With the purpose of estimating the Hurst parameter of the generated traces, the fBm-SGA and 4 different *retrospective* estimation algorithms were implemented in Java programming language: EBP, following the suggestions in [69], and according with what was said in chapter 2; VT, as described in e.g. [38, 82]; RS, as described in [38]; and DFA, as described e.g. in [85, 86].

In this section, some examples of long-range dependent sequences with predetermined Hurst parameter, generated using the fBm-SGA, are presented and some considerations about them are drawn. After that, a study about the accuracy of the method is included.

The three charts included in Figure 4.5 depict three approximate fBms with different Hurst parameters. The graphical representation is obtained by plotting 1000 points of the sequences generated using the fBm-SGA against their *indexes*, which emulate the time domain. From careful observation, it is possible to differentiate them by a *roughness factor*, which is a direct consequence of their self-similarity degree. The weaker the dependence from the past is, the more *irregular* the line in the graphical representation looks like. One can also notice that the spatial span of the charts (the y-axis range) increases with the value of the Hurst parameter, in accordance with (2.20).

The fBm-SGA simulates the fBms in Figure 4.5 by generating the individual steps of the motions. In the charts of Figure 4.6, the steps of the previously depicted process are plotted against the order by which they were generated. Self-similarity properties are also reflected in these charts. For instance, the sign change frequency (from positive to negative and vice versa) decreases for higher values of the Hurst parameter. This rate can be understood as a measure of the random nature of the samples and it is easily explained, if one takes into account that the *persistence probabilities* increase with the value of the Hurst parameter. In other words, the probability of a step of the generated sequence being equal to the previous ones increases with the Hurst parameter. Consequently, the moving average value of the first order differences process is affected (biased) by the value of the Hurst parameter for relatively small numbers of steps.

Two different kind of tests were conducted for each one of the implemented estimators. Initially, the accuracy was analysed for the *scales of type* 2×2^{N_P}, for which the

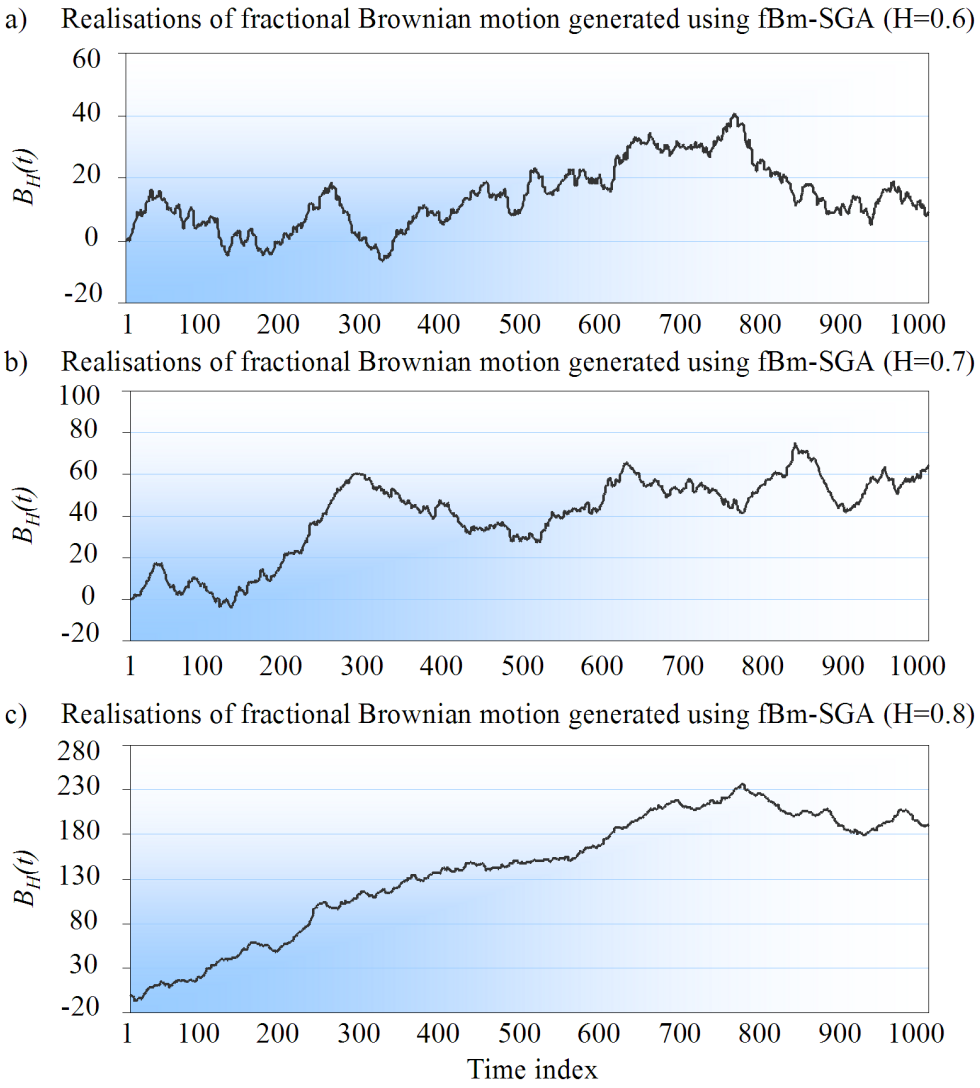

Figure 4.5: Examples of fBm processes generated using the fBm-SGA and exhibiting persistent behaviour with pre-determined Hurst parameter: a) $H = 0.6$, b) $H = 0.7$ and c) $H = 0.8$.

algorithm is supposed to be asymptotically exact. After that, to prove that the synthesized processes maintain the self-similarity properties, at least in an approximated sense for the *scales of type* 3×2^{N_P}, the accuracy was also analysed for this type of scales. A total of 100 simulations were performed for each kind of scales, and the results were statistically compiled in the form of a duple (sample average and variance) and included below. All the generated processes had 10^6 points and the *number of precision scales parameter* has been set to 18 for all simulations.

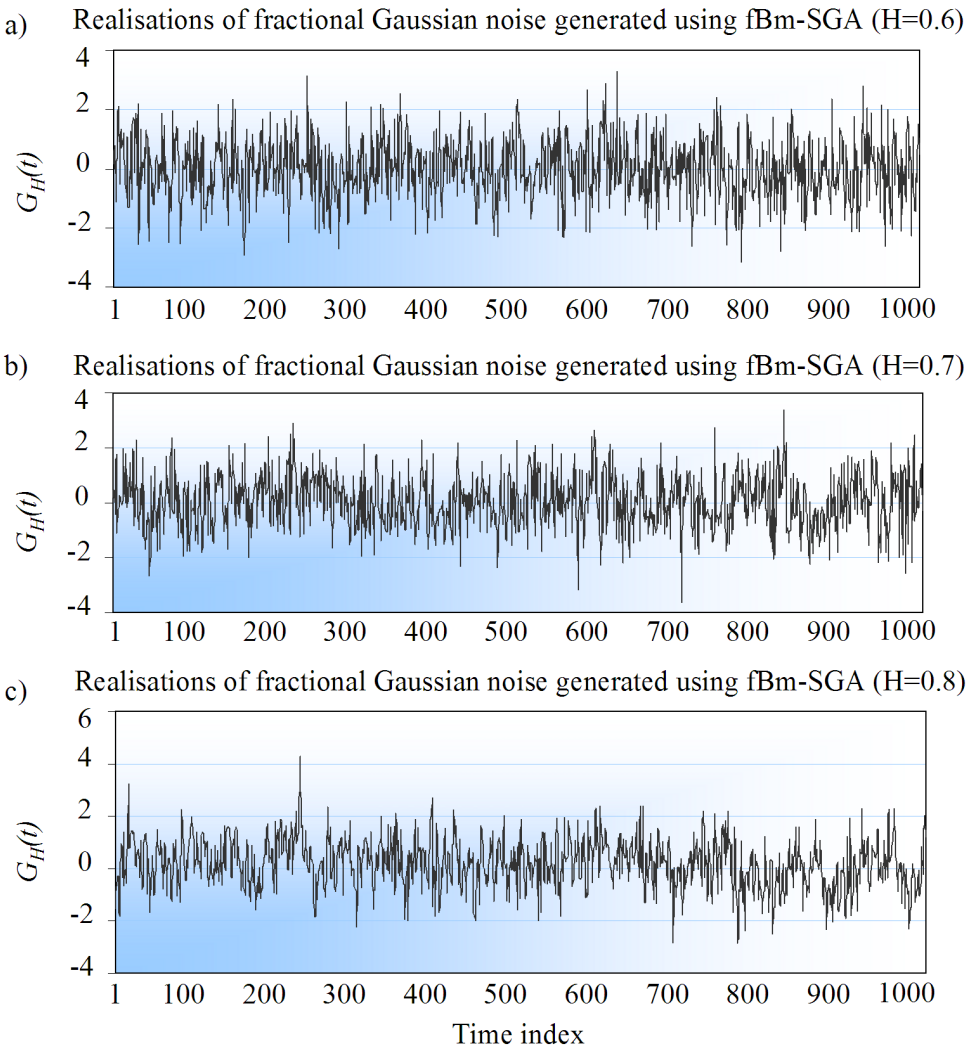

Figure 4.6: Examples of fGn processes, generated using the fBm-SGA, with pre-determined Hurst parameter: a) $H = 0.6$, b) $H = 0.7$ and c) $H = 0.8$.

Embedded Branching Process

EBP is suitable for estimating the values of the Hurst parameter of processes exhibiting persistent behaviour. Contrarily to the other implemented methods, this one does not use directly any type of aggregation of the self-similar process for estimation of the Hurst parameter value. Therefore, it is not possible to adjust the type of the scales. By looking into its definition, it can be said that this method deals with temporal scales that vary from 2 to 4, as the Hurst parameter varies from 0.5 (inclusively) to 1 (exclusively). Consider referring to [69] or to chapter 2 for further information on this topic.

Notice that the way EBP is used herein is different from the way it was used in

chapter 3 in three different aspects: first of all, it is being used retrospectively; secondly, it was implemented according with [69], where μ is calculated using formula (2.51); and thirdly, the series is not being normalised on-the-fly (conversely to what was done in chapter 3). In chapter 3, the series was being normalised not because it was required to, but because during normal operation, a point-by-point or windowed estimator should be capable of handling shifted and scaled series.

Variance Time

The VT estimation method is based on condition (4.6), which is the same condition that inspired the development of the proposed algorithm. Therefore, this was the method for which the best performance was to be expected, and the results corroborate this belief. The charts in Figure 4.7 contain a representation of the VT log-log plot (see section 2.4.2. for more information on this subject). As it was explained with detail in chapter 2, the Hurst parameter value can be obtained from these plots by finding the line that best fits the plotted values and by calculating its slope (for instance, for the DFA method, the slope of the line is the Hurst parameter itself). Finding the lines for the presented charts is not difficult, and in this particular case such task could actually be reduced to the simple exercise of connecting all of the individual points. Because these charts constitute one of the best means to depict the scaling properties of the sequences synthesised with the fBm-SGA, it was decided to include a similar graphical representation for each one of the next two methods.

Rescaled Range Statistics

The chart in this subsection (Figure 4.8) concerns the RS analysis. As before, the Hurst parameter value is obtained from the slope of the line that best fits the plotted values resulting from the RS analysis. Because a small bias was noticed for relatively small aggregation scales, the scales of 2 and 4 were not considered in the line fitting procedure and, as for higher aggregation scales only a few samples of the aggregation process are available, they were not taken into account either. The lines represented in the charts are, therefore, the ones that best fit the 15 intermediate points of the log-log

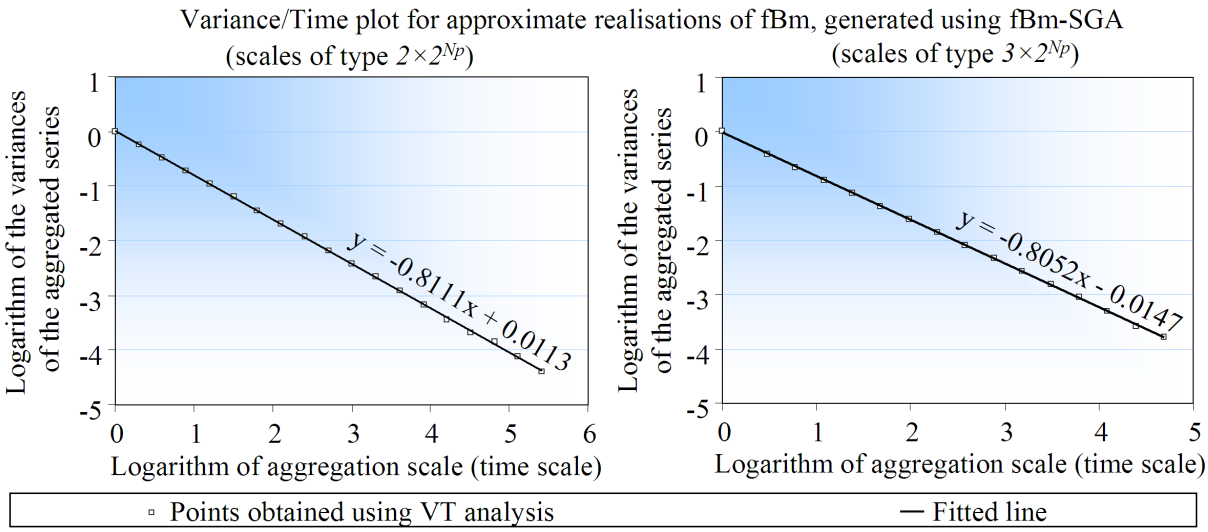

Figure 4.7: VT log-log plots for a sequence with 10^6 points, generated with the fBm-SGA. In this case, the expected Hurst parameter was equal to 0.6, the aggregation *scales are of type* 2×2^{N_p} for the log-log plot on the left, and *of type* 3×2^{N_p} for the chart on the right. The estimated Hurst parameter was 0.60 for the first, and 0.61 for the second.

plot.

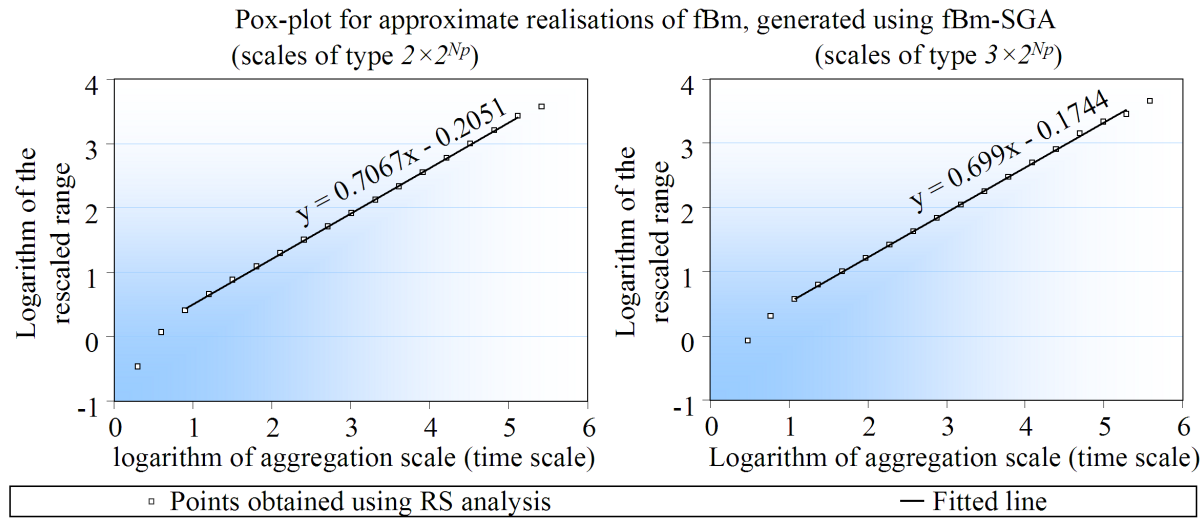

Figure 4.8: RS log-log plots for a sequence with 10^6 points, generated with the fBm-SGA. In this case, the expected Hurst parameter was equal to 0.70, the aggregation *scales were of type* 2×2^{N_p} for the log-log plot on the left, and of type 3×2^{N_p} for the chart on the right. The estimated Hurst parameter was 0.71 for the first, and 0.70 for the second.

Detrended Fluctuation Analysis

The results obtained via utilisation of the DFA method were surprisingly good (Figure 4.9). In fact, they were the best if compared with the ones returned by the others estimators used. An interesting remark that can be made at this point is that the oscillation of the points resulting from the DFA method around the fitted line is bigger when the *scales are of type* 3×2^{N_p}. This statement is valid for the VT and RS methods. Also true and applicable to these last three methods is the fact that the Hurst parameter value estimated for the *scales of type* 3×2^{N_p} differs from the one estimated for the *scales of type* 2×2^{N_p} in approximately 0.01 units.

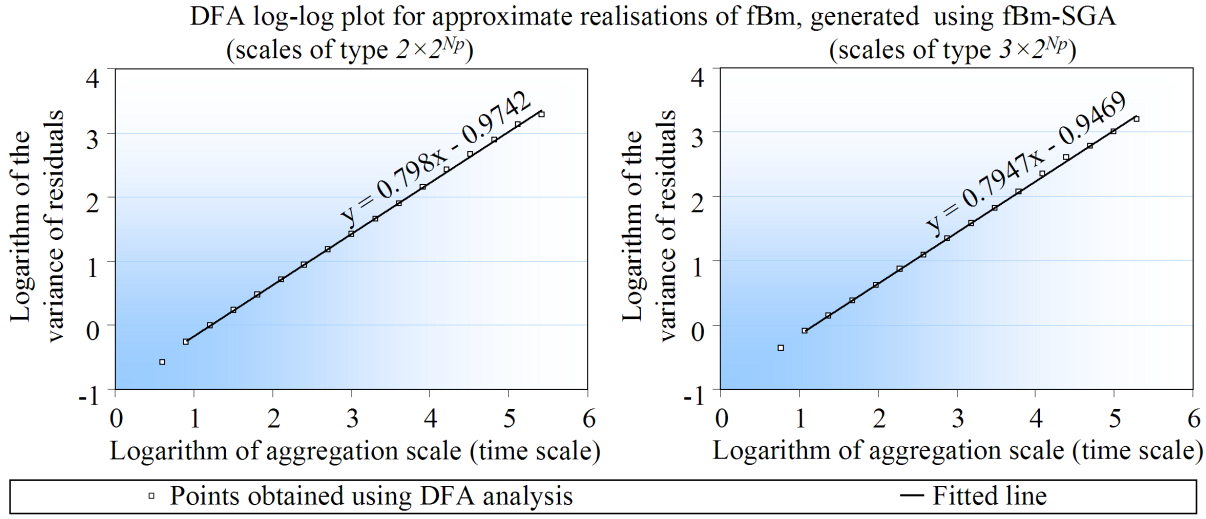

Figure 4.9: DFA log-log plots for a sequence with 10^6 points, generated with the fBm-SGA. In this case, the expected Hurst parameter was equal to 0.80, the aggregation *scales were of type* 2×2^{N_p} for the log-log plot on the left, and of type 3×2^{N_p} for the chart on the right. The estimated Hurst parameter was 0.80 for the first, and 0.79 for the second.

Discussion of the Results

The accuracy of the fBm-SGA was analysed until the hundredth of the value of the Hurst parameter, because it was soon noticed that no valuable conclusion could be drawn beyond that degree of precision. One of the factors that can explain this fact is the usage of a computer-based PRNG to emulate the probabilities co-domain. It is obvious that even guaranteeing the quality of the generator, the algorithm will always be affected by a small correlation between the values the PRNG returns. Another aspect that must

be taken into account is that the accuracy of the presented algorithm benefits from the number of points generated, since its rationale is heavily dependent on the *law of large numbers*.

Hurst parameter values ranging from 0.5 to 0.99 (with increments equal to 0.01) were tested through computer-based simulation. The next two tables contain a subset of the numerical results obtained for the *scales of type* 2×2^{N_p} (Table 4.1) and for the *scales of type* 3×2^{N_p} (Table 4.2) for the four above mentioned estimators. All the values returned by the estimators were averaged and rounded off to the hundredth. The variances were rounded and written in scientific notation (with two decimal places). The graphical representation in Figure 4.10 and Figure 4.11 compress a bigger set of results. In the charts, the average of the estimates of the Hurst parameter is plotted against the expected value, for each estimation method. The line with equation $y = x$ was also plotted in the charts as a comparison reference.

Table 4.1: Target and estimated Hurst parameter values for scales of type 2×2^{N_p}. Each cell contains the average and the variance (in brackets) of the results of 100 simulations.

Hurst	**VT**	**DFA**	**RS**	**EBP**
0.50	0.50(1.77E-05)	0.50(1.86E-04)	0.50(1.02E-04)	0.50(7.41E-06)
0.55	0.55(1.96E-05)	0.55(2.18E-04)	0.55(9.44E-05)	Table 4.2
0.60	0.60(1.89E-05)	0.60(3.16E-04)	0.60(1.25E-04)	Table 4.2
0.65	0.65(1.58E-05)	0.65(3.12E-04)	0.64(9.44E-05)	Table 4.2
0.70	0.70(1.18E-05)	0.69(3.21E-04)	0.68(1.16E-04)	Table 4.2
0.75	0.75(1.55E-05)	0.75(3.43E-04)	0.73(1.37E-04)	Table 4.2
0.80	0.80(3.12E-05)	0.80(5.96E-04)	0.77(1.68E-04)	0.79(1.07E-05)
0.85	0.85(3.83E-05)	0.85(6.31E-04)	0.82(3.82E-04)	0.84(7.35E-06)
0.90	0.90(1.56E-04)	0.90(1.18E-03)	0.87(5.54E-04)	0.89(7.40E-06)
0.95	0.94(3.99E-04)	0.95(2.32E-03)	0.93(1.12E-03)	0.95(4.29E-06)

Based on the presented results, one can safely conclude that the fBm-SGA is very accurate, in terms of expected Hurst parameter. In average, the estimated values do not differ from the expected values more than 0.02 for any of the estimation methods used. The DFA estimator is the one that most favours the precision of the algorithm. It returns

Table 4.2: Target and estimated Hurst parameter values for scales of type 3×2^{N_p}. Each cell contains the average and the variance (in brackets) of the results of 100 simulations.

Hurst	**VT**	**DFA**	**RS**	**EBP**
0.50	0.50(2.24E-05)	0.50(1.24E-04)	0.51(1.29E-04)	see Table 4.1
0.55	0.55(2.79E-05)	0.54(1.81E-04)	0.55(1.31E-04)	0.54(7.08E-06)
0.60	0.60(3.78E-05)	0.60(2.29E-04)	0.60(1.75E-04)	0.58(7.40E-06)
0.65	0.64(2.03E-05)	0.64(1.71E-04)	0.64(1.64E-04)	0.63(9.16E-06)
0.70	0.69(2.05E-05)	0.69(2.02E-04)	0.69(2.17E-04)	0.68(1.21E-05)
0.75	0.74(1.86E-05)	0.74(3.23E-04)	0.73(2.42E-04)	0.73(1.05E-05)
0.80	0.79(3.81E-05)	0.79(4.96E-04)	0.78(3.67E-04)	see Table 4.1
0.85	0.84(8.70E-05)	0.85(5.78E-04)	0.83(5.77E-04)	see Table 4.1
0.90	0.88(2.17E-04)	0.90(9.02E-04)	0.86(9.43E-04)	see Table 4.1
0.95	0.93(5.64E-04)	0.95(1.80E-03)	0.91(1.18E-03)	see Table 4.1

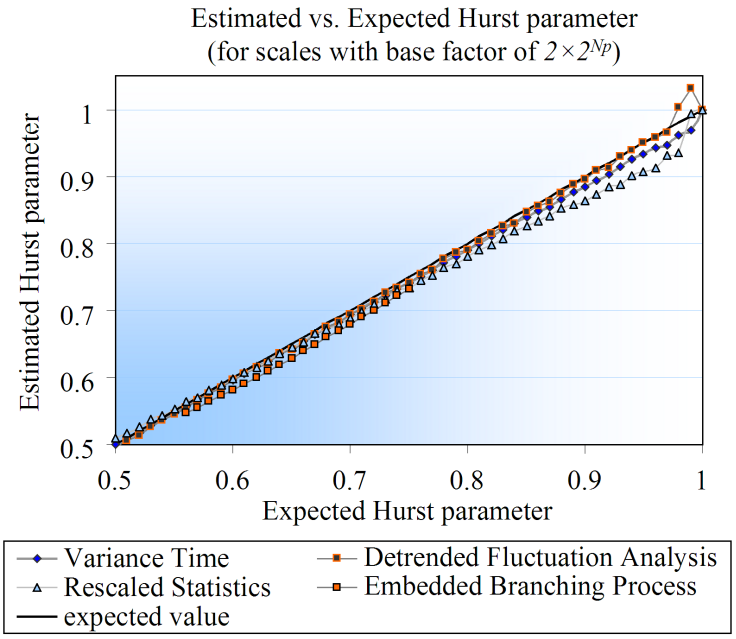

Figure 4.10: Comparison between the expected and the estimated Hurst parameter values. The VT, the DFA and the RS estimators were testing *scales of type* 2×2^{N_p}.

values that are exact (on average) 94% of the times for the *scales of type* 2×2^{N_p}, and 82% of the times for the *scales of type* 3×2^{N_p}. Because of this, the lines concerning equation $y = x$ and the estimated values for the mentioned method are almost indistinguishable in both charts. The same happens with the VT method for *scales of type* 2×2^{N_p}. However,

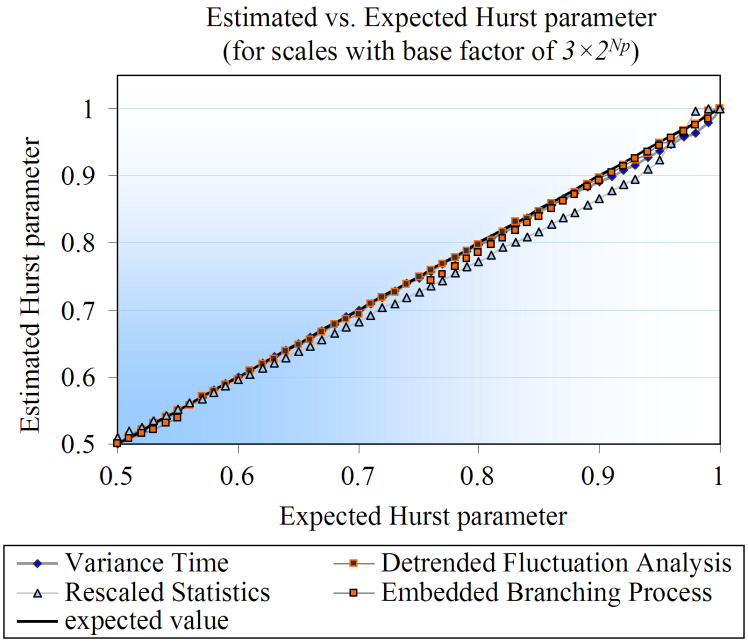

Figure 4.11: Comparison between the expected and the estimated Hurst parameter values, this time for *scales of type* 3×2^{N_P}.

this last method results are not as good as the first ones for *scales of type* 3×2^{N_P}. On the other hand, the RS method was the one with the worst results obtained for both types of scales. Nevertheless, despite showing an error bigger than 0.03 for Hurst parameter values bigger than 0.85, it can be seen that the estimated values follow the trend of the values obtained by the others estimators, and that for smaller values of the Hurst parameter, the same error is again not bigger than 0.001, on average.

Given what was said previously, the values obtained by the EBP estimator were, somehow, expected and they are easy to explain. There is a decrease of the accuracy as the estimated Hurst parameter values vary from 0.5 to 0.63 (which corresponds, lets say, to a variation of the observation scale from 4 to 3) and again an increase, as the estimated values tend to 1 (the temporal observed scale tends to 2). By these reasons, the values obtained for the aforementioned estimator were respectively distributed by the two tables and charts. The bias that was emphasised during the analysis to the point-by-point implementation of the method is not noticeable when the series of values is not being iteratively normalised.

The variance of the estimated values seems to increase with the Hurst parameter value for all methods. The difference between the expected and the estimated Hurst pa-

rameter values increases in the same manner. For *scales of type* 3×2^{N_p}, all the estimators underestimate the value of the Hurst parameter, for which it may be concluded that, for types of scale different from 2×2^{N_p}, the estimated values of the Hurst parameter will always be smaller than the expected ones (the series is thus *attracted* to randomness).

4.3.3. Computational Performance and Memory Requirements of fBm-SGA

The following two sections are dedicated to the discussion of the computational requirements of fBm-SGA. As the computational complexity of this algorithm is equal to the one of the second approximate self-similar sequences generator presented in this chapter, the theoretical demonstration of this aspect is not included for now, but the reader may refer to section 4.4.3. for more details on this subject.

Computational Performance of fBm-SGA

The tests concerning the fBm-SGA computational speed were performed in a non-dedicated, though controlled, system with the following specifications: a 2.8 GHz Pentium IV processor with 504 Mb of effective RAM, running Microsoft Windows XP with Service Pack 2. In order to reduce to the minimum all possible external influences, there were no other processes running on the computer by the time this simulation was performed. As before, the presented conclusions are the result of the statistical treatment of a total of 26000 simulations, conducted for 260 different configurations.

From all the possible data representations, the chart in Figure 4.12 was selected for two main reasons: (i) besides emphasising the impact that the number of precision scales has on the efficiency of the algorithm, (ii) it is also the one that best depicts the linear relation between the number of points generated and the time spent to generate them. The shape of the curves in the chart of Figure 4.12 are a reflex of the computational complexity of the algorithm. Since they are straight lines, instead of logarithmic look-alike curves, they sustain the conclusion about the computational complexity being of $O(n)$. The generation of an high quality correlated fractional motion with 10^6 samples

does not take more than 0.3 s. Therefore, the aforementioned Java implementation of the fBm-SGA operates at an average rate of 3500 points per millisecond (in the simulation machine).

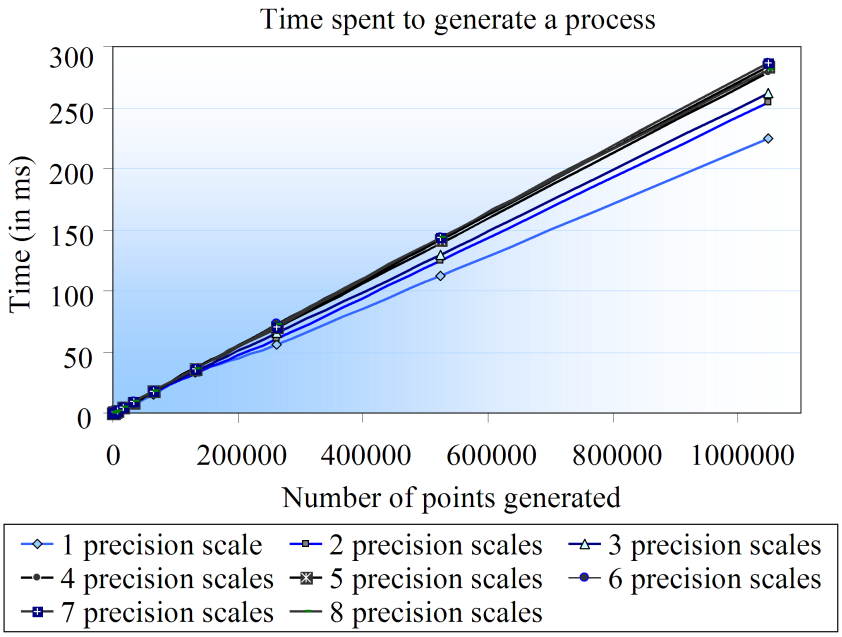

Figure 4.12: Chart where the average time spent by fBm-SGA to generate an fBm process is plotted against the total number of points generated, for different levels of quality (expressed in terms of number of precision scales supported).

Memory Requirements of fBm-SGA

It comes as no surprise that, in terms of memory requirements, fBm-SGA is not pretentious either. Its biggest demand concerns the storage of the N_p (*past*) points, which are necessary to generate *the future* values of the sequence. These values may be stored in *float* type variables, whose number is never required to be bigger than $\log_2(n)$, where n is the length of the series to be generated. Thus, if the Size of the type *Float* representation in the memory, in Bytes, is given by SFB, the codification and posterior instantiation of this algorithm does never take more than $SFB \times \log_2(n)$ bytes to generate a sequence with n points. In the case of a Java implementation of the method, 64 *double* type variables (512 B) would be sufficient to generate a data series with, at least, $2^{65} \approx 36 \times 10^{18}$ points.

4.3.4. Usefulness of fBm-SGA Within the Scope of the Thesis

As it was already mentioned, the previously described algorithm was particularly useful to test the modified and the windowed modified estimators, presented in chapter 3. During the explanation of those estimators, the usage of fBm-SGA was signalised, but the underlying algorithm was not named. At this point, the author would like to emphasise that the tests herein were conducted so as to demonstrate the quality of the generator and, consequently, of the estimators. The outcomes of the generator were first submitted to several retrospective implementations of the estimators, to conclude about its precision. The precision of the windowed estimators was only tested afterwards.

Because of its origins, the fBm-SGA could have been actually used to test MVT without too many quality concerns. Its construction guarantees that the fabricated series should respect the *laws* that VT aims to check, when assessing the self-similarity degree, at least for the aggregation scales of type 2×2^{N_p}. Nevertheless, to demonstrate that the algorithm was duly implemented and that the reasoning behind it was correct, the retrospective version of VT was implemented and used to evaluate the generator. As it was stressed out, this algorithm may not be held responsible for not assuring the exact scaling properties for other aggregation scales. For those, the Hurst parameter is expected to be *lightly* smaller than the predefined one. This artefact is immediately noticed by the usage of e.g. windowed estimators.

One of the best properties of fBm-SGA is that it returns the points the same way the estimators process them. This simple fact enabled the simulations to be conducted for longer series, without any memory problem (no need to store the sequences of values in e.g. files), and to repeat them at will, without being affected by severe time limitations.

4.4. The Simple Self-Similar Sequences Generator

Incapable to abstract himself from the major drawbacks of fBm-SGA, and inspired by the new *probabilistic perspective* provided by the *persistence probabilities*, the author designed a different algorithm that inherits the qualities of fBm-SGA, and improves it in two different aspects: the asymptotic precision, and the range of Hurst parameter values

it supports. The algorithm entitled herein of *Simple Self-Similar Sequences Generator (4SG)* is a simulation method for fGns, capable of efficiently producing long sequences of numbers, exhibiting *persistent or anti-persistent* behaviour (Hurst parameter values ranging from 0 (exclusively) to 1 (exclusively)). Notice that fBm-SGA is only capable of generating *persistent* fGn or fBm processes.

The next description follows the template provided by the previous section. 4SG is formally presented in the first subsection, and then tested in the subsequent ones. As the need to evaluate the estimators was satisfied by fBm-SGA, the major application for this generator was the production of high quality self-similar traces of traffic. Therefore, to test the performance of 4SG, the author has decided in a more detailed evaluation that could prove that besides being accurate *enough*, the computational efficiency is also outstanding. For that, the author opted for testing 4SG along with an $O(n^2)$ and *exact* method (Hosking), and with an $O(n \log(n))$ and *approximate* method (Wavelets-based with FFT).

4.4.1. The 4SG Algorithm

The name of this algorithm has more to do with the belief that enabled its creation than with its mathematical foundations and explanation. 4SG was inspired in the belief that any self-similar stochastic process could be described resorting to the definition of two main components: a *constant* and a *variable* one. The duration and contribution of these components would be dependent on the Hurst parameter and on the aggregation scales, and would actually dictate the *amount that remains* (self-)similar during several (and consecutive) occurrences of the stochastic process. All that was left to be done was to characterise mathematically those contributions.

Along the maturing process, it became obvious to the author that the variable part depends on the actual step of the process and that, therefore, the whole series would logically be constituted by *several* constant parts (and not only one), that change their sign at different moments it time. This conclusion is trivial, when compared to what is said in [19], where Willinger, Taqqu, Sherman and Wilson proved that several streams of bits (constant components) following a heavy tail distribution (which defines the length of

the constant part) could all be contributing to the creation of a self-similar series. While such property was on the basis of a proposal to a generation procedure that simulates differently sized (*On/Off*) contributions during runtime, in here, the focus is placed upon the weights that *fixed* sized contributions should have in order to produce approximate realisations of an fGn. The size of the contributions is always a power of 2 and the design of the algorithm is oriented towards the efficient generation of values in a sequential manner.

Figure 4.13 provides the subject at hand with a graphical representation. In the figure, the several components of the sequences appear as blocks with the words *sum* and *subtract*. For the sake of this explanation, the biggest scale represented was the one of 16. The weights of the several contributions are also there, in anticipation to the subsequent mathematical explanation. The *sum* and *subtract* words mean that during their life period, the values of the blocks are either summed or subtracted to the other values, in order to obtain the point of the forged fGn. Consider complementing the following explanation with frequent observations of the figure, while making analogies between what is said in the text and what is represented in the figure.

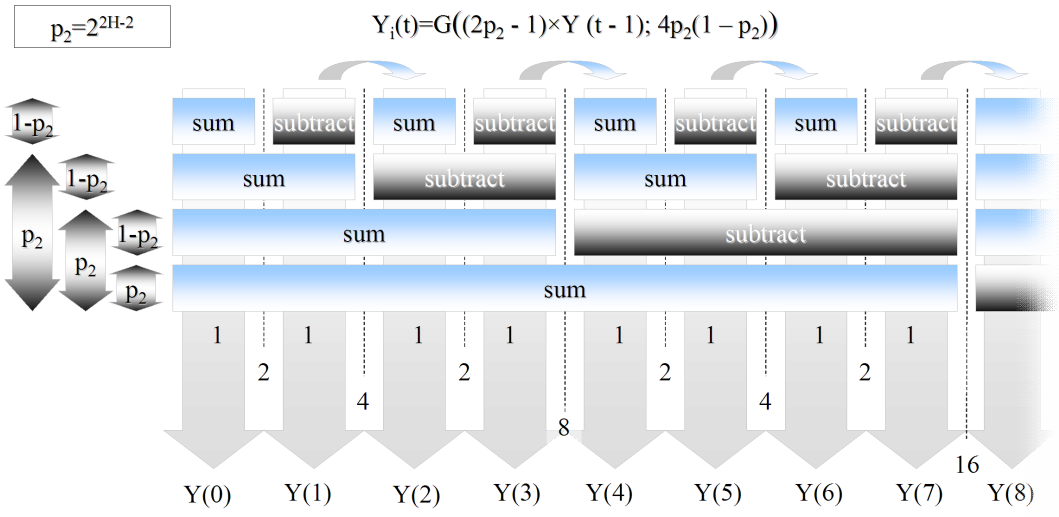

Figure 4.13: Graphical representation of the reasoning behind 4SG.

Starting at 2

Consider that, for each $i = 1, ..., N$, $\{S_i(t)\}_{t\in\mathbb{N}}$ is the first differences process of a random walk $\{R_i(t)\}_{t\in\mathbb{N}}$, respecting the relation defined in (4.32), where H is the Hurst

parameter and $P(S_i(t) = S_i(t+1), \forall i = 1, ..., N, t \in 2\mathbb{N})$ denotes the probability of $S_i(t)$ being equal to its successor, for any $i = 1, ..., N$ and any $t \in 2\mathbb{N}$:

$$P(S_i(t) = S_i(t+1), \forall i = 1, ..., N, t \in 2\mathbb{N}) = 2^{2H-2}. \tag{4.32}$$

Notice that the expression 2^{2H-2} on the right side of (4.32) was already inferred earlier in this chapter, and that its appearance there assures that the processes follow a scaling rule that is consistent with the self-similarity assumptions, for the non-overlapping blocks of size 2.

Consider the time shift from t to $t+1$ and observe that, thanks to the definition of probability, the *average* number of realisations of $S_i(t+1)$ at $t \in 2\mathbb{N}$ that is going to remain equal to its predecessor is given by $2^{2H-2} \times N$. For convenience, and without loss of generality, assume that $\{S_i(t)\}_{t\in\mathbb{N}}$ are i.i.d. and take only the integer values 1 and -1 (i.e., for each $i = 1, ..., N$, the expected value of $\{S_i(t)\}_{t\in\mathbb{N}}$ is 0 and the variance is 1). Focus on the points of $\{S_i(t)\}_{t\in\mathbb{N}}$ with an even time index $t \in 2\mathbb{N}$, and on the relations between one of those points and its successor. Allow $D_{2^k}(t)$ with $k = 1$ to denote the set of the values at time $t \in 2\mathbb{N}$ of all the processes $\{S_i(t)\}_{t\in\mathbb{N}}$ that change their sign (from positive to negative or vice versa), during the transition from t to $t+1$, in accordance with the following expression (valid for $k = 1, 2, ...$):

$$\begin{aligned} D_{2^k}(t) = \{&S_i(\tau), i \in \mathbb{N} : \tau = \lfloor t/2^k \rfloor, \\ &S_i(\tau) = ... = S_i(\tau + 2^{k-1} - 1) \neq S_i(\tau + 2^{k-1}) = ... = S_i(\tau + 2^k - 1)\}. \end{aligned} \tag{4.33}$$

Let $d_{2^k}(t)$ with $k = 1$ be the value obtained by summing and normalising all the values in $D_2(t)$ at time t, as specified by equation (4.34):

$$d_{2^k}(t) = \frac{\sum_{j=0}^{\#D_{2^k}(t)} S_j(t)}{\sqrt{\#D_{2^k}(t)}}, \text{ where } S_j(t) \in D_{2^k}(t),\ k = 1, ..., N_p. \tag{4.34}$$

The normalisation requires dividing the sum by the square root of the cardinality of $D_2(t)$, denoted by $\#D_2(t)$. However, in this particular case, $\#D_2(t)$ can be safely substituted by

its probabilistic equivalent, given by $(1-p_2)\times N$, where $p_2 = 2^{2H-2}$. Notice that $D_{2^k}(t)$ and $d_{2^k}(t)$ are both defined for $t \in \mathbb{N}$ though, by construction, neither of them changes during the lifetime of the aggregation block to which they refer to. In other words, they are constant except for t multiple of 2^k (recall that τ is the largest integer not greater than $t/2^k$).

