UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

RobotChain: a Blockchain for Registering Robot Events using Tezos Technology

Miguel Ivo Ferreira Fernandes

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Luís A. Alexandre

Covilhã, Junho de 2019

Acknowledgements

This project, and most of my academic life would not be possible without the only element of my direct family, my Mother, supporting and giving the help that an academic student needs throughout its academic career.
And also my advisor for the confidence in me and my abilities and the opportunity given in this and previous works.
Finally but not least I would like to thank my friends and the various members of the SOCIA lab.
Supported by the Tezos Foundation through a grant for project RobotChain.

Resumo alargado

A robótica é um fator importante do dia-a-dia de uma fábrica moderna. Como tal, é importante manter estes *robots* a funcionar com o melhor desempenho possível e para fazer isto, uma forma é a utilização dos registos dos próprios robots de forma a identificar falhas ou comportamentos não intencionais sejam eles de natureza maliciosa ou não. As *Blockchains* são um tipo de base de dados eletrónica que previne a modificação de registos já inseridos. Esta tecnologia é interessante no contexto industrial a fim de prevenir alterações dos registos de robôs, sejam estas alterações não intencionais ou alterações maliciosas.

Nesta dissertação é criada a RobotChain, uma *blockchain* para trabalhar com *robots* industriais, assente na *blockchain* Tezos. É apresentada também uma técnica de segmentação temporal de uma *blockchain* a fim de se poder fazer uso de dispositivos de computação mais fracos, com armazenamento mais limitado, de forma eficiente. Estes dispositivos de computação, referidos como *compute devices*, são módulos utilizados para fazer interface entre os robôs e a *blockchain*, evitando percas de desempenho dos robôs com a execução do novo programa associado, e, servem para tornar uniforme a informação enviada para a rede. É considerada também a existência de nodos de armazenamento, denominados *cold storage nodes* onde é guardada informação completa da rede, nodos de pedidos, *query nodes*, nodos que fazem interface entre operadores humanos e a *blockchain*, permitindo o acesso a informação contida nesta e nodos oráculos, nodos que interagem com a rede a partir de contratos inteligentes.

São apresentados também resultados experimentais resultantes das varias alterações feitas à *blockchain*, assim como a funcionalidade de segmentação temporal.

Abstract

Robots are important equipment in the modern day factory environment. To maintain and improve factory productivity, ledgers containing robotic actions may be used to identify possible bottleneck points in a assembly line or to serve as a record of in unintentional behaviours, be it of a malicious nature or not. Blockchains are a type of secure ledger, that prevent unwanted changes. These blockchains, during their lifetimes, record large amounts of data, that in a common usage its kept on its entirety. This dissertation presents RobotChain, a possible solution using blockchain technology that prevents unwanted changes in a robotic action ledger, and provides a way to use the said ledger in order to aid in production efficiency or other management requirements, and presents a time-segmentation solution for devices with limited storage capacity, integrated in RobotChain. It also presents various experiments related to the performance of Tezos blockchain network with the various modifications.

Keywords

Keywords in English

1. blockchain
2. robots
3. Tezos
4. time-segmentation

Contents

1 Introduction **1**
1.1 Objectives . . . 1
1.2 Main Contributions . . . 1
1.3 Dissertation Organization . . . 2

2 Related Work **3**
2.1 Papers . . . 3
2.2 Companies and Whitepapers . . . 4
2.3 News Articles . . . 6
2.4 Conclusions . . . 7

3 The Tezos Blockchain **9**
3.1 Introduction . . . 9
3.2 Economy Concepts . . . 9
3.2.1 Gas . . . 9
3.3 Network Concepts . . . 9
3.3.1 Node . . . 9
3.3.2 Client . . . 9
3.4 Protocol Genesis 000_Ps9mPmXa . . . 10
3.5 Protocol Demo . . . 10
3.6 MainNet Protocol 002_PsYLVpVv . . . 10
3.6.1 Context . . . 10
3.6.2 Operation . . . 11
3.6.3 Block Creation . . . 11
3.6.4 Baker . . . 11
3.6.5 Endorser . . . 11
3.6.6 Block . . . 11
3.7 Tezos Updates . . . 13
3.7.1 Protocol 003_PsddFKi3 . . . 13
3.7.2 History Mode . . . 14
3.8 Conclusions . . . 14

4 RobotChain **15**
4.1 Introduction . . . 15
4.2 Addition of a Description Field to the RobotChain Protocol . . . 15
4.3 Modification of the Gas Interpretation . . . 16
4.4 Allow Multiple Operations per User in the Same Block . . . 17
4.5 Time-Segmentation . . . 17
4.5.1 Network Structure . . . 17
4.5.2 The Idea . . . 18
4.5.3 Implementation . . . 19
4.6 Conclusions . . . 20

5 Experiments **21**
5.1 Introduction . . . 21
5.2 Sandbox Experiment . . . 21
5.2.1 Experimental Setup . . . 21
5.2.2 Trial Description . . . 21
5.2.3 Results and Discussion . . . 22
5.3 CPU Performance Experiments . . . 24
5.4 I/O Performance Experiments . . . 25
5.4.1 Storage speed evaluation . . . 26
5.4.2 Network Speed Evaluation . . . 28
5.4.3 Tezos I/O Evaluation . . . 29
5.5 Time-Segmentation with Protocol 1 . . . 29
5.6 Time-Segmentation with Protocol 4 . . . 30
5.6.1 Regular Node Storage . . . 30
5.6.2 Cold Storage Node Storage . . . 31
5.7 Conclusions . . . 32

6 Conclusions and Future Work **35**
6.1 Conclusions . . . 35
6.2 Future work . . . 35

Bibliography **37**

A Tezos Blockchain **39**
A.1 Block . . . 39
A.2 Transactions . . . 40

List of Figures

3.1 Node initialisation schematic. 10

4.1 Overview of the RobotChain. The compute modules are devices that serve as interface between robots and RobotChain, Cold Storage nodes will save all the segments of the blockchain, Query nodes allow queries to the blockchain and Oracles are external entities that interact with the blockchain through smart contracts. One Cold Storage will also serve as the Genesis Node. 16
4.2 Screenshot of the command *transfer* after the changes presented in section 4.2. The shown modification is the addition of *"with <dsc>"*, where the *with* is a non-variable keyword and the *"<dsc>"* the parameter itself. 16
4.3 Example of the process of doing a transaction, transferring 1 tz from bootstrap1 to bootstrap2, with the description *"Transaction Description"*. The symbol -> is the terminal command prompt. 17
4.4 Screenshot of the receipt generated after the transaction. 17
4.5 Visual representation of the second approach of the time-segmentation process. The red arrows represent the use of hashes to allow inter-segment connectivity. 18

5.1 Average transaction time as a function of the number of nodes. 22
5.2 Average transaction time with the 6 tested protocol parameters, with the parameter "time per block" being 1,2,3,5,10,60 respectively, with the left referring the hard drive results and in the right the Tmpfs. Note the different "Transaction time" scales. 24
5.3 Storage speed on various storage devices. Ramfs corresponds to RAM FileSystem, Tmpfs corresponds to Tmpfs, a Ramfs derivate that uses swap memory. SSD correspons to a Samsung SSD 970 EVO 500GB, and the HDD is a Seagate Barracuda ST3000DM007-1WY10G, where both SSD and HDD using ext4 filesystem. 28
5.4 Storage size on each regular node (not cold storage), for three versions of the blockchain, as a function of the number of blocks. For the unsegmented blockchain (normal), every node contains the complete blockchain. 30
5.5 Storage size on each compute device node (not cold storage), for four versions of the blockchain, as a function of the number of blocks. The segmented blockchain running mode is archive. The left sub-figure presents the results for the four tests, while the right sub-figure presents only the results for the rolling mode and the segmentation version, for improved comparison. 31
5.6 The left sub-figure presents the compute device storage size as a function of the number of blocks per segment for a total of 1000 blocks, compared with the corresponding rolling mode storage size of the Tezos blockchain. The right sub-figure presents the cold storage size as a number of blocks per segment for a total of 1000 blocks, compared with the archive mode of the Tezos blockchain. Results for the full mode are not presented since they are not comparable to any of the node types on our approach, regarding storage capabilities. 32