Allow for the variables $E_2(t)$ and $e_2(t)$ to be *complementary in definition* to $D_2(t)$ and to $d_2(t)$, respectively. I.e. $E_2(t)$ is the set of all $\{S_i(t)\}_{t\in\mathbb{N}}$ that do *not* change their value during the referred time shift, and $e_2(t)$ is the normalised sum of the values $S_i(t)$ in $E_2(t)$, at time t. Given that $e_2(t)$ and $d_2(t)$ result both from the sum of i.i.d. variables, the process $\{Y(t)\}_{t\in\mathbb{N}}$, given by (4.35), tends to a Gaussian process as N tends to infinity. Considering the definition of each one of the processes in the referred equation, equation (4.36) and the respective simplification in (4.37) hold true. For the same reasons, $e_2(t)$ and $d_2(t)$ converge also to Gaussian variables, as N tends to infinity:

$$Y(t) = \frac{e_2(t) + d_2(t)}{\sqrt{2}}, \tag{4.35}$$

$$Y(t) = \frac{\sqrt{p_2 N}\times e_2(t) + \sqrt{(1-p_2)N}\times d_2(t)}{\sqrt{N}}, \tag{4.36}$$

$$Y(t) = \sqrt{p_2}\times e_2(t) + \sqrt{(1-p_2)}\times d_2(t). \tag{4.37}$$

It was already mentioned that $e_2(t)$ does not change when time shifts from t to $t+1$ and, as so, it follows that (for even t) $Y(t+1)$ can be written in terms of $Y(t)$ and $d_2(t)$, as suggested by equation (4.38):

$$Y(t+1) = Y(t) - \sqrt{(1-p_2)}\times d_2(t) - \sqrt{(1-p_2)}\times d_2(t), \text{ if } t \in 2\mathbb{N}. \tag{4.38}$$

At this point, an *archaic* (and still incomplete) form of the generator can already be designed: if t is an even number, sum and normalise $e_2(t)$ and $d_2(t)$, which are occurrences of two independent Gaussian variables; if t is odd, get the previously generated point, and reverse the effect of the inconstant part given by $d_2(t)$. A generator like this would conserve

the scaling properties until the scale of 2 only. The *weights* of the components would be given by $\sqrt{(1-p_2)}$ and $\sqrt{p_2}$.

Passing Through 4

The previous progression shows that it really is possible to decompose the self-similar process into constant and mutable parts. Lets take that demonstration one step further, and consider now the scales of 4. The reader may notice that a special attention is given to the scales that are powers of 2. This is mainly due to the two following motives: first of all, the development of 4SG was influenced by the way fBm-SGA was thought and developed, and second of all, it was already shown that the relations between points following that scales scheme are easier to *work out*. For now, consider that t is a multiple of 4 ($t \in 4\mathbb{N}$) and that the process is *preparing itself* to transit from $t+1$ to $t+2$. According to what was previously said, during the time shift from t to $t+1$, approximately $p_2 \times N$ of the $S_i(t)$ remained equal, while the others shifted their sign. This means that, after being *aggregated* for the scale of 2, approximately $p_2 \times N$ of the $S_i^{(2)}(t)$ processes will be equal to 1 or -1, and the rest will be equal to 0, as depicted by (4.39). The same holds for the transition between $t+2$ and $t+3$:

$$S_i^{(2)}(t) = \frac{S(t)+S(t+1)}{2} = \begin{cases} S(t) & \text{if } S(t) = S(t+1) \\ 0 & \text{if } S(t) = -S(t+1) \end{cases} . \tag{4.39}$$

The $(1-p_2) \times N$ processes, that change their value inside the scale of 2, have fundamentally *nothing* to say about the self-similar properties of the underlying process for other scales which are powers of 2 (because their sum (aggregation) for those scales is equal to 0). Therefore, one just needs to assess how many of the $\{S_i(t)\}_{t\in\mathbb{N}}$ in $E_2(t)$ *must* remain equal and how many can change their sign during the time being of a scale of 4. It is critical to mention that the penultimate sentence is valid within the context of this explanation, since it applies to non-overlapping aggregation blocks.

A self-similar process should follow the same scaling rules, independently of the aggregation it has suffered. As the processes resulting from the aggregation for the scale of 2 of the walks in $E_2(t)$ are indistinguishable from the original $S_i(t)$ (i.e. $S_i^{(2)}(t) \overset{d}{=} S_i(t)$,

for $S_i(t) \in E_2(t)$), the very same probability p_2 that applies to the latter, during the transition from even to odd indexes, applies to the transition at hand for $S_i^{(2)}(t)$. Thus, for $t \in 2^2\mathbb{N}$, the total number of processes that *persist* after 2 shifts is given by $(p_2)^2 \times N$, and the number of processes that switch their value (from positive to negative or vice versa), from $t+1$ to $t+2$ is given by $p_2(1-p_2) \times N$.

As before, let $d_4(t)$ denote the normalised sum of the $p_2(1-p_2) \times N$ processes that change their value every two points *only*, and let $e_4(t)$ be the immutable part so far. Notice that (4.33) and (4.34) are consistent with the definitions of $D_4(t)$ and $d_4(t)$ also, as for $k=2$, $D_{2^k}(t) = \{S_i(\tau), i \in \mathbb{N} : \tau = \lfloor t/4 \rfloor, S_i(\tau) = S_i(\tau+1) \neq S_i(\tau+2) = S_i(\tau+3)\}$. $Y(t)$ is now defined as the contribution of 3 different components (equation (4.40)): $d_2(t)$, which reverts its value every 1 point; $d_4(t)$, which reverts its value every 2 points; and $e_4(t)$, which remains constant during all the transitions considered. $Y(t+1)$, $Y(t+2)$ and $Y(t+3)$ can now be obtained as indicated by equations (4.41), (4.42) and (4.43), where w_2, w_4 and w_8 are the *weights* identified so far, duly concretised in (4.44). By construction, the distribution of $d_2(t)$, $d_4(t)$ and $e_4(t)$ tends to the one of a Gaussian variable, as $N \to \infty$:

$$Y(t) = w_2 \times d_2(t) + w_4 \times d_4(t) + w_8 \times e_4(t) \; t \in 4\mathbb{N}; \tag{4.40}$$

$$Y(t+1) = Y(t) - 2 \times w_2 \times d_2(t), \tag{4.41}$$

$$Y(t+2) = Y(t+1) - w_2 \times d_2(t) + w_2 \times d_2(t+2) - 2 \times w_4 \times d_4(t), \tag{4.42}$$

$$Y(t+3) = Y(t+2) - 2 \times w_2 \times d_2(t+2), \tag{4.43}$$

$$\begin{aligned} w_{2^1} &= \sqrt{1-p_2} = \sqrt{(p_2)^{1-1}(1-p_2)}, \\ w_{2^2} &= \sqrt{p_2(1-p_2)} = \sqrt{(p_2)^{2-1}(1-p_2)}, \\ w_{2^3} &= \sqrt{p_2 \times p_2} = \sqrt{(p_2)^{3-1}}. \end{aligned} \tag{4.44}$$

$Y(t+1)$, $Y(t+2)$ and $Y(t+3)$ (with $t \in \mathbb{N}$) are all written in terms of their predecessors. As previously hinted, the value of $d_2(t)$ changes after 2 points, reason by which its contribution is deleted from (4.42), and the one of $d_2(t+2)$ is added.

(And) Generalising to Larger Scales

The three *weights* shown in the equation array (4.44) already uncover the pattern that defines the several contributions for scales that are powers of 2. In here, it will only be said that, if N_p is allowed to denote the (*finite*) number of contributions (one for each scale) to be taken into consideration, and k is allowed to denote the power of 2 of the aggregation scale, the several *weights* w_{2^k} can be generically defined using expression (4.45) and (4.46). These values result naturally from the iterative application of the reasoning described in the previous subsection, for scales of aggregation that are larger than 4:

$$w_{2^k} = \sqrt{(1-p_2)(p_2)^{k-1}}, \text{ if } k < N_p, \tag{4.45}$$

$$w_{2^{N_p}} = \sqrt{(p_2)^{N_p-1}}. \tag{4.46}$$

The generalisation of the algorithm follows from the same iterative procedure, but before continuing into that, recap the most important facts of the previous explanation. It was said that it is possible to approximate a self-similar process by decomposing it into *several* components $d_{2^k}(t)$, that shift their contribution, from positive to negative, in different moments of time. Additionally, it was argued that the components that shift their contribution, *within* a given aggregation scale, have *no active voice* in what concerns the self-similar properties of the aggregated processes for higher powers of 2, which basically means that, once a given contribution $d_{2^k}(t)$ has fulfilled its purposed inside an aggregation block of size 2^k, it may be safely substituted by another value. The *weights* of each one of the $d_{2^k}(t)$, with $k = 1, ..., N_p$, were already defined.

At this point, the formal description of 4SG can be effectively completed using the

following set of expressions:

Start with

$$Y(0) = \sum_{k=1}^{N_p} (w_{2^k} \times d_{2^k}(0)), \text{ where}$$

$$w_{2^k} = \sqrt{(1-p_2)(p_2)^{k-1}}, \text{ if } k < N_p,$$

$$w_{2^{N_p}} = \sqrt{(p_2)^{N_p-1}},$$

For each $k = 1, ..., N_p$, set

$$d_{2^k}(0) = G(0,1), \text{ and}$$

$$s_{2^k} = 1.$$

To get $Y(t)$, $t > 0$, do the following

$$Y'(t) = Y(t-1) - \sum_{k=1}^{\min(N_p,K)} \left(s_{2^k} \times w_{2^k} \times d_{2^k}(t-1)\right),$$

For each $k \leq \min(N_p, K)$, set

$$s_{2^k} = -s_{2^k}.$$

For each $k \leq \min(N_p, K-1)$, set

$$d_{2^k}(t) = G\left((2p_2-1) \times d_{2^k}(t-1), 4p_2(1-p_2)\right),$$

$$Y(t) = Y'(t) + \sum_{k=1}^{\min(N_p,K)} \left(s_{2^k} \times w_{2^k} \times d_{2^k}(t)\right).$$

To get K, start with $K = 1$, increment $K \in \mathbb{N}$, until

$$t \bmod 2^K = \frac{2^K}{2}, \text{ for each } t \in \mathbb{N},\ t > 0.$$

The algorithm deals with the possibility that, at a given iteration, K may be larger than the number of contributions declared in the initialization of the procedure. In such cases, *all* $d_{2^k}(t), k = 1, .., N_p$ are renewed simultaneously, bringing the longest dependence to an end. The only dependencies that are preserved are the ones implied by (4.47), which is part of 4SG:

$$d_{2^k}(t) = G\left((2p_2-1) \times d_{2^k}(t-1), 4p_2(1-p_2)\right). \tag{4.47}$$

This type of construction was already known from the *inner workings* of fBm-SGA, being responsible for guaranteeing the self-similar properties for the scales of 2 for a given

series generated using the last mentioned method. In the case of 4SG, the same equation is collateral, as it is not a crucial factor in the assurance of the self-similar properties for the scales which are powers of 2. On the other hand, as it was said that $d_{2^k}(t-1)$ could be replaced by $d_{2^k}(t)$ after having contributed to the generation of the series, it seems more logical to keep updating these values according to rules of persistence or anti-persistence. Nonetheless, this collateral factor benefits 4SG, in regards to fBm-SGA, since the self-similar properties can also be enforced for scales of type different than 2^k.

It is obvious that a computational implementation of 4SG depends on the capability to produce sequences of high-quality pseudo random numbers following a Gaussian distribution. This dependence however, is not as critical as it is for fBm-SGA. That is why the last main section of this chapter is dedicated to the subject of generation of pseudo random numbers.

One may notice that the representation in Figure 4.13 follows closely the description of the algorithm, at least till the scale of 16. Nonetheless, the values of the *weights* in the figure are not equal to the deduced ones. In there, the several *portions* are written in terms of probabilities, instead of variances (which are the real *weights*), for the sake of clarity.

The most interesting remark that came out of the design of 4SG is the *ambivalence* of the expression given by 2^{2H-2}, where H is the Hurst parameter. Herein, this expression is interpreted as a probability that ranges from 0.25 (exclusively) to 1 (exclusively), when H varies from 0 (exclusively) to 1 (exclusively), that defines the resemblance between points and aggregated blocks. Alone, it defines all the *weights* of the several contributions for the scales of 2 and the means to update the values of $d_{2^k}(t)$. The *simplicity* of this important quantity has also influenced the designation given to the generator.

4.4.2. Quality Assessment via Hurst Parameter Estimation

To obtain a confirmation of the precision of 4SG (and to later use it as traffic generator), the algorithm was implemented in the object oriented programming language Java. To provide the quality assessment and the computational efficiency analysis with

a baseline comparison, all the tests that were performed to the proposed generator were also conducted for other two generation procedures, well known from the literature. The Hosking and the Wavelets-based on FFT generators were chosen for this analysis (and implemented in the aforementioned computer language) because of their distinct characteristics: the first is exact while the latter is approximate; the first requires $O(n^2)$ computations, while the calculation complexity of the second is of $O(n \log(n))$.

As in section 4.3.2., the accuracy of 4SG was assessed by simulating several data series with different Hurst parameter values, and by submitting the latter to different Hurst parameter estimators. The methods chosen to participate in this analysis were: MEBP, as described in section 3.2.1.; MVT, as described in section 3.2.2.; DFA, as described in e.g. [67, 86]; RS, as described in [38]; Absolute Moments Time with $n = 1$ ($\text{AMT}_{n=1}$), as described in [38]; and the retrospective version of the AV estimator, as described in e.g. [38, 91].

As the reader may notice, the accuracy assessment of 4SG was made by using more estimators than for fBm-SGA, mostly because some of the methods used in the quality evaluation of fBm-SGA are not well suited for the analysis of series exhibiting anti-persistent behaviour (namely MEBP and RS). One of the novelties of the analysis is the usage of the $\text{AMT}_{n=1}$ method, which basically explores the expansive (spatial) properties of the generated series. Another novelty of the evaluation is the usage of the AV estimator, which was implemented according to some of the specifications in [91], for the Daubechies 10 family of Wavelets.

The large set of results obtained from this multifaceted simulation is going to be divided into three major parts, depending on the generator (Hosking, Wavelets-based or 4SG) that indirectly originated them. No log-log plots will be included in this section, because they where already included in different sections of chapter 2. The reader is invited to (re)visit that chapter, and to notice that each one of those log-log charts concerns the analysis of series with Hurst parameter values equal to 0.3, 0.5 and 0.7, that could only have been created resorting to 4SG, or to other algorithm capable of forging anti-persistence, but never via the usage of fBm-SGA. In this particular case, the reader may be certain that they are one of the results of the analysis to the series that 4SG has generated. No figures of sequences with different Hurst parameter values are going to be

included either because, this time, the explanation is to be focused on the results solely.

The simulation apparatus can be described as follows. Each generator was asked to output 50 series of points, for each one of the Hurst parameter values ranging from 0.01 to 0.99, with step equal to 0.02. The newly created series were then taken as *inputs* of the implemented Hurst parameter estimators, and processed in conformity. The average and the variance of the 50 estimates of the Hurst parameter were then calculated and stored in suitable records. These last two steps (statistical treatment and storage of the values) were repeated for each possible combination of estimator / generator. The results were finally plotted against the expected values of the Hurst parameter, for comparison reasons.

Because the Hosking and the Wavelets-based generators had their own computational limitations (memory and time consumption related), the length of the analysed series was dependent on the generator used: the sequences generated by the Hosking method contained 2^{17} points, and the sequences produced by 4SG or by the Wavelets-based method were 2^{19} points long. It is worth to mention that, despite the manifestly smaller size of the sequences produced by Hosking, the simulations concerning the latter were still the ones that took more time to be performed. A total amount of 58982400 points were generated and analysed. The configurable estimators were all set to start aggregating at the base scale of 2, with a multiplication factor of 2. For the Hosking related simulations, the maximum considered scale was of 2^{12}, and for the remaining ones it was of 2^{14}. For some of the methods (RS and DFA), some of the initial scales were not considered in the fitting procedure. The δ parameter of MEBP was adjusted to 2. The simulations were run on several computers: two Personal Computers (PCs) and four servers. One of the PCs is a desktop computer with 512 MB of RAM, running at 2.8 GHz. The other is a laptop with 992 MB of RAM, and a dual core processor running at 1.61 GHz, each. The four servers contain the Intel Pentium D processor, performing at 3 GHz, and 2 GB of RAM. All the machines were running the *Open Source Eclipse Integrated Development Environment* on top of the Microsoft Windows XP (with Service Pack 2) during the simulations.

The next two subsections contain the results obtained for the Hosking and for the Wavelets-based generator, while the quality analysis of 4SG is postponed to the third

subsection.

The Hosking Method

The Hosking method is a retrospective technique for the simulation of self-similar processes. Its definition enables it to be classified as an *exact* method, and the sequences it produces are perfect to test the estimation capabilities of the estimators or assess their behaviour for different Hurst parameter values. Unfortunately, its quality is counterbalanced by its computational complexity ($O(n^2)$), which renders the generation of moderately long sequences of points a tedious task, and makes any long-term simulation impractical (see section 4.4.3.).

The charts in Figure 4.14 and the values in Table 4.3 summarise the results obtained for the Hosking method. To enhance visibility, it was decided to separate the results for the several estimators in two different plots, based on a precision criteria. The estimators that returned values closer to the expected ones are represented on the right side of Figure 4.14, while the others are positioned on the left. The values in the table represent a smaller subset of the results, but contain additional information about the variance (in brackets) of the estimates for each one of the assessment methods.

Table 4.3: Target and (average of) estimated Hurst parameter values for the sequences generated using the Hosking method. Each cell contains the average and the variance (in brackets) of the results of 50 simulations.

Hurst	**MEBP**	**MVT**	**DFA**	**RS**	**AV**	**AMT**$_{n=1}$
0.01	0.16(4.14 E-03)	0.01(3.48E-05)	0.07(1.23E-07)	0.13(1.03E-05)	0.19(2.20E-04)	0.01(8.07E-05)
0.11	0.24(4.10E-04)	0.11(4.21E-05)	0.15(1.14E-06)	0.21(1.84E-05)	0.06(1.19E-04)	0.11(5.29E-05)
0.21	0.30(2.25E-04)	0.21(4.08E-05)	0.24(3.69E-06)	0.29(2.36E-05)	0.19(8.40E-05)	0.21(1.07E-04)
0.31	0.37(2.27E-04)	0.31(3.57E-05)	0.33(4.47E-06)	0.37(2.91E-05)	0.30(4.63E-05)	0.31(8.49E-05)
0.41	0.45(1.79E-04)	0.41(3.04E-05)	0.42(7.23E-06)	0.46(2.57E-05)	0.41(6.62E-05)	0.41(3.05E-05)
0.51	0.55(6.87E-05)	0.51(3.73E-05)	0.51(1.05E-05)	0.54(4.71E-05)	0.51(5.30E-05)	0.51(8.09E-05)
0.61	0.64(5.26E-05)	0.61(3.82E-05)	0.60(1.47E-05)	0.63(5.66E-05)	0.61(9.04E-05)	0.61(1.07E-04)
0.71	0.73(4.60E-05)	0.71(3.87E-05)	0.70(1.21E-05)	0.71(7.71E-05)	0.71(8.01E-05)	0.71(9.50E-05)
0.81	0.82(2.84E-04)	0.80(1.21E-04)	0.79(1.49E-05)	0.79(7.89E-05)	0.81(7.14E-05)	0.80(1.20E-04)
0.91	0.91(4.18E-04)	0.88(1.32E-04)	0.89(1.78E-05)	0.86(6.31E-05)	0.91(9.29E-05)	0.88(1.61E-04)

As previously stated, the Hosking method is an excellent benchmark in terms of

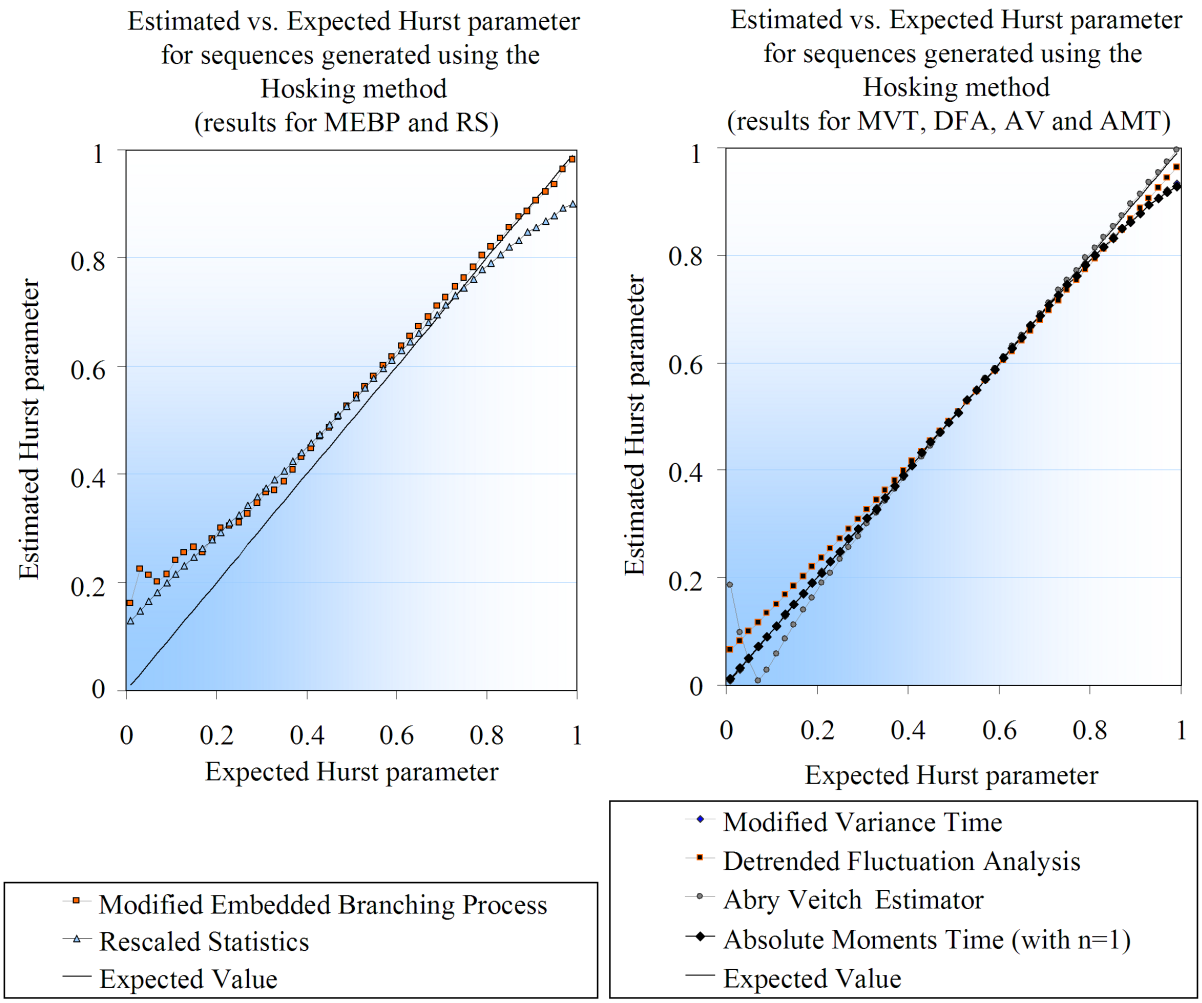

Figure 4.14: Charts where the estimated Hurst parameter values are plotted against their expected value, for different estimators. In this particular case, the data series were generated using the Hosking method. The chart on the left reflects the results for the estimators that performed *worst* (generally speaking); the chart on the right reflects the results for the estimators that were (generally) *closer* to the expected values.

quality. Therefore, this method can be used to test the estimators. By observing the charts in Figure 4.14, one can see that the MVT and the $\text{AMT}_{n=1}$ estimators assess the low Hurst parameter values with superb accuracy, whereas the DFA and the AV estimators start matching the expected Hurst parameters around the values of 0.3 and 0.2 respectively. Up until the Hurst parameter of 0.8, these four methods follow the line given by $y = x$ very closely. The divergence occurs when the Hurst parameter surpasses the said value. After 0.8, DFA, $\text{AMT}_{n=1}$ and VT tend to underestimate the *real* value of the Hurst parameter with almost the same level of inaccuracy. The best results for the high values of Hurst are clearly obtained by the application of AV.

The two remaining estimators, MEBP and RS, supply poor assessments for low Hurst values. MEBP is affected by the incremental normalisation procedure, and it only starts returning good estimates for the values above 0.6, whereas RS only manages to approach the expected Hurst values in the interval $]0.6, 0.8[$. For this reason, the values obtained using these estimators are plotted on the left side of Figure 4.14.

To sum up, MVT, DFA, AV and $\text{AMT}_{n=1}$ methods assess the Hurst parameter well over the bigger part of the possible Hurst parameter values, with the MVT and the $\text{AMT}_{n=1}$ achieving the best results for the smaller Hurst parameter values, and the Wavelets-based estimator for the higher ones. $\text{AMT}_{n=1}$ performed superbly, following the footsteps of MVT. It is curious to notice how close the values obtained by the two last mentioned estimators are, in Table 4.3. From the theoretical plane, this fact is not hard to explain, as both estimators capture the dispersion of the aggregated processes, though using different statistical means. RS proved itself to be a rather poor estimator (at least in this simulations), whereas MEBP can be relied upon for the Hurst values above 0.6. The conclusion that MEBP is not specially useful for the estimation of the Hurst parameter of *anti-persistent* processes is inevitable. The reasons for such to happen were mentioned before (see section 3.4.4.), and lie on the fact that the spatial span of the fBm decreases with the Hurst parameter, diminishing the statistical significance of the average number of crossings and, consequently, its ability to produce accurate results. The variance of all the estimates decreases as one moves from persistent to anti-persistent. If misunderstood, this small remark may lead to the conclusion that the precision of the estimators is inversely proportional to the Hurst parameter but, in fact, it just means that the statistical properties of the aggregated processes (or Wavelets decomposition), on which the several methods are based on, do not vary as much as the fBm tends to *white noise* ($H \approx 0$).

Wavelets-Based Generator

It will be proven afterwards that, in terms of computational performance, none of the implemented procedures is actually a match for 4SG (or fBm-SGA). Hence, the main motivation for the analysis of the Wavelets-based generator is to provide the accuracy evaluation with the perception of *what can be expected* from an approximate method. As

before, the whole set of results was partitioned into two charts (included in Figure 4.15) and one table (Table 4.4).

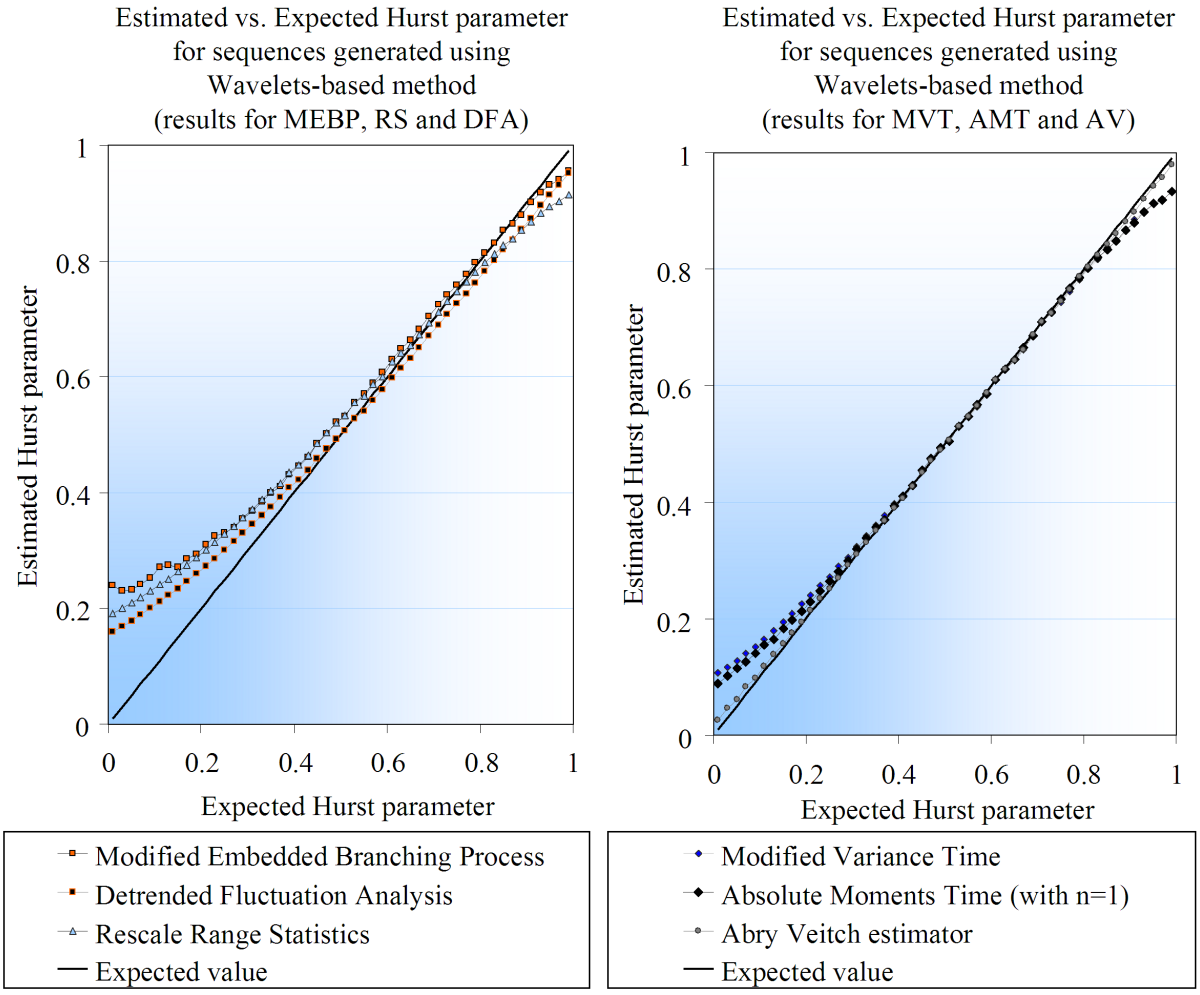

Figure 4.15: Charts where the estimated Hurst parameter values are plotted against their expected value, for different estimators. The data series were generated using the Wavelets-based method. Once more, the results were partitioned into two charts, being the estimates that most closely follow the line of expected values plotted in the chart on the right.

While the previous measurements may be perceived as tests to the quality of the estimators, here it is the quality of the method that is under the test. As expected, the most accurate estimates are achieved when using the AV estimator, since the philosophy behind both the generator (being the tested) and the estimator (being the tester) is the same. The estimated Hurst parameter is consistently good over the whole range of possible values, with some small discrepancies for low Hurst values. The estimates provided by MVT and $\text{AMT}_{n=1}$ are similar to those made for the Hosking method, with exception of the low Hurst parameter values. Below 0.2 and 0.3 (respectively), the estimators

Table 4.4: Target and (average of) estimated Hurst parameter values for the sequences generated using the Wavelets-based method. Each cell contains the average and the variance (in brackets) of the results of 50 simulations.

Hurst	**MEBP**	**MVT**	**DFA**	**RS**	**AV**	**AMT**$_{n=1}$
0.01	0.24(1.46E-03)	0.11(2.42E-05)	0.16(5.86E-07)	0.19(6.23E-06)	0.03(7.19E-05)	0.09(6.63E-05)
0.11	0.27(2.03E-04)	0.17(2.61E-05)	0.21(1.15E-06)	0.24(1.08E-05)	0.12(6.31E-05)	0.15(5.85E-05)
0.21	0.31(1.76E-04)	0.24(3.62E-05)	0.27(3.23E-06)	0.30(1.43E-05)	0.21(3.95E-05)	0.23(6.35E-05)
0.31	0.37(2.21E-04)	0.32(2.03E-05)	0.35(3.26E-06)	0.37(1.71E-05)	0.31(3.41E-05)	0.32(6.09E-05)
0.41	0.45(1.90E-04)	0.41(2.36E-05)	0.42(9.25E-06)	0.45(2.43E-05)	0.41(3.90E-05)	0.41(5.55E-05)
0.51	0.53(8.95E-05)	0.51(3.86E-05)	0.51(7.44E-06)	0.53(2.66E-05)	0.51(5.65E-05)	0.51(5.86E-05)
0.61	0.63(4.64E-05)	0.61(2.89E-05)	0.60(8.28E-06)	0.63(3.38E-05)	0.61(4.25E-05)	0.61(6.44E-05)
0.71	0.72(3.71E-05)	0.71(4.29E-05)	0.69(1.16E-05)	0.71(5.48E-05)	0.71(6.15E-05)	0.71(6.75E-05)
0.81	0.81(6.04E-05)	0.80(6.42E-05)	0.78(9.41E-06)	0.80(4.52E-05)	0.80(6.38E-05)	0.80(1.64E-04)
0.91	0.90(2.33E-04)	0.88(1.68E-04)	0.87(1.56E-05)	0.86(6.31E-05)	0.90(5.07E-05)	0.88(1.61E-04)

affirm that the generated series is less anti-persistent than the generator claims it is. For values bigger than 0.85, the deviation is somewhat similar to the one verified for the Hosking method, where the two last mentioned estimators tend to retrieve Hurst parameter values that are lower than the expected ones. MEBP, RS and DFA provide less accurate estimates for low values of the Hurst parameter but, while RS and MEBP start improving only after 0.5, DFA starts providing values close to the expected ones starting from 0.4. After AV, MEBP is the estimator that better *defends* the Wavelets-based generator for higher values of the Hurst parameter.

To sum up, the sequences generated by the Wavelets method exhibit similar behaviour as those obtained by the exact Hosking method, except for the low Hurst values, where the MVT and AMT$_{n=1}$ estimators demonstrate poorer results, and AV superior results than those obtained for the Hosking method. Since Hosking method is exact, it is more likely that the Wavelets-based generator shows diminished performance for lower Hurst values, than that the MVT and AMT$_{n=1}$ estimators are biased.

Simple Self-Similar Sequences Generator

Now that the behaviour of the different estimators is known (from the Hosking method simulations) and the vicissitudes of a *moderately* good approximate method have

been studied, it is time to analyse the proposed generator. As can be seen in the chart of the right side of Figure 4.16 and in Table 4.5, the estimators that favour most 4SG are AV, MVT and $\text{AMT}_{n=1}$. Notice that this is very similar to what happened in the previous section but, in this case, the related estimates seem to follow the line of expected values more closely than for the previous case. From this angle, 4SG seems slightly better than the Wavelets-based one. While the AV method returns estimates that are bigger than the expected ones for values of the Hurst parameter in the interval $]0, 0.32]$, the MVT and the $\text{AMT}_{n=1}$ return exact values for the same domain, and keep that degree of precision until $H = 0.79$. After 0.32, the AV estimator curve is completely indistinguishable from the line $y = x$, providing additional confirmation that 4SG is very accurate for those values.

Table 4.5: Target and (average of) estimated Hurst parameter values for the sequences generated using 4SG. Each cell contains the average and the variance (in brackets) of the results of 50 simulations.

Hurst	**MEBP**	**MVT**	**DFA**	**RS**	**AV**	**AMT$_{n=1}$**
0.01	0.28(7.30E-04)	0.01(2.44E-05)	0.18(7.96E-07)	0.22(5.78E-06)	0.06(7.73E-05)	0.01(6.93E-05)
0.11	0.27(5.75E-04)	0.11(4.19E-05)	0.22(1.69E-06)	0.26(9.21E-06)	0.14(7.70E-05)	0.11(6.79E-05)
0.21	0.31(5.05E-04)	0.21(4.24E-05)	0.28(3.42E-06)	0.32(1.22E-05)	0.22(4.38E-05)	0.21(6.05E-05)
0.31	0.37(2.99E-04)	0.31(2.09E-05)	0.35(3.99E-06)	0.38(1.35E-05)	0.32(6.69E-05)	0.31(6.10E-05)
0.41	0.44(1.79E-04)	0.41(3.09E-05)	0.42(5.58E-06)	0.46(2.03E-05)	0.41(5.25E-05)	0.41(4.60E-05)
0.51	0.54(8.16E-05)	0.51(3.22E-05)	0.51(5.56E-06)	0.54(1.96E-05)	0.51(3.96E-05)	0.51(4.20E-05)
0.61	0.63(6.00E-05)	0.61(4.91E-05)	0.60(8.77E-06)	0.62(3.47E-05)	0.61(5.00E-05)	0.61(7.01E-05)
0.71	0.72(5.66E-05)	0.71(3.77E-05)	0.69(1.34E-05)	0.71(5.84E-05)	0.71(2.89E-05)	0.71(8.45E-05)
0.81	0.82(2.46E-04)	0.80(8.39E-05)	0.79(1.97E-05)	0.79(5.94E-05)	0.80(5.05E-05)	0.80(1.48E-04)
0.91	0.90(5.59E-04)	0.88(2.59E-04)	0.88(4.09E-05)	0.85(1.37E-04)	0.91(1.19E-04)	0.89(1.49E-04)

The three remaining estimators behave like they have done for the Hosking and Wavelets-based methods, except that they start from higher estimates, when $H = 0.01$, than for the same value of the Hurst parameter in the previous sections. As the Hurst parameter increases, the estimators tend to return values closer to the expected ones. The RS estimator describes the exact same conduct for the three simulation scenarios: it overestimates the Hurst parameter for values smaller than 0.73, and underestimates it for the rest of the interval $]0.73, 1[$. This *tendency* is known from the literature [75] and, as so, it should not be used against or in favour of 4SG. MEBP and DFA clearly backup the capability of 4SG to simulate high quality persistent processes.

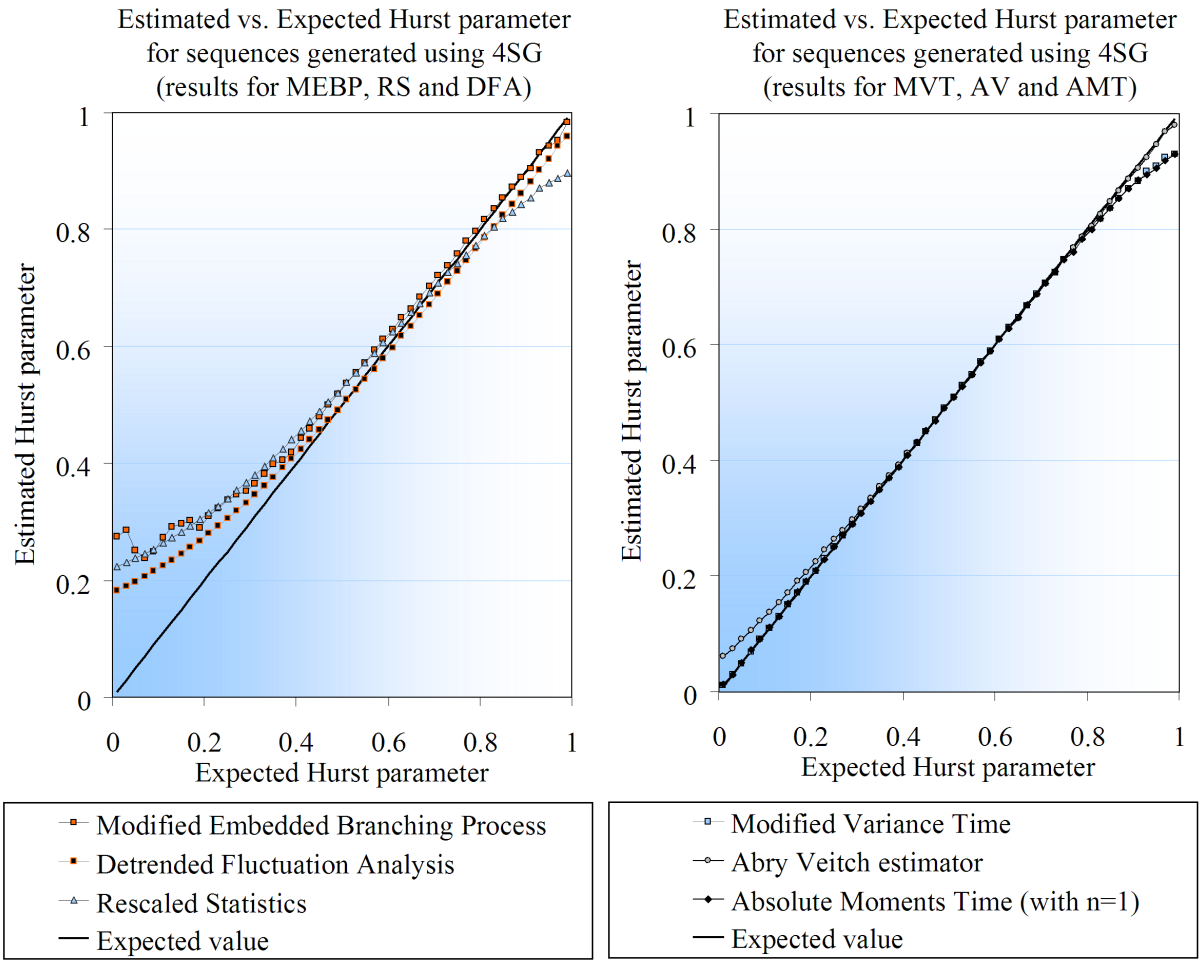

Figure 4.16: Charts where the estimated Hurst parameter values are plotted against their expected value, for different estimators. In this particular case, the data series were generated using 4SG. The chart on the left reflects the results for the estimators that performed *worst* (generally speaking); the chart on the right reflects the results for the estimators that were (generally) *closer* to the expected values.

All the results lead to the conclusion that, in terms of quality, 4SG is (at least) as good as the Wavelets-based method, but obviously not as good as the retrospective one. They also provide support for the conclusion that the reasoning that led to the development of the algorithm and the subsequent deduction of the weights of the contributions are correct. For some of the estimators, 4SG was actually performing near perfection (see chart on the right side of Figure 4.16). The next subsection brings focus to the aspect in which 4SG (and also fBm-SGA) truly distinguishes itself from the other generators.

4.4.3. Computational Performance and Memory Requirements of 4SG

This section aims to explore the computational performance of the three implemented generators, and contains a brief discussion about the memory requirements of 4SG.

Computational Performance of 4SG

The results concerning the computational performance of the three generators are summarised in the two charts shown in Figure 4.17 and in Table 4.6. The first chart (left side of Figure 4.17) displays the time required by the generators to produce a given series, against the number of points created. Because the computational complexity of the Hosking technique is of $O(n^2)$, the time it spends to create a process grows exponentially, and the logarithmic scale had to be used for the y-axis of that chart, so as to produce a readable plot. The second chart of the figure compares only the two fastest generators, 4SG and the Wavelets-based. The values represented in the charts and in the table are the average of ten measurements on the laptop with a dual-core processor, with X2 Mobile Technology TL-50, performing at 1.61 GHz, and with 1 GB of RAM. The OS installed on the computer was Microsoft Windows XP, with Service Pack 2. As before, the simulation scripts were written in Java, and run in the *Open Source Eclipse Integrated Development Environment*. It should be mentioned that up to 32768 points, some of the measurements returned values that were smaller than one millisecond for the 4SG. The values above 0 ms can probably be attributed to the instability introduced by the OS itself, since no other applications were running during simulation time.

Table 4.6: Time taken by Hosking, Wavelets-based, 4SG and fBm-SGA (for comparison) to generate self-similar processes with different lengths. The time unit is millisecond.

Number of points	256	1024	4096	8192	16384	32768	65536	131072	262144
Hosking	1.5	18.7	315.7	1265.6	4740.6	18936	98832.8	378920.3	1241156.5
Wavelets	160.9	159.4	189	423.4	456.2	1101.6	2898.4	6342.2	14065.6
4SG	1.6	1.6	3.1	6.3	12.5	23.4	50	100	200
fBm-SGA	0	0	3.5	8.75	19.25	43.75	82.25	157.5	322

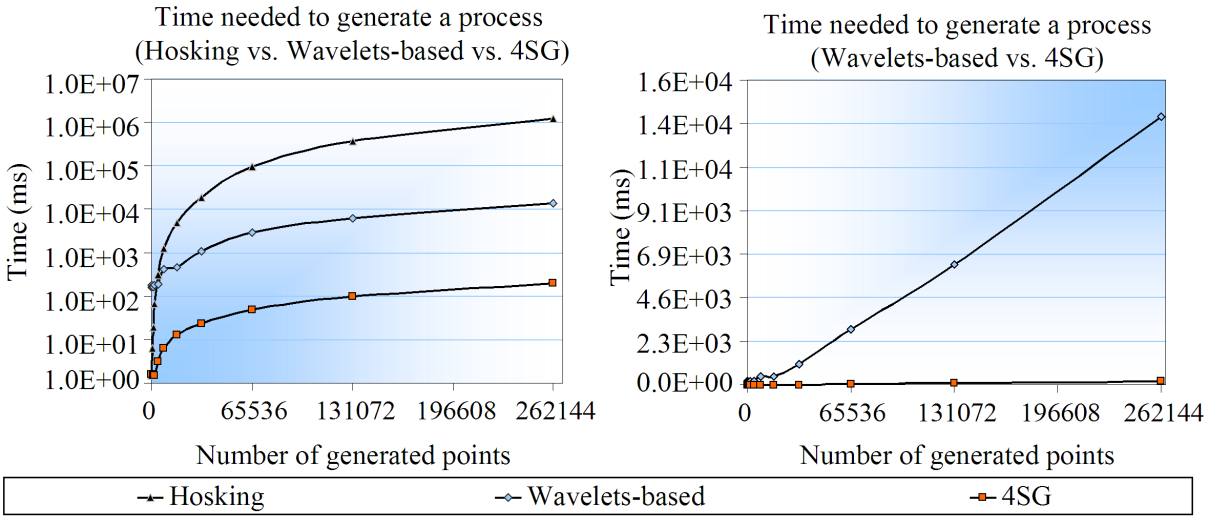

Figure 4.17: Performance comparison between different generators: on the left, the average time spent by Hosking, Wavelets-based generator and 4SG (in logarithmic scale) is plotted against the total number of points generated; on the right, the same analysis is conducted only for Wavelets-based generator and 4SG.

From the observation of the charts in Figure 4.17 and the values in Table 4.6, it may be concluded that 4SG is the fastest of the generators. The discrepancy between 4SG and the Hosking method is very high for the maximum number of points considered in the simulations: the Hosking method takes 20 minutes to generate a series of 262144 points, whereas 4SG requires only 200 ms. The difference between 4SG and the Wavelets-based generator is also noticeable, and specially emphasised by the chart on the right side of Figure 4.17. In the referred plot, the computation time of the Wavelets-based method *seems* to increase linearly, as the number of points generated increases. That observation is, nevertheless, misleading and owns itself to the representation only, because the computational complexity of this particular implementation is of $O(n \log(n))$, meaning that the points concerning the Wavelets-based method are better approximated by a $n \log(n)$ curve than by a line, and that the time spent by the generator increases (much) *faster* than the linear progression given by $y = n$.

These results deserve to be critically observed from the theoretical perspective. Notice that, as defined, both fBm-SGA and 4SG make use of a - lets say - *sub-algorithm* to get the value of the variable K, that indicates the *dependency from the past*, in the first, or the *number of components that have to change their contribution*, in the second. For each point to be generated, the number of operations to be performed is given by the

number K, deduced by means of (4.30) and (4.31). The average value of this variable defines the computational complexity of the algorithms.

Notice that 1/2 of the times, K is equal to 1 (and the algorithms perform 1 *composite operation*), 1/4 of the times, K is equal to 2 (thus, the algorithms perform 2 *composite operations*), 1/8 of the times, K is equal to 3, and so on. The average number of *composite operations* per point generated (C_p) is, therefore, given by equation (4.48), where N_p is the *number of precision levels considered* (if one is referring to fBm-SGA), or the *number of contributions* (if one is referring to 4SG):

$$C_p = \sum_{K=1}^{N_p} \frac{K}{2^K}. \tag{4.48}$$

The expression *composite operation* is used herein to denominate a *finite* and *constant* number of floating point computations (normally encompassing no more than 10 sums or multiplications). Because the sum in (4.48) tends to the *Gabriel's staircase series* which, by its turn, converges to 2 (see expression (4.49) and [118]), the order of complexity of the algorithm is (under *pessimistic* assumptions) given by $O(2n)$:

$$C_p \xrightarrow[N_p \to \infty]{} \frac{2^{-1}}{(1-2^{-1})^2} = 2. \tag{4.49}$$

This linear relation between the size of the series and the time used to generated them is perfectly seen on Table 4.6.

Within the approximate methods to simulate fGn (or fBm), 4SG constitutes a very good choice, presenting an excellent trade-off between precision and computational complexity. It is theoretically faster than any other method described in the literature, as the ones in [115] or in [112], claimed to run with a complexity of $O((n+2)\times\epsilon^{-2})$ and $O(w\times n)$ (respectively), where ϵ is an *accuracy parameter* in the interval $]0,1[$ and w is the size of the Discrete Wavelet filter. In the studies included in [115], the method was said to produce fairly good approximations of fGn for $\epsilon = 0.1$, for which the latter embodies a manifestly slower choice (at least 50 times slower). In [112], the several studies conducted for more than one type of Wavelets and filter lengths, led to the conclusion that the accu-

racy of the procedure is improved by the usage of Daubechies Wavelets (instead of Haar) and by choosing bigger filters (e.g with lengths of 16, 32 or 64). Nevertheless, the results presented in [112] show that the DWT based method is not particularly suited for the generation of highly persistent fGns. A good implementation of the method would be at least 16 times slower than the one presented herein.

The computational complexity of 4SG is not dependent of the quality of the approximations, and its improved dexterity is achieved by defining components that may last for long periods of time, rather than by means of *truncation* of the farthest past of the series. This fact comes as a direct consequence of the approach taken for devising the algorithm. 4SG was not the first algorithm the author has proposed and, as such, this second construction was the result of a *reverse engineering process*, inspired by an empirical knowledge of the problems that an implementation of this kind of algorithms may present. To economise the resources, the generation of self-similar sequences was observed from a perspective of continuity. The evolution of the processes in time is seen as the cadence of *simple transitions*, which can be efficiently coded in a small number of sums (apart from the operations used in the Gaussian numbers generator).

Memory Requirements of 4SG

4SG is extremely modest in terms of memory requirements, for they can be expressed in terms of a multiple of the integer part of $\log_2(n)-1$, where n is the number of generated points. Obtaining the *fixed* amount of variables required by the algorithm is the same of identifying the number of components that are *essential* to the generation of the referred quantity of values. The essential components are the ones that change their sign (their contribution) at least once during the execution of the algorithm (i.e. they are the ones whose longevity is smaller than the size of the synthesised sequence), since all the others remain constant during the entire span of the sequence. Notice that the longevity of the contributions defined by 4SG is given as a power of 2, and that their value may be renewed after being used twice (hence, the variable may be reused). In practice, this means that the *biggest* essential component of a series with n known occurrences, has the maximum duration of $2^K < n$, and that, consequently, one does not require more than $K = \lfloor \log_2(n) - 1 \rfloor$ variables to store the several values the referred components

may take. As an example, it could be said that a Java implementation of 4SG would take approximately 128 B of memory to produce a series with 2^{17} values on-the-fly (i.e. without having to store all the generated values in memory), while the Hosking method would require approximately 1 MB (8192 times more) to perform exactly the same task.

4.4.4. Usefulness of 4SG Within the Scope of the Thesis

At this moment, the author considers that the two major contributions that the development of 4SG has brought to this thesis are still difficult to grasp, but also that that will change in the following chapter. While the worth of fBm-SGA was proven even before its explanation, the worth of 4SG will only become clear when the time to generate self-similar traffic, inject attacks into the simulated traces, and interpret the results comes.

More than a fast algorithm to simulate fGns, 4SG constitutes a *personal* interpretation of self-similarity, which came in handy during the remaining part of the investigation. While the fact that this algorithm is capable of generating *anti-persistent* processes is completely irrelevant in the scope of this work, its capability to generate arbitrarily long self-similar series in an on-demand basis (while assuring a reasonably good quality of the outputs) is going to be largely explored to create traces of traffic in a packet-by-packet manner. This enables a more *realistic* emulation of an online analysis, avoiding having to store the traces before processing them. The decision of using this algorithm for the traffic generator is principally founded on the fact that 4SG preserves better the long-range dependencies than fBm-SGA.

4.5. The Source of Randomness: Prime Remainder Revolution Pseudo Random Number Generator

During the design and development of the previously described algorithms, the author was constantly wondering about the perfect means to provide them with the *pseudo random* data they require (i.e. series of *uncorrelated* values following a Gaussian distribution). The prerequisites to be fulfilled were, once more, *quality* and *low computational*

cost.

Motivated by the necessity to generate arbitrarily long sequences of pseudo random numbers, and suspecting that the Java in-built PRNG was not good enough for the task (specially in terms of quality), the author conducted a search for a high quality PRNG, which led first to the substitution of the previous generator by the well-known Mersenne Twister (MT) [119], and later on, to the creation of a new PRNG, entitled herein of *Prime Remainder Revolution Pseudo Random Number Generator (PRR PRNG)* or (more compactly) *Prime Remainder Revolution (PRR).* The proposed algorithm is not better than MT in terms of computational complexity, but it is equally capable of generating high quality sequences of pseudo random numbers. Because the author trusts on the quality of the newly devised and tested algorithm, it was used during all the experiments conducted for 4SG and for the simulation of the network traces. This choice was facilitated by the fact the proposed algorithm is only two and a half times slower than the fastest Java implementation of MT.

This section describes that part of the research work, by briefly explaining the main lines that define the novel PRNG, and the tests it was submitted to. At the end of the section, the preferred method to convert uniformly distributed values into occurrences of a Gaussian variable is also briefly criticised. Notice that the following description constitutes a simplified version of the one in [33].

Pseudo Random Number Generators

Computer-based simulation embodies nowadays a very attractive working tool for researchers. The biggest advantage of such simulations over *traditional* experimentation draws on their efficiency. Using a computer, one can simulate an enormous pool of *logically represented experiments* and analyse them. Transposing that procedure to real life requires may take too much time, rendering the empirical observation infeasible.

A PRNG is the procedure by which a computer creates an *apparently* uncorrelated and uniformly distributed sequence of numbers. As computer programs, PRNGs can only be considered within the limitations in which they are defined: their output is deterministic, periodic and depends of the initial value that initialises the procedure, typically

termed *seed* [120]. PRNGs constitute a rather attractive solution over real sources of random numbers, based on the transformation of physical events into a computer representation (see e.g. [121, 122, 123]), as they are often faster than the latter.

PRNGs are crucial components of any computer-based simulation, as they enable the virtual representation of the random conditions by which an event is affected in the real world. Because they constitute basic components of the simulation apparatus, a fault in terms of quality of randomness can result in a biased reproduction of a model, possibly leading to false or deficient conclusions. An analysis of some implementations of Linear Congruential Generators (LCGs), for example, concluded that for some parameters, an LCG epitomises a poor quality generator and that the simulations based on them can be wrong and should be re-evaluated [124]. PRNGs are also used in gambling machines, in online gambling games software, video games and computer security related applications. In the computer security field, PRNGs are often the source of *encryption keys*, *nonces*, *grains of salt* or *stream ciphers*.

Methods to create uncorrelated sequences of numbers exist since ancient times. Generation of random events was done with resort to simple techniques as dice throwing or coin flipping. The Middle Square Number PRNG, proposed by Von Newman in 1946 [125], is the first reference in the history of a computer-based PRNG, and embodies the perfect example of what was said previously for PRNGs. According to this method, the next (pseudo) random number of a sequence is formed by the middle numbers of the square of the previously outputted number. Since that date, many other true and PRNGs were proposed, used and studied in detail. The exhaustive enumeration of them all would not be useful in the scope of this thesis and, for the sake of this explanation, only the three PRNGs that somehow were used during the research work, are to be mentioned with more or less detail. Refer to [126], for instance, for an interesting qualitative analysis to a fairly big list of PRNGs and to [127] for more information on this subject.

One well-known family of generators is the aforementioned LCGs family of generators [128]. As the algorithm proposed in this section makes extensive use of this type of generators, a more detailed insight to their operational model is provided in one of the following subsections. Another popular reference when it comes to PRNGs is MT [119], developed by Matsumoto and Nishimura. Having the purpose of providing the reader with

a baseline comparison, it was decided to use a fast Java implementation of MT, available at [129], and run it in *parallel* with the algorithm proposed in this part of the thesis. As requested by the authors of the said PRNG, here it is stated that the procedure in [129] is the translation into Java of the original MT written in C++.

Another PRNG that is going to be used with the same objective as MT is the embedded Java PRNG, represented by the Java Random Class [130, 131], mainly because it did not involve any additional integration or implementation concerns and because it was the first source of randomness of 4SG and fBm-SGA. For a quick insight on the quality of this generator, refer to [132].

Evaluation of Pseudo Random Number Generators

The evaluation of a PRNG follows normally three major criterion: (i) the computational requirements, which includes the computational complexity of the algorithm and the amount of memory it requires; (ii) the quality of its outputs, which measures how close the procedure imitates *real randomness*; and (iii) its period, which is one of the most clear indicators of the limitations of the PRNG, and determines its usage in applications requiring long sequences of pseudo random data. While cryptographic applications favour the *unpredictability* of the produced sequences over the computational speed of the method, simple applications of PRNGs tend to require fast generators that behave well in a sufficiently large number of empirical tests [120, 126].

The speed of a PRNG can be assessed by deducing the computational complexity or by setting up a simple computational simulation scenario. Testing the quality of a PRNG however, is not as straightforward and requires one to submit the PRNG candidate to a battery of tests of randomness. The number of tools designed for that purpose reflects somehow the importance PRNGs gained along the years. *Knuth's collection* [133], *ENT- A pseudo random Number Sequence Test Program* [134], *pLab's Tests for Random Numbers* [135], *Crypt-X* [136] and the *DIEHARD Test Suite* [137] are examples of such tools or collection of tests. Some of them are available in the Internet in the form of downloadable packages, along with some of the qualifications an algorithm must possess in order to be considered a good PRNG.

4.5.1. The Prime Remainder Revolution Pseudo Random Number Generator

The main idea behind PRR is to use a set of normal LCGs instead of only one, structured in such a way that the generation is performed by choosing coordinates of a disordered table of numbers. In such conceptual reality, selection of random sequences of numbers is performed in 2 dimensions, instead of e.g. 1 (like LCGs) or 624 (like MT), although in a different sense than these last two. With this type of construction, the author aims to avoid some of the known (randomness) problems of LCGs [124], while keeping the computational complexity as low as possible (LCGs embody fast means to create sequences of numbers, specially for some combinations of their parameters).

After getting to know the theory of LCGs a little better, it will also be described how one can make use of the concept of constantly changing some of the parameters of LCGs in order to increase the period of the sequences produced and, in some way, improve its randomness. Take into consideration that the notation used in this section is self-contained (i.e. the definitions provided for mathematical symbols or concepts should locally superimpose any other made until this moment).

The origin of the expression *Prime Remainder*

Consider the formula given by (4.50), where R_n denotes the result of the modulo operation for a given index $I_n \in \{0, 1, ..., M-1\}$, and F, M are positive integer values respecting condition $F < M$. In the case of M being a prime number, the set defined by $\{R_n\}_{n=\{0,...,M-1\}}$ contains all the numbers between 0 and $M-1$, inclusively (this property can be derived from the combination of the *Euclid's lemma* and with the mathematical theorem known as *cancellation law*). These values constitute all possible *remainders* of M:

$$R_n = (F \times I_n) \bmod M. \tag{4.50}$$

It is not difficult to see that, in the case of having $M = 7$ and $F = 1$, for example, $\{R_n\}_{n=\{0,...,M-1\}} = \{0, 1, 2, 3, 4, 5, 6\}$. Ordering $I_{n=\{0,...,6\}}$ results in the ordered sequence

$[R_n]_{n=[0,\ldots,M-1]} = [0,1,2,3,4,5,6]$, and repeating the same procedure for $F = 3$ (for example), results in a differently ordered sequence $[R_n]_{n=[0,\ldots,M-1]} = [0,3,6,2,5,1,4]$. For any prime number M, one can always obtain a total of $M-1$ differently ordered sequences. Notice that $[R_n]_{n=[0,\ldots,M-1]}$ (with square brackets) denotes an *ordered sequence* and not a set.

Consider now that some of these sequences constitute the columns of a table, as depicted in Figure 4.18. By construction, this table will always contain all the possible remainders of M, dispersed through its lines and columns. Furthermore, the number of incidences of a given value is equal to the number of incidences of any other value of the referred structure. In terms of distributions, this means that one can obtain a uniformly distributed sequence of numbers just by randomly choosing coordinates of the table and returning the value to which they point to.

In Figure 4.18, it is also depicted how each number composing the table can be obtained. As can be seen, the referred structure does not need to be entirely stored in memory since its values can be retrieved by indicating the index of the line and the factor that originated the sequence displayed in one particular column. An ordered array containing all the considered factors (F) is thus sufficient to calculate every single value in the table. Because these factors constitute the origin of all the sequences in the matrix, they will be called *seed factors*, from this moment on.

The fact that the first line of the table contains only 0's (zeros) creates a sense of order, in regards to the whole structure. The *stratagem* schematised in Figure 4.19 and formalised in (4.51) can be used to lessen that sense:

$$R_n = (F \times I + c^2) \bmod M. \tag{4.51}$$

The square function summed to the left side of the modulo, in expression (4.51), can be safely substituted by any other non linear function of the column index (symbolised by a c) to achieve the same effect.

After the application of this additional step, the numbers on the table seem to be more randomly scattered. The author would like to bring the attention to the fact that

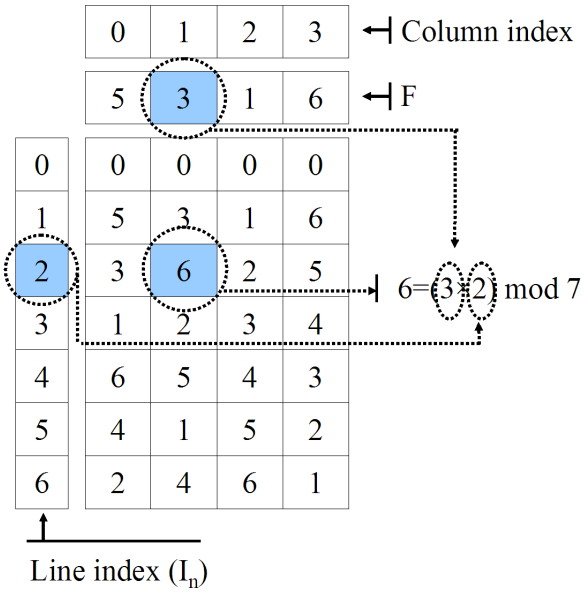

Figure 4.18: Example of a 7×4 table containing the remainders of the prime 7. The sequence of numbers in each column is the result of the application of $(F \times I_n) \bmod 7$ to different values of F (one per column), where I_n denotes the index of the line.

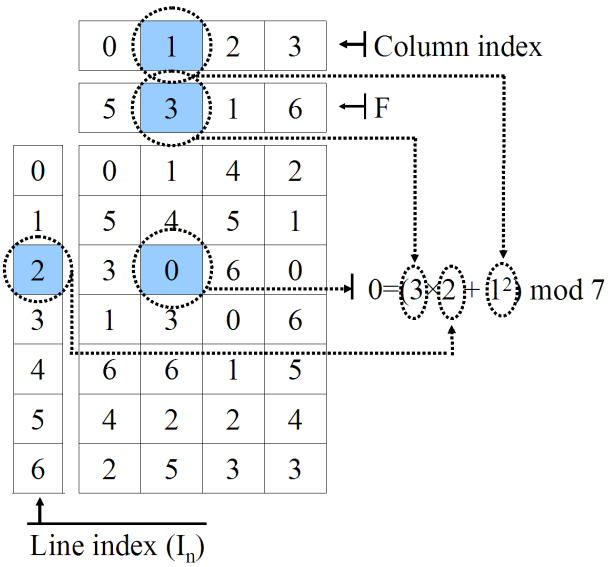

Figure 4.19: Example of a 7×4 table containing the remainders of the prime 7, disposed in an order different than in Figure 4.18. In this table, the sequences are obtained via the application of $(F \times I_n + c^2) \bmod 7$, where c denotes the index of the column.

the prime considered in the example is a Mersenne prime on purpose, as it will be shown that this choice impacts the performance of an implementation of the algorithm.

The name of the proposed PRNG is inspired in the means used to construct this

table of numbers (the numbers in the table are the *remainders* of a *prime*). Nonetheless, the procedure discussed in this section may be simply thought as a comfortable and efficient means to store and rapidly construct (meaning during *execution time*) a table with numbers, since the storage of the referred structure would result in impractical or onerous implementations for large values of M. Having this said, the next problem to be addressed is how to get pseudo random values of this structure, and that is exactly where the *old* LCGs are proven useful.

Linear Congruential Generators

An LCG is a PRNG based on the recursive formula given by (4.52), where X_n denotes the previously generated value and X_{n+1} the next value of the pseudo random sequence. A, B and M should be integer values respecting conditions (4.53):

$$X_{n+1} = (A \times X_n + B) \bmod M. \tag{4.52}$$

$$0 < A, B < M. \tag{4.53}$$

For a fixed M, the sequence of numbers produced via repetitive application of (4.52) is uniquely determined by A, B and by the positive integer number $X_0 < M$ used as the seed of the generator. For obvious reasons, an LCG only embodies a fairly good PRNG for some combinations of the parameters A, B and M (for example, if A, B and M are all even, the generator retrieves only even numbers, which are not pseudo random). Moreover, it can be proven that the period of a given LCG is maximum when conditions (4.54), (4.55) and (4.56) are met [128]. When such is the case, the period of the generator is M:

$$A - 1 \text{ is divisible by all prime factors of } M; \tag{4.54}$$

$$\text{If } M \text{ is a power of 2, } A - 1 \text{ must be a multiple of 4;} \tag{4.55}$$

$$B \text{ and } M \text{ are relatively prime.} \tag{4.56}$$

By construction, LCGs are incapable of returning the same value consecutively prior to its period has ended, which is in part related with the lattice structure of the sequence they produce. This was one of the limitations the author aimed to solve with PRR. The replication of each one the values of the space of occurrences in the *table of remainders*, combined with the procedure described below, enables the state of the generator to essentially evolve differently than LCGs, and output sequences that could not be attained using the last mentioned type of PRNGs.

Putting it Together: the PRR Algorithm

PRR relies on the extraction of numbers from the *table of (prime) remainders* which, in this case, is done by assigning the responsibility of choosing the column and the line indexes to LCGs. Observe Figure 4.20 for an illustration of these words, and notice that the randomness will be the result of the conjunction of two different factors: (i) the way the numbers are scattered in the table (which results from the selection of the *seed factors* at the initialization stage of the algorithm, and from the means used to scatter the numbers in the table, given by (4.51)); and (ii) the arbitrariness introduced by (at least) two LCGs working simultaneously. Through experimentation, it was concluded that, apart from conditions (4.54), (4.55) and (4.56), the parameters of the LCGs do not require any special treatment and they can contain different values every time the generator is initialised. This conclusion finds theoretical support in the previously described factors.

Using equations (4.51) and (4.52), and the notation introduced previously, the algorithm in Figure 4.20 can be formalised by expressions (4.57), (4.58) and (4.59), where (4.58) and (4.59) are LCGs with $B = 1$, sometimes referred to as *column* or *line selectors*, and $F(X_n)$ is the factor F that defines the sequence of numbers on a given column index X_n:

$$R_{n+1} = (F(X_{n+1}) \times W_{n+1} + X_{n+1}^2) \bmod M, \tag{4.57}$$

$$\text{Where } X_{n+1} = (A_1 \times X_n + 1) \bmod M_1 \text{ and} \tag{4.58}$$

$$W_{n+1} = (A_2 \times W_n + 1) \bmod M_2. \tag{4.59}$$

In the particular example depicted by Figure 4.20, M_2 and M are both equal to 7, and M_1 is equal to 4. M_2, M_1 and M do not have to exhibit any direct relation. However, as (4.59) defines univocally the sequence of indexes that are going to be generated (and their range), the value of M_2 should be either equal to M, or a transformation that maps the output of (4.59) into the desired range should be provided. Because of the same reason, M_2 cannot be smaller than M either (if $M_2 < M$, the values of W_{n+1}, retrieved by (4.59), would only cover a small portion of the interval $[0, M-1]$, and the congruence in (4.57) would not be capable of returning the entire sequence of *remainders* of M).

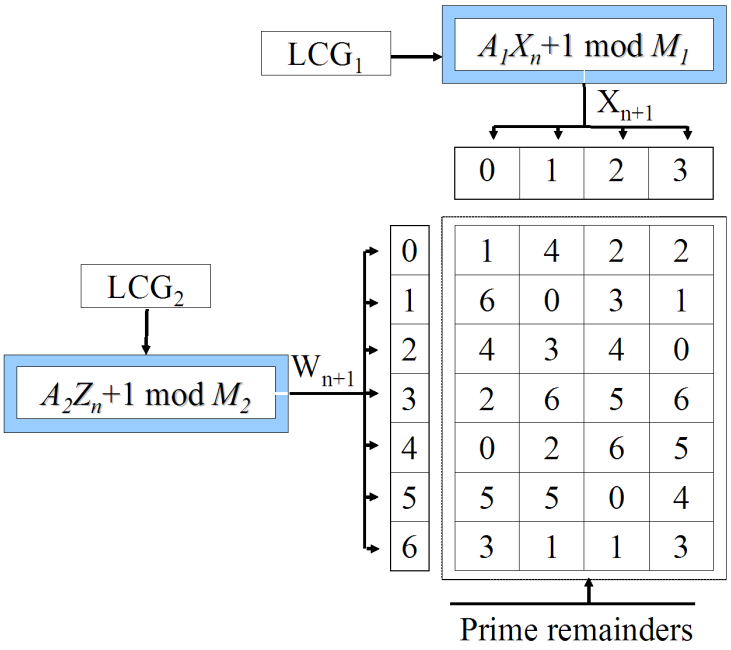

Figure 4.20: Graphical representation of the PRR algorithm: *the simple version*.

As the model adopted is not very stringent in terms of the parameters of the LCGs, offering the possibility to explore that aspect, the author decided to take advantage of that fact in the following manner. Consider that M_1 and M_2 are both powers of 2. In such case, the only condition the LCGs defined in (4.58) and (4.59) must met is that $A_1 - 1$ and $A_2 - 1$ must be multiples of 4. These numbers can therefore be created each time (4.58) or (4.59) are called or simply renewed frequently using *other* PRNGs.

Figure 4.21 illustrates the procedure by which each one of the LCGs depicted in Figure 4.20 is regularly updated by a herein termed of *master* LCG. The indexes of these

LCGs are intentionally different than n, to emphasise that these sequences may evolve at a slower rhythm than the one of the slaves. Notice that condition (4.55) is completely satisfied when $M_4 = \lfloor (M_2 - 1)/4 \rfloor$ and $M_3 = \lfloor (M_1 - 1)/4 \rfloor$ (where $\lfloor . \rfloor$ is, as before, the *floor function*), and that Figure 4.21 can be mathematically formalised by the following set of expressions:

$$R_{n+1} = (F(X_{n+1}) \times W_{n+1} + X_{n+1}^2) \bmod M, \tag{4.60}$$

$$\text{Where } X_{n+1} = ((4 \times Y_{i+1} + 1) \times X_n + 1) \bmod M_1, \tag{4.61}$$

$$W_{n+1} = ((4 \times Z_{j+1} + 1) \times W_n + 1) \bmod M_2, \tag{4.62}$$

$$\text{Update } Y_{i+1} \text{ by using } Y_{i+1} = (A_3 \times Y_i + 1) \bmod M_3, \tag{4.63}$$

$$\text{Update } Z_{j+1} \text{ by using } Z_{j+1} = (A_4 \times Z_j + 1) \bmod M_4, \tag{4.64}$$

$$M_4 = \lfloor (M_2 - 1)/4 \rfloor, \text{ and} \tag{4.65}$$

$$M_3 = \lfloor (M_1 - 1)/4 \rfloor. \tag{4.66}$$

These conditions present PRR under general terms only. In order make it more concrete, some specifications can be made at this point. For instance, the Mersenne prime $2^{31} - 1$, which is represented by 31 1s in the binary base constitutes a good choice for M (focus on expressions (4.68) and (4.69) and on the explanation that precedes them). $M_1 = 2^{10}$, $M_3 = 2^8$, $M_2 = 2^{31}$ and $M_4 = 2^{29}$ comprise particularly interesting choices also, if the computational performance of the algorithm is already being taken into consideration, since the modulo operation is easily optimized for values that are powers of 2. In such cases, the operation a mod b can be replaced by a logical *AND*, if the second input of the

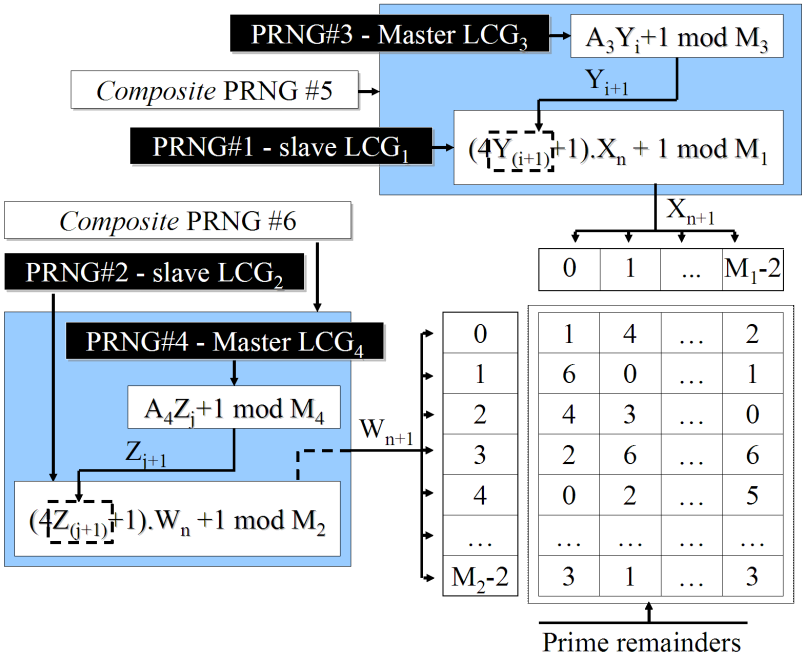

Figure 4.21: Graphical representation of the PRR algorithm: *the less simple version*.

congruence is substituted by its predecessor (equation (4.67)):

$$a \bmod b = a\&(b-1), \text{ if } b = 2^k, \text{ for some } k \in \mathbb{N}. \tag{4.67}$$

The modulo operation $a \bmod P$, where $P = 2^p - 1$ is a Mersenne Prime can be also optimized [138] using the procedure described by expressions (4.68) and (4.69), where the symbol $\gg$ denotes the bitwise shift operation and & denotes the bitwise operation AND:

$$a \bmod P = \begin{cases} d, \text{ if } & d < P \\ d - P, \text{ if } & d \geq P > 0 \end{cases} \tag{4.68}$$

$$\text{where } d = (a\&P) + (a \gg p) \text{ and } P = 2^p - 1. \tag{4.69}$$

The Period of PRR

The period of the proposed generator can be found by following a simple reasoning. Consider only the last mentioned version of PRR. For a fixed set of *seed factors*, the sequence of numbers produced by PRR depends directly from two different LCGs and

indirectly from another two. The number of iterations the algorithm takes until both line and column index selectors start repeating themselves simultaneously constitutes the period of PRR.

Without loss of generality and for the sake of simplicity, consider the composite generator defined by equation (4.70) and condition (4.71), where M is a power of 2. As mentioned in the previous section, PRR allows the modification of some of the parameters of the LCG, as long as condition (4.55) is respected. Consider now that the referred alteration is to be performed by the time the period of the selectors is reached, formalised as follows:

$$X_{n+1} = ((4Y_i + 1)X_n + 1) \bmod M, \text{ where} \tag{4.70}$$

$$Y_{i+1} = (A_3 Y_i + 1) mod \lfloor \tfrac{M-1}{4} \rfloor, \text{ if } n \bmod M = 0. \tag{4.71}$$

It is relatively simple to conclude that for each value of $[Y_i]_{i\in\mathbb{N}}$, the LCG (4.70) will generate a unique sequence of M numbers, before its formula is updated. Having this in mind, it is equally simple to infer that the period of the generator composed by LCG (4.70) and condition (4.71) is the product of the periods of the two LCGs constituting it. The exact value of the period of the *composite generator* is therefore given by

$$\lfloor M \times \frac{M-1}{4} \rfloor. \tag{4.72}$$

The period of PRR can never be smaller than the maximum period of the two composite generators it includes, because the sequences it produces only start repeating themselves after both generators start doing the same simultaneously. By construction, the composite line index generator is the one with better chances of having the largest period, since the number of columns of the *table of remainders*, M_1, is associated with the amount of memory required for the storage of *seed factors* and, as such, it has to be a relatively small value. According to this and to formula (4.72), the period of PRR for the aforementioned example (where $M_1 = 2^{10}$, $M_3 = 2^8$, $M_2 = 2^{31}$ and $M_4 = 2^{29}$) is approximately $2^{60} = 2^{31} \times 2^{29}$.

Seeding PRR

Seeding PRR corresponds to the adjustment of a *large* set of parameters. First of all, the *seed factors* have to be chosen. Ideally, all values should be different to enhance the arbitrariness of the *table of remainders*. The parameters (A_1 and A_2) of LCG_1 and LCG_2 depicted in Figure 4.21 must also to be chosen randomly, while taking into account conditions (4.54) or (4.55). Finally, the starting points of the four LCGs have to be decided.

Initializing PRR in the particular case where $M_1 = 2^{10}$, is the same of assigning $2^{10}+6 = 1030$ values to the parameters of the algorithm. This can be carried out minutely, or by letting another (weaker) PRNG fill up the critical parameters. Such PRNG can be another LCG, seeded with a unique value, in resemblance with what happens with the initialization procedure of MT (see [129]).

The initialization procedure of the implementation of PRR used in the scope of this research work, accepts a 64 bit long seed and divides it into two 32 bit long integers. Each integer seeds two instances of the LCG given by equation (4.73), which are then used to generate all the previously mentioned variables of PRR. The particular instantiation of an LCG in (4.73) is known to produce *reasonably good* pseudo random numbers from [139]:

$$X_{n+1} = (1664525X_n + 1013904223) \bmod 2^{32}. \tag{4.73}$$

4.5.2. Analysis of the Pseudo Random Number Generator

As previously explained, PRR is the result of a naive approach to the subject of PRNGs, which evolved later to a more serious research work. To test the algorithm in terms of the quality of the sequences produced, the author opted for the *Diehard Battery of Tests of Randomness*, which was developed by George Marsaglia, and is pointed out by many authors (see e.g. [140]) as a reliable means to test PRNGs. The mentioned set of tests has the advantage of being easy to use, since an executable file is available for download at [137]. After downloading and unzipping the package, the only thing one

has to do is to generate files containing a fairly long pseudo random sequence of numbers and write their name in the command line after the name of the executable Diehard file (e.g. `C:\Diehard_Dir\Diehard.exe rng_filename.dat`). The version of the software used herein already contains the *tough tests* described in [141].

The computational performance of PRR was estimated via empirical observation and analysis of a possible implementation of the algorithm on an object oriented language. The results for the computational speed and for the memory requirements apply solely to that implementation.

Tests of Randomness

Some of the tests of the Diehard battery require the file containing the random sequence of bits to have at least 270 Mb of data. As the considered implementation returns 31 bit long integers only, the algorithm was forced to generate at least 75497472 integers each time a file was to be tested.

In this section, it will only be included a brief summary of the results, obtained for the three considered generators. No detailed description of what each test does will be provided, since that falls out of the scope of this thesis. It will just be referred that the tests of randomness take advantage of statistical models the random bits in the file must follow, after being transformed into different representations. Normally, such resemblance is assessed using *goodness-of-fit* tests, which return *p-values* (probability values) that should be uniformly distributed in the interval $[0, 1[$. As in [126], a given test is tagged herein as *passed* if all the *p-values* within a test were greater than 0.01 and less than 0.99. If some of the *p-values* fall out of the aforementioned interval, the tests are repeated for another four times and considered as *passed* when the majority of the four tests are tagged as *passed*, according to the previous rule. Otherwise, the test is tagged as *failed*. Refer to [141] or to the documentation on [137] for more details on each individual test of the Diehard battery.

Table 4.7 assembles the results obtained for the tests of randomness. Its contents attest the quality of PRR and MT, in terms of randomness of the sequences generated, and at the same time, the failure of the Java PRNG in some of the tests, namely in the *tuff*

Table 4.7: Summary of the quality tests results for the three considered PRNGs.

Test	**PRR**	**MT**	**Java PRNG**
Birthday Spacings	passed	passed	passed
GCD*	passed	passed	passed
Gorilla	passed	passed	**failed**
Overlapping Permutations	passed	passed	**failed**
Ranks of 31x31 and 32x32 matrices	passed	passed	passed
Ranks of 6x8 Matrices	passed	passed	passed
Monkey Tests on 20-bit Words	passed	passed	passed
Monkey Tests OPSO†	passed	passed	**failed**
Monkey Tests OQSO‡	passed	passed	**failed**
Monkey Tests DNA	passed	passed	**failed**
Count the 1‘s in a Stream of Bytes	passed	passed	passed
Count the 1‘s in Specific Bytes	passed	passed	**failed**
Parking Lot Test	passed	passed	passed
Minimum Distance Test	passed	passed	passed
Random Spheres Test	passed	passed	passed
The Squeeze Test	passed	passed	passed
Overlapping Sums Test	passed	passed	passed
Runs Up and Down Test	passed	passed	passed
The Craps Test	passed	passed	passed

*Greatest Common Divisor (test) (GCD).

†Overlapping-Pairs-Sparse-Occupancy (OPSO).

‡Overlapping-Quadruples-Sparse-Occupancy (OQSO).

Gorilla test. This conclusion corroborates what was said in [132] about the last mentioned algorithm and, most importantly, these results substantiate the idea that ultimately gave origin to the algorithm presented herein.

Computational Performance of PRR

The PRR algorithm was implemented in Java programming language, following all the specifications from section 4.5.1.(including the optimization procedures). The following results apply to the optimized code of PRR of the *less simple* version described before. The experiments to assess the computational speed of the algorithm were conducted on a desktop computer, running the *Open Source Eclipse Integrated Development Environment* on a 2.4 GHz Pentium IV machine.

The chart in Figure 4.22 contains the comparative analysis results obtained by means of computer simulation. They concern the time a computer program implementing each PRNG takes to generate a sequence of 70000000 integer values. (The length of the sequence has to do with the number of values required by the Diehard battery of tests, and not with any other particular reason.) During execution time, no other applications were running on the machine (except for the *Eclipse Integrated Development Environment*) to reduce the possibilities of having the results biased by any unknown tentative to access the processor. Because of the same reason, the experiments were repeated 30 times and the average and variance were calculated and included in the plot. To overcome any possible initial instability, the first run of each experiment was not taken into account when calculating the aforementioned statistics.

PRR is represented by the upper line of the chart, which means that, in terms of computational speed, it is the slowest PRNG of the set of 3. The fastest of them all is MT, which takes only an average time of 2 s to produce the 70000000 numbers, performing at approximately 33777038 points per second. The implementation of PRR used in the scope of this work is thus 2.54 times slower than MT and 1.01 times slower than Java PRNG.

Notice that, in order to level all PRNGs implementations, an additional method that concatenates several outputs from PRR in such a way it produces 32 bit long integers

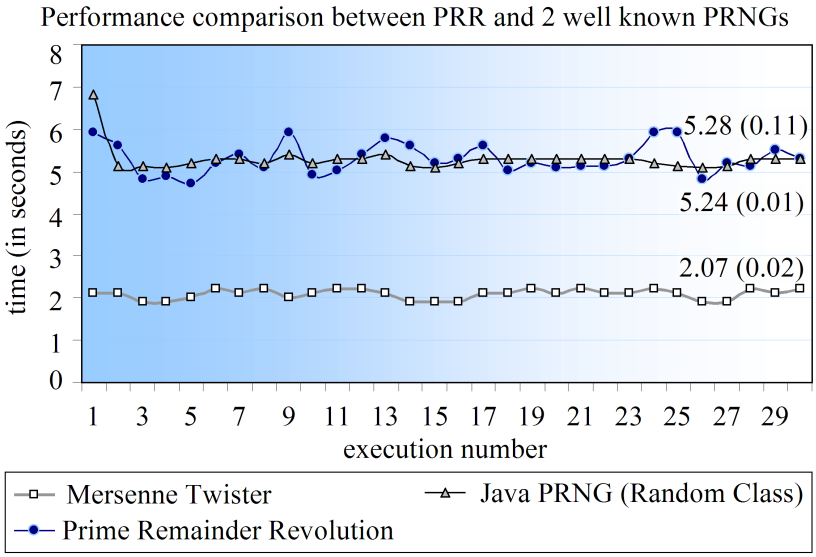

Figure 4.22: Time spent (in seconds) to generate a pseudo random sequence containing 70000000 integer values using PRR, Java Class PRNG and MT.

(instead of 31 bit long integers) is required, because the other two are already designed to output numbers with 4 bytes resolution. As the considered implementation of PRR is only capable of generating 31 bit long integers, generating 32 bit integers requires 1.03 more function calls than any of the other two. Using a different set of parameters should enhance the performance of the algorithm, by allowing longer bit streams to be generated, but it would certainly require the variables to be handled differently, since the longest native type in Java supports only 64 bit operations.

Memory Requirements

During some of the qualitative experiments conducted to the algorithm, the author of this thesis came to the conclusion that, in order to be able to produce high quality sequences of numbers, PRR requires the width of the table to be bigger than 512. The optimized version of the algorithm used herein utilises a total of 1034 integer variables (1024 *seed factors* plus LCGs variables plus local scope variables), represented by 1033 normal type *int* variables and by a type *long* variable. The computational representation of the *int* type in Java is 32 bits long, while the *long* type extends the representation for another 32 bits (64 bits in total). Therefore, generating pseudo random sequences of numbers using a Java implementation of PRR encompasses an inherent cost of, at least,

4140 B of memory. On the other hand, the considered MT implementation makes use of only 634 integer values, represented by the Java *int* type. That configuration requires solely 2536 B of memory. The third PRNG considered uses a completely negligible amount of memory, when compared with the other two, since it requires only storage capabilities for 3 integer values (12 B).

Notice that the number of *seed factors* is always given in terms of a power of 2. In this case, such fact is strictly related with the processing proficiency of potential implementations of the algorithm, since the arithmetic operations *modulo a power of 2* are more computational acquiescent.

4.5.3. The Generation of Normally Distributed Numbers

Now that the theory that supports the most primitive parts of the pseudo random sequences generator is exposed, it is time to describe the set of operations that enable the transformation of uniformly distributed numbers in a sequence of values with a Gaussian distribution. This transformation constitutes the link between PRR and each one of the previously described algorithms, since the latter are built on top of Gaussian Random Numbers Generators (GRNGs), and not directly over a Random Number Generator (RNG).

As brilliantly stressed out by the survey made by David *et al.* [142], a fairly big panoply of methods to replicate Gaussian variables is available in the literature. After investigating the several methods pros and cons, the author of this thesis decided for the implementation of the Polar method [143], not only because of its popularity and easiness of implementation, but also because it offers a good trade-off between complexity and quality [142]. Additionally, its implementation for the programming language in which all the procedures developed along this work were codified was already present in the class of the fast version of the MT generator, previously downloaded from [129].

The description of the viscera of the Polar method is not critical to this presentation, for its rationale can be found in the World Wide Web (WWW) or in the literature very easily (for example in [142, 143, 144]). Either way, the author would like to briefly discuss

its implementation, to provide closure to the subject of the computation complexity of fBm-SGA and 4SG.

The procedure that may be used to obtain a sequence of values following a Gaussian distribution from an uniform variable using the Polar method is described by (4.74), where $U(-1,1)$ denotes *an* occurrence of an uniformly distributed process, taking real values in the interval $]1,-1]$:

$$\begin{aligned}
&\text{Set} \quad s = 1 \text{ and} \\
&\text{While } s \geq 1 \text{ or } s = 0 \text{ do} \\
&\qquad X_1 = U(-1,1) \text{ and} \\
&\qquad X_2 = U(-1,1). \\
&\qquad s = X_1 \times X_1 + X_2 \times X_2, \\
&\text{Return } M \times X_1 \text{ and } M \times X_1, \\
&\text{Where } M = \sqrt{-2\frac{\ln(s)}{s}}.
\end{aligned} \tag{4.74}$$

The implementation of the algorithm in computer language is normally done by codifying the expression $X_1 = U(-1,1)$ as $X_1 = 2 \times U(0,1) - 1$, and by transforming the outputs of the PRNG into floating point values in the interval $]0,1]$. This last task is performed by dividing the pseudo random number with a predefined length n_b (number of bits), by the successor of the maximum value it may get (given by 2^{n_b+1}), and by casting the result into the *float* representation. Notice that the multiplication $2 \times U(0,1)$ incurs in the *infinitesimal* loss of information of one bit, as the mantissa of the number resulting from that operation will always be even (i.e. the less significant bit is equal to 0).

As can be seen in (4.74), the Polar method is capable of returning *two different* values of a Gaussian variable at the cost of a *finite* number of operations, as S is known to fall into the interval $[0,1[$ with fix probability $\pi/4$ [142, 143]. This means that the complexity of this method is of $O(4/\pi \times n)$.

Because the maximum resolution of the mantissa of a *double* type variable in Java is 53 bits, the implementation of this procedure in the referred object oriented language

requires the generation of (at least) 53 bit long numbers, which basically means that the particular instantiation of PRR described above has to be called approximately 1.71 times per point generated. It is possible to achieve the highest resolution possible if, instead of codifying X_1 as $2 \times U(0,1) - 1$, it is directly codified as $X_1 = U(-1,1) - 1$. That type of output is obtained by first constructing a 54 bit long integer via the application of the xOR operation to two 31 bit long integers (one of them shifted 23 times), and by (secondly) dividing the number by 2^{53}, casting it to a float variable. Notice that the floating point retrieved by this procedure has 54 bits of resolution (one more than when using $2 \times U(0,1)$). The values of X_1 and X_2 are obtained by subtracting 1 to the pseudo random uniform $U(-1,1)$. As a note of curiosity, this particular implementation of $U(-1,1)$ speeds up the referred computations in approximately 4%, when compared to the direct codification of $X_1 = 2 \times U(0,1) - 1$.

4.6. Conclusion

While the previous chapters of this thesis were focused on the estimation of the Hurst parameter of a given time series, this chapter brought the spotlight precisely to the reverse of the medal. After having enunciated the main reasons for this *change of perspective* (the necessity to test the modified estimators, and the need to simulate self-similar traffic), the most interesting features of a self-similar sequences generator were identified, by briefly discussing the several simulators of fBm proposed in the literature.

The directly proportional relation between computational complexity and the quality of most of the generators became clear after the overview of the state of the art. The description of two novel algorithms for synthesis of sequences with the approximate properties of an fGn was then included, proving that it is possible to create approximate long-range dependent series with good quality in a sequential and efficient manner, while avoiding having to store the entire past of the data series during the generation procedure, and preventing processing an increasing set of data, each time a point is to be produced. Contrarily to the methods that try to reduce complexity by *truncating the impact of the past to a certain extent*, the two proposed algorithms achieve the same purpose either by selecting particular points of the (arbitrarily long) past of the series, or by decomposing

them into a finite number of components that store the immutable part of the series for a virtually infinite period of time.

The first algorithm presented was fBm-SGA, which is inspired in the VT method, and is capable of producing arbitrarily long sequences with properties similar to the ones of an fGn exhibiting persistent behaviour (i.e. long-range dependence). The true novelty of the procedure resides on the so-called *persistence probabilities*, derived along its mathematical explanation. The said probabilities describe self-similarity as a *measure of a given point being equal to a point of the past* which, after all, constitutes a very intuitive way of seeing that property. The fBm-SGA was used on the experiments taken for the modified estimators. The results of those experiments may be found on chapter 3. The quality of the algorithm was assessed by using impartial estimators, implemented in their retrospective form (VT, DFA, EBP and RS).

While the fBm-SGA is only suitable for the generation of long-range dependent processes, the second algorithm presented is not. 4SG elaborates on a different ideology, by stating that the self-similar sequence can be decomposed into components that change their contribution from time to time and their weight on the overall *mass* of the process. The weight of the components depends on the Hurst parameter: the more self-similar the process is, the more substantial the contribution of the heavy-tailed components is. The real challenge behind this way of thinking was on the calculation of the weights of the components, which turned out to be all dependent from the value given by 2^{2H-2}. Within a fairly complex theory, this simple expression defines the relations of all the aggregation blocks with sizes that are powers of 2. Due to its reduced computational complexity and improved quality, 4SG was used in the simulations concerning self-similar traffic. The evaluation of 4SG was performed using six different Hurst parameter estimators (RS, DFA, MEBP, AV, MVT and $\text{AMT}_{n=1}$), being the results obtained for the AV, MVT and $\text{AMT}_{n=1}$ specially favourable, arguing clearly in benefit of the capability of the algorithm to generate sequences exhibiting persistent and anti-persistent behaviour. MEBP and DFA also agree with the remaining ones, but only for Hurst parameter values bigger than 0.6 (in the case of MEBP) and 0.4 (in the case of DFA).

The computational complexity of both algorithms was theoretically assessed as being of $O(n)$, and corroborated by empirical observation. The performance of 4SG surpasses

theoretically the performance of most of the exact or approximate methods in the literature, and outperforms by far the Wavelets-based generator used in the experiments described herein. The method is capable of producing approximately 1310720 points per second, while the Wavelets-based method would not produce more than a centesimal part of that number. The modesty of the procedures is also written in terms of the amount of memory they require, for they are satisfied with the space to store only a small multiple of $\log_2(n)$ variables, where n is the length of the series to be generated.

In the final part of the chapter, a PRNG was explained with detail, and tested using a well-known and *stringent* battery of tests of randomness. The development of PRR *per se* is a side effect of the objective to get the most reliable and fastest means to feed the simulation procedures of self-similar processes with random sequences of values following a Gaussian distribution. While PRR draws on a relatively simple rationale and trusts on the *old* LCGs to retrieve values from a table, during all the (endurance) experiments conducted by the author, it passed all the tests of randomness to which it was submitted. Even though its current implementation in Java programming language is not as fast as the optimized version of the well known MT (PRR is approximately 2.5 slower than the latter), its performance is not prohibitive at all, and the method was used during all of the experiments reported in this thesis. As the randomness of the outputs of PRR results from a combination of different factors, the initialization of the parameters of the algorithm can be carried out without many concerns. The same fact offers also the possibility to change the parameters of the column and line index selectors during runtime, which may be explored in such a way as to increase the period of the PRNG. Moreover, the design of PRR enables the adjustment of the quality of the produced series, via the specification of the width of the *table of remainders*.

The responsibility to transform the outputs of PRR into occurrences of a Gaussian variable is herein attributed to the mechanism known as the Polar method. During a brief discussion on how the procedure was implemented for the sake of this research work, it was shown that its computational complexity is in average of $O(4/\pi \times n)$.

Chapter 5

Traffic Simulation and Study of the Impact of Network Intensive Attacks

5.1. Introduction

When the initial guidelines of this work were drawn, the possible *static* definition of the behaviour of the self-similarity degree during a network attack was included as an objective. The problem however, turned out to be more fundamental than that, as such behaviour had still to be understood, before jumping to (perhaps) precipitate conclusions. During this chapter, it will become clear that the self-similarity degree is, in fact, affected by some kind of attacks and that such behaviour can be theoretically described. Nevertheless, this behaviour is written in statistical terms, which vary with the intensity of the attack and with the traffic load.

The following section of this chapter (section 5.2.) is dedicated to the delineation of the application scope of a possible IDS based on the analysis of the self-similarity degree of the network traffic. The discussion is directed towards the clarification of some of the limitations imposed by the nature of the approach. After that, a brief overview of the most important network based IDSs and Traffic Monitoring Systems (TMSs) is included. The enumeration of the several open source or commercial products is made so as to point out their most interesting features. A small section where the related works are analysed from a critical perspective is then included.

The detailed description of the means used to create legitimate traffic traces (at least

from the self-similarity point-of-view) constitutes the starting point for an in-depth analysis of the impact an attack has in the local scope Hurst parameter values. Though the pertinence for developing a simulator of network traffic and of attacks becomes clearer after the reports on the experiments conducted for the well known traces from the Massachusetts Institute of Technology / Defense Advanced Research Projects Agency (MIT/DARPA) data sets [145], included in the beginning of section 5.5., it was decided to describe the simulator in section 5.4., for the sake of the organization of this chapter. The type of attacks taken into consideration within the scope of this work and the way they were injected into the generated traces are also discussed in some of the subsections of that part. The study of the evolution of the local scope Hurst parameter values is then made by means of computer-based simulation, and discussed after the brief subsection dedicated to the MIT/DARPA data sets. Sections 5.5.2., 5.5.3. and 5.5.4. tackle the three possible general scenarios that the windowed-modified estimators may face during an attack. The subject of the *loss of self-similarity* is addressed in section 5.5.6., after the proposal of an anomaly detector based on the findings of the previously referred sections. A theoretical perspective of the results is then provided in section 5.5.7..

This chapter is partially based on papers [29, 30, 31].

5.2. Application Scope of an Intrusion Detection Method based on Self-Similarity Analysis

The main purpose of this section is to delimit the application scope of an intrusion detection tool based on self-similarity analysis within the IDSs general framework. The description of the most common ways to classify IDSs is, henceforth, structured so as to converge to the categories that best suit the referred approach, without going into too much detail.

An IDS is the network device or the software module specialised in the detection of security events. Its main functions include the inspection of files and logs, or of all inbound and outbound network activity that may indicate a network or system attack. The commercial attractiveness of this kind of systems is due to the increased importance

that security incidents are gaining, namely in the enterprise environment [146]. There are several ways to categorise an IDS [7, 8, 9, 10, 25], as further discussed in the following subsections.

5.2.1. Information Sources

In terms of *information sources*, an IDS can be classified as a *Network Intrusion Detection System (NIDS)*, if it monitors traffic from more than one network node (these systems are usually placed in or near a network gateway, router or switch); or as a *Host Based Intrusion Detection System (HIDS)*, which is often an user-end monitor software, that runs in the background of an OS checking for suspicious behaviour of applications, or unwanted modifications to key system files. More recently, the so-called *Application based Intrusion Detection System (AIDS)* [147] category was added to this particular way of classifying IDSs, so as to tackle the systems that use the *transaction logs* between applications, and the feedback obtained from the interaction with the application itself, to detect suspicious behaviour related with non-authorised attempts to gain root privileges (e.g. *user to root* non-authorised behaviour).

As previously stressed out (see section 2.3.2. of chapter 2), self-similarity is a property of the network traffic, and reaches its biggest expression in *popular* aggregation nodes, fed by the several tributes coming from the hosts or other aggregation nodes. As so, and from the *information sources* (or system location) perspective, the self-similarity analysis tool is restricted to NIDSs.

5.2.2. Analysis Approach

From the point of view of the *analysis*, an IDS can be classified as a *Signature-based IDS*, if it relies on the identification of sequences of bits that describe a known attack, being therefore connected to a typically large database, which is updated with new signatures of attacks every time a new threat is detected (or in a regular time spaced manner); or as *Anomaly-based IDS*, if it basis the detection of possible menaces to the integrity of the network, or system, in the extraction and analysis of standard features of

normal behaviour, and on the detection of suspicious deviations to that normality.

Signature-based NIDS, also known as *Misuse-based NIDS* [8, 25] are typically based on a complex protocol data unit filtering function. Every time one of those units of information reaches the NIDS, it is cross-checked with the signatures contained in the database to verify whether it contains any traces of a malicious program, or if a given succession of values representing the traffic match the ones of a previously identified attack. If the protocol data unit is considered part of an intrusion, it is discarded or submitted to further analysis. This operational model relies mainly on the assumption that most attacks have clearly defined signatures, and its efficiency (in terms of detection rate and computational performance) is dependent on the amount of information available for examination. In the case of network systems, packet inspection techniques and, more recently, DPI mechanisms [11] are used to achieve higher detection performance, for they enable the system to parse, organise and process the payload generated at the application layer of the TCP/IP architecture, maximising the amount of information subdued to signature matching. However, this kind of inspection can impose significant delay to traffic flow/system response functions.

Signature-based IDSs are typically updated by a central server or through a self-learning procedure. In the latter case, the IDS has the capability to auto detect unknown attacks and to infer fingerprints for them, updating its own database. Construction of new signatures can be done using some kind of heuristics (Bayesian, neural, artificial intelligence algorithms [148], etc.) but they are normally difficult to arrange automatically, and typically they require administrator intervention (either local or remote).

Some attacks are designed to bypass the intrusion detection or prevention system through exploitation of known network mechanisms, as fragmentation or end-to-end encryption, or through illegal manipulation of the information of the flows (e.g. spoofing the source address of the protocol data unit). Because of their operational model, systems based on packet inspection are particularly susceptible to encryption or obfuscation mechanisms [149], as the latter randomise the contents of the packets, frustrating any attempt to act upon their analysis.

The biggest disadvantage of Signature-based NIDS (and of Signature-based IDS, in

general) is related with the computational effort to which the system is subdued. Cross-checking an input stream of values, or a string of bits, with an increasing list of signatures may prove itself to be an inglorious task in the long term, specially when the proliferation of bandwidth intensive services and the standardisation activities of the 40 Gbps and 100 Gbps Ethernet technology [13, 14] promise to bring more data to the aggregation nodes. In spite of their disadvantages, the signature-based systems constitute the niche in terms of network security systems, and the DPI techniques are seen as the most reliable means to assure security and QoS to the users of a network, for they enable a more accurate management of the traffic [11, 149].

As previously said, Anomaly-based IDSs (a.k.a. *Behaviour-based IDSs*) [8, 148, 150, 151] draw on the definition of normality, and on the investigation of the best means to enhance the divergence from that normality to perform the task of identification of malicious activities. The philosophy of Anomaly-based IDSs is opposite to the one of Signature-based IDSs in the sense that the former strive for the codification of what is *good* in an information system, rather than aiming for the enumeration of *all* the possible instances of misconduct. Because of that, anomaly based mechanisms require less computational resources.

Within the scope of action of NIDSs and, more particularly, of traffic monitoring and analysis tools, the prime source of information are the protocol data units flowing through the devices. Hence, the anomaly-based detection methods have to be built on top of that constraint. The task of *profiling* draws on the specification of a mathematical model for one of more traffic aspects, which may either be extrapolated from observation of historical data, or explicitly given by the constraints of the system. The model may e.g. be the expected probabilistic distribution of the values representing the referred traffic aspect or a more complex conjugation of several rules of *good statistical behaviour*. In such case, an infraction occurs when the empirical distribution ceases to comply with the expected one. The normality may also be delivered in the form of a statement, being the developer the one responsible for the formalisation of the best means to verify the validity of the statement during online analysis.

An anomaly-based IDS has many advantages when compared with the signature-based IDSs, but also some drawbacks. Depending on how good the laws defining the

behaviour are, the system in question can suspect about *new* attacks without additional reconfigurations or updates. These systems calculate several statistics for a single or multiple packets and draw conclusions about their fitness by testing the resulting metrics: if they are out of a predefined confidence interval or surpass a given threshold, the traffic is considered abnormal and further measures should be taken. In theory, the better the traffic description model and its dimensions are, the better and more effective the developed system is. A greater share of the existing anomaly-based NIDSs use classic statistical metrics and counters for detection of traffic anomalies and draw often on heuristics when there is a need to dynamically update the values of the thresholds [9].

Some of the anomaly-based IDSs require the characteristics or the models to be adjusted before being used. In those cases, the learning phase is critical. Assuming that, during the learning phase for such a system, the latter is fed with a huge amount of abnormal data, the NIDS can become vitiated and potentially subverted. Additionally, if the system features a continuous adjustment loop and traces the current state of the network (or host) communications, a number of false attacks can lead the NIDS to alter detection thresholds, in such a way that a real attack passes unnoticeable or is classified as a false positive alarm. Also with the subversion in mind, some probes are intentionally dispersed in the time domain (e.g. Portsweeps), to dilute possible patterns. These last type of probing activities are called *stealth probes* [152].

A third type of *analysis approach* called *Stateful Protocol Analysis* is specified in [153]. It requires the IDS to understand the state machines of the protocol being used in the communications and the detection of an anomaly is performed by checking if the states (composed by one or more events and other significant information) are rightfully following the flow diagram of the model (e.g. the detection procedures may check if a given sequence of protocol data units is respecting the protocol state machines specified in a given Request For Comments (RFC)). The occurrence of a non listed state is indicative of a potential anomaly. Though this particular type of analysis is not applicable in terms of what is studied in this thesis (see next paragraph), it comprises a feature of many of the commercial IDSs described below.

A tool for intrusion detection inspired in real-time self-similarity analysis embodies a traffic characterisation mechanism, as the assumption of normality is placed upon statis-

tical properties of the network traffic. As those statistics apply solely to some of the most general aspects of the traffic (the bit count per time unit process), it should be emphasised that, actually, the method strives for *categorising traffic in the dark*. TCD techniques draw on less (and different) assumptions and rely on less information than other types of techniques. Their scope is limited to the information the flow collectors may retrieve, and to the header of the protocol data units under analysis [15], which travels in plain text more often than the payloads. The major drawback of these techniques is that their operational model can only be used to *suspect* about the illegitimacy of a given traffic segment or flow, and not always to point out the specific details of an attack. Therefore, while the traffic characterisation methods can pose an attractive choice because of their efficiency (the statistics can normally be calculated using low computational resources), they cannot be always considered as a complete detection mechanism alone. The task of intrusion detection based on self-similarity analysis is therefore circumscribed by the advantages and disadvantages of TCD.

5.2.3. Response Type

From the point of view of system activity, a NIDS can be considered as a *passive NIDS*, that logs and emits an alarm upon detection of a potential attack, or as an *active NIDS*, which responds to a potential attack by automatically triggering further investigation procedures, by disconnecting a user or by dropping suspicious packets. As the two categories do not necessarily exclude each other, there are also *hybrid response type systems* [9], which are capable of issuing actions belonging to both of the above mentioned categories.

As a consequence of what was said in the previous section, the response type of a NIDS built on top of the means for fast Hurst parameter estimation may be considered an *active response type system*, but its actions should be confined to the possibility to trigger further investigation procedures and emit alarms during a potential intrusion. As previously explained, the output of anomaly-based techniques (and more particularly of TCD techniques) should be carefully considered before starting to drop protocol data units. These methods should complement (or be complemented by) other detection techniques,

or if such is infeasible or not applicable, their classification should be submitted to human discernment.

5.2.4. Analysis Timing

A final remark regarding the *analysis timing* of the means used to assess the self-similarity degree may still be made before ending this section. While the examination of audit logs may be performed in a periodic or offline manner, real-time operation is of critical importance in network security equipment [9]. Thus, one of the major objectives of this thesis was to find the means that could enable the estimation of the Hurst parameter to be conducted in real-time. The discussion in chapter 3 proves that that objective can indeed be achieved, and that a possible IDS based on the fractal structure of the traffic would actually fulfil the prerequisites for an online implementation.

5.3. Overview of Open Source and Commercial NIDSs, and Critical Analysis of the Related Works

In the previous section, it was shown that the conditions where a tool for intrusion detection based on self-similarity analysis has the best chances of succeeding are met within the application field of NIDSs. This section contains the compilation of the results of a search for open source or commercially available NIDSs and TMSs (first subsection), and the critical analysis of the works that are more closely related with the subject of this thesis (second subsection). The initial search for the commercial security products was specially motivated by the fact that the work was being conducted inside a company developing network equipment.

5.3.1. Overview of Open Source and Commercial NIDSs

The main purpose of this section is to provide the subject at hand with a general perspective of which are the most common technologies employed by NIDS and TMS developers. The following list contains the description of some of those systems and of

their main features (a broader list of NIDSs may be found in [154]), but it does not include the detailed explanation of their inner workings. While the analysis approach behind some of the referred features is easily obtained from the description included in the previous section, others are not. Unfortunately, the details of a small part of them are not discriminated in the products documentation either, due to e.g. industrial secrecy policies.

1. Cisco has IDS and Intrusion Prevention System (IPS) solutions [155]. The vast portfolio includes systems for hosts and network aggregation points, as well as the systems or software modules for their centralised management, and correlation of the information coming from distributed security sensors. In [156], it is claimed that their solutions are capable of defending the networking devices against known and unknown threats like worms, network viruses and application level intrusions. The Cisco IPS is conceived for online operation, and it supports protocol parsing, IDS/IPS evasion protection, active response actions, active notification actions, web administration interface and additional support for secure communications between security systems (via compliance with the Internet Protocol Security (IPSec) framework), apart from application misuse, stateful pattern recognition, anomaly-based detection, heuristics and standard signature-based techniques (the complete list of features may be found on table 9 of the data sheet of the product in [156]). Cisco solutions are also known for their advanced and intuitive management front-end, and for complementing their offer with additional services, which include *near real-time* actualisation of the database of attacks.

2. Allot Communications claims to have a TMS solution (NetEnforcer [157]) capable of providing more than one hundred metrics of the network traffic and of its performance. Their major business area comprises network traffic classification, prioritisation, and shaping on a per application or on a per user basis. Their most prominent brand promoter is the DPI acronym, which is mostly used to show the granularity that the system is capable of achieving. As far as the author could assess, the set of metrics provided by the TMS precludes self-similarity analysis in real-time. Apart from the referred characteristics, NetEnforcer comprises also a scanner for mitigation of known Distributed Denial of Service (DDoS) attacks and

Worms, and offers monitor and control capabilities for P2P and VoIP related traffic.

3. NIKSUM NetDetector [158] seems to be an emergent security solution for information networks. The product is presented as a *full-featured appliance for network security surveillance, detection, analysis, and forensics*, and it is claimed to draw simultaneously on signature- and on statistical anomaly-based techniques to perform intrusion detection. One of the most publicised features of the solution is related with its capability to reconstruct the streams of well-known protocols / applications, (as Hypertext Transfer Protocol (HTTP), e-mail, File Transfer Protocol (FTP), Telnet, etc.), being therefore suitable for the identification of malicious code flowing within application layer requests or responses. Once deployed, NetDetector is *continuously* capturing and storing the network traffic, for future comparison and for real-time reaction to attacks. This operational model could actually embody a scalability problem, if it was not for the explicit suggestion of the manufacturer to improve the storage capabilities of the solution as the network grows. The interface of the system is said to be highly intuitive and accessible via web-browser (using an authenticated and encrypted channel). NetDetector is presented as being compatible with other security solutions (e.g. with Cisco IPS), and as a useful forensics tool.

4. Appliance [159, 160], from SecurityMetrics, is a security solution that combines *correlated intrusion detection* with *vulnerability assessment* to achieve higher performance and decrease the false positive alarms rate. By actively exploiting the several components of the network in a timely and controlled manner, the Appliance is capable of constructing a database of the attacks to which the systems are susceptible to. It may then be configured to raise alarms for the subset of the attacks that actually constitute a threat, effectively decreasing the number of *false positives* notifications sent to administrators. Apart from the signature matching detection techniques, the solution also incorporates a *frequency based detection method*, that aims to discover unusual amounts of a given type of protocol data units, during predefined periods of time. The system is suitable for the detection of known Denial of Service (DoS) attacks and worms, as well as for the protection against viruses and cross-site scripting attacks. Because the equipment acts as a layer 2 bridge, its integration in the existing infrastructure is simple and does not require too much

effort, in terms of configurations. The update of the software of the solution is made automatically during the night.

5. Check Point presents itself as the owner of the intellectual property rights of the *stateful protocol analysis* [161], a technology that decides on the legitimacy of the protocol data units by checking weather they respect the premisses of the protocols they carry. Stateful inspection presumes the monitoring of both directions of the connections / sessions at the device where it is applied, namely via the construction and storage of a *table of states*, which enables the security solution to observe if the incoming data falls within one of the possible paths of the state machines of a protocol, otherwise declaring an attack. The factor that most favours the performance of the referred procedure is that the context information table can be efficiently constructed in real-time, and it does not require as much storage space as e.g. a signatures database. The Check Point IPS/IDS solution is termed IPS-1 [162], and it includes several technologies built on top of *stateful inspection*, namely detection of known or unknown protocol breaches and proactive defence against probing (the IPS maintains all the layer 3 ports closed while no rightful connection has been established to one of them). The number of protocols that are natively supported by the solution covers all the 6 top layers of the OSI model. Besides employing the signature-based techniques, the IPS-1 is said to be able to seamlessly correlate the alarms emitted by different security entities in the same network, detect and detain unknown but rapidly spreading menaces, and perform *smart* IP reassembly, prior to data examination. Automatic updates are available as a service through the *Check Point SmartDefence Services*, and the management is said to be intuitive and centralised, so as to benefit scalability.

6. The Juniper Networks Intrusion Detection and Prevention solutions present an impressive list of features. In [163], the several series of the products are said to implement eight different detection methods that cooperate to achieve higher detection ratio. One of the most interesting features is termed *Stateful Signature Detection*, and consists of the procedure by which small portions of the traffic are selected prior to the application of signature-based methods. The selection of the portion of the traffic is determined by the protocol context. Apart from that, all the security systems include *protocol and traffic anomaly detection* and the obligatory protection

against DoS attacks. According to the data sheet of the products, the detection of unknown threats is covered by the usage of techniques based on heuristics, and by the *same day coverage* service, which is concretised on the commitment of the company to update the signatures database as soon as the fingerprints of a new attack are discovered. Some of the IDS/IPSs include also an *Honeypot* function, so as to actively track probing activities and to gain insight into real-world threats. The management of the systems is centralised and role-based, enabling the assignment of different responsibilities to different administrators. Additionally, the continuous support of the *Juniper Security Team* and of their *security suggestions service* spare the network administrators from the daily research task for new threats.

7. The Enterasys NIDS is named Dragon IDS/IPS [164], and it is built on top of a technology that was distinguished with a quality and with a trust award in 2004 and in 2006, respectively [165, 166]. Enterasys claims that their solution has active response capabilities that can be used to dynamically tune the policies of the firewalls to which the system is connected to, in order to frustrate and/or stop intrusions. The NIDS is said to be capable of interacting with other NIDS or with several sensors located at the given network, enabling a more accurate identification and isolation of the source of the attacks. The problem of the scalability is addressed by the multithreaded architecture and via the support of virtual sensors. Its detection methods include the ones based on signatures matching, and on anomaly, protocol and behaviour analysis. By the time this thesis was written, the most recent release of the product was publicised as being capable of reconstructing data flows (e.g. TCP or HTTP sessions) at a data rate of 10 Gbps, and of processing them in real-time. The specifications of the system included also the response capabilities to DDoS attacks and the transparent operation of the NIDS within the network. It is equally emphasised that Dragon is capable of looking for vulnerabilities within VoIP calls, and of performing Zero Day detections (probably thanks to non-specified anomaly based techniques). The front-end of the Enterasys security solution is written in Java, and the management console may be accessed using a web browser.

8. Bro is the name of the Vern Paxson open source IDS proposal [167]. It targets high-speed, high-volume networks, and it is conceived to run in (cost effective) commercially available PC hardware running the UNIX OS. Bro is said to be

specially useful for sites requiring flexible and customisable detection capabilities, mostly because its *specialised policy language* allows it to make the operation of the IDS dependent from the site policies and from the discovery of new attacks. While its monitoring functions are passive, the IDS may be configured to issue active responses through the execution of commands, or to merely generate and send a security log. Bro parses the traffic to extract application level semantics, which may then be submitted to the so-called *event-oriented analysers*. It is actually this particular mode of operation that enables the intrusion detector to run so efficiently. Thus, the IDS is not only suitable for the identification of attacks with a predefined signature, but also of the ones that may be defined in terms of events. Its biggest disadvantage is that the system has to be handled by experts. Additionally, as it is stated in [167], Bro should be understood as a solution for those who seek an IDS that has to be built as need arises, rather than an *out of the box solution*.

9. The most popular open source network intrusion prevention and detection system is SNORT, which may be freely downloaded from [168], where it is publicised as the *de facto* standard NIDS for the industry. The system utilises a rule-driven language that combines the benefits of several approaches for the identification of threats, namely a signature matching engine, protocol and anomaly inspection methods. SNORT follows the market trends very closely and, at the time this thesis was written, its most recent release was already capable of natively supporting Internet Protocol version 6 (IPv6) and Multi Protocol Label Switching (MPLS) (which is a major feature for carrier networks). The architecture of the *SNORT Security Platform* favours online deployments of the NIDS, and its *multithreaded execution modes* were listed as one of the solutions for the scalability problems.

As it was previously stressed out, most NIDSs are built on top of signature-based detection techniques, and use technologies as protocol parsing, stateful protocol analysis and anomaly-based mechanisms to enhance the attractiveness of the product. Most of the developers of these kind of systems seem to be investing a lot of effort in the intuitiveness of their interface, as well as in the scalability problem. *Multithreaded execution* support appears to be one of the solutions for such a problem, since the urge to go deep into the protocol data units contents encompasses a computational cost that cannot be avoided

nor neglected. In the case of commercial systems, the actualisation of the software and of the database of the signatures is often sold as a management service. Most systems act simultaneously as IDSs and as IPSs, and some of them draw on the correlation of the information coming from host security sensors, or from other security equipment, to detect and isolate sources of attacks, namely of DDoS attacks. When applied, the statistical treatment of the captured traffic is confined to the calculation of the most common metrics, as the average or standard deviation of several networking aspects. At the time this thesis was written and to the best of the knowledge of the author, there were no security solutions implementing real-time self-similarity analysis as a means to perform intrusion detection. The conclusions of this chapter provide a possible explanation to that fact.

5.3.2. Critical Analysis of the Related Works

Since the developments that unfolded the self-similarity nature of network aggregated traffic [16, 39], the concept of *self-similarity* has gathered special interest from the networking research community, being the subject of many contributions along the years. During the initial years after its discovery, the phenomenon of self-similarity was mostly observed from the perspective of its impact in the network resources utilisation, and only after the year 2000 one may find contributions that relate the peculiar statistical behaviour of the traffic with the subject of intrusion detection. References like [22, 23, 24, 169, 170, 171], for example, discuss the importance of guaranteeing the self-similar properties of the background traffic used in the evaluation of intrusion detection mechanisms [22, 23, 24, 169, 170], and the problem of the possible loss of the fractal structure during an attack or during an erroneous state of the network [170, 171], but not on the concrete behaviour of the self-similarity degree during an intrusion. The last mentioned topic is addressed in a smaller number of contributions [20, 21, 25, 26], which are going to be discussed below with more detail. Please notice that the following statements represent a critical perspective of the author of this thesis and that they must always be understood while having that consideration in mind.

The first work the author would like to address in this section is the one of Li

[25], dated from 2006. As far as the author of this thesis could assess, in [25], long-range dependence is presumed to be a property of the size of the protocol data units, rather than of the amount of information per time unit. The reasoning is then built on top of that assumption, and the practical demonstration that the Hurst parameter values *decrease* during DDoS (flood) attacks is made via the analysis of the MIT/DARPA traces. The Hurst parameter is assessed by means of a parametric minimisation of the difference between the empirical and the expected autocorrelation function of self-similar series. The window sizes used in the study are not discriminated, though one might extrapolate from the few references to those details that the non-overlapping sections to which the analysis was applied to were of size 2^{14}, and that the maximum lag for which the autocorrelation was calculated was of 16. The theoretical proof of the results obtained from the referred analysis is made by enunciating two corollaries which, in the opinion of the author of this thesis, are demonstrated in an extremely terse manner.

One of the conclusions that could be drawn from [25] is that a DDoS decreases the *burstiness* of the traffic (since it causes the Hurst parameter to decrease), which could consequently lead to the conclusion that the quality parameters of the network could actually improve during a flood attack. Such conclusion is wrong, and the explanation for the apparently consistent results included in the paper is difficult to grasp from the description *per se*, but it must be related with its initial assumptions or with the utilised estimation method. In this chapter, it will be shown that the local scope Hurst parameter estimates obtained from the implementations of WMVT and WMEBP may increase for moderately intense flood attacks (depending on the traffic load), and that self-similarity is actually lost during higher intensity floods, contrarily to what is suggested in [25]. The change in the Hurst parameter estimates happens during the transition period where the attack is either entering or leaving the observation window, and it is even shown empirically (and later corroborated theoretically) that the self-similarity is locally preserved while the attack lasts. Nonetheless, it should be mentioned that the series to which the analysis is applied to is different from the one in [25], and that the estimation methods are different also.

The paper entitled *Iterative Window Size Estimation on Self Similarity Measurement for Network Traffic Anomaly Detection* [21] aims for the assessment of the *window*

size that most favours the detection of anomalies based on self-similarity analysis. Their proposal consists of an iterative (brute force type) algorithm that applies the HEAF estimator to the same data set sampled for different window sizes, ranging from e.g. 500 s to 2000 s (with a predefined step size of 100 s). The criteria the algorithm aims to maximise is the inversely proportional relation between the number of anomalies detected and the number of false alarms. Analogously to what was done by Li [25] and to what is initially done in this thesis, the analysis was conducted for the 1999 MIT/DARPA data sets. The main focus of [21] is thus not directed towards the analysis of self-similarity. After discussing the results, the sample size of 1400 s is indicated as the one with the best detection rate, for intrusions with duration larger than 500 s. No theoretical support is provided for such conclusion.

The research work described in [26] starts by referencing the previously discussed work of [25], apparently supporting the conclusions of Li, but in their report, they show that the estimates of the Hurst parameter increase during the anomalies they examined (recall that in the work of Li [25], the value of the Hurst parameter was said to decrease). In [26], observation windows with a length of 30 minutes are used, and the traces of traffic are aggregated for time units varying between the 10 ms and the 1000 ms. The loss of self-similarity is assessed via the evaluation of the average deviation between the theoretical and the empirical autocorrelation functions (the autocorrelation function is calculated for lags up to 200 units). For departures larger than 10^{-3}, an anomaly is signalled, but nothing is added in [25] regarding its intensity or duration. The results contained in [25] show that the values of the Hurst parameter obtained for abnormal traffic is never larger than 1, but that the difference between the autocorrelation functions may indicate *loss of self-similarity* (expressive departures are achieved during some of the attacks analysed).

The work of Allen and Marin [20] is, in the opinion of the author of this thesis, the most concrete and coherent of all the studies discussed in this section. The type of attacks they tackle in their investigation coincides with the ones considered in this study. The size of the sample pools varies between the 10 and the 30 minutes, depending on the size of the traces and on the traffic load. The value of the Hurst parameter is calculated every 5 minutes. The 1999 MIT/DARPA data sets were, once more, used in the evaluation of this particular proposal. A *DoS - Traffic Exploit* attack (designation used in the reference) is

signalled when the value of the Hurst parameter, estimated by the Periodogram *or* the Whittle estimator, is either superior to 1 or inferior to 0.5. The paper does not contain a study to the evolution of the self-similarity degree, and does not consider the possibility of having DoS - Traffic Exploit attacks that do not result in the loss of the self-similarity.

The definition of *loss of self-similarity* presented in [20] is not correct, but it seems to fulfil the objectives of the proposers. It is true that the Hurst parameter of self-similar series should be confined to $]0, 1[$, and the one of long-range dependent series should be larger than 0.5, but such does not necessarily mean that the estimators will return values out of the referred interval for all non self-similar series. Sastry *et al.* [172], for example, discussed how Wavelet analysis and *single value decomposition* could be used to specify the number of scales in which the series under analysis remains self-similar, and in which ones the anomaly is reflected. Though their approach is different from the one in [20] and from the one taken in here, it shows that the subject of the loss of self-similarity during an anomaly may be analysed from several angles, and that a clear line between the two states is not that easy to draw.

Notice that the concepts of *observation window*, *sample size* or *non-overlapping segments or blocks*, utilised in the references discussed herein, are not equivalent to the one of *observation window* formalised in chapter 3. In [21], the term *window* is normally used to refer the block size (or the *sample time*, according to the reference) taken by the HEAF estimator which, according to the authors, can be chosen by an iterative mechanism, that aims for the optimization of the relation between *data insufficiency* and *sensitivity*. In the remaining research works, the concept is used to refer to subsets of values to which the analysis was applied to. Additionally, the movement of the observation window is defined by jumps.

It is of the opinion of the author that the work reported in this thesis differs from the aforementioned works in the following three aspects:

1. First of all, in this work it is not presumed, nor concluded, that the presence of some kinds of attack results *inevitably* in the loss of self-similarity. It is concluded that, depending on the relative location of the observation window during the time span of an intrusion, the self-similarity may be either preserved at a local scope or lost.

For some situations, the estimators signal an increase of the self-similarity degree.

2. Secondly, the self-similarity analysis is herein made by using well-known estimators that were strictly modified for the purposes of this research, enabling a more granular (i.e. step-by-step) examination of the evolution of the Hurst parameter values during an intrusion. The modified estimators enable operation over significantly smaller observation windows than in the previously discussed works. Additionally, the window step can be as small as the smallest aggregation scale considered (e.g. it can be as small as a millisecond), without having any problem concerning the computational feasibility of the calculations.

3. The validity of the hypothesis was pursued via the application of two completely different (and independent) estimation methods, for the sake of impartiality. Furthermore, all the assumptions and results are coherent with the theory developed along the research work, and the analysis was conducted in such a way it enables the results to be generalized to scenarios different from the ones considered herein.

5.4. Self-Similar Traffic Generation and Attacks Simulation

During the initial phase of this research work, it was concluded that there were several shortcomings for conducting the tests for real traces of traffic. On the one hand, there was an inherent lack of control over basic, yet critical, aspects of the experimental scenarios (e.g. the network load, the degree of self-similarity); on the other, it was noticed that some pre-collected traces containing attacks suffered from flagrant flaws in terms of the self-similarity properties, most certainly because that particular characteristic was thought as being of secondary relevance for the purposes of such resources.

One of the main applications of the generators of occurrences of fBm (or fGn) processes is the simulation of traffic in network aggregation points. This section contains the description of a possible means of using 4SG as a network traffic generator with self-similar characteristics. The procedure draws on the sequential generation capability of 4SG, allowing for the synthesis of a trace on a packet-by-packet and real-time basis.

Note that the following description is also valid for fBm-SGA, since the latter fulfils the most critical prerequisites of the proposed procedure. The edge of 4SG is on its higher quality for scales of type different from 2^k.

5.4.1. Model Description and Formalisation

The process in which self-similarity has been proven to exist is the bit count per time unit process, and it is where the impact of the network attacks is to be assessed. Thus, the simulation of that particular aspect of a legitimate trace of traffic could be directly performed via the generation of an fGn and of its posterior adaptation to fit the specifications of the network aggregation point, in terms of average and variance. The effect of the attacks could then be inserted by defining an additional process, which would only take values different from 0 by the time of the intrusion, and that would actually be summed to the shifted fGn. Nonetheless, it was decided that the simulation of the traces via synthesis of the packet size and inter-arrival processes would improve the realism of the model, making the insertion of malicious traffic a more straightforward task (see next sections).

The challenge behind the approach taken was on how to generate consistent occurrences of the processes of the inter-arrivals and of the packet sizes, while assuring that the underlying process of the amount of information per time unit possesses a fractal structure, with predefined Hurst parameter. Furthermore, the procedure had to provide the means to control other statistical characteristics of the referred series. As so, the set of parameters, assumptions and statistical properties that were taken into account when devising this simulator were the following:

- the intended self-similarity degree of the network trace (given by the Hurst parameter H);
- the available bandwidth BW at the given network point;
- the intended load L (a fraction of the total available bandwidth);
- the minimum inter-arrival time IA_{min};

- and a known (e.g. empirical) probability distribution for the sizes of the packets arriving to a given network aggregation point.

Accordingly to what was said, a trace consists of a set of packet sizes and of inter-packet gap values. The processes representing those dimensions are hereinafter denoted by $\{PS(t)\}_{t\in\mathbb{N}}$ and $\{IA(t)\}_{t\in\mathbb{N}}$, respectively. Assume that each packet size is a random number between $PacketSize_{min}$ and $PacketSize_{max}$, which respects an empirical packet size distribution. This means that no special measures are to be taken for the generation of the packet sizes, and that these values may or may not present correlation (no self-similarity) between them. In other words, only the inter-arrival times will somehow reflect (or be affected by) the self-similarity phenomenon. Given the packet size distribution, one can calculate the average packet size value $\mathbb{E}(PS)$. Fix a timescale ts and suppose that the available bandwidth BW is already in bits per timescale ts. Assume also that the IA_{min} parameter is expressed in bits (the explanation is easier under that assumption). To recover the inter-arrival in time units, divide it by BW. According to this and to the definition of *traffic load*, at a given moment of time, the average bandwidth used is given by $BW_U = L \times BW$ and the average number of packets that *fill* that bandwidth is given by

$$n_{packets} = \frac{BW_U}{\mathbb{E}(PS)}. \tag{5.1}$$

As each packet is followed by an inter-arrival time, there will exist $n_{packets}$ inter-arrival values. It is assumed herein that the inter-arrival times are realisations of a shifted and scaled fGn process (because there are no negative inter-arrival times), with Hurst parameter H. The expected value and variance of this process must fulfil certain criterion. Since the used bandwidth is *filled* with the packet sizes, the unused part must be *filled* with the inter-arrivals. To assure that, at least in average, the amount of traffic that arrives to the given network point in the considered timescale is given by BW_U, the average inter-arrival value must be defined as in (5.2):

$$\mathbb{E}(IA) = \frac{BW_{NU}}{n_{packets}}, \text{ where } BW_{NU} = BW - BW_U. \tag{5.2}$$

The previous condition also assures that, in average, the generated values for the inter-

arrival times are positive numbers, but it does not assure that all of the samples are so, nor that the minimum inter-arrival time is seldom retrieved by the generation procedure. This problem can be solved (and, in some way, controlled) if the variance of the process is defined as in (5.3) or (5.4):

$$\sqrt{\mathbb{V}(IA)} = \frac{\mathbb{E}(IA) - IA_{min}}{2}; \tag{5.3}$$

$$\sqrt{\mathbb{V}(IA)} = \frac{\mathbb{E}(IA) - IA_{min}}{3}. \tag{5.4}$$

In order to understand better what expressions (5.2), (5.3) and (5.4) mean, the reader is asked to observe Figure 5.1. The choice of the $\mathbb{V}(IA)$ value ((5.3) or (5.4)) provides a way to choose *how many* values of the generated Gaussian process are going to be smaller than IA_{min} (0.1% *or* 2.2% *of the whole population*). Ideally, one would think that this value should only be retrieved by the generation algorithm with infinitesimal probability (except when one is simulating heavy load conditions), but empirical analysis show that the IA_{min} is regularly measured, even under light load conditions, for it separates the packets of the bursts sent by terminal or other multiplexing nodes.

In order to avoid inter-arrival values smaller than IA_{min}, the generated values smaller than that value should be discarded and the predefined value of IA_{min} should be retrieved instead. Unfortunately, this last step is the same as truncating the generated values of the fGn when they are smaller than IA_{min}, i.e.

$$\text{Return } IA_{min}, \text{ if}$$
$$IA(t) < IA_{min}. \tag{5.5}$$

To avoid having a slight bias in terms of the target traffic load (and in terms of the self-similarity degree), one might want to define a symmetrical upper bound truncation as well, as suggested by the following reasoning:

$$\text{Return } \mathbb{E}(IA) + (\mathbb{E}(IA) - IA_{min}), \text{ if}$$
$$IA(t) > \mathbb{E}(IA) + (\mathbb{E}(IA) - IA_{min}). \tag{5.6}$$

Having defined the average and the variance of the inter-arrival times process $\{IA(t)\}_{t\in\mathbb{N}}$, it is easy to write the law that, according to this model, confers self-similar properties to the traffic trace (recall that, in the next equation, $G_H(t)$ denotes an fGn process with Hurst parameter H):

$$IA(t) = G_H(t) \times \sqrt{\mathbb{V}(IA)} + \mathbb{E}(IA). \tag{5.7}$$

Figure 5.1 provides the graphical representation of the idea behind the simulation apparatus. It depicts where the long-range dependent Gaussian variable is to be *placed* in order to assure the self-similar properties of the bit count per time unit. Please take into account that, for the sake of simplicity (and without loss of generality), some of the previously described concepts were *relaxed* to produce the illustration in the referred figure, which, consequently, does not exactly reflect what was previously said. To be exact, it must be said that the illustration is valid for $ts = \frac{\mathbb{E}(PS)}{L}$ (i.e. the time scale for which the figure is valid for is the one in which, in average, fits one packet only).

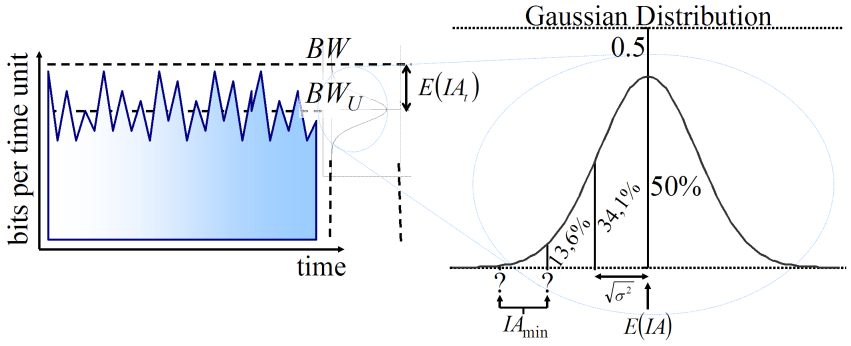

Figure 5.1: The *position* of the fGn in the bit count per time unit process, and the illustration of the limits after which the value of IA_{min} is retrieved. (This figure is valid for $ts = \frac{\mathbb{E}(PS)}{L}$.)

The responsibility of assuring the fractal structure of the amount of information per time unit is laid upon the procedure for the generation of inter-arrival times, which basically places the generation of the packet sizes in a second plane, as long as the average value of the retrieved values is known and fixed *a priori*. This particular design favours the specification of the parameters of the simulator, for it is possible to precisely adjust the *amount* of the unused part of the bandwidth, thus enabling the control over the simulated traffic load.

5.4.2. Implementation Details and Pictorial Proof of Self-Similarity

The network traffic generator was implemented according to the specifications in the previous section in the object oriented programming language Java, and tested on a desktop computer equipped with a PIV 2.4 GHz processor and 1 GB of RAM. To generate the packet sizes, a file containing a packet size distribution of an aggregation network point was downloaded from one of the sites of National Laboratory for Applied Network Research (NLNAR) [173], and later read by the traffic generator. Each time a packet size was to be returned, the probability domain was emulated by a pseudo random number between 0 and 1 (PRR was used to this purpose), and the respective packet size value located in the empirical distribution.

In [31], the previously described simulation model was presented as a possible application of fBm-SGA. In there, the generation of the inter-arrival times process was done by transforming the outputs of the algorithm according to (5.8), where fBm-SGA$_H(t)$ represents the sequence of values returned by fBm-SGA for a predetermined Hurst parameter H:

$$IA(t) = \text{fBm-SGA}_H(t) \times \sqrt{\mathbb{V}(IA)} + \mathbb{E}(IA). \tag{5.8}$$

Herein, the simulation of the inter-arrival times is done resorting to 4SG, due to the already enumerated reasons.

The traffic generator was tested for several network parameters combinations (diverse loads and Hurst parameter values). For each scenario, the number of bits generated was aggregated for three different time scales (0.1 s, 1 s and 10 s - see charts in Figure 5.2) and the self-similarity degree of the resulting processes was analysed. This analysis was similar to what was made in section 4.3.2. in all aspects, including in the specifications of the computer were the simulations were conducted. In average, the estimated Hurst parameter values were *slightly smaller* than the intended ones, and their variance was almost 9 times bigger then the ones presented in section 4.3.2. of the referred chapter, for the biggest time scale used. This particular behaviour was somehow expected, since the core of the simulation apparatus is an *approximate* method for the synthesis of fGn, which results in minor lacks of correlation (and thus, in a smaller degree of self-similarity).

To overcome this minor flaw, the dependencies of the fGn must be strengthened by declaring an Hurst parameter value that is actually bigger than the expected one. Table 5.1 contains some of the compensation addends that may be used to control the self-similarity degree of the produced traces. (The values in the table were taken from a larger set of results, which were obtained from the examination of 3000 synthetic traces.)

Table 5.1: The difference between what should be declared, and what should be expected, in terms of the self-similarity degree of the traces generated using 4SG.

Expected Hurst Parameter Value (std. dev.)	**Declared Hurst Parameter Value**	**Compensation Addend**
0.6(5.59E-04)	0.65	0.05
0.65(4.21E-04)	0.69	0.04
0.7(5.81E-04)	0.73	0.03
0.75(6.71E-04)	0.78	0.03
0.8(1.48E-03)	0.84	0.04
0.85(1.09E-03)	0.89	0.4
0.90(1.09E-03)	0.95	0.5
max: 0.92(0.00125)	0.98	0.6

The traffic load was also subjected to experimentation, since it is one of the main input parameters of the generator. The absolute error between the intended and the measured load was always smaller than 10^{-3}, which is a direct consequence of how accurate 4SG is, in terms of the average and variance of the generated sequences. Because the proposed simulation model preserves the main statistical properties of the approximate fGn, the amount of information per time unit is well centred at the expected position.

The charts in Figure 5.2 are the graphical representation of the bit count per time unit process, and they were obtained during the simulations to the scenario with the following combination of network parameters: $H = 0.8$; $BW = 1$ Gbps; $L = 0.4$; $PacketSize_{min} = 360$ b; $PacketSize_{max} = 12000$ b; $IA_{min} = 96$ b. As can be concluded from careful observation of the charts, the variance of the aforementioned processes is slowly decaying as one moves from chart a) to chart c), as a consequence of the depicted aggregation procedure. However, contrarily to what would happen with a completely

random variable, the reduction of the range of occurrences is not inversely proportional to the time scale considered, otherwise the chart in c) would be much less variable than it is. This fact is owed to the scaling properties of self-similar processes (the general aspect of the graphical representations of the process seems to be *independent* from the distance it is observed), and thanks to Leland and Taqqu [39], it is currently known as the classical pictorial proof of self-similarity of the network traffic (please refer to section 2.3.3. for further details on this subject).

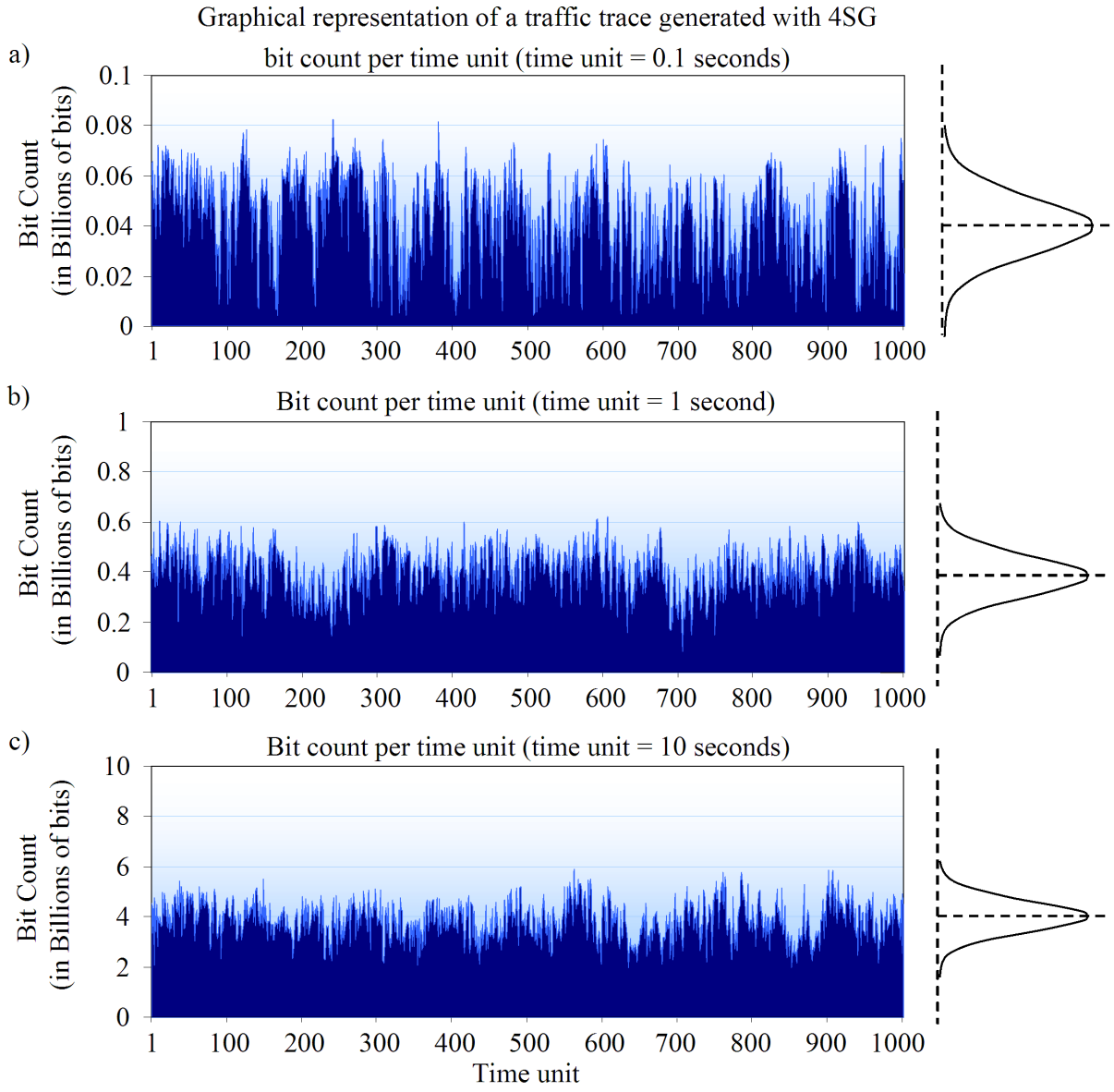

Figure 5.2: Classical pictorial representation of self-similarity in network traffic: the aggregation scale increases from 0.1 s to 10 s as we move from chart a) to chart c). All traces were generated using the 4SG with a predefined Hurst parameter equal to 0.8 and load parameter equal to 0.4.

The simulator does not replicate all the particularities of the traffic for a given network aggregation point. One of the major faults of the presented model is its incapacity to accurately simulate the phenomenon of packet trains. This fact is explained by two main reasons: (i) the generation of the trace is being made at the point where self-similarity is to be assured, emphasising only this objective and forgetting each one of the individuals (remote nodes) that contribute for the generation of the trace and; (ii) the packet sizes are modelled as realisations of a random process and do not necessarily obey to any rule of persistence. Nevertheless, this fault is of secondary relevance in the scope of this thesis, since the study it reports is conducted for statistics taken from a series that, in this case, is not affected by the order of the size of the packets. If strictly required, the fractal structure may be embedded in the sequence of packet sizes by generating a self-similar series following the uniform distribution, and by using it to choose the values from the empirical distribution.

5.4.3. Demonstration of the Fractal Properties of the Bit Count per Time Unit Through VT Analysis

In the final part of the previous section, the scaling properties of the generated traces are demonstrated by means of a pictorial proof. It is also said that the self-similarity degree of the amount of information per time unit is slightly smaller (in average) than the one of the generated fGn, due to minor lacks of correlation of the approximate series. Thus, the doubt on how well the fractal structure is embedded in the generated traces is not completely unfounded. To prove that the referred characteristic is, in fact, preserved by the previously presented procedure, and that the uncorrelated selection of the packet sizes does not render the referred process random, the Hurst parameter of the bit count per time unit was calculated for different traces using the retrospective version of VT. Several traces (with Hurst parameter varying from 0.70 to 0.95) were created, and their VT log-log plots were subject to human observation. The compilation of the statistical treatment for 100 simulations of traces with *expected Hurst parameter equal to 0.83* (using a compensation addend of approximately 0.04) was included in Figure 5.3 for exemplification purposes. The fact that the points in the plot are so well fitted by a line, added to the fact that the value retrieved by the formula $H = 1 + slope/2$ is 0.83,

provide that the traces generated by these means are of good quality, at least from the self-similarity perspective.

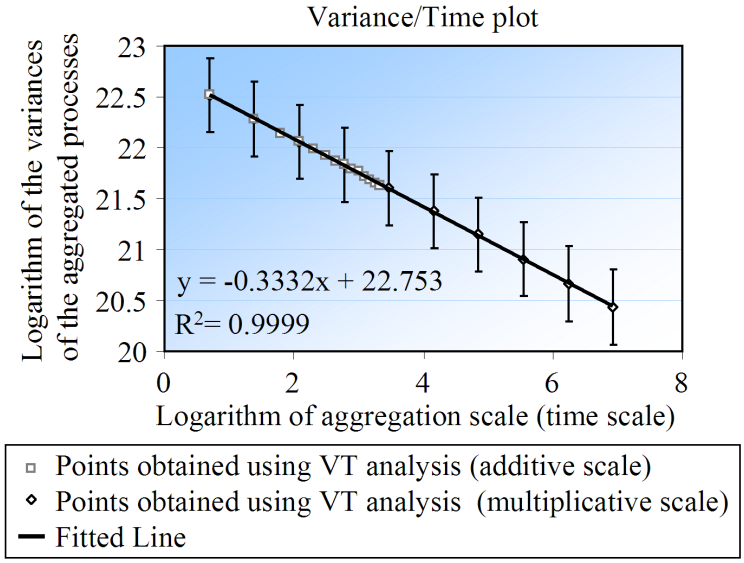

Figure 5.3: Demonstration that the procedure described in section 5.4.1. can, in fact, be used to generate self-similar traces. The value of the Hurst parameter for the particular trace is 0.83.

The scaling law that defines the variance/time structure of the analysed process is printed in Figure 5.3. The justification for that lies again on the inner workings of the adopted model, and in the procedure that was used to obtain the bit count per time unit values. As the several packet sizes and inter-arrival values are being summed to fulfil a *fixed* threshold, the scaling law of the approximate fGn is being *evenly* distributed along the several occurrences of the bit count per time unit, thus *equally* affecting all the aggregation blocks considered during the VT analysis.

5.4.4. Definition and Simulation of Network Intensive Attacks

Before proceeding with the description of the means used to simulate attacks, it is pertinent to comment on the exact type of anomalies with potential of being detected by self-similarity analysis. Since long-range dependence reflects the relations of a necessarily large pool of samples, any attack constituted by a relatively small number of protocol data units has little chances of being detected by the referred approach. Such fact immediately excludes intrusions based on single malformed packets (e.g. the *Land* attack [174]), or the transfer of a small amount of malicious data, resulting from the spread of malicious

software as virus, worms or trojan horses. On the other hand, the epidemic spread of such menaces may be expressive enough to raise an alarm.

Malicious activities that are randomly sparse in the time domain embody a second type of activities that should not produce, at least theoretically, any effect in the scaling exponents defining the trace. Thus, stealth probing activities [152] (as Portsweeps or IPsweeps) intentionally dispersed in time, are not detectable either, because the effect of the packets that actually materialise the probe procedure gets statistically diluted in the presence of a *large enough* quantity of legitimate protocol data units.

The previous remarks direct this study towards the attacks that, at some point of their intrusion, occupy a non negligible amount of bandwidth, if only for a moment. These attacks are here designated by *network intensive attacks*. Note that several types of DoS attacks fall precisely into the indicated category, and that the previous reasoning goes to the encounter of what was said by Allen and Marin [20] and by Li [25] about the type of attacks they were targeting in their analysis. In [25], the designation of *bandwidth attacks*, suggested by CERT in e.g. [175], is actually emphasised and utilised along the paper.

DoS attacks based on massive amounts of messages are amongst the menaces that most preoccupies network administrators, especially the ones for which the network is assuring a given service. More recently, new forms of performing such intrusions using botnets (DDoS, Distributed Reflected Denial of Service (DRDoS) [176, 177]) increased their effectiveness, being now much easier to emulate a large number of apparently legitimate connections. The target of such malicious activities is not always the node where one is trying to discover them. The significance that, for example, a DDoS attack may achieve is highly dependent on how close to the address of the victim the detector is, since the sources of the packets are spatially distributed. On high-debit aggregation nodes, the last type of attacks may be on the origin of intrusions with moderate expressiveness, in terms of network load, which encapsulate situations where it is hard to assess the loss of self-similarity.

The simulation of network intensive attacks was done via specification of the attack *Intensity* (I) parameter, which basically determines the amount of packets that arrive at a given network aggregation point, per time unit and in function of the *available* bandwidth.

The packet generation frequency is then calculated in relation to the predetermined size of the malicious protocol data units, so that the intrusion fills up the right load. For example, to simulate a TCP SYN flood, the size of the packets is set to 54 B and the average number of incidences per time unit is given by $I \times \frac{BW_{NU}}{54 \times 8}$. Due to the fact the legitimate traffic trace is generated in a packet-by-packet manner (as described in the previous section), the injection of the traffic related with the malicious intent may be achieved by a direct implementation of the procedure illustrated in Figure 5.4. The packets belonging to a malicious attempt are orderly inserted in between the genuine ones, which may result in delays for both types of protocol data units. In the figure, the packets belonging to an attack (represented by black rectangles) are intentionally separated by a constant size interval, precisely to emphasise that they are generated in a timely manner, being that an adjustable parameter of the simulation. The equipment responsible for the aggregation of the data is depicted as the device that accepts two streams of data and outputs a single stream only.

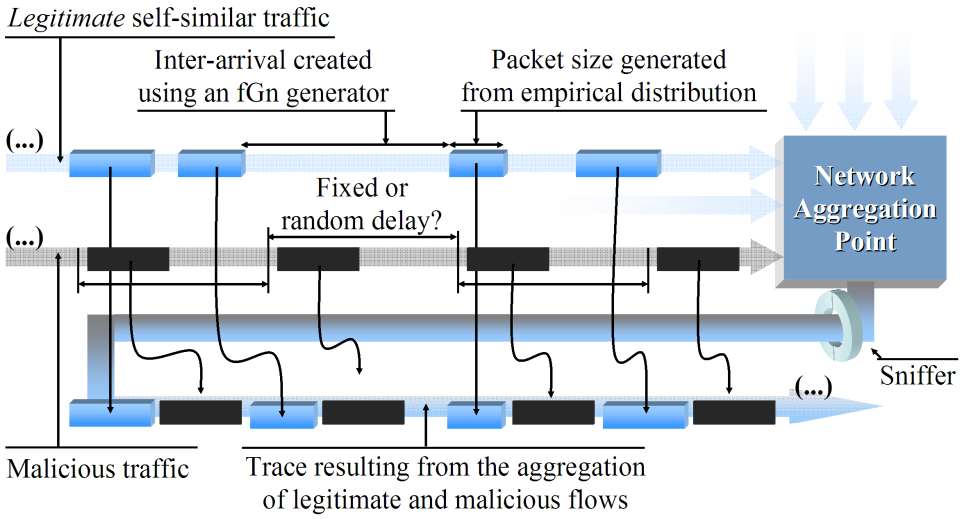

Figure 5.4: Graphical representation of the procedure used to inject a network intensive attack into self-similar traffic.

The simulation procedure is not a perfect replication of the reality, mostly because it was built at the abstraction level required in this research work. For the sake of the analysis, it was considered that the aggregation equipment was capable of storing several milliseconds of data before being forced to drop packets. In other words, it was assumed that an attack could affect the delay introduced to the legitimate packets, but not their late processing. Furthermore, it is assumed that the traffic trace under observation is the

one exiting the single interface of the network aggregation system (where the sniffer is placed), meaning that it is assumed that it is impossible to be analysing a bit count that surpasses the maximum bandwidth of that port. (In terms of the analysis reported herein, the scenario where the network is overloaded is equivalent to the one where the load is maximum.) This means also that all possible scenarios are limited by the one where an anomaly is actually so intensive that the flow under examination is *almost* constant bit rate.

5.5. Analysis of the Impact of an Attack in the Self-Similarity Degree of the Network Traffic

Now that most of the tools developed and used along this work are duly presented, it is time to discuss the impact of an attack in the self-similarity degree of the network traffic, and report on how the analysis was conducted. The next four subsections discuss the results of the application of the modified Hurst parameter estimators to real (i.e. simulated in a laboratory) and synthetic traces. A method to identify network intensive attacks based on the behaviour of the estimates is then described. A section dedicated to the subject of loss of self-similarity as a consequence of an anomaly may be found in the next-to-last subsection, right before the discussion on the theoretical framework of the results.

5.5.1. Analysis of the MIT/DARPA Traces

Before developing the self-similar traffic simulator and 4SG, the author searched for a quick way of obtaining results using the recently implemented windowed-modified estimators. At the time, the utilisation of a collection of traces from a well known experiment conducted by the MIT/DARPA laboratory seemed like the best way of doing so. The data sets [145] are constituted by several traffic traces captured in 1998 and in 1999, in a simulated network with real equipment. After their creation, these data sets have been used as a standard corpora for the evaluations of intrusion detection methods in multiple scientific contributions, inclusively in some of the references mentioned in section 5.3.2..

The details of the MIT/DARPA experiment are widely discussed in the literature and in the Internet [23, 145, 178, 179], but for the explanation included below, it suffices to know that the traces contain several types of attack, some of which fall within the category of network intensive attacks. The several files representing the experiment were downloaded from [145] to the local machine, and analysed by a procedure that may be explained as follows. First, the script processing a given trace initialises a time counter to zero, and sums the size of the packets that occupy 1 ms, incrementing the counter accordingly to obtain the bit count per millisecond series. These values are then fed to the incremental and windowed-modified versions of VT and EBP and the respective point-by-point estimations of the Hurst parameter values are plotted against the time they were assessed. For this analysis, the author was particularly interested in the data subset concerning the second week of experiments as, for those days, the time of occurrence of each one of the attacks is known *a priori* (the data sets containing labelled attacks were created with the purpose of providing the creators of behaviour-based intrusion detection mechanisms with training data). During this analysis, the size of the observation window of the local context estimators was set to 2^{14} samples and the number of aggregation levels was set to 9 (meaning that the estimators were looking for relations up to the time scale of 512 ms).

The charts included in Figure 5.5 depict some of the results obtained when using the MVT and WMVT methods. They were chosen for being the perfect graphical representation of the manifestation of the network intensive probe SATAN and of the SYN flood Neptune attack [178]. The two particular incidences shown here were captured during the fourth day of the second week, around 09:33 and 11:04, respectively. The plotted values show that the local context Hurst parameter values surpass the horizontal line defined by $y = 0.8$ during the attacks, while the incremental estimates are close to approximately 0.76 during the time span of the first chart and 0.79 during the time interval of the second.

Apart from the SATAN and Neptune intrusions, it was equally possible to identify several incidences of the Mailbomb attack, some Portsweeps and IPsweeps (the manifestation of the first of those three is depicted on the right side of Figure 5.6). The attacks denominated by Smurf, Apache, Back and UDPstorm were also detected when some of the files of the remaining weeks of the experiment were examined. Notice that all of

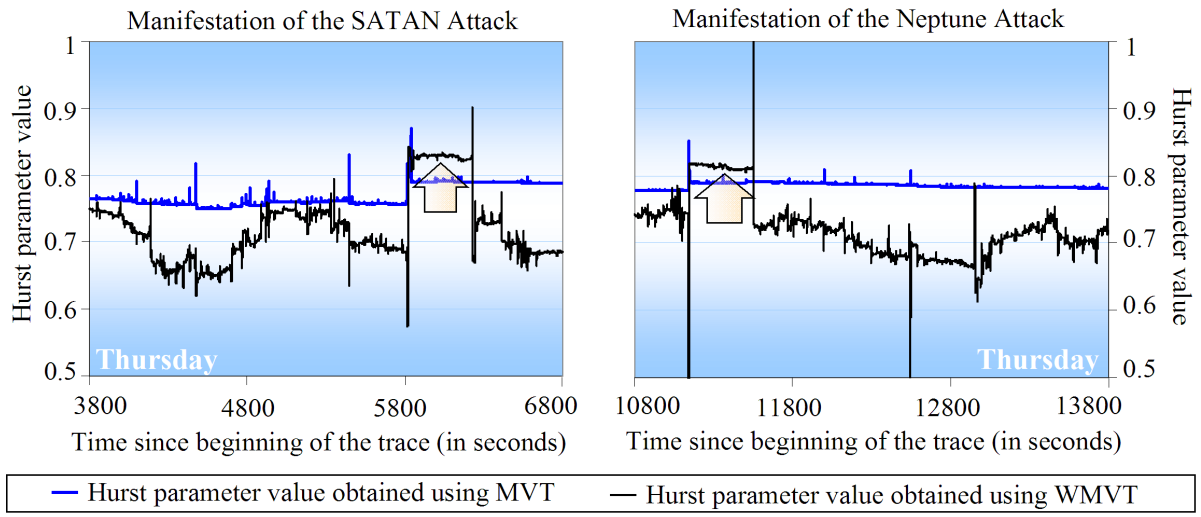

Figure 5.5: Manifestation of the intensive probe SATAN (Thursday) and of the Neptune attack (Thursday), when observed from the self-similarity analysis perspective. Values of the Hurst parameter were obtained using MVT and WMVT for the byte count per millisecond, being the size of the observation window of 2^{14} samples.

the aforementioned intrusions fall into the network intensive attacks category, since all of them rely on a considerably large number of protocol data units to achieve their purpose. As expected, no effect was noticed at the time of the occurrence of more modest attacks (in terms of amount of bandwidth), as e.g. the *Ping of Death*, the *Chrassis* or the *Land* attacks.

The impact of the SATAN probe or of the Mailbomb attack in the estimates of WMEBP is depicted in Figure 5.6 (the first intrusion was captured at 12:02 of the third day of the second week, while the second was identified at 14:25 of the same day). The two charts were included with the intention of showing the difference between the VT and the EBP based analysis, and also summarise the experiments conducted using MEBP and WMEBP. During this part of the work, over 250 views of the MIT/DARPA traces were created and carefully studied. As can be seen, for this particular collection of traces, the curves produced by the distinct estimation methods are different (the lines concerning EBP analysis are smoother), but the behaviour of the local context Hurst parameter value, when facing a network intensive attack, is essentially the same for both cases: the windowed-modified estimators seem to agree that, in such circumstances, the degree of self-similarity increases.

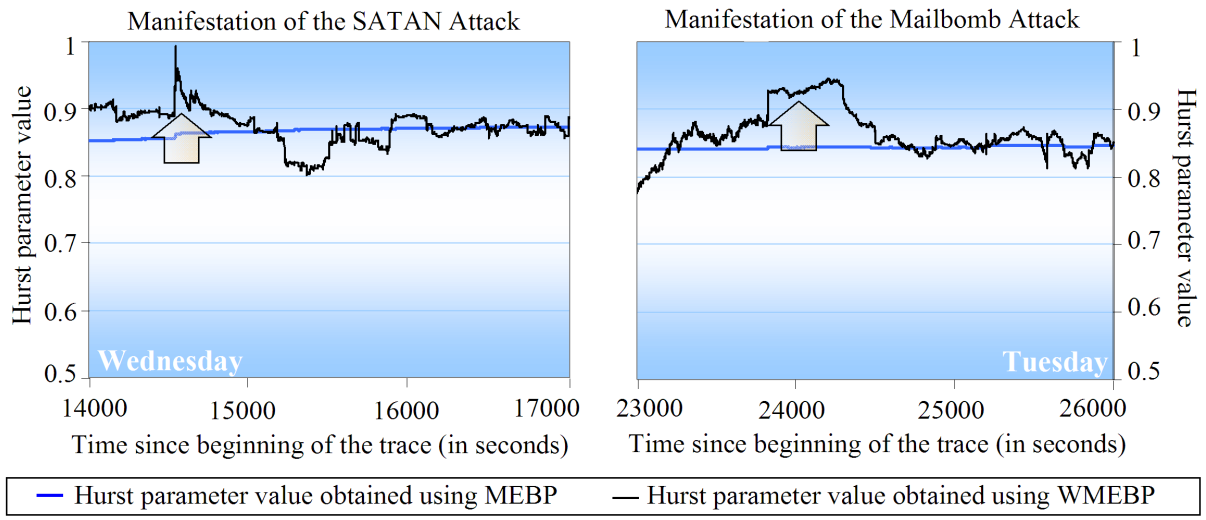

Figure 5.6: Manifestation of the intensive probe SATAN (Wednesday) and of the Neptune attack (Tuesday), when observed from the self-similarity analysis perspective. Values of the Hurst parameter were obtained using MEBP and WMEBP for the byte count per millisecond, being the size of the observation window of 2^{14} samples.

The explicit difference between the way WMEBP and WMVT react to the intrusions is mainly due to three factors. Firstly, the operational model of the former estimator is less susceptible to abrupt changes in the process under investigation, because losses of stationarity propagate better through the variables of WMEBP. Secondly, the windowed philosophy of WMVT is not exactly equal to the one of WMEBP. Recall that, to avoid possible erroneous states during runtime, it was decided that the effect of a leaving point would only be removed from the several variances at the moment the aggregation block is completely fulfilled. This causes the method to react more suddenly, but it also implies that the impact of the attack lasts longer. Lastly, these traces are not particularly fortunate in terms of self-similarity and the network load is often too small [23]. Additionally, the MIT/DARPA researchers utilised repetitive scripts to generate the background traffic, which reduce the confidence on any subsequent statistical analysis. The periodicity of the scripts can be observed if higher aggregation scales are used.

5.5.2. Analysis of Completely Synthetic Traces - Length of the Attack is Smaller Than the Observation Window Size

The experiments taken over completely synthetic traces were divided according to the most general situations that the windowed-modified estimators can face. This subsection discusses the case where the length of a network intensive attack is strictly smaller than the size of the observation window, while the antithetical scenario is explored afterwards. Though the investigation took most of the parameters of the simulation to be particularised, the conclusions included below can be extrapolated to different scenarios as well, as long as the underlying assumption of self-similarity is guaranteed.

During this part of the research, several synthetic traces were generated, transformed into suitable graphical representations and carefully observed. Figure 5.7 and Figure 5.8 contain the graphical representation of one of those traces, obtained when the simulation parameters were set to emulate an aggregation point performing at 1 Gbps, receiving approximately 10% of that as effective load, during a period of time of 30 s. The Hurst parameter of the inter-arrivals generator was set to 0.75 (refer to section 5.5.5. for a brief explanation of this value). Notice that a 4 s long network intensive attack was injected when the clock ticked the 10 s mark, effectively increasing the bit count per time unit during that period of time. As one can observe, the values of the Hurst parameter, estimated using the MVT and WMVT were also included for scrutiny. In this particular instantiation of the simulation, the observation window size was set to 2^{13} points (approximately 8 s), and the respective estimator was looking for the relations within the *last* 256 ms, using the usual aggregation scales of type $\{2^k\}_{k=0,1,\ldots,8}$.

Figure 5.8 brings the focus to three key moments of the histogram depicted in Figure 5.7. In there, three sliding observation windows are represented up to scale as semi-transparent boxes, for which the respective local context Hurst parameter estimates are emphasised as grey dots. After an initial period of instability, the estimated values of the Hurst parameter tend to 0.75, being that the value with which the self-similar traffic generator was initialised (accounting with the compensation). As the observation window slides from the left to the right part of the figure, it is possible to notice that: a) as soon as the attack starts, the estimates of WMVT increase significantly (to approximately 0.86);

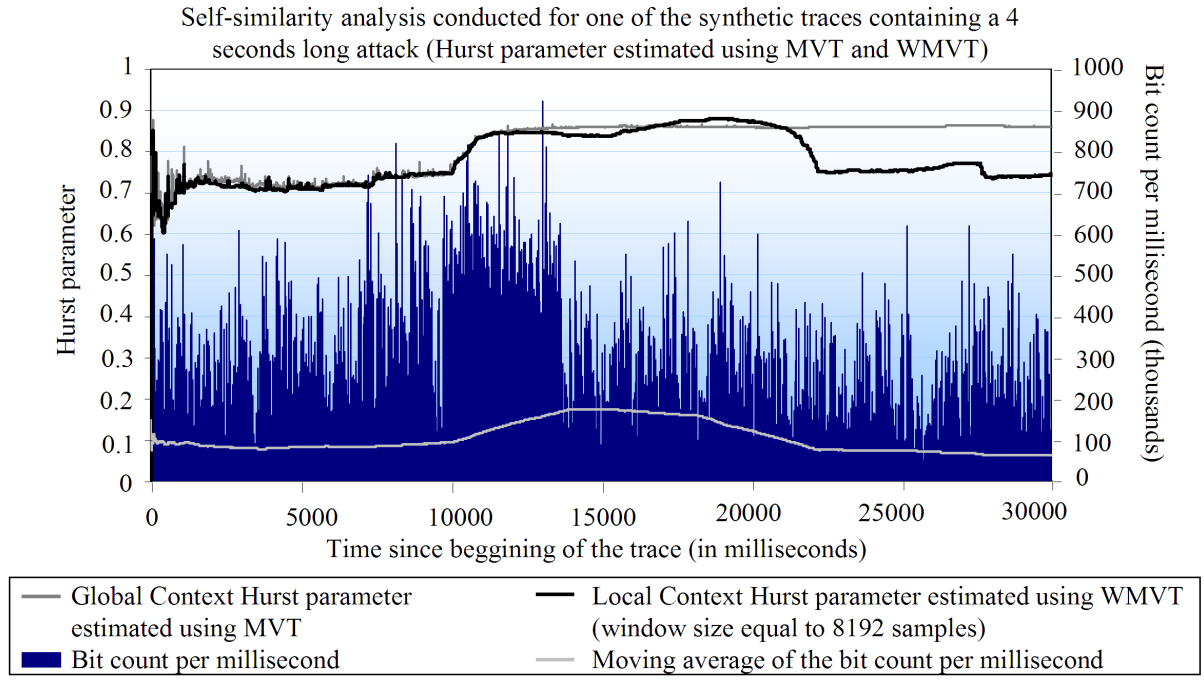

Figure 5.7: Histogram that reflects the self-similarity analysis conducted using MVT and WMVT for one of the many synthetic traces generated during this work. In this particular instantiation, the legitimate traffic simulator was initialised with $BW = 1$ Gbps, $L = 10\%$, $I = 10\%$ and $H = 0.75$. A 4 s long attack was injected at the 10th second.

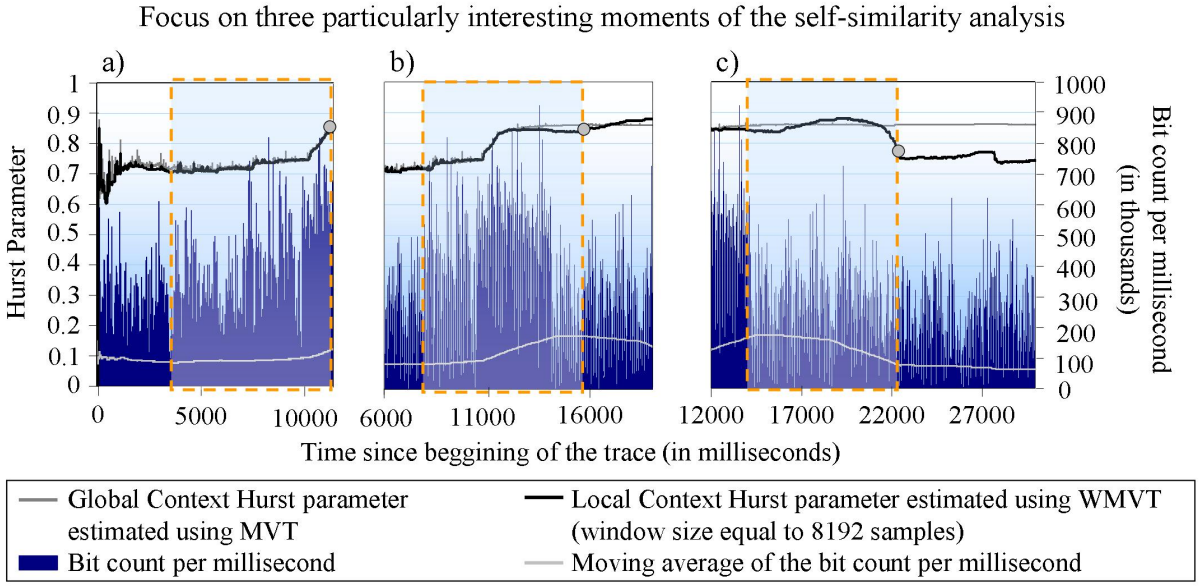

Figure 5.8: Focus on three key moments of the histogram depicted in Figure 5.7: a) the traffic concerning the attack enters the observation window; b) the intrusion is completely inside of the observation window; and c), the effect of the attack disappears from the estimates of WMVT.

b) they are bigger than the legitimate traffic self-similarity degree during approximately $4000 + 2^{13} \approx 12000$ ms (see below for the explanation of this value); and c), the values decrease to 0.75 once the observation window leaves the traffic concerning the attack behind.

This behaviour can be explained as follows. When the sliding observation window arrives to the point where the attack actually begins, the smaller scales are affected first, slightly *bending* the line used to estimate the Hurst parameter, instigated by the slight increase of the bit count per millisecond, resulting in a smooth increase of the local scope Hurst parameter values. As soon as the observation window *absorbs* the attack, it remains in an apparent higher state of self-similarity until the malicious traffic begins to exit the estimator. Thus, the total amount of time an abnormal activity affects the local scope Hurst parameter is expected to be approximately $D + w$, where D is the duration of the attack and w is the size of the observation window. The relatively fast decrease of the local scope Hurst parameter estimates depicted in Figure 5.8 c) reflects, once more, the slightly different window philosophy of the implementation of WMVT. The effect of the attack takes longer to be removed from the variables of the method than to be included. The increase of the WMVT estimates seen in a) is thus smoother than the decrease in c).

The global context Hurst parameter value remains high even after the attack ends, because the effect of the latter does not disappear easily from the *memory* of the incremental estimator. With time, the numbers retrieved by the point-by-point method would decrease to a value slightly higher than the expected one, possibly converging to it in the infinity (assuming that no other anomalies happen in the meanwhile). These comments enhance one of the best properties of the windowed-modified estimators: their *robustness*. The behaviour of the estimates of MVT is, in this case, highly dependent of the time frame of the attack within the experimental scenario. The incremental estimator had only the *time* to process 10000 values before the arrival of the intrusion, reason by which the method still reacted promptly to the attack. If the attack was inserted e.g. at the 22th second, then no abrupt change would be noticed.

To provide this explanation with an idea on how the Variance/Time plots change during an anomalous state of the network traffic, it was decided to construct and include Figure 5.9. The top chart of the figure enables one to compare the log-log plot of legitimate

traffic with the one where an intrusion is being analysed. The three smaller charts embody more detailed perspectives, where it is possible to detect the small departures from the exponential laws that define VT. As can be seen, during a moderate intensive attack, the logarithms of the variances of the aggregated processes are still best fitted by a line, though not as perfectly as before. The small decrease of the R^2 values during the phases depicted by b) and c) are consequence of an attack with an intensity of 100 Mbps, but they are not sufficient to irrevocably reject linearity. During this experiment, the local scope Hurst parameter evolved from 0.75 (Figure 5.9 a)) to 0.96 (Figure 5.9 c)). In the intermediary phase between those extremes (Figure 5.9 b)), the value returned by WMVT was 0.81.

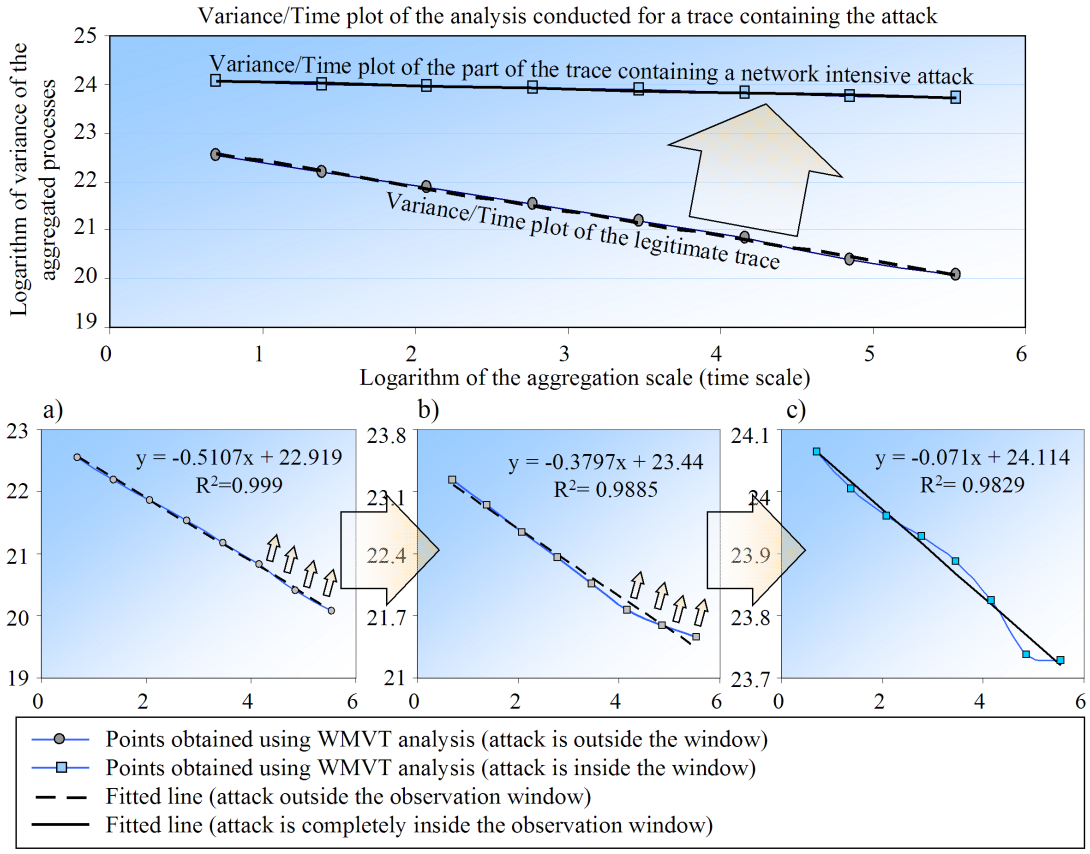

Figure 5.9: Evolution of the Variance/Time plots during the continuous analysis to a trace containing a network intensive attack (for exemplification purposes, $BW = 1$ Gbps, $L = 50\%$, $I = 20\%$ and $H = 0.75$): a) only legitimate traffic is being analysed; b) the attack is entering the observation window; and c), the attack is completely inside of the observation window.

Notice that the points in the chart of Figure 5.9 c) are not so well fitted by a line as the ones in Figure 5.9 a). This is one of the signs that self-similarity is not so well

embedded in the series under observation, since an attack occupying 10% of the total bandwidth already produces a considerable shift to the average bit count per millisecond.

5.5.3. Analysis of Completely Synthetic Traces - Length of the Attack is Bigger Than the Observation Window Size

The second type of simulated scenarios concerns the situation where the length of the attack is bigger than the observation window size. A possible illustration of such scenarios is included in Figure 5.10. To produce this illustration, the traffic generator was instructed to simulate a network aggregation point performing at 1 Gbps with an initial load of 70% and a network intensive attack at the 8th second with an intensity of 10% (or the available bandwidth). The Hurst parameter of the legitimate trace was adjusted to 0.80 (refer to section 5.5.5. for a brief explanation of this value). All the simulations were conducted under equal conditions for both estimation methods. However, after getting to the conclusion that the outcomes were similar, it was decided not to repeat the same graphical representation for the VT and for the EBP based analysis in both subsections. This time, WMEBP and MEBP were the ones responsible for the estimates of the local and global context Hurst parameter values. During this peculiar set of experiments, the size of the observation window was worth half of the length of the intrusion.

Once more, the results depicted in Figure 5.10 show that, when the traffic of the network intensive attack enters the observation window, the local context Hurst parameter estimates increase. The same happens to the incremental ones, due to previously discussed reasons. The main difference between the situation where the attack is shorter than the observation window and the one under investigation in this subsection happens after that initial transition period, as the outputs of the windowed-modified estimators return to the predefined value of 0.80 still during the time frame of the intrusion. In Figure 5.10, the semi-transparent box is precisely located at the point where its respective estimate is below the aforementioned mark.

The reasons that explain the initial increase of the Hurst parameter are already known. After the initial period, where the windowed-modified estimator is observing a trace constituted by two distinct parts (one containing legitimate traffic only and another

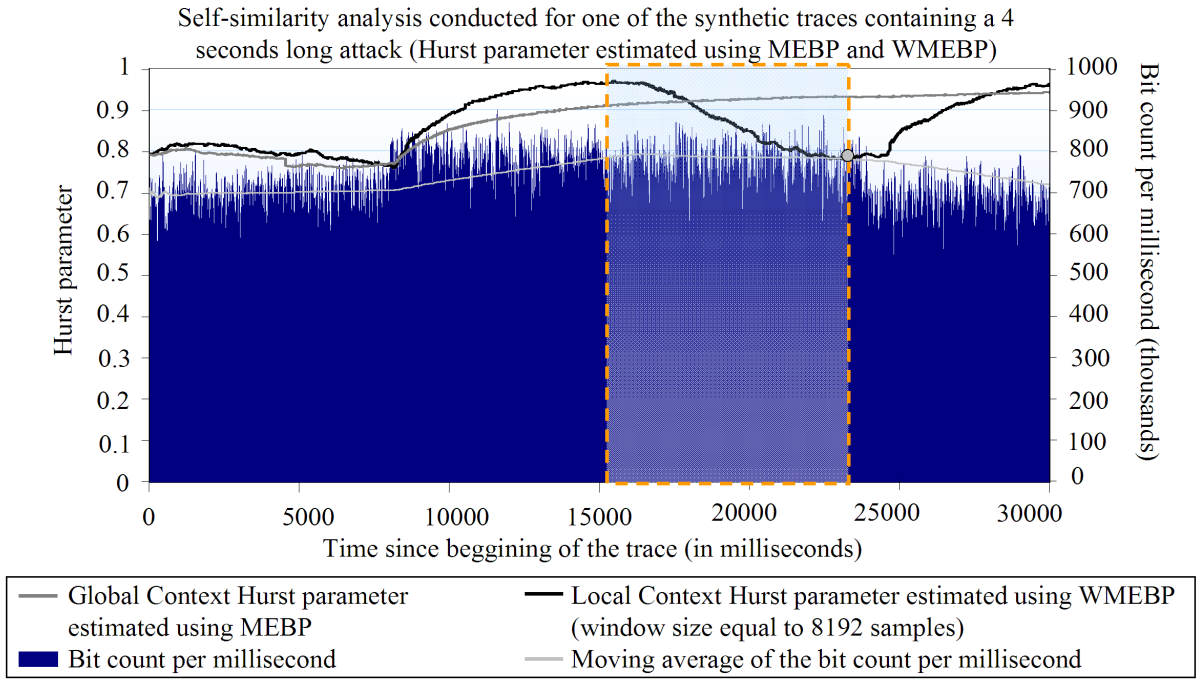

Figure 5.10: Histogram that reflects the self-similarity analysis conducted using MEBP and WMEBP for one of the many synthetic traces generated during this work. In this particular instantiation, the legitimate traffic simulator was initialised with $BW = 1$ Gbps, $L = 10\%$, $I = 10\%$ and $H = 0.80$. A 16 s long attack was injected at the 8th second.

one containing a mixture of legitimate and malicious data units), the scope of analysis moves to the portion containing the mixture of legitimate and malicious data units only. Both methods (WMVT or WMEBP) are then incapable of distinguishing that portion of the trace from a shifted version of the underlying legitimate trace. This means that self-similarity is preserved within the time frame of the attack. If the property is lost, such can only happen when the observation scope of the analysis includes the beginning or the end of the attack (or both) and part of the legitimate trace.

When the attack ends, the estimates of WMVT or WMEBP increase once more, mostly motivated by the change in the traffic load, and decrease right after that. Normality is restored as soon as the malicious traffic exits the observation window completely. Thus, the impact in the local scope Hurst parameter is noticed, for the last time, $D+w$ ms after the start of the anomaly. Within that time interval, the values of the windowed-modified estimators might come close to the values assessed prior to the beginning of the intrusion for a maximum period of $D - w$ ms (recall that, in this subsection, $D \geq w$).

5.5.4. Reaction of the Windowed-Modified Estimators to High Intensity Attacks

The network intensive attacks depicted in Figure 5.7 and in Figure 5.10 comprise intrusions with relatively moderate intensity. Figure 5.11 contains a representation similar to the previous ones, but for a network intensive attack that takes 3/4 of the available bandwidth at the given node (a very high intensity attack that injects approximately 1000 packets per millisecond). This attack does not provoke congestion yet, but it forces the moving average of the bit count per millisecond to rapidly shift to an higher value, causing an expressive loss of stationarity, and the loss of self-similarity. The plots on the figure where produced by initialising the simulator with $BW = 1$ Gbps, $L = 30\%$, $I = 75\%$ and $H = 0.75$. The size of the observation window (also schematised in the figure) was set to 2^{13}, and the WMVT method was looking the relations up to the scale of 2^8.

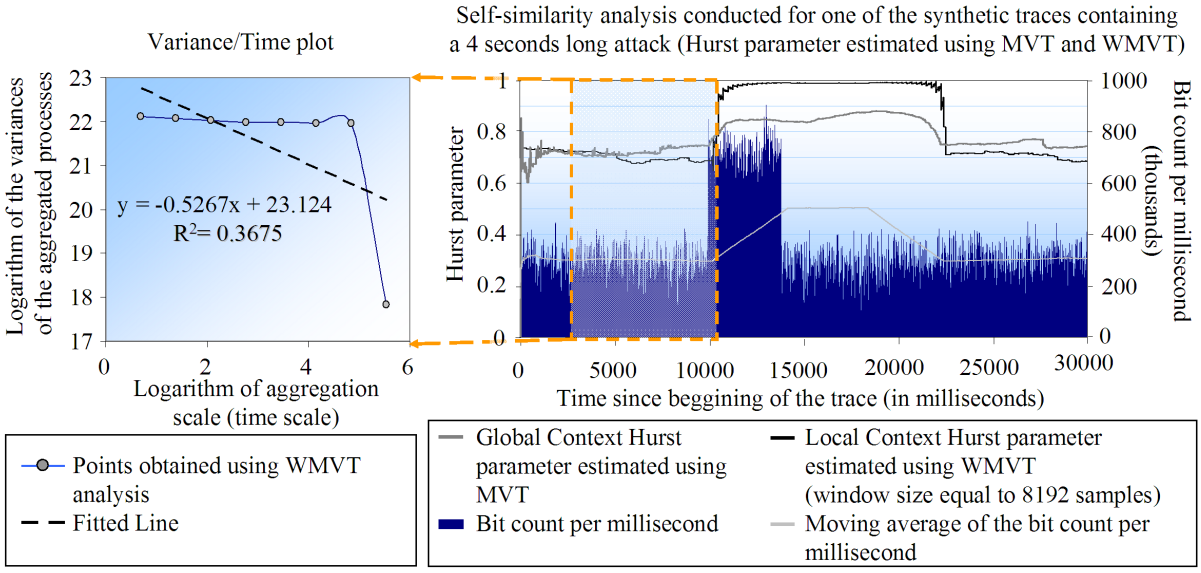

Figure 5.11: Variance/Time plot and histogram reflecting the self-similarity analysis conducted using MVT and WMVT for a synthetic trace containing a high intensity attack. The Variance/Time plot was obtained immediately before of the sudden increase of the estimates. The legitimate traffic simulator was initialised with $BW = 1$ Gbps, $L = 30\%$, $I = 75\%$ and $H = 0.75$, and a 4 s long attack was injected at the 8th second.

The reaction of the windowed-modified estimators to the sudden shift of the series is reflected in the rapid increase of the estimates to values close or equal to 1. The post-conditions of the procedures that assess the local scope Hurst parameter value in the

implementations of WMEBP and WMVT do not allow them to retrieve values bigger than 1 (nor smaller than 0), but on the left side of Figure 5.11 one may observe the ominous impact of high intensity attacks in the Variance/Time plot. This plot was obtained immediately before WMVT started to return values closer to 1 and, as it may be noticed from careful observation, the points obtained using the VT method are no longer over a line. The change of the statistical properties of the series under analysis was so quick and significant that the original linear law of the logarithms of the variances was *bended* in the higher aggregation scale considered. The largest aggregation scale was the last to be affected by the change, which explains why all the other statistics seem to be already in a different state than the last one. This effect could be noticed in other aggregation scales if the log-log plots were constructed several moments before the one in Figure 5.11. The value retrieved using the formula $H = 1 - \beta/2$ is 0.73 but, according to the value of R^2 included in the chart, the estimate should no longer be termed of Hurst parameter, even under loose terms (for more on this matter, refer to section 2.4.2. or to section 5.5.6.).

5.5.5. Detection of Network Intensive Attacks Based on Self-Similarity Analysis

Based on the previously described analysis, an attack detector was constructed and used to evaluate some of the most important properties that define an intrusion detection technique: accuracy, capability to point out the beginning and the duration of the anomaly, etc. This detector draws on the increase of the estimates returned by the windowed-modified estimators implemented within the scope of this work, and not on the loss of self-similarity, because it was noticed that the former was the main effect of any network intensive attack.

One may notice that the traffic simulator has been set to mimic systems performing at 1 Gbps along the previous sections. As mentioned in section 5.5.2., this was made so as to enable the empirical analysis, but the results may be extrapolated for other workload scenarios as well. Theoretically, self-similarity is a property that should be reflected in all aggregated series, meaning that if the bit count per time unit is long-range dependent for the seconds time unit, it should also be long-range dependent when aggregated for 10 s,

100 s, etc. This explains why modelling was performed for an abstract timescale ts in section 5.4.1.. Nonetheless, when dealing with empirical sequences, the referred property may not necessarily hold for all aggregation scales. Because network aggregation points handle nowadays debits of the order of the Gbps, and partially based on the aggregation scales used in the study [180], it was decided that the smallest aggregation scale used herein would be one millisecond, and that the simulator would try to reproduce 1 Mbit for that time quanta. The values assigned to the intended Hurst parameter of the traffic simulator were also inspired by the findings in the literature, where it is shown that it is common to find empirical traces with the self-similarity degree ranging from 0.75 to 0.85 (see, e.g. [39]).

Herein, the size of the observation window was chosen with the following two constraints in mind. On the one side, the referred parameter should be large enough to enable the analysis of *long*-range dependencies of the series under observation. On the other, the scope of the windowed-modified estimator should not be excessively large, at least not to the point of seeing the responsiveness to changes being jeopardised by the *law of large numbers*. By setting the referred parameter to 2^{14} in the simulations described in this section, it was possible to analyse the dependencies between samples separated by more than 511 units, while supporting the calculations in at least 8 (multiplicative) aggregations of the series. When operating below the seconds unit, 2^{14} may not be enough to embrace most types of network intensive attacks (the intrusions may last longer than 16 s [20, 21], and the self-similarity analysis falls into the case discussed in section 5.5.2.). If the same observation window is applied to higher time units (e.g. 10 s or 100 s), then the same analysis falls into the case studied in section 5.5.3.. Thus, it makes more sense to, for example, apply the same instantiation of the windowed-modified estimators to different time scales of an empirical bit count, than to try to adjust the size of the observation window to the type (or length) of the network intensive attack one wants to detect (adjusting the observation window size parameter could actually narrow the application scope of the estimator). One might even run several instantiations of the same detector for different time scales (or to different observation window sizes) in parallel, as a result of the low computational requirements of WMVT and WMEBP. This approach would nevertheless require a more specific adjustment of the parameters of the detector.

One of the notions that simplified the development of this procedure is the one of the continuous perspective over the local scope Hurst parameter values. The step-by-step estimates returned by the windowed-modified estimators are herein denoted by $H_w(t)$. The functioning of the detector and the experimental setup can be explained as follows. The detector was ran 50 times for each one of the 324 different combinations of the pair *traffic Load / Intensity* (L, I). The L parameter ranged between 8% and 93% (inclusively), and the I parameter varied between 5% and 95% (inclusively). Each trace was divided into different parts: an initial phase, not containing attacks, which is used to stabilise the windowed-modified Hurst parameter estimator; a second phase, also without attacks, which is used to calculate the average of the local scope Hurst parameter estimates, herein denoted by $\mathbb{E}(H_w)$, and of their absolute deviation to the average value, i.e. $\mathbb{E}_a(H_w) = \mathbb{E}\|H_w - \mathbb{E}(H_w)\|$; a third phase containing a 4 s attack; and a last phase constituted, once again, by legitimate traffic. With basis on the comments included in the previous sections and on the graphical perspectives provided by Figure 5.7, Figure 5.10 and Figure 5.11, the method was set to raise an alarm each time the local scope Hurst parameter estimates rose above $\mathbb{E}(H_w) + 2 \times \mathbb{E}_a(H_w)$ during, at least, 100 ms.

During a possible *status of anomaly*, none of the aforementioned values is updated by any means. The detector indicates if it has detected an anomaly or not and, in the affirmative case, it returns the time it started, and the difference between the maximum value retrieved by the windowed-modified estimator and the average one $(H_w^{max} - \mathbb{E}(H_w))$. This difference is then normalised (divided by $1 - \mathbb{E}(H_w)$), and used as a measure of the *intensity* of the attack.

The 3D charts in Figure 5.12 depict precisely the average values of the maximum (normalised) difference obtained for each one of the two estimation methods (the ones concerning WMVT may be found on the left side of the figure, while the ones concerning WMEBP are located on the right side). They were obtained by setting the intended Hurst parameter of the traffic simulator to 0.80 and by initialising the size of the observation window of both modified estimators with the value 2^{14}. Though the surfaces exhibit several resemblances, it is noticeable that the estimates of WMEBP react faster to low intensity attacks than WMVT. The implementation of WMVT seems to take longer to reflect the effects of the shift caused by the attack, because the variances are not as

susceptible to changes in the input series as the numbers of crossings are. Though, for each value of L, there is not a linear relation between I and $(H_w^{max} - \mathbb{E}(H_w))/(1 - \mathbb{E}(H_w))$, the latter complements the idea of the statistical weight that the particular intense flow has achieved in terms of the bandwidth. In the chart concerning the WMVT analysis, and for higher values of bandwidth utilisation, the values of $(H_w^{max} - \mathbb{E}(H_w))/(1 - \mathbb{E}(H_w))$ seem to react almost linearly to the increase of the intensity, but that behaviour vanishes as the load decreases.

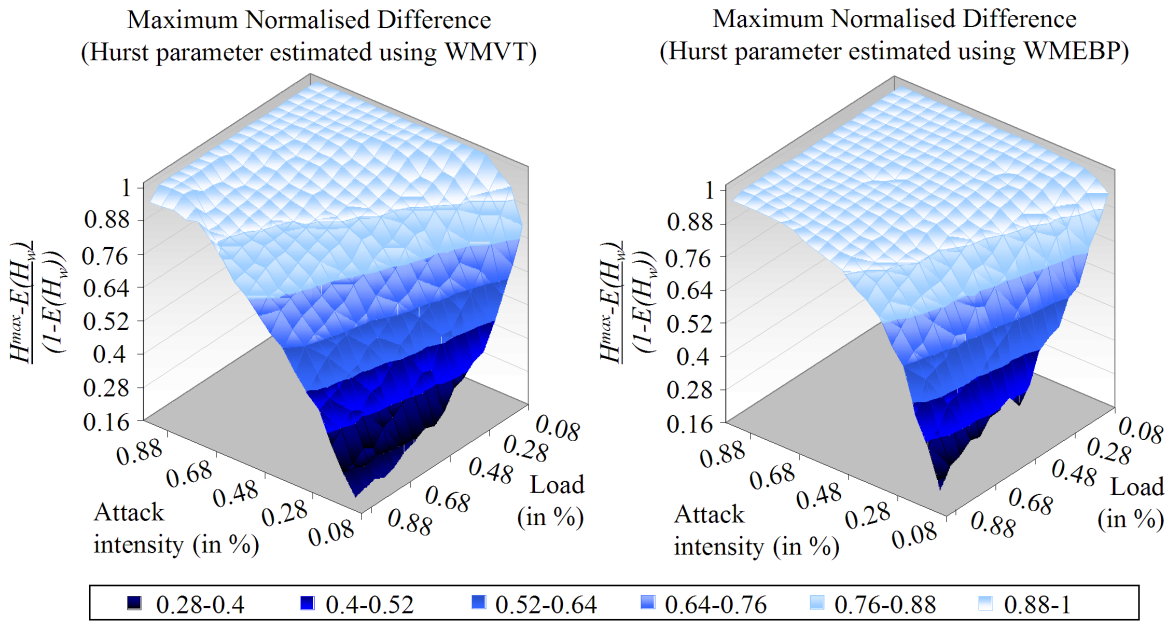

Figure 5.12: Maximum (normalised) difference between estimates of the local scope Hurst parameter taken before and during a network intensive attack. The chart on the left concerns the analysis conducted with WMVT; the chart on the right concerns the analysis conducted with WMEBP.

The approach taken was capable of identifying approximately 76.6% of the 16700 simulated anomalies. In the detected ones, the time span between the 16 s and the 27 s (average value of 17199 s and a standard deviation of approximately 973 ms) was pointed out as the first time that the local Hurst parameter values surpassed the predefined threshold, which may be considered to be the beginning of the attack. The left side of Figure 5.13 contains the graphical representation of the moments estimated as being the starting times of the anomalies as a function of L and I. In the depicted instantiation of the simulation, the attack was set to start exactly at the 16384th millisecond. The average value of the local scope Hurst parameter estimates was approximately 0.79 and

the average value for the absolute deviations was always smaller than 0.01, meaning that an alarm was raised each time an estimate was bigger than 0.81 for 100 ms. The shape of the surface suggests that the procedure described in this section is particularly accurate in pointing out this aspect of the anomaly, since excepting the cases where the proportion Intensity / Load is too small, the 16485 ms ($= 2^{14}+101$ ms) are indicated as the beginning of the anomaly. Notice that the values represented in both charts of Figure 5.13 concern both types of estimators (WMEBP and WMVT), since they were obtained via statistical treatment of the entire set of results.

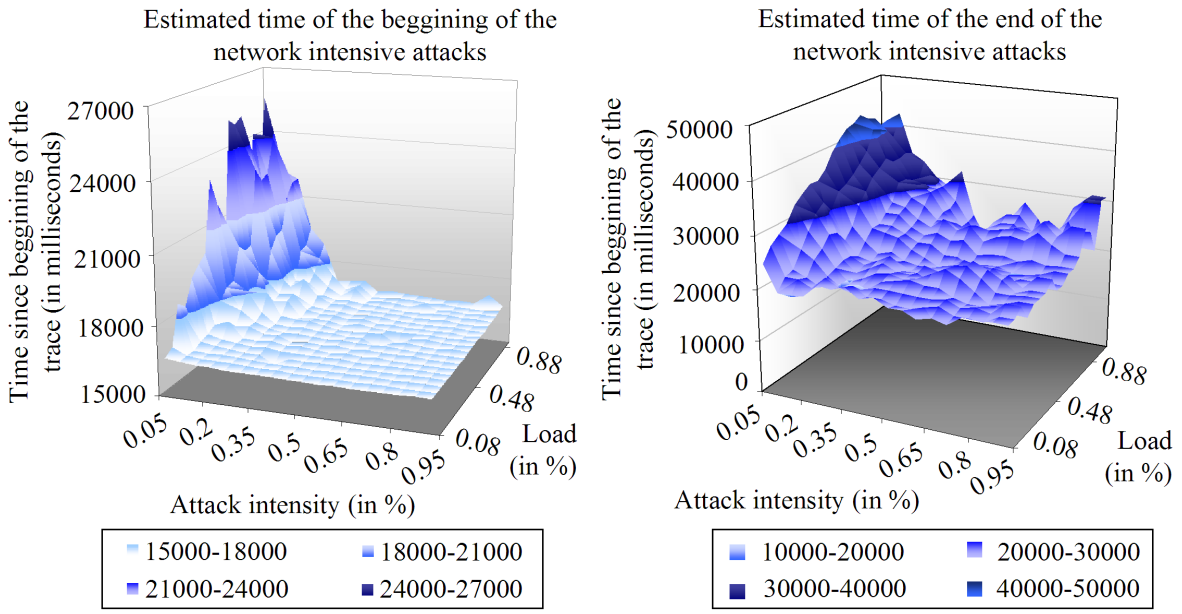

Figure 5.13: Graphical representation of the moments that the detector based on self-similarity analysis indicated as being the beginning (left side of the figure) and the end (right side of the figure) of the network intensive attacks.

The 3D surface on the right side of Figure 5.13 demonstrates that the end of the attack is not as easy to point out as its beginning. The end of the attack was set to be at the 20384 ms, but the detector returned values ranging between the 18600 ms and the 44251 ms (with an average value of 24746 ms and a standard deviation of 5278 ms). From careful analysis of the figure, it seems clear that the values tend to stabilise as the attack intensity increases, but that trend is not so strict as for the previously discussed aspect. The fact that the detector seems to be overestimating the end of the attacks is mostly due to random states of congestion induced by the injection of malicious traffic, which causes the anomaly (or its effect) to last longer. While the intensity of the attack

is herein defined with respect to the available bandwidth, the burstiness of the traffic may eventually induce states of congestion when interacting with the injected data units. According to what was said in section 5.4.4., this forces the simulator to retard all the protocol data units, effectively pushing the end of some anomalies further into the future.

The results included in this section, however, have to be understood under the premisses in which they where produced. In this case, the traces were synthetic, thus bearing a strong self-similar structure and the anomalies where simulated by inserting a considerable amount of packets into the traces. All these conditions led to a detection rate of 76.6%. Nevertheless, it should be stressed out that any traffic flow arriving to the network aggregation point with the aforementioned characteristics would always be tagged as an anomaly, even when coming from a legitimate source or application. However, for high utilisation rates, having an application behaving like a network intensive attack would comprise a rather unusual situation, and one could even wonder if such flow comprises a threat to the network.

5.5.6. The Loss of Self-Similarity

As stressed out in section 5.3.2., most of the references in this area draw on the loss of the property of self-similarity to construct a detector for network intensive attacks. In e.g. [20, 25], the loss of the referred property is often associated with estimates of Hurst parameter that are smaller than 0.5, or larger than 1. By definition of self-similarity, such association is not necessarily true, since it is possible to obtain values of the Hurst parameter within those limits, independently of the process being self-similar or not. In order to assess if the loss of the referred property is an unavoidable consequence of the above mentioned malicious activities, two different statistical tests were implemented in the scope of this research work. A test based on the well known *Kolmogorov-Smirnov goodness of fit test* (KS test), and another one based on the *coefficient of determination* (the R^2 value) of the linear regression used in the VT method (for more on this subject, refer to section 2.4.2.). The former was used to evaluate if the distribution function of a given trace containing an attack is similar to the distribution function of several aggregations of the series under analysis, while the latter measures the quality of the

estimates outputted by MVT and WMVT.

The KS test elaborates directly on the definition of self-similarity for discrete time series, which states that the (finite dimensional) distribution of $\{Y(t)\}_{t\in\mathbb{N}}$ should be equal to the (finite dimensional) distributions of the aggregated series $\{m^{1-H}Y^{(m)}(i)\}_{i\in\mathbb{N}}$. In resemblance to what was done for the estimation of the Hurst parameter and for the generation of self-similar series, the aggregation scales considered for this test were the ones of type $m_k = 2^k$, with $k = 1, 2, ...$ For each m_k, the distribution of $\{Y(t)\}_{t\in\mathbb{N}}$ is compared to the one of $\{m_k^{1-H}Y^{(m_k)}(i)\}_{i\in\mathbb{N}}$, in accordance to the guidelines of the KS test (explained below).

The occurrence of a network intensive attack corresponds to the introduction of a shift in the self-similar series and to a consequent change of its statistical properties. Before the beginning of the attack, and even during the intrusion, self-similarity is kept at a local level, but the same may not happen during the period in which the observation windows of WMVT and WMEBP move from the part containing legitimate traffic only, to the one containing malicious data units also. It was precisely in that period of time that the subsequent analysis was focused on.

Several data series that mimic the time span in which the beginning of the attack is somewhere in the middle of the observation window were simulated. After that, the sequences of values were normalised and aggregated, being the respective distribution functions constructed to each one of them. All the data series were 2^{19} points long to obtain a strict idea of the distribution functions (see plot in the right side of Figure 5.14), and the empirical frequencies were taken by dividing the interval of occurrences into 400 equal buckets, and by counting the incidences falling in each subinterval. In order to apply the KS tests, the largest distances D_k (with $k = 1, 2, ...$) between the cumulative distribution functions of the aggregated series and the one of the initial series were taken and compared with the critical values in [181]. When D_k was smaller than the critical value (for the degree of significance of 0.05), the test was considered as being *passed*. Simultaneously, it was asked to the class implementing MVT to return the R^2 statistic of the of the analysis of the lengthy sequences. This was made so as to obtain a more accurate idea of what might happen when a windowed-modified estimator is analysing a potentially smaller number of values. All the conclusions and results of this section were

consolidated by repeating each particular experiment 30 times.

The charts depicted in Figure 5.14 summarise the investigation concerning the KS tests and apply to the scenario were the size of the observation window was of 8192 ms. The KS tests were applied up to the 10th aggregation scale. The results presented in the left side of the figure show that the resilience to shifts depends of the self-similarity degree. Though the dependence does not seem to be explained by an explicit formula, it is noticeable that, as the Hurst parameter increases, the larger is the shift supported by the sequence, before the hypothesis of self-similarity can be reject. If the criteria for loss of self-similarity was considered to be the failure of two of the KS tests implemented herein, it would be possible to conclude that for values of the Hurst parameter between 0.75 and 0.85, a self-similar trace could handle attacks with intensity as large as the standard deviation of the legitimate bit count per time unit process, before loosing the self-similarity structure.

The particular case where the Hurst parameter of the generated series is 0.85 is represented by the chart in the right side of Figure 5.14. This plot was randomly selected from the set of results obtained when the shift was set to be 0.5 (half the standard deviation), and the main purpose of its inclusion here is to show that such shift does not produce a drastic impact in the several distribution functions depicted in there. The utilised goodness of fit test signalled loss of self-similarity for the first aggregation scale only, since it is for that scale that a larger number of samples is available (and, as such, the test is more stricter). As stressed out in chapter 2, the last phase of almost all of the estimators of the Hurst parameter draws on the approximation of a given set of coordinates by a line, using regression analysis. The value of the *coefficient of determination* may thus be understood as a measure of how well self-similarity is reflected in the realisations of the process under analysis. R^2 varies between 0 and 1, arguing in favour of the quality of the approximation for values closer to the superior limit. The chart located in the left side of Figure 5.15 shows that the presence of network intensive attacks affects the exponential law in which the general VT method is based on. Nonetheless, if the failure of the linear regression was defined in terms of the so called *F test* (refer to [80] or to section 2.4.2. for more on this matter), then only after considerable intensities one could say that self-similarity was lost. For reading commodity, the surface represented in the

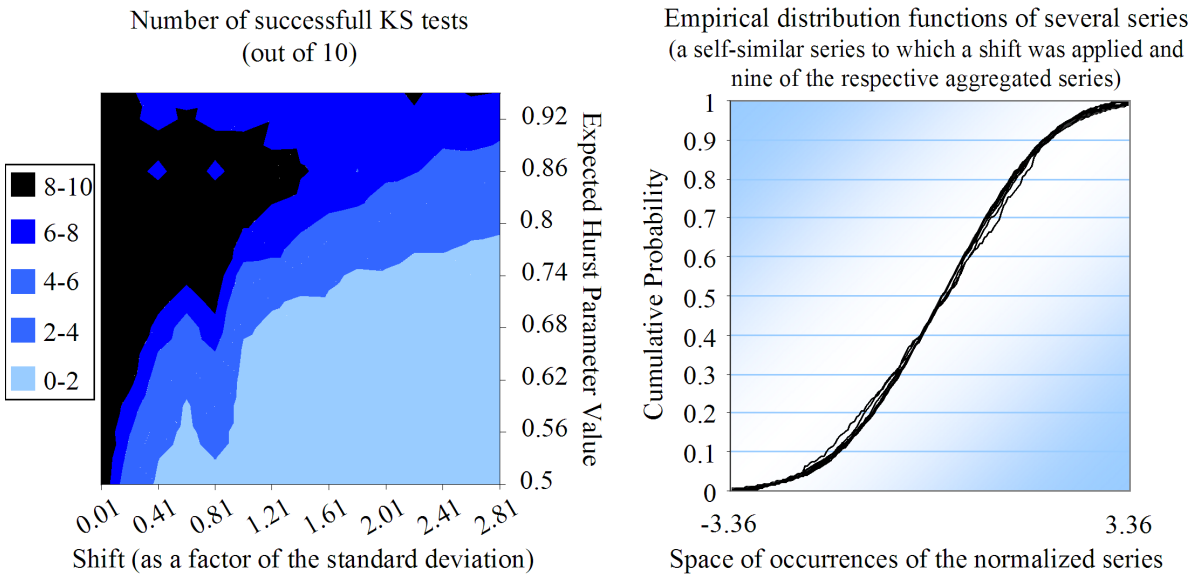

Figure 5.14: Part of the results that summarise the tests made to the resilience of self-similarity. On the left, the average number of well succeeded KS tests (out of 10) is represented as a function of the Hurst parameter and of the shift induced to the series under analysis; on the right, one may find the representation of the cumulative distribution functions of 10 aggregations of a normalised self-similar series with 2^{19} points (with an intended Hurst parameter of 0.85), to which has been applied a shift of 0.5 units.

chart is differently coloured depending on the failure (dark blue) or non failure of the F *test* (light blue). During anomalies with moderate intensity, and under the assumption that the background traffic presents a reasonably good self-similar structure (with an Hurst parameter of 0.85), the value of R^2 (and consequently of F) is kept high, being impossible to reject the hypothesis of existing an exponential (fractal) relation between the process and its aggregations.

The chart on the right of Figure 5.15 was included with the purpose of showing the behaviour of the estimates of the Hurst parameter, during the previously described and tested scenarios. As one may observe in this figure, the success or failure of the KS tests, or the quality of the fitting used in MVT (and in WMVT), do not seem to directly influence the values outputted by the estimators, which tend stringently to 1, as the shift increases. As a final remark, it should be said that the implementations of the several methods used herein do not allow the estimators to output values out of the range $[0, 1]$, for coherence reasons. While by definition, the methods VT and EBP are in

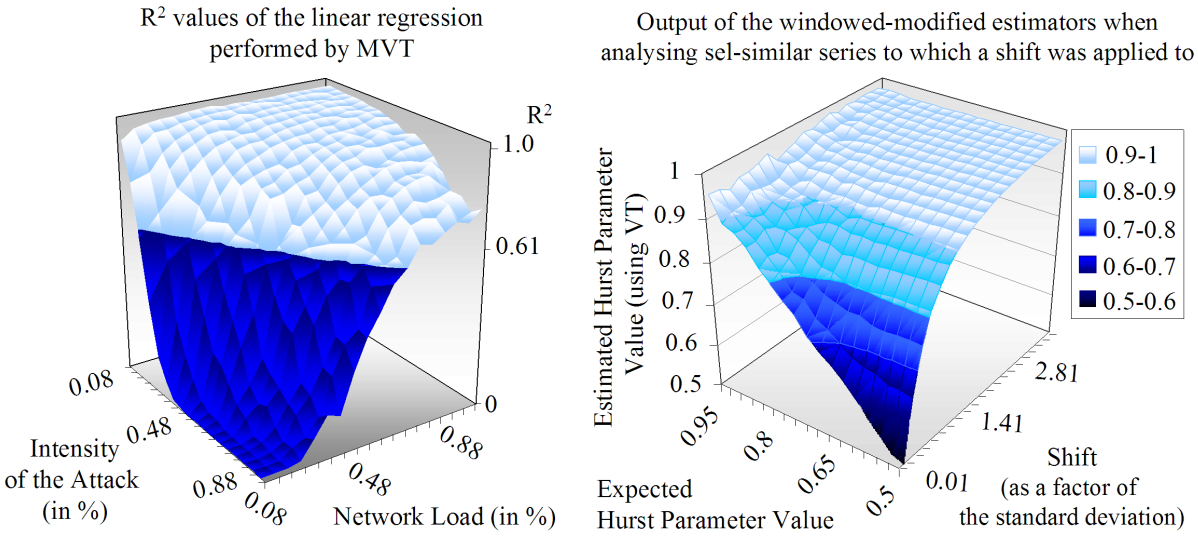

Figure 5.15: On the right side of the figure, the minimum values of the R^2 statistic are plotted against the network load and attack intensity parameters. The plotted values of R^2 concern the linear regressions performed by the MVT method during the analyses of the simulated traces; the surface on the left shows the relation between the expected and the estimated Hurst parameter values, as a function of the shift induced to series with 2^{19} points.

accordance with such post-conditions, there are specific circumstances that could induce the methods to return values out of the range, specially in the cases where the properties of the empirical series do not comply with the ones of self-similar processes. Nonetheless, those states are erroneous and, as such, the author decided not to let them happen, and assess the loss of self-similarity using other means. Under expressive lacks of stationarity, the two estimators used within the scope of this work interpret the series under analysis as being a shifted and scaled 0/1 process, returning estimates close to 1.

The results prove that the task of deciding that self-similarity is lost during a network intensive attack is not straightforward. The impact on the statistical distribution itself is highly dependable of the variability of the traffic, and the tolerance to small lacks of stationarity increases with the Hurst parameter (at least for values smaller than 0.9). Notice that high-debit aggregation nodes constitute scenarios particularly favourable to the preservation of self-similarity, since all the aforementioned factors (mostly variability and network load) tend to be larger in such points of the network. Drawing on the loss of self-similarity for detection of intrusions in a real network environment is thus even more difficult.

5.5.7. Discussion on the Theoretical Framework of the Results

From the results included in the previous sections, it is possible to conclude that (i) the property of self-similarity is not necessarily destructed by network intensive attacks and that (ii) the values outputted by the local scope estimators increase every time a constant flow of traffic is injected into the network. In this section, a theoretical framework for these facts is proposed.

As demonstrated by the rationale behind 4SG, the first order differences process of a self-similar series may be approximated using a sum of several constant components. The duration, amplitude and sign of which determine the fractal behaviour of the respective process. The Hurst parameter increases from 0 to 1 as the extension and amplitude of the constant parts increase. During periods of normal functioning of the network, the constant components are nothing more than the aggregation of several information flows, remotely (and randomly) generated, and directed by the forwarding systems (routers or switches) to the point where they continuously feed the process of the quantity of information per time unit, conferring it with a more or less persistent behaviour. During a network intensive attack, the two most important factors of the three above mentioned ones (duration and amplitude) are both affected positively, resulting in the reinforcement of the (local) constant part of the analysed series.

The aforementioned effect is illustrated in Figure 5.16, where it is shown that the insertion of an intense flow of traffic may be seen as the addition of two new symmetrical and constant components, each with duration equal to the one of the flow. The symmetry occurs when the flow starts. Even though the two concepts are not directly comparable in the general sense, in this particular case, an increase of the *constancy* may either strengthen or destroy *self-similarity*. Either way, it can never result in the decrease of its degree. The insertion of malicious traffic results always in a loss of stationarity, but that loss may not necessarily result in the destruction of self-similarity. It is known (for example from [38]), that the stationarity of the processes exhibiting long-range dependencies is hard to evaluate, since the very nature of self-similarity the local scope drifts of their statistics. If observed from a naive perspective, those drifts may be confused with lack of stationarity while, in fact, they are nothing more than manifestations of the bias introduced by the

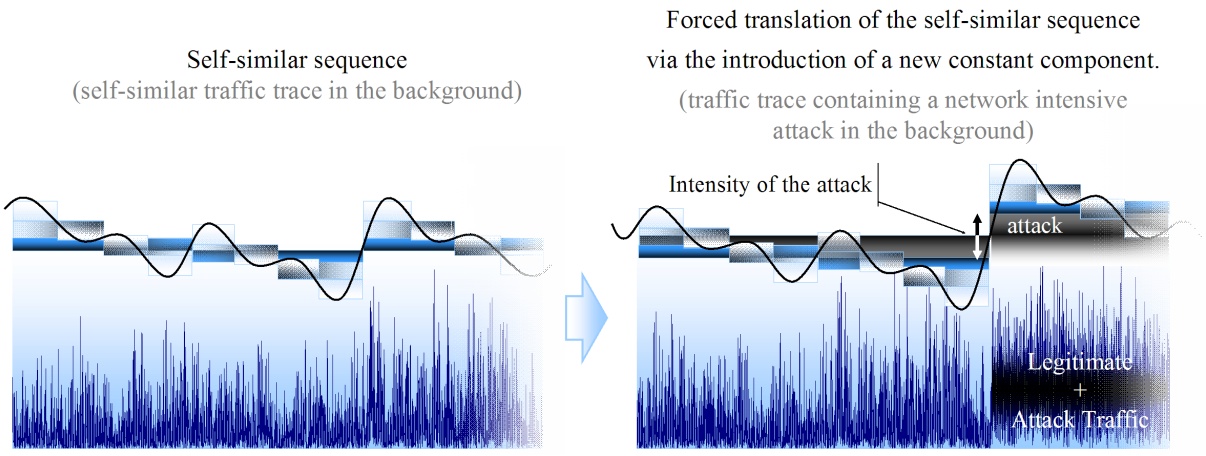

Figure 5.16: Illustration of the effect of summing a constant component to part of a self-similar signal. Analogy with the anomalous situation that may induce such effect in the network traffic trace.

(self)correlations. The introduction of relatively modest (in relation to the network traffic load) malicious traffic flows may eventually result in small and ephemeral shifts of the bandwidth occupied, which increases the self-similarity. Incapable of assessing how well the input series exhibits self-similarity, all that the implemented estimators can detect is, basically, the transformation of a variable process into a more stable one, for which the estimates of the Hurst parameter can never be smaller.

5.6. Conclusion

This chapter discussed the evolution of the self-similarity degree during intrusions with non-negligible expression at the network traffic load level, and it described how the investigation was conducted. Most of the results contained herein were obtained by simulation, but supported by two different Hurst parameter estimators and by empirical observations. The possible loss of self-similarity during the referred anomalies was tested using two completely different statistical approaches, and all results are discussed from the theoretical point of view.

As far as the author could assess, self-similarity analysis is not currently part of any network security solution. On the one hand, the efficient construction of the continuous perspective of the local scope Hurst parameter is not that common; on the other, such per-

spective may embody a more attractive feature for real-time network traffic management mechanisms, where it may be applied as a measure of burstiness. The factor mentioned in last is, actually, the one with which self-similarity has been associated to for the longest time. Some of the literature references concerning the application to the network security topic define the status of loss of self-similarity as being the one where the estimates of the Hurst parameter fall outside of the interval $[0.5, 1[$, and base the detection of network intensive attacks on that assumption. The analysis described in those references is often based in a single Hurst parameter estimator, which is either applied to the entire data series or to non-overlapping intervals.

The intermediate part of the chapter was dedicated to the presentation of the means used to generate synthetic traffic traces and the (herein denominated) *network intensive attacks*. While the analysis of pre-collected traces may be sufficient to tackle the general problem of the thesis, the detailed observation of the evolution of the self-similarity degree during a network intensive attack can only be attained in a fully controlled scenario. The synthesis of the traces is mostly focused on the simulation of the *unused* part of the available bandwidth at a given network aggregation point. The proposal draws on the sequential generation capability of 4SG to enable the creation of the traces in a packet-by-packet manner, which suits well the purposes of this work, easing the task of simulating attacks as well. To enable the usage of an empirical packet size distribution, the inter-arrival times are modelled as occurrences of a shifted and scaled fGn, which confers the self-similarity property to the synthetic trace. The modelling and discussion of the results were oriented towards the possibility to generalise the conclusions taken from the simulations.

The four modified estimators were first applied to the traces of the MIT/DARPA set containing labelled attacks. Regardless of the evident flaws in terms of the load and self-similarity structure of those traces, several intrusions resulted in the increase of the estimates of the Hurst parameter, providing for a first impression of what the behaviour of the self-similarity degree could be. The very specific perspective given by the windowed-modified estimators delimit the two scenarios such procedures may face. For the case in which the size of the observation window is smaller than the length of the attack, the estimates of the local scope Hurst parameter are larger than the ones of the legitimate

trace while the time span of the attack is inside the observation window. In the opposite case, the values returned by the windowed-modified estimators increase as the traffic concerning the attack enters their processing scope, but they decrease to values close to the ones of the legitimate trace as soon as the observation window is completely absorbed by the mixture constituted by malicious and legitimate traffic. As soon as the attack ends, there is another increase of the estimates, which return to normal levels when the traffic concerning the attack leaves the observation window completely.

Based on the results, a method for the detection of intensive flows of data that draws on self-similarity analysis was implemented. Though simple, the detector is fast and adaptable to the network conditions. Due to the nature of the analysis itself, the approach is unsuitable of looking into the contents of the packets and decide about their validity, but the implementations of the estimators proposed herein are capable of raising useful alerts, that may be used to trigger other in depth investigation procedures. As the computer-based simulations were made as if being done online, they provide for an immediate idea of the computational performance of monitoring traffic using the modified estimators. The simulations were run in a Pentium IV 2.8 GHz computer with 504 MB of RAM, and the results of the generation and analysis of traces with 30 s were delivered in less than 1 s. Because the time unit was the millisecond, and because the apparatus involved the generation of the legitimate trace and the injection of the attacks, it may be concluded that it would be possible to process the byte count at a granularity smaller than the tenth of millisecond using a dedicated monitoring system.

It was equally demonstrated that, in the highly dynamic and demanding network environment, it may be difficult to conclude that the presence of a network intensive attack results inevitably in the loss of self-similarity. It was concluded that the complete destruction of such property depends not only of the self-similarity degree of the traffic by the time of the intrusion, but also on the expressiveness of the loss of stationarity which, by its turn, depends of the ratio between the intensity of the attack and the amount of bandwidth occupied by the legitimate traffic. Independently of the loss or preservation of the property, the values returned by the utilised instantiations of WMVT and WMEBP increase in the presence of a network intensive attack, always signalling an apparent increment of the self-similarity degree.

Such behaviour may be summarily explained as follows. During normal operation, the several individual flows arriving to the aggregation point contribute equally to an highly dynamic process of the bit count per time unit. Depending of the time span from which the process is observed, it may be said that the series is constituted by two different type of components: a constant and a variable one. The occurrence of an intensive attack increments the constant component for some moments, inherently increasing the (self-)similarity between the values fed to the windowed-modified estimator. With the advent of bandwidth demanding services and faster transmission technologies, the scenario where network intensive attacks actually provoke loss of self-similarity may potentially become less common. According to the results in this chapter, the bandwidth occupied by the malicious data units would have to be as high as 2.5 Gbps (∼312500 SYN packets per second), in a 10 Gbps network point with 50% of effective bandwidth utilisation, so as to produce the above mentioned effect.

As shown in this chapter, the behaviour of the Hurst parameter during network intensive attacks and the tools developed along this research work can be used to construct a detector for such kind of intrusions. Nonetheless, this type of analysis is constrained by several factors. First of all, it relies on the assumption that the bit count per time unit of the legitimate traffic is (and will be) asymptotically self-similar, at least in an approximate sense. Such may not hold under light load conditions or in points where the statistical characteristics of the aforementioned process are changed by specific control mechanisms. Secondly, the application scope of such detector is limited to points with high level of networking, thus embodying a possible solution for NIDSs only. Lastly, it should be mentioned that any particularly intense flow (even a legitimate one) will produce the same kind of effect that network intensive attacks do, being impossible to distinguish them just by self-similarity analysis. Normal traffic flows are not expected to achieve the same order of magnitude that intentional DoS or DDoS attacks do, and should be treated as threats otherwise.

Amongst the best advantages of the presented approach is the fact that it concretises a TCD mechanism, being thus completely independent of the contents of the protocol data units. It is also protocol agnostic and operates at the highest layer of abstraction one could define for the network stack. The continuous perspective over the evolution of the local

Hurst parameter values is new, mostly because the proposed modified estimators are very efficient in terms of computational resources.

The massive adoption of the computer as a working tool, the proliferation of novel telematic applications and the ease to connect to the Internet, reinforce one of the factors that makes the network traffic self-similar: the number of renewal processes. The evolution of the services will dictate to which extent the property studied herein will be reflected in the network traffic. By now, the common practice of using multiple web based applications simultaneously (e.g. tabbed web browsing, mail) along with VoIP, IPTV and file sharing software has result in the increase of the number and heterogeneity of independent flows near the terminal machines, inherently reinforcing the referred property.

Chapter 6

Final Conclusions and Future Work

This chapter assembles the most important conclusions taken from the research work this thesis represents. It also presents some topics / problems the author would like to address in the future. Part of those topics were identified as potential subjects of interest at least in one of the several areas related with this work, others result from the natural development of the ideas discussed herein.

6.1. Main Conclusions

This thesis describes a research work in the area of traffic monitoring and statistical analysis. Its main purpose was to study the behaviour of the statistic known as Hurst parameter during network attacks, so as to evaluate to what extent that knowledge can be used to detect network intensive attacks. The four intermediate chapters that constitute the body of this thesis explain with detail all the phases of the work and constitute, simultaneously, one of the most systematic and complete studies conducted in the context of the self-similarity analysis of the network traffic. The scientific contributions of this thesis span from the area of statistical analysis to the ones of intrusion detection, computer-based simulation, traffic monitoring, and algorithm development and optimization.

During the course of the work, several modifications to estimators of the self-similarity degree were proposed and tested. The incremental and windowed versions of VT and EBP are algorithms with $O(n)$ complexity and relatively scarce memory requirements. The algorithms may be seen as excellent real-time network monitoring tools

and, even though that specific usage was the main motivation behind their development, their application scope is not confined to that area, as further elaborated below.

The adaptation of the Hurst parameter estimators to windowed calculation procedures encompasses some disadvantages, mostly related with the temporary instability caused by the incremental construction of the aggregated blocks, but it also results in some advantages. The windowed estimators enable one to adjust the trade-off between the sensitivity to ephemeral changes of the self-similarity degree and the statistical significance of the estimates. They are also more *resilient* to momentary losses of stationarity because, contrarily to the point-by-point or retrospective estimators, their operational model enables the complete readjustment of the statistics on which they are based on during runtime. This fact comes in favour of their application in the online analysis of empirical data series, which rarely possess the characteristics of a normalised stochastic process (e.g. real network traffic traces).

Motivated by the need to test the modified estimators and simulate online traffic conditions, two new methods to approximate fGn and fBm processes exhibiting self-similar behaviour were proposed. One of them (4SG) is presented as being amongst the most efficient algorithms of its class of precision. Its $O(n)$ computational complexity is hard to surpass, since it requires only an average of four sums, two multiplications and two calls to a GRNG per point generated. On a Pentium IV processor running at 1.61 GHz, it is able to produce over 1300 points per millisecond and it does not require more than the space for storing 127 floating point values (approximately 1 KB of memory) to generate a data series with $2^{128} \approx 3.4 \times 10^{38}$ points.

Unlike other approximate methods for the generation of self-similar sequences, the low memory prerequisites of the two proposals are related with the strategy adopted for the construction of the approximated fGn *per se*, and not with any type of *truncation mechanism* that reduces the extension of the dependencies of the points of the fGn process. 4SG stores only the immutable parts of the series, and regenerates regularly the ones for which the life cycle ends, while fBm-SGA elaborates on a probabilistic technique, generating future points from strategically selected points of the past.

The tests to the quality of 4SG show that the method is worthy of trust for simu-

lations built on top of self-similarity related models. It generates points on demand, and the performance of the algorithm is not degraded as the number of outputted values increases. Thus, the algorithm embodies a specially attractive solution for the reproduction of situations where the analysis is made as the points become available (online conditions), or for the fast generation of arbitrarily long sequences of data. Due to the low memory usage and efficiency, 4SG is perfect for the simulation of traffic with the self-similarity structure. In the opinion of the author of this thesis, 4SG is one of the best outcomes of this research work.

Included in this thesis is also the proposal of a new means to generate pseudo random numbers. The algorithm referred to as PRR is one of the embodiments of a family of generators, whose development is inspired on the numbers theory and on the codification of the fBm-SGA as an array of GRNGs. It was used as the primitive of the self-similar sequences generators, whose operation depends strictly of the quality of the underlying source of randomness. PRR is simple to understand and implement, and also highly portable. Amongst its best features lies its fairly good computational efficiency and the possibility to improve the quality of the outputs without impacting its performance. The drawback is that there is an inversely proportional trade-off between the referred aspect and the requisites of memory. The implementation of the PRNG used within the scope of this thesis passed all the tests of randomness to which it was subdued to, and its period was assessed as being equal to 2^{60}. PRR is capable of producing a sequence of approximately 11×10^6 high quality pseudo random numbers in about 1 second on a Pentium IV 2.4 GHz processor, using 4140 B of memory.

The network traffic generator developed along the research work is suitable for the synthesis of packet-by-packet traces, meaning that it is able to simultaneously output occurrences of the packet size and inter-arrival times processes. The simulator is presented as a direct application of fBm-SGA or of 4SG, from which it inherits their sequential generation capabilities and computational performance. As defined, the method can be used to generate long traces exhibiting the property of self-similarity, without imposing a severe burden to the machine running the simulation software. Just to give a practical idea, the implementation of the simulator used in the course of this work was able to output network traffic at a data rate higher than 8 Gbps, and it is capable of producing

a 100 GB trace with less than 340 B of memory. In spite of its strict application scope, the generator can be used in situations were some of the characteristics of the aggregated traffic can be neglected in favour of the self-similar properties of the trace, namely in the assessment of the impact of persistence in the utilisation of resources, during longer periods of time and under controllable load conditions.

The description in section 5.4.1. comprises the modelling of important statistical properties and network parameters, but it does not take into account all the characteristics of the traffic in a given aggregation point, since that level of detail is out of the scope of this thesis. The presented model does not include the formalisation of the contents of the protocol data units, nor hints the means to generate them. The synthesised traces are hollow, in terms of payload, and the produced series may be seen as the most classical way to represent data communications (packet sizes *and* inter-arrival times). As further elaborated below, the reproduction of other aspects of the traffic requires further research, and implies the increment of the complexity of the model. Nonetheless, as the produced trace is statistically correct and conceptually acquiescent, it is ideal to create background traffic for network aggregation points simulators. The generation of the traces in a packet-by-packet and sequential manner mimics perfectly the online data capture procedure. The information provided by the traces is enough to construct the data structures arriving to the lower end of the data link layer of the network card of the aggregation device, easing the inclusion of data units designed for the purpose of testing the emulated scenario.

The complete development of a simulator of network traffic traces, with long-range dependent properties, was one of the biggest challenges of this work, but from it resulted a different insight of the self-similarity theory. The scheme behind 4SG was of special relevance, since it was used as the basis for the theoretical justification of the results obtained.

According to the scheme used in 4SG, a self-similar process may be approximated by the weighed sum of several components, whose (time) duration and statistical weight depend of the Hurst parameter. The occurrence of some types of attack corresponds to the injection of a new constant component, which can only result in the increase of the persistence of the series under analysis, or in the destruction of the referred property. Both cases where analysed in this thesis.

More than with any other common statistics, the Hurst parameter estimation deals with a necessarily large pool of samples, since the latter compresses the information about the scaling properties of the sequence being processed. As such, it is not affected by sporadic, diluted or statistically insignificant variations from normality. In terms of network level intrusions, this means that attacks based on single data units (e.g. malformed IP packets) or intentionally diluted in the time domain (e.g. stealth probes) will not be detected by means of self-similarity analysis. As the analysis is made *in the dark*, host level intrusions (e.g. malicious code, virus, trojans, spyware, etc.) are not expected to produce any effect in the Hurst parameter values either. The only intrusions that may affect the statistic are the ones whose operational model falls into the category of *network intensive attacks*, as is the case of DoSs (and DDoSs) attacks based in massive quantities of service requests.

One of the major novelties of this work concerns the continuous perspective of the local scope Hurst parameter values given by the point-by-point and windowed estimators. The results of the analysis conducted with such methods, for traces containing *network intensive attacks*, show that the latter impact the estimates in three different manners, depending mostly on the intensity and duration of the anomaly:

1. If the occurrence of an attack results in an expressive loss of stationarity, the implementations of the windowed estimators are completely worthless because, in such case, the method is erroneously led to the conclusion that the sequence under investigation is merely a shifted and scaled version of an 0/1 process. The windowed estimators return values near to 1 in such situations. The loss of stationarity results, in that case, in an irrefutable loss of self-similarity.

2. If the temporal span of the attack is smaller than the size of the observation window of the local scope Hurst parameter estimator, and if the intensity of the attack does not incur in an expressive loss of stationarity, the Hurst parameter value returned by the referred estimator increases while the attack endures, decreasing to the *normal* degree after the traffic concerning the attack has *completely* left the observation window.

3. If the temporal span of the attack is bigger than the size of the observation window

of the local scope Hurst parameter estimator, and if the intensity of the attack does not result in an expressive loss of stationarity, then the estimates of the Hurst parameter increase only during the transitory phase where the traffic concerning the attack is entering or leaving the observation window.

During the phase where the windowed estimator is *observing* the mixture constituted by the sum of legitimate and illegitimate traffic, the Hurst parameter value decreases to the value assessed prior to the attack. This happens because, as soon as the attack is completely *absorbed* by the observation window, the estimator updates the entire set of statistics it is based on, to the ones of the shifted series, which are approximately equal to the ones of the original series. In such a case, the windowed estimator cannot distinguish the shifted part of the process under analysis from the legitimate process itself.

The impact in the global scope estimates is mostly dependent of the quantity of data previously processed by the method. For small quantities, the behaviour of the values of the Hurst parameter is similar to the one of the windowed estimates, while for the opposite case, the reaction may be almost null. Moreover, the effect of any ephemeral modification of the self-similarity degree takes longer to disappear from an incremental estimator, than from its reciprocal windowed version. Under certain restrictions, the values returned by the point-by-point estimators may then be used as baseline comparison for the previously mentioned ones.

It was proven that the resilience of a given process to the complete loss of the self-similarity structure increases with the Hurst parameter. Moreover, loss of self-similarity depends on the ratio between load and intensity of the attack, because the higher the load is, the more prominent the attack needs to be so as to produce enough damage to the legitimate trace. Thus, high-debit aggregation nodes concretise scenarios particularly favourable to the preservation of that property and, as the networks speed and processing power increases, the more favourable the circumstances tend to be.

Based on the results, one may conclude that the self-similarity analysis can be used to detect anomalies whose nature *may* be related with network intensive attacks. For that, a technique that elaborates on the continuous monitoring of the values of the

Hurst parameter, detecting abnormal variations of the estimates during predefined periods of time, may be used. Using the size of the observation window as an input variable of the detector, it is possible to point out the beginning and approximate duration of any particularly lengthy and intense flow of traffic. As suggested by the discussion in chapter 3, it would be actually possible to apply the estimators to the byte count per *tenth of millisecond* using a 2.8 GHz processor and less than 4 KB of memory. *Per se*, these numbers are enough to justify the implementation of these real-time indicators of the self-similarity degree of the traffic, if only for the perspective they provide. Because of the metrics on which this type of analysis is based on, a method that draws on the self-similarity degree estimation can be placed immediately after the most basic network information collectors, and used to raise alerts about potentially harmful data flows or to trigger further investigation procedures (e.g. DPI techniques).

Regardless of the promissory results, in the increasingly complex area of information networks, the type of analysis investigated herein cannot be seen as a sole nor as an extraordinary solution for the problem of intrusion detection. The application scope of any method based in self-similarity analysis lies in the intersection of the action areas of NIDSs and *traffic characterisation in the dark* mechanisms, being thus limited. Additionally, rather intense data flows produce the same kind of impact in the estimates of the Hurst parameter that network intensive attacks do, rendering impossible to decide about the legitimacy of the traffic only based on that metric. Preprocessing of the empirical input series is required in network points where transfers of large amounts of data (e.g. due to activities of backup) are known to exist. Finally, it has to be taken into account that the assumptions behind this type of detection are rather strong, since it is necessary to presume that the traffic under analysis exhibits (and will continue to exhibit) the self-similarity property. Such assumption may not hold true e.g. under erroneous or congestion states of the network, light load conditions or in points where control mechanisms uniformise the way data is transmitted. Along this work, it became obvious to the author that the self-similarity degree of the traffic is better defined if understood as one of the indicators of the health of the network, at a given aggregation point. Under such terms, the point-by-point and windowed estimators comprise efficient tools for the real-time observation of the evolution of that statistic, enabling administrators to manage the network while taking that factor into account also.

6.2. Directions for Future Work

The author is particularly receptive to all the possibilities to apply some of the concepts and implement some of the mechanisms presented herein in a real network traffic monitoring equipment. He believes that the empirical observation of the behaviour of these mechanisms in a real network environment will enable one to optimise them in a practical and useful manner. It is also planned to continue the line of research of chapter 5 with the analysis of different traces of traffic (pre-collected or in real-time). The observation of the online reaction of the windowed-modified estimators to attacks *in the wild* would certainly add an entirely new perspective to this investigation.

Future Use of the Modified Hurst Parameter Estimators

It is of the opinion of the author that the future of the modified Hurst parameter estimators is more promissory in the area of real-time traffic monitoring and analysis, than in the network security field. This belief is based on the conclusions included herein, and on the review of the literature for those particular fields. When compared with the incremental estimators, the *windowed modified* version of the procedures embody a more attractive solution for any kind of system operating in a real-time basis. Thus, the last *type* of estimators has more chances to find applications in the future, either in the networking area, or in many other areas of knowledge (medicine, hydrology, weather prediction, etc.).

Within the area of the information networks, the possible repercussions of assessing the Hurst parameter within a window of values in real-time are still to be fully examined, specially in terms of the impact it may have on queue management and (consequently) on QoS improvements. Recall that the degree of self-similarity may be understood as a measure of the *burstyness* of the traffic: the higher the Hurst parameter value is, the higher is the probability of having problems with buffer overflow. Hence, the research of this particular topic should address the feasibility of assessing the Hurst parameter for different types of flow separately (divided by priority or classes), and of using that information to allocate more memory and computational resources to the traffic with higher level of burstyness. Online estimation of the Hurst parameter opens the possibility to create proactive provisioning mechanisms for network elements.

Future Use of the Self-Similar Sequences Generators

The most prominent advantages of the generator entitled fBm-SGA were on the basis of an invention report (*Method for On-line Simulation of Traffic Conditions on Network Aggregation Points* [182]), which addresses the problem of anticipating the network behaviour, in terms of amount of traffic. Because the fBm-SGA is sequential and requires low computational resources, it was transformed into an online network traffic simulator, that receives the *real trace of traffic* and the associated Hurst parameter value as inputs, and rapidly returns one (or several) paths as a prediction of what might happen in the following moments.

Some of the schematics that were used to explain the idea in the referred document are included in Figure 6.1. The use case represented in the bottom part of the figure should be understood as one of the possible embodiments of the idea, as it could be easily generalized to other applications requiring parallel simulation of self-similar processes. While the report was specific on how to adapt the internal data representations of fBm-SGA to an incoming set of values (e.g. the network trace), the interpretation of the retrieved values and the actions to be taken upon that interpretation were not yet defined, being that a potential topic of future research. Meanwhile, stimulated by this possible application and aiming for the resolution of some of the disadvantages of fBm-SGA, the author has already designed the lines that improve further the quality of the generator, without damaging its computational efficiency. Such improvements are yet to be formally described, but they were already tested.

As a future improvement for 4SG, the author would like to mention the possibility to generate series of values with multi-fractal (or multi-scaling) structure, which can be carried out by introducing a *slight* modification to the means used to calculate the weights of the contributions (see section 4.4.). This modification affects only the way the several aggregation blocks scale individually, leaving intact the core of the algorithm. The scope of application of a multi-fractal sequences generator is obviously larger, and may be of particular relevance in e.g. the financial [183] or the physiological signal research areas [184].

The author also plans to describe how the philosophy behind 4SG can be adapted

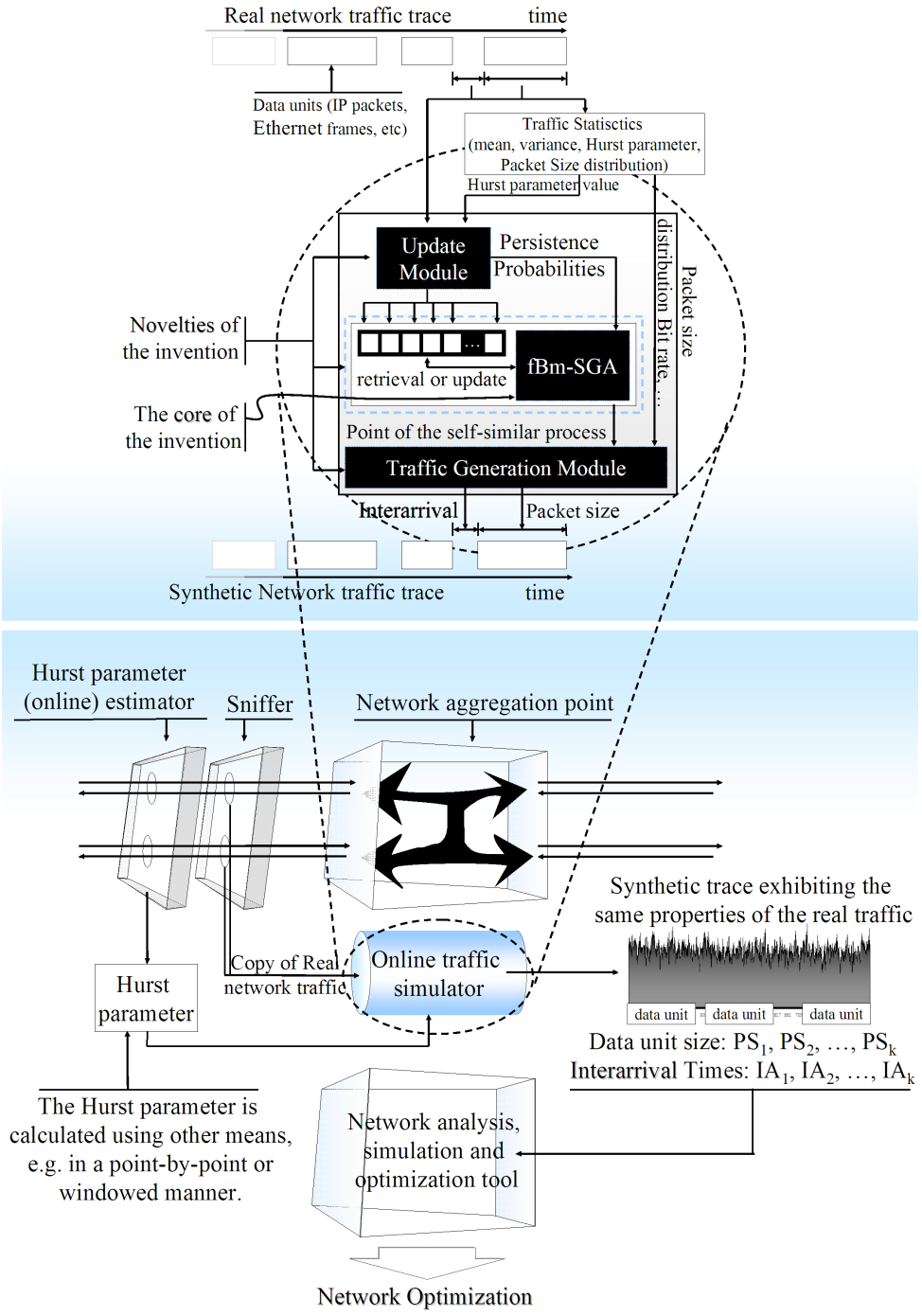

Figure 6.1: The schematics of the concept machine of the network online simulator. The top of the figure illustrates the inner workings of the machine, while the bottom part presents a possible use case scenario.

to generate realisations of a Gaussian variate from uniforms. In spite of the variety of GRNGs proposed in the literature (see e.g. [142]), it seems worth to explore the fact that 4SG requires only 1 call to a PRNG per point generated, and that the respective GRNG would actually inherit its superb performance. The novel algorithm would be at least 4

times faster than the *Polar method*, whose implementation was discussed in section 4.5.3.. Additionally, the statistical properties of the produced sequences (as the average and the variance) are expected to be very stable.

The Future of the Prime Remainder Revolution

It was mentioned that PRR was developed for simulation purposes, reason by which its cryptographic application was not considered nor evaluated. The worth of the algorithm as a cryptographic primitive will be the subject of future research work.

Another topic that deserves attention is the investigation of the impact of considering a bigger Mersenne prime (for instance $2^{61} - 1$), in terms of performance of the algorithm. At this point, it would be logical to expect that such a change would render PRR *almost* twice as fast as its actual implementation, in machines capable of natively supporting 64 bit long operations, while increasing its period to $2^{61} \times 2^{59} = 2^{120}$. The author is equally interested in submitting the algorithm to other batteries of tests, namely to the recently updated *TestU01: A C library for empirical testing of random number generators* [120, 185].

The Future of the Network Traffic Simulator

The network traffic simulator presented in chapter 5 was build strictly with the purposes of this thesis in mind. As so, its structural blocks are mostly related with the traffic load of a given network aggregation point, and with the self-similarity properties of the trace produced. Two possible future directions for the simulator include (i) the modelling of other *aspects* of the network traffic, as for example the dynamism of higher layer protocols (IP, TCP, User Datagram Protocol (UDP), VoIP or P2P file exchange protocols, just to name a few), and (ii) the inclusion of more complex schemes, that unify different models depending on the aggregation scale considered (see section 2.3.4.). The hypothesis to integrate the traffic generator (or of some of its underlying algorithms and theory) into already developed and functional simulators is not excluded either.

The emulation of the messages of higher layer protocols is important for the area

of QoS, principally the simulation of the most recent services running on the Internet: VoIP, IPTV, P2P file sharing, etc. As the means described herein apply to the network aggregation point *only*, their usefulness is written in terms of the analysis of the effect the said types of traffic have on the network aggregation point, and of the analysis of the delays one particular flow suffers when passing through one of those nodes. The research effort may be significantly reduced by considering only two distinct flows of traffic, one coming from the source, and the other already aggregated at the intermediary node. By these means, it is possible to concentrate on a particular protocol, avoiding the burden of having to simulate too many sources or services. Nevertheless, the validity of this approach has to be further investigated before proceeding.

Network Traces with Attacks Data Base

The subject of *intrusion detection* gathers as much interest from the academic community, as from the industry (telecommunication products sellers) and Service Providers (SPs). Despite that, it is rather difficult to find up-to-date traces of traffic containing *real* attacks captured in a controlled environment. The traces at MIT/DARPA [179] date from 1998 and 1999, and are still used to test new candidates to intrusions detection mechanisms. Because those traces are becoming *old* and do not precisely reflect all the characteristics of aggregated traffic [23], some [20] even try to emulate the attacks by injecting packets into known traces.

A project aiming for the creation of new data sets containing traces of traffic with attacks is thus of relevance, and it would be welcomed in the research community of the network security area. The details of such an endeavour are still to be determined, but the usefulness of its outputs are depend on whether or not the following guidelines are met.

Availability and Anonymity. The data resulting from this project would have to be available for others to use. This means that the traces have to be made anonymous prior to be stored online (preferred) or in other specific format. The collected data should be saved in *standard* or well documented file formats (e.g. tcpdump, tsh), to enhance portability.

Correctness and Temporal Appropriateness. The main objectives of collecting new traces is to make them reflect the dynamics of real traffic of *current* networks, as well as to include some of the most *recent forms of attacking them*. If simulated, the traces or the attacks must meticulously meet the characteristics (to be subject of research) of the network they are trying to reproduce. These characteristics include aspects related with self-similarity (in the case of aggregation points), network load, types of application, etc. Given the nature of these experiments, their validity is limited to the temporal span in which they were conducted. Hence, one must not be overly concerned in constructing *future proof* traces, since the evolution of the attacks, networks and counter-measures is hard to predict.

Controlled and Structured. In order to be really useful, the traces should be categorised and the attacks labelled. The occurrence of a given attack should be discreetly regulated, so as to imitate an uncontrolled attacking source. All the information concerning the traces and the attacks considered relevant, should also be attached to the data sets. This information should include several statistical measures (e.g. length and number of packets of the trace, average load, self-similarity degree), the topology and the specifications of the (simulated) network in which the traces were obtained, and the details of the attacks it includes (type, severity, and time of occurrence, attacking source and destination, etc.).

Other Lines of Research

One of the most prominent topics of the area of network security is the one of DPI mechanisms. While TCD techniques may embody a good first line of defence, it is on DPIs that systems vendors and buyers trust the most [11, 149, 186]. As previously said, the integration of the particular set of monitoring mechanisms described herein, in real network systems, is something that deserves to be explored. Furthermore, the cooperation between the windowed Hurst parameter estimators and the DPI module is also worth of the effort. The definition of that cooperation and how the different modules can interact to improve their performance may embody a particularly defying task.

Because the trend for security systems is to go deeper and deeper into the packets

contents, which may be on the basis of discussions regarding the *privacy* topic, the future of the research in this area should tackle this tendency, and explore efficient ways of pointing out a malicious activity (network attack, virus, worm, trojan horse, etc.) while analysing potentially huge amounts of data. The persons conducting research in this particular field should also keep in mind that there is an actual need for simplified (yet efficient) IDSs. Most of the times, it is requested to the network elements to perform simple tasks of security, which are not their main functionality and, as so, the resources allocated to that part of the system should be minimum. Additionally, over complicated mechanisms require often large amounts of memory and high computational power, which render the devices (or the software solution *per se*) more expensive.

For a network administrator, all the menaces to the integrity of the network or of its components have to be handled with extreme caution, for their severity may be misleading. Nevertheless, the attacks based on *apparently legitimate connections*, that act like normal for undetermined periods of time, until they reach the critical attacking point, constitute a good topic of research, specially when non-spoofed DDoS attacks are becoming more common (for that matter, refer to [187]).

Bibliography

[1] J. B. Postel, "RFC 791: Internet Protocol," September 1981, obsoletes RFC0760. See also STD0005. Status: STANDARD. [Online]. Available: ftp://ftp.internic.net/rfc/rfc760.txt,ftp://ftp.internic.net/rfc/rfc791.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc760.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc791.txt

[2] ——, "RFC 793: Transmission control protocol," 1981, status: STANDARD. [Online]. Available: ftp://ftp.internic.net/rfc/rfc793.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc793.txt

[3] Federal Networking Council (FNC), "Definition of "Internet"," October 1995, accessed July 26, 2008. [Online]. Available: http://www.nitrd.gov/fnc/Internet_res.html

[4] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. B. Postel, L. G. Roberts, and S. S. Wolff, "A Brief History of the Internet," *CoRR*, vol. cs.NI/9901011, 1999, a Brief History of the Internet, version 3.32, Last revised 10 Dec 2003.

[5] S. Gibson, "DRDoS Distributed Reflection Denial of Service," Gibson Research Corporation, Tech. Rep., 2002. [Online]. Available: http://www.grc.com/dos/drdos.htm

[6] CERT® Coordination Center, "Computer Emergency Response Center," July 2008, accessed 18 July, 2008. [Online]. Available: http://www.cert.org/

[7] Jupitermedia Corporation, "Intrusion Detection System Definition," December 2002, accessed March 23, 2008. [Online]. Available: http://wi-fiplanet.webopedia.com/TERM/I/intrusion_detection_system.html

[8] A. Kanaoka and E. Okamoto, "Multivariate Statistical Analysis of Network Traffic for Intrusion Detection," in *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*. Washington, DC, USA: IEEE Computer Society, 2003, p. 472.

[9] R. Bace and P. Mell, "NIST Special Publication on Intrusion Detection Systems," National Institute of Standards and Technology, Tech. Rep., 2004. [Online]. Available: http://www.21cfrpart11.com/files/library/reg_guid_docs/nist_intrusiondetectionsys.pdf

[10] Information Technology Department, Harvard College, "Glossary - Computer Security Terminology," May 2007. [Online]. Available: http://hms.harvard.edu/hmsit/pg.asp?pn=security_glossary.

[11] R. Grigonis, "Securing Carrier Networks," January 2009, accessed March 18, 2009. [Online]. Available: http://www.tmcnet.com/voip/0109/securing-carrier-networks.htm

[12] F. Yu, "High Speed Deep Packet Inspection With Hardware Support," Ph.D. dissertation, Berkeley, CA, USA, 2006, adviser-Katz, Randy H.

[13] J. McDonough, "Moving Standards to 100 GbE and Beyond," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 6–9, November 2007.

[14] IEEE P802.3ba Task Force, "IEEE P802.3ba 40 Gb/s and 100 Gb/s Ethernet Task Force," July 2008, accessed July 26, 2008. [Online]. Available: http://www.ieee802.org/3/ba/index.html

[15] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," in *SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, vol. 35, no. 4. New York, NY, USA: ACM Press, 2005, pp. 229–240.

[16] I. Norros, "Studies on a Model for Connectionless Traffic, Based on Fractional Brownian Motion," COST24TD(92)041, 1992.

[17] W. E. Leland, W. Willinger, M. S. Taqqu, and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic," in *Proceedings of the Conference on Communications*

Architectures, Protocols and Applications, vol. 25, no. 1. New York, NY, USA: ACM, September 1993, pp. 183–193.

[18] M. E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE /ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, 1997.

[19] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 71–86, 1997.

[20] W. H. Allen and G. A. Marin, "The LoSS Technique for Detecting New Denial of Service Attacks," in *SoutheastCon*, Florida Institute of Technology. IEEE, March 2004, pp. 302–309.

[21] M. Y. Idris, A. H. Abdullah, and M. A. Maarof, "Iterative Window Size Estimation on Self Similarity Measurement for Network Traffic Anomaly Detection," *International Journal of Computing & Information Sciences*, vol. 4, no. 4, pp. 88–91, August 2005.

[22] D. Nash and D. Ragsdale, "Simulation of Self-Similarity in Network Utilization Patterns as a Precursor to Automated Testing of Intrusion Detection Systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 31, no. 4, pp. 327–331, July 2001.

[23] W. H. Allen and G. A. Marin, "On the Self-similarity of Synthetic Traffic for the Evaluation of Intrusion Detection Systems," in *Proceedings of the Symposium on Applications and the Internet*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 242–248.

[24] Y. Hua and C.-L. Wu, "Intrusion Detection Based on Artificial Immune System With Self-Similar Traffic," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, vol. 4, November 2003, pp. 2437–2441.

[25] M. Li, "Change Trend of Averaged Hurst Parameter of Traffic Under DDOS Flood Attacks," *Computers & Security*, vol. 25, no. 3, pp. 213–220, 2006.

[26] M. F. Rohani, M. A. Maarof, A. Selamat, and H. Kettani, "Uncovering Anomaly Traffic Based on Loss of Self-Similarity Behavior Using Second Order Statistical Model," *International Journal of Computer Science and Network Security*, vol. 7, no. 9, September 2007.

[27] W. Willinger, "On Internet Traffic Dynamics and Internet Topology II - Internet Model Validation," January 2004, accessed March 5, 2008. [Online]. Available: http://www.ima.umn.edu/talks/workshops/1-7-9.2004/willinger/willinger1.2.pdf

[28] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic Classification on the Fly," *SIGCOMM Computer Communications Review*, vol. 36, no. 2, pp. 23–26, April 2006.

[29] P. R. M. Inácio, M. M. Freire, M. Pereira, and P. P. Monteiro, "Analysis of the Impact of Intensive Attacks on the Self-Similarity Degree of the Network Traffic," in *Proceedings of The 2nd. International Conference on Emerging Security Information, Systems and Technologies*, Cap Esterel, France, 2008, pp. 107–113.

[30] ——, "A Evolução do Parâmetro de Hurst e a Destruição da Auto-Semelhança Durante um Ataque de Rede Intenso," in *Proceedings of Segurança Informática nas Organizações*, Paulo Simões e Edmundo Monteiro, Ed., Coimbra, Portugal, November 2008, pp. 63–74.

[31] ——, "Fast Synthesis of Persistent Fractional Brownian Motion," *ACM Transactions on Modelling and Computer Simulation*, 2009, accepted for publication with minor changes.

[32] ——, "The Design and Evaluation of the Simple Self-Similar Sequences Generator," *Elsevier Information Sciences*, 2009, accepted for publication with minor changes.

[33] ——, "Remainders Revolution Pseudo Random Number Generator," *ACM Journal of Experimental Algorithmics*, 2009, submitted for publication.

[34] F. Harmantzis and D. Hatzinakos, "Heavy Network Traffic Modeling and Simulation Using Stable FARIMA Processes," in *Proceedings of The 19th International Teletraffic Congress*, Beijing, China, September 2005, pp. 393–413.

[35] J. Liu, Y. Shu, L. Zhang, F. Xue, and O. T. Yang, "Traffic Modeling Based on FARIMA Models," in *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 1, Edmonton, Alta., Canada, 1999, pp. 162–167.

[36] D. Cox, *In Statistics: An Appraisal*. Iowa State Statistical Library: The Iowa State University Press, 1984, ch. Long-Range Dependence: a Review, pp. 55–74.

[37] T. Karagiannis, M. Molle, and M. Faloutsos, "Long-Range Dependence: Ten Years of Internet Traffic Modeling," *IEEE Internet Computing*, vol. 8, no. 5, pp. 57–64, 2004.

[38] T. Dieker, "Simulation of Fractional Brownian Motion," Master's thesis, University of Twente, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, 2004. [Online]. Available: http://www.proba.ucc.ie/~td3/fbm/thesis.pdf.

[39] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, February 1994.

[40] W. Sun, S. Rachev, F. Fabozzi, and P. Kalev, "Fractals in Trade Duration: Capturing Long-Range Dependence and Heavy Tailedness in Modeling Trade Duration," *Annals of Finance*, vol. 4, no. 2, pp. 217–241, March 2008.

[41] G. Samorodnitsky, "Long Memory and Self-Similar Processes," *Annales de la faculté des sciences de Toulouse*, vol. 6, no. 1, pp. 107–123, 2006.

[42] C. M. Grinstead and J. L. Snell, *Introduction to Probability*. American Mathematical Society; (July 1, 1997), 1997, vol. 2 Revised edition, ch. Random Walks, p. 510.

[43] B. B. Mandelbrot and J. W. V. Ness, "Fractional Brownian Motions, Fractional Noises and Applications," *SIAM Review*, vol. 10, no. 4, pp. 422–437, 1968.

[44] V. Paxson, "Fast, Approximate Synthesis of Fractional Gaussian Noise for Generating Self-Similar Network Traffic," *ACM SIGCOMM Computer Communications Review*, vol. 27, no. 5, pp. 5–18, 1997.

[45] V. Paxson and S. Floyd, "Wide Area Traffic: the Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995.

[46] K. G. Coffman and A. Odlyzko, "The size and growth rate of the internet," Tech. Rep. 99-11, 21, 1999. [Online]. Available: http://www.dtc.umn.edu/~odlyzko/doc/internet.size.pdf

[47] A. M. Odlyzko, "Internet Growth: Myth and Reality, Use and Abuse," *Journal of Computer Resource Management*, no. 102, pp. 23–27, Spring 2001.

[48] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a Fundamental Result in Self-similar Traffic Modeling," *SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 5–23, April 1997.

[49] B. A. Cipra, "Oh, What a Tangled Web We´ve Woven. . . ." *SIAM News*, vol. 33, no. 2, 1999.

[50] H. J. Fowler and W. E. Leland, "Local Area Network Traffic Characteristics, With Implications for Broadband Network Congestion Management," *IEEE Journal of Selected Areas in Communications*, vol. 9, no. 7, pp. 1139–1149, 1991.

[51] A. Erramilli, O. Narayan, and W. Willinger, "Experimental Queueing Analysis With Long-Range Dependent Packet Traffic," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 209–223, April 1996.

[52] P. R. Morin, "The Impact of Self-Similarity on Network Performance Analysis," Computer Science 95.495, Carleton University, Tech. Rep., December 1995. [Online]. Available: http://coblitz.codeen.org:3125/citeseer.ist.psu.edu/cache/papers/cs/6423/http:zSzzSzfermat.eup.udl.eszSz~cesarzSzTFCzSzRobert_SallazSzpaper2.pdf/morin95impact.pdf

[53] K. Park, G. Kim, and M. Crovella, "On the Effect of Traffic Self-Similarity on Network Performance," in *Proceedings of Winter Simulation Conference*, 1997, pp. 989–996.

[54] D. A. Rolls, G. Michailidis, and F. Hernández-Campos, "Queueing Analysis of Network Traffic: Methodology and Visualization Tools," *Computer Networks*, vol. 48, no. 3, pp. 447–473, 2005.

[55] J. Cao, W. Cleveland, D. Lin, and D. Sun, "Internet Traffic Tends to Poisson and Independent as the Load Increases," Bell Labs, Tech. Rep., 2001. [Online]. Available: http://www.stat.purdue.edu/~wsc/papers/lrd2poisson.pdf

[56] N. Wisitpongphan and J. Peha, "Effect of TCP on Self-Similarity of Network Traffic," in *Proceedings of the 12th International Conference on Computer Communications and Networks*, October 2003, pp. 370–373.

[57] A. Veres and M. Boda, "The Chaotic Nature of TCP Congestion Control," in *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, March 2000, pp. 1715–1723.

[58] O. Rose, "Estimation of the Hurst Parameter of Long-Range Dependent Time Series," *Research Report*, vol. 137, 1996.

[59] R. G. Clegg, "A Practical Guide to Measuring the Hurst Parameter," *Science and Technology*, vol. 7, pp. 3–14, 2006.

[60] K. M. Rezaul and V. Grout, "An overview of long-range dependent network traffic engineering and analysis: characteristics, simulation, modelling and control," in *ValueTools '07: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 1–10.

[61] M. S. Taqqu, V. Teverovsky, and W. Willinger, "Estimators for Long-Range Dependence: An Empirical Study," *Fractals*, vol. 3, pp. 785–798, 1995.

[62] J. Strecker, "Fractional Brownian Motion Simulation: Observing Fractal Statistics in the Wild and Raising Them in Captivity," Master's thesis, The College of Wooster, April 2004.

[63] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, "A Nonstationary Poisson View of Internet Traffic," *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1558–1569, March 2004.

[64] W. Willinger, M. Taqqu, and A. Erramilli, *Stochastic Networks: Theory and Applications*, ser. Royal Statistical Society Lecture Note Series. Oxford University

Press, September 1996, no. 4, ch. A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks, pp. 339–366.

[65] T. Karagiannis and M. Faloutsos, "SELFIS: A Tool For Self-Similarity and Long-Range Dependence Analysis," in *Proceedings of the 1st Workshop on Fractals and Self-Similarity in Data Mining: Issues and Approaches (in KDD)*, Edmonton, Canada, July 2002, pp. 81 – 93.

[66] T. Karagiannis, M. Faloutsos, and M. Molle, "A User-Friendly Self-Similarity Analysis Tool," *SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 81–93, 2003.

[67] C.-K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger, "Mosaic Organization of DNA Nucleotides," *Physical Review E (Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics)*, vol. 49, no. 2, pp. 1685–1689, February 1994.

[68] P. Abry and D. Veitch, "Wavelet analysis of long-range-dependent traffic," *IEEE Transactions on Information Theory*, vol. 44, no. 1, pp. 2–15, 1998.

[69] O. Jones and Y. Shen, "Estimating the Hurst Index of a Self-Similar Process via the Crossing Tree," *IEEE Signal Processing Letters*, vol. 11, no. 4, pp. 416–419, 2004.

[70] H. Kettani and J.A.Gubner, "A Novel Approach to the Estimation of the Long-Range Dependence Parameter," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 6, pp. 463–467, June 2006.

[71] J. Sutcliffe, "Obituary Harold Edwin Hurst: 1 January 1880-7 December 1978," pp. 16–20, December 1979, accessed January 12, 2007. [Online]. Available: http://www.cig.ensmp.fr/~iahs/hsj/240/hysj_24_04_0539.pdf

[72] H. E. Hurst, "Long-Term Storage Capacity of Reservoirs," *Transactions of the American Society of Civil Engineers*, vol. 116, pp. 770–799, 1951.

[73] G. Sakalauskiene, "The Hurst Phenomenon in Hydrology," *Environmental research, engineering and management*, vol. 25, no. 3, pp. 16–20, 2003.

[74] A. W. Lo, "Long-Term Memory in Stock Market Prices," *Econometrica*, vol. 59, no. 5, pp. 1279–313, September 1991.

[75] A. A. Anis and E. H. Lloyd, "The Expected Value of the Adjusted Rescaled Hurst Range of Independent Normal Summands," *Biometrika*, vol. 63, no. 1, pp. 111–116, April 1976, biometrika Trust.

[76] W. Feller, "The Asymptotic Distribution of the Range of Sums of Independent Random Variables," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 427–432, 1951.

[77] T. Dieker and M. Mandjes, "On Spectral Simulation of Fractional Brownian Motion," *Probability in the Engineering and Information Sciences*, vol. 17, no. 3, pp. 417–434, 2003.

[78] H. Zhang, Y. Shu, and O. Yang, "Estimation of Hurst Parameter by Variance-Time Plots," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, vol. 2, August 1997, pp. 883–886.

[79] E. W. Weisstein, "Variance," 2008, accessed April 12, 2008. [Online]. Available: http://mathworld.wolfram.com/Variance.html

[80] M. P. Allen, *Understanding Regression Analysis*, ser. Humanities, Social Sciences and Law. Springer US, November 2007, ch. The coefficient of determination in multiple regression, pp. 91–95, free Preview.

[81] N. Christou, "The True R^2 and the Truth About R^2," UCLA Department of Statistics, UCLA Center for the Teaching of Statistics, University of California, Los Angeles, California, USA, Tech. Rep., 2005.

[82] M. Trovero, "Long Range Dependence: a Light Tale for the Practitioner," August 2003, statistics students seminar at UNC. Accessed December 11, 2007. [Online]. Available: http://www.stat.unc.edu/students/statlunch/2003fall/trovero_10_08_03_screen.pdf

[83] O. Jones and Y. Shen, *Fractals and Engineering: New Trends in Theory and Applications*, J. Levy-Vehel and E. Lutton ed. Springer London, December 2005,

ch. A non-parametric test for self-similarity and stationarity in network traffic, pp. 219–234.

[84] ——, "Matlab Code for Estimating the Hurst Index H of a Self-Similar Process - Function get_hits," 2005, accessed February 15, 2008. [Online]. Available: http://www.ms.unimelb.edu.au/~odj/EBPHest/get_hits.m

[85] C.-K. Peng, S. Havlin, H. E. Stanley, and A. L. Goldberger, "Quantification of Scaling Exponents and Crossover Phenomena in Nonstationary Heartbeat Time Series," *Chaos*, vol. 5, no. 1, pp. 82–87, March 1995.

[86] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioNet, the Research Resource for Complex Physiologic Signals - Fractal Mechanisms in Neural Control: Human Heartbeat and Gait Dynamics in Health and Disease," March 2003, accessed January 23, 2008. [Online]. Available: http://www.physionet.org/tutorials/fmnc/

[87] A. Heckert and D. J. J. Filliben, *Dataplot Reference Manual*. National Institute of Standards and Technology, 1976, vol. 1: Commands, ch. 2. Graphics Commands - PERIODOGRAM. [Online]. Available: http://www.itl.nist.gov/div898/software/dataplot.html/refman1/ch2/periodog.pdf

[88] M. Taqqu, "Local Whittle," acessed July 19, 2007. [Online]. Available: http://math.bu.edu/people/murad/methods/locwhitt/

[89] M. Roughan, D. Veitch, and P. Abry, "Real-Time Estimation of the Parameters of Long-Range Dependence," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, pp. 467–478, 2000.

[90] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[91] D. Veitch and P. Abry, "A wavelet based joint estimator of the parameters of long-range dependence," *IEEE Transactions on Information Theory*, vol. 45, no. 3, pp. 878–897, 1999.

[92] D. N. Veitch, M. Roughan, and P. Abry, "Real-Time Estimation of Long Range Dependent Parameters," Patent PCT/AU1999/000 077, October, 1999, WO/1999/040703.

[93] T. Higuchi, "Approach to an Irregular Time Series on the Basis of the Fractal Theory," *Physica D Nonlinear Phenomena*, vol. 31, no. 2, pp. 277–283, June 1988.

[94] K. M. Rezaul and V. Grout, "Exploring the Reliability and Robustness of HEAF(2) for Quantifying the Intensity of Long-Range Dependent Network Traffic," *International Jornal of Computer Science and Network Security*, vol. 7, no. 2, pp. 221–229, February 2007.

[95] S. Stoev, M. S. Taqqu, C. Park, G. Michailidis, and J. Marron, "LASS: a Tool for the Local Analysis of Self-Similarity," *Computational Statistics & Data Analysis*, vol. 50, no. 9, pp. 2447–2471, May 2006.

[96] M. Bartolozzi, C. Mellen, T. D. Matteo, and T. Aste, "Multi-Sscale Correlations in Different Futures Markets," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 58, no. 2, pp. 207–220, July 2007, in collection: Physics and Astronomy.

[97] T. Hagiwara, H. Doi, H. Tode, and H. Ikeda, "High-Speed Calculation Method of the Hurst Parameter Based on Real Traffic," in *Proceedings of the 25th Annual IEEE Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2000, p. 662.

[98] Sun Microsystems, Inc., "Primitive Data Types (The JAVA Tutorials>Learning the Java Language>Language Basics)," February 2008, acessed February 19, 2008. [Online]. Available: http://java.sun.com/docs/books/tutorial/java/nutsandbolts/datatypes.html

[99] N. C. Kenkel and D. J. Walker, "Fractals in the Biological Sciences," *Coenoses*, vol. 11, pp. 77–100, 1996.

[100] B. B. Mandelbrot, *The Fractal Geometry of Nature*. San Francisco: W. H. Freeman Company, August 1982, no. 1.

[101] R. L. Oldershaw, "Nature Adores Self-Similarity," April 2002. [Online]. Available: http://www.amherst.edu/~rloldershaw/nature.html.

[102] R. Lathrop and D. L. Peterson, "Identifying Structural Self -Similarity in Mountainous Landscapes," *Landscape Ecology*, vol. 6, no. 4, pp. 233–238, 1992.

[103] M. Kallache, H. W. Rust, and J. Kropp, "Trend Assessment: Applications for Hydrology and Climate Research," *Nonlinear Processes in Geophysics*, vol. 12, no. 2, pp. 201–210, 2005.

[104] J. R. M. Hosking, "Fractional Differencing," *Biometrika*, vol. 68, no. 1, pp. 165–176, April 1981.

[105] ——, "Modeling Persistence In Hydrological Time Series Using Fractional Differencing," *Water Resources Research*, vol. 20, no. 12, pp. 1898–1908, 1984.

[106] Y.-Q. Lu, D. W. Petr, and V. Frost, "Characterization and Modeling of Long-Range Dependent Telecommunication Traffic," Telecommunications and Information Sciences Laboratory, Tech. Rep., August 1994, department of Electrical Engineering and Computer Science, University of Kansas. [Online]. Available: http://www.ittc.ku.edu/publications/documents/Lu1994_tr-tisl-10230-04.pdf

[107] T. Dieker, "Master's Thesis Ton Dieker," September 2004, acessed February 20, 2008. [Online]. Available: http://www.proba.ucc.ie/~td3/fbm/

[108] G. Kramer and G. Pesavento, "Ethernet Passive Optical Network (EPON): Building a Next-Generation Optical Access Network," *IEEE Communications Magazine*, vol. Topics in Lightwave, pp. 62–73, February 2002.

[109] G. Horn, A. Kvalbein, J. Blomsk, and E. Nilsen, "An Empirical Comparison of Generators for Self Similar Simulated Traffic," *Performance Evaluation*, vol. 64, no. 2, pp. 162–190, 2007.

[110] A. Erramilli, "Chaotic Maps as Models of Packet Traffic," in *Proceedings of ITC 14, The Fundamental Role of Tele-traffic in the Evolution of Telecommunications Networks*, J. Labetoulle and J. Roberts, Eds. Antibes Juan-Les-Pins, France: Elsevier Science Publishers B.V. (North-Holland), June 6-10 1994, pp. 329–338.

[111] I. Norros, P. Mannersalo, and J. L. Wang, "Simulation of Fractional Brownian Motion With Conditionalized Random Midpoint Displacement," in *Proceedings of Advances in Performance Analysis*, 1999, pp. 77–101.

[112] H.-D. J. Jeong, D. Mcnickle, and K. Pawlikowski, "Fast Self-Similar Teletraffic Generation Based on FGN and Wavelets," in *Proceedings of the IEEE International Conference on Networks*, September - October 1999, pp. 75–82.

[113] S. Rambaldi and O. Pinazza, "An Accurate Fractional Brownian Motion Generator," *Physica A Statistical Mechanics and its Applications*, vol. 208, pp. 21–30, July 1994.

[114] N. Enriquez, "A simple Construction of the Fractional Brownian Motion," *Stochastic Processes and their Applications*, vol. 109, pp. 203–223, February 2004.

[115] M. Caglar, "Simulation of Fractional Brownian Motion With Micropulses," *Advances in Performance Analysis*, vol. 3, pp. 43–69, 2000.

[116] D. Ostry, "Synthesis of Accurate Fractional Gaussian Noise by Filtering," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1609–1623, April 2006.

[117] C. M. Jarque and A. K. Bera, "Efficient Tests for Normality, Homoscedasticity and Serial Independence of Regression Residuals," *Economics Letters*, vol. 6, no. 3, pp. 255–259, 1980.

[118] E. W. Weisstein, "Gabriel's Staircase," 2008, acessed April 26, 2008. [Online]. Available: http://mathworld.wolfram.com/GabrielsStaircase.html

[119] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, 1998.

[120] P. L'Ecuyer and R. Simard, "TestU01: A C Library for Empirical Testing of Random Number Generators," *ACM Transactions on Mathematical Software*, vol. 33, no. 4, p. 22, August 2007.

[121] A. Zúquete, "An Efficient High Quality Random Number Generator for Multi-Programmed Systems," *Journal of Computer Security*, vol. 13, no. 2, pp. 243–263, 2005.

[122] id QUANTIQUE, "id QUANTIQUE - Quantum Random Number Generator (RNG)," July 2007, acessed July 19, 2007. [Online]. Available: http://www.idquantique.com/products/quantis.htm

[123] M. Haahr, "RANDOM.ORG - Introduction to Randomness and Random Numbers," July 2007, accessed July 18, 2007. [Online]. Available: http://www.random.org/

[124] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in Fortran 77: The Art of Scientific Computing.*, C. Univ, Ed. Press, Cambridge, 1992.

[125] J. von Neumann, "Various Techniques used in Connection With Random Digits," in *John von Neumann, Collected Works*, A. H. Taub, Ed. Oxford: Pergamon Press, Oxford, 1951, vol. 5, pp. 768–770.

[126] "Vulnerability Analysis Tools for Cryptographic Keys," accessed July 20, 2007. [Online]. Available: http://www.cs.hku.hk/cisc/projects/va/index.htm

[127] A. Orhon and J. E. Magen, "Dissonant Numbers," 12 2006, acessed October 27, 2007. [Online]. Available: http://yonkeltron.com/code/randomtest/dissonant-numbers-handout.pdf

[128] P. L'Ecuyer, "Uniform Random Number Generation," *Annals of Operations Research*, vol. 53, no. 1, pp. 77–120, 1994.

[129] S. Luke, "MersenneTwisterFast.java," The GNU Scientific Library - a free numerical library licensed under the GNU GPL, October 2004, accessed June 5, 2007. [Online]. Available: http://cs.gmu.edu/~sean/research/mersenne/MersenneTwisterFast.java

[130] P. Chan, R. Lee, and D. Kramer, *The Java Class Libraries:* `java.io`*,* `java.lang`*,* `java.math`*,* `java.net`*,* `java.text`*,* `java.util`, 2nd ed., 1999, vol. 1.

[131] Sun Microsystems, Inc., "Random (Java 2 Platform SE 5.0)," 2007, accessed July 27, 2007. [Online]. Available: http://java.sun.com/j2se/1.5.0/docs/api/java/util/Random.html

[132] N. Beebe, "Nelson Beebe's Java Notes," March 2004, accessed July 20, 2007. [Online]. Available: http://www.math.utah.edu/~beebe/java/

[133] D. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, November 1997.

[134] J. Walker, "ENT - A Pseudorandom Number Sequence Test Program," October 1998, accessed July 26, 2007. [Online]. Available: http://www.fourmilab.ch/random/

[135] P. Hellekalek, "Random Number Generators - the pLab Project - Tests," The GNU Scientific Library - a free numerical library licensed under the GNU GPL, 2006, accessed July 27, 2007. [Online]. Available: http://random.mat.sbg.ac.at/tests/

[136] QUT's Information Security Institute (ISI), "QUT — ISI — Crypt-X," The GNU Scientific Library - a free numerical library licensed under the GNU GPL, May 2006, accessed July 20, 2007. [Online]. Available: http://www.isi.qut.edu.au/resources/cryptx/

[137] G. Marsaglia, "Diehard Battery of Tests of Randomness v0.2 Beta," 1997, accessed July 21, 2007. [Online]. Available: http://i.cs.hku.hk/~diehard/cdrom/

[138] A. Hidayat, "D.C.T.W.Y.C.D.T: Modulus With Mersenne Prime," February 2007, accessed July 28, 2007. [Online]. Available: http://www.cs.hku.hk/cisc/projects/va/index.htm

[139] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1992.

[140] P.-H. Lee, Y. Chen, S.-C. Pei, and Y.-Y. Chen, "Evidence of the Correlation Between Positive Lyapunov Exponents and Good Chaotic Random Number Sequences," *Computer Physics Communications*, vol. 160, no. 3, pp. 187–203, July 2004.

[141] G. Marsaglia and W. Tsang, "Some Difficult-to-Pass Tests of Randomness," *Journal of Statistical Software*, vol. 7, no. 3, pp. 1–8, 2002.

[142] D. B. Thomas, W. Luk, P. H. Leong, and J. D. Villasenor, "Gaussian Random Number Generators," *ACM Computing Surveys*, vol. 39, no. 4, p. 11, 2007.

[143] J. R. Bell, "Algorithm 334: Normal Random Deviates," *Communications of the ACM*, vol. 11, no. 7, p. 498, 1968.

[144] E. F. C. Junior, "Generating Gaussian Random Numbers," acessed January 09, 2008. [Online]. Available: http://www.taygeta.com/random/gaussian.html

[145] Lincoln Laboratory, Massachusetts Institute of Technology, "MIT Lincoln Laboratory - 1999 DARPA Intrusion Detection Evaluation Training Data Week 2," 2001, accessed February 15, 2008. [Online]. Available: http://www.ll.mit.edu/IST/ideval/data/1999/training/week2/index.html

[146] T. Espiner, "The Worst IT Security Incidents of 2007," November 2007, accessed July 26, 2008. [Online]. Available: http://resources.zdnet.co.uk/articles/features/0,1000002000,39290745,00.htm

[147] *Application Intrusion Detection Using Language Library Calls.* Los Alamitos, CA.: IEEE Computer Society Press, December 2001.

[148] G. Jian, L. Da-Xin, and C. Bin-Ge, "An Induction Learning Approach for Building Intrusion Detection Models Using Genetic Algorithms," *Fifth World Congress on Intelligent Control and Automation*, vol. 5, pp. 4339–4342, June 2004.

[149] Allot Communications, "Digging Deeper into Deep Packet Inspection," White paper, April 2007, accessed May 21, 2008. [Online]. Available: http://www.getadvanced.net/learning/whitepapers/networkmanagement/Deep%%20Packet%20Inspection_White_Paper.pdf

[150] S. Farraposo, P. Owezarski, and E. Monteiro, "A Multi-Scale Tomographic Algorithm for Detecting and Classifying Traffic Anomalies," in *Proceedings of the IEEE International Conference on Communications*, Glasgow - UK, 24 to 28 June 2007, pp. 363–370.

[151] ——, "NADA - Network Anomaly Detection Algorithm," in *Proceedings of the 18th IFIP IEEE International Workshop on Distributed Systems Operations and Management*, San Jose - USA, 29 to 31 October 2007, pp. 191–194.

[152] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA Off-Line Intrusion Detection Evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579–595, 2000.

[153] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDSP)," National Institute of Standards and Technology, Tech. Rep., February 2007, special Publication 800-94. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf

[154] Computer Network Defence Ltd., "Network IDSs," August 2008, accessed October 5, 2008. [Online]. Available: http://www.networkintrusion.co.uk/index.php/component/mtree/IDS-and-IPS/Network-IDS.html

[155] Cisco Systems, Inc., "Cisco Intrusion Prevention System - Products & Services - Cisco Systems," 2008, accessed July 7, 2008. [Online]. Available: http://www.cisco.com/en/US/products/sw/secursw/ps2113/index.html

[156] ——, "Cisco IPS Advanced Integration Module for Cisco 1841, 2800 and 3800 ISR [Cisco Intrusion Prevention System] - Cisco Systems (Data Sheet)," 2008, accessed July 7, 2008. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/routers/ps5853/ps5875/product_data_sheet0900aecd806c4e2a.pdf

[157] Allot Communications Ltd., "Allot Communications Ltd. - Products: NetEnforcer," 2008, accessed July 8, 2008. [Online]. Available: http://www.allot.com/index.php?option=com_content&task=view&id=45&Itemid=88888966

[158] NIKSUN, "NIKSUN NetDetector (Data Sheet)," NIKSUN, 1100 Cornwall Road, Monmouth Junction, NJ 08852 U.S.A., Tech. Rep., 2007. [Online]. Available: www.insacorp.com/Documents/niksun-datasheet-netdetector.pdf

[159] SecurityMetrics, "SecurityMetrics Appliance (Data Sheet)," SecurityMetrics, Victory House, 400 Pavillion Drive, Northampton Business Park, Northampton, NN4 7PA, Tech. Rep., 2008.

[160] ——, "Integrated Vulnerability Assessment - Intrusion Detection and Prevention - A Technical White Paper - Introduction, Implementation and Technology," SecurityMetrics, Victory House, 400 Pavillion Drive, Northampton Business

Park, Northampton, NN4 7PA, White Paper, 2008. [Online]. Available: http://www.securitymetrics.com/securitymetricsappliance.adp

[161] Check Point Software Technologies Ltd., "Stateful Inspection Technology - The industry Standard for Enterprise-Class Network Security Solutions," Check Point Software Technologies Ltd., 3A Jabotinsky Street, 24th Floor Ramat Gan 52520, Israel, White Paper, August 2005. [Online]. Available: http://www.checkpoint.com/products/downloads/Stateful_Inspection.pdf

[162] ——, "IPS-1 - Robust and Accurate Intrusion Prevention (Data Sheet)," Check Point Software Technologies Ltd., 3A Jabotinsky Street, 24th Floor Ramat Gan 52520, Israel, Tech. Rep., May 2008. [Online]. Available: http://www.checkpoint.com/products/downloads/ips-1_datasheet.pdf

[163] Juniper Networks, Inc., "Juniper Networks IDP 75/250/800/8200 (Data Sheet)," uniper Networks, Inc. , 1194 North Mathilda Avenue, Sunnyvale, California 94089-1206 U.S.A., Tech. Rep., 2008. [Online]. Available: http://www.juniper.net/products_and_services/intrusion_prevention_solutions/index.html

[164] Enterasys Networks, Inc., "Enterasys Dragon® 10 Gigabit Intrusion Detection and Prevention System (Data Sheet)," 2008, accessed July 3, 2008. [Online]. Available: http://www.enterasys.com/company/literature/dragon-10gIDS-ds.pdf

[165] ——, "Dragon Named IDS Product of the Year by Information Security Magazine," April 2006, accessed June 17, 2008. [Online]. Available: http://www.enterasys.com/company/press-release-item.aspx?id=667

[166] ——, "Enterasys - Company - News - Press Releases," December 2004, accessed October 19, 2005. [Online]. Available: http://secure.enterasys.com/corporate/pr/releases/2004/dec/12-15.html

[167] Lawerence Berkeley National Laboratory, "Bro Intrusion Detection System - Bro Overview," 2007, accessed July 8, 2008. [Online]. Available: http://bro-ids.org/

[168] Sourcefire, "Snort - the De Facto Standard for Iintrusion Detection/Prevention," 2008, accessed February 24, 2008. [Online]. Available: http://www.snort.org/

[169] H. Hu, W. Guo, B. Zhang, and X. Chen, "A Method of Security Measurement of the Network Data Transmission," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, April 2005, p. 8.

[170] J. Caberera, B. Ravichandran, and R. Mehra, "Statistical Traffic Modeling for Network Intrusion Detection," in *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, California, USA, 2000, pp. 466–473.

[171] W. Schleifer and M. Mannle, "Online Error Detection Through Observation of Traffic Self-Similarity," *IEEE Proceedings Communications*, vol. 148, no. 1, pp. 38–42, February 2001.

[172] C. S. Sastry, S. Rawat, A. K. Pujari, and V. P. Gulati, "Network Traffic Analysis Using Singular Value Decomposition and Multiscale Transforms," *Information Sciences*, vol. 177, no. 23, pp. 5275–5291, 2007.

[173] NLANR, "NLANR - National Laboratory for Applied Network Research - Internet measurement, Internet analysis," 2005, accessed March 29, 2008. [Online]. Available: http://moat.nlanr.net/

[174] CERT Coordination Center, "CERT® Advisory CA-1997-28 IP Denial-of-Service Attacks," 2008, accessed March 30, 2008. [Online]. Available: http://www.cert.org/advisories/CA-1997-28.html

[175] F. Gong, "Deciphering Detection Techniques: Part III Denial of Service Detection," McAfee Network Security Technologies Group, White Paper, January 2003. [Online]. Available: www.mcafee.com/us/local_content/white_papers/wp_ddt_dos.pdf

[176] B. Todd, "Distributed Denial of Service Attacks," February 2000, accessed March 30, 2008. [Online]. Available: http://www.linuxsecurity.com/resource_files/intrusion_detection/ddos-whitepaper.html

[177] V. Paxson, "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks," *ACM Computer Communications Review*, vol. 31, no. 3, 2001.

[178] Lincoln Laboratory, Massachusetts Institute of Technology, "MIT Lincoln Laboratory - Intrusion Detection Attacks Database," 2001, accessed February 15, 2008. [Online]. Available: http://www.ll.mit.edu/IST/ideval/docs/1999/attackDB.html

[179] ——, "MIT Lincoln Laboratory: Information Systems Technology - DARPA Intrusion Detection Evaluation," 2001, accessed February 15, 2008. [Online]. Available: http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/index.html

[180] W. Willinger and V. Paxson, "Where mathematics meets the internet," *Notices of the American Mathematical Society*, vol. 45, no. 8, pp. 961–970, 1998.

[181] Paul Wessel, "Critical Values for the Two-sample Kolmogorov-Smirnov Test (2-sided)," accessed August 27, 2008. [Online]. Available: www.soest.hawaii.edu/wessel/courses/gg313/Critical_KS.pdf

[182] P. R. M. Inácio, M. M. Freire, M. Pereira, and P. P. Monteiro, "Method for online simulation of traffic conditions on network aggregation points," Nokia Siemens Networks S.A., Rua Irmãos Siemens, no. 1, Amadora, Invention Report, June 2007.

[183] B. Mandelbrot, A. Fisher, and L. Calvet, "A Multifractal Model of Asset Returns," Cowles Foundation, Yale University, Cowles Foundation Discussion Papers 1164, September 1997.

[184] N. Scafetta, R. E. Moon, and B. J. West, "Fractal Response of Physiological Signals to Stress Conditions, Environmental Changes, and Neurodegenerative Giseases: Essays and Commentaries," *Complexity*, vol. 12, no. 5, pp. 12–17, 2007.

[185] P. L'Ecuyer and R. Simard, "Empirical Testing of Random Number Generators," April 2007, acessed May 10, 2008. [Online]. Available: http://www.iro.umontreal.ca/~simardr/testu01/tu01.html

[186] eSoft, "Modern Network Security: The Migration to Deep Packet Inspection," White paper, 2006, accessed March 29, 2008. [Online]. Available: http://www.esoft.com/pdf/whitepaper/DPI_white_paper.pdf

[187] P. R. M. Inácio, J. J. V. Gomes, M. M. Freire, M. Pereira, and P. P. Monteiro, "Zombie Identification Port," in *Proceedings of the Third International Conference on Internet Monitoring and Protection*, Bucharest, Romania., 2008.