List of Tables

5.1 Transaction statistics for the trials in the experiments with four blocks per cycle and eight blocks per cycle. Using a virtual machine with four threads allocated. Each experiment has represented the Total transaction time, Average transaction time and Timed-out transactions. 23
5.2 Average transaction time with the 6 tested protocol parameters, with the parameter ”time per block” being 1,2,3,5,10,60 respectively, with the left referring the hard drive results and in the right the Tmpfs. 24
5.3 Priority block value for each of the variations of the parameter ”Time between blocks”, in the experiment on Hard Drive storage device. The priority of a block corresponds to the baker’s baking right rank in the priority list. The Quantity is the Quantity of blocks of a given priority, the Sum is the sum of blocks of the given priority and previous priorities, and the percent is the percentage of blocks of the given priority. 26
5.4 Priority block value for each of the variations of the parameter ”Time between blocks”, in the experiment on Tmpfs storage device. The priority of a block corresponds to the baker’s baking right rank in the priority list. The Quantity is the Quantity of blocks of a given priority, the Sum is the sum of blocks of the given priority and previous priorities, and the percent is the percentage of blocks of the given priority. 27
5.5 Data transfer between two computers, using iperf3, with left table using PC A as server and PC B as client, and the right table using PC B as server and PC A as client, running for a total of 10 seconds. 28
5.6 IOtop results and transaction statistics for the 5 executions of the experiment in the HDD storage device, where TpB corresponds to Transactions per Block. . . . 29
5.7 IOtop results and transaction statistics for the 5 executions of the experiment in the SSD storage device, where TpB corresponds to Transactions per Block. 29

Chapter 1

Introduction

1.1 Objectives

This dissertation presents RobotChain, a private blockchain aimed for robotic enabled factories as a method for keeping a ledger of each and every action performed by robots. This ledger is useful in order to understand production line performance, where are possible points of improvement, and where are possible faulty or under-performing robots. Said under-performing or faulty robots are a hindrance for the processing line, since they are high cost machines that have to pay back their investment in a short amount of time. With a crypto-ledger, posterior changes to the ledger in order to hide under-performing robots are hard to perform, requiring vast amounts of resources.

Blockchain technology is desired in order to prevent malicious entities from modifying the ledger, preventing manipulation of information coming from robots.

This project will use *Tezos*[Goo08] technology in order to develop the blockchain. Since it will work in a contained and limited environment such as a factory, this blockchain network is going to be private and since robots can produce a large number of actions in a small amount of time, performance for the blockchain is going to be a priority.

Smart contracts will also be integrated into the blockchain since they allow embedding code on the blockchain and execute said code with the data contained by the blockchain.

1.2 Main Contributions

Two main contributions are made in this dissertation. The first one is the addition of a way to register a robotic log entry per transaction together with the performance improvements done to allow multiple log entries of a robot inside a single block, independently of the size occupied by said logs (sections 4.2 to 4.4). The second contribution is a solution named time-segmentation that allows for cheap nodes to use this blockchain, reducing storage costs and allowing faster deployment of new nodes without compromising all the benefits that arise from the use of blockchains in these contexts.

The results of this dissertation were published a conference article[FA18], on the First Symposium on Blockchain for Robotic Systems, in the MIT Media Lab, where RobotChain is defined and described along with the initial experiments done on the blockchain, (section 5.2). An additional journal paper is also under review on Robotics and Computer-Integrated Manufacturing, also available on Arxiv[FA19], where the time-segmentation is described. It is presented in this document in section 4.5 and its results are in section 5.6.

1.3 Dissertation Organization

The remainder of this document is organized as follows: chapter 2 presents work related to the dissertation, regarding blockchains with robotic applications, monitoring systems, and company products related to the presented work. Chapter 3 presents the Tezos blockchain technology, describing its main features and the updates it has received. Chapter 4 presents the RobotChain, the modifications to the Tezos blockchain to allow it to register robotic events with an improved data throughput and with cheap compute devices. Chapter 5 presents experiments regarding performance of the Tezos blockchain with and without the RobotChain modifications. Chapter 6 presents the conclusions and future work related to this dissertation.

Chapter 2

Related Work

This chapter presents several works related to robotics and blockchain technology. It is divided into three sections, the first one containing scientific papers, the second containing company whitepapers and the third containing news articles related to the dissertation.

2.1 Papers

Ferrer[Fer16] presents how blockchain can improve robotic swarm systems by solving some existing issues. Those issues are data confidentiality, distributed decision making, ability to work in different and dynamic environments without changes to the control program and a way to ensure safety and legal responsibility for the robotic nodes in the swarm in order to be ”integrated” with the human society. It is also referred that by adding blockchain in the concept of swarm robotics, it is a deviation of the minimalism of the common research practices used in swarm robotics.

Byzantine Fault tolerance is the concern for fault-tolerance in distributed computer systems where components may fail or be unreliable, a component or system can be unreliable in a form that it responds correctly or incorrectly to other systems or fail-safe systems. The name refers to the ”Byzantine General’s Problem”, where a group of generals and their armies encircle a city and try to make a plan whether to attack or retreat from the city, and without a consensus, the result would be worse than a combined assault or combined retreat. If all the generals are loyal, the problem is easy to solve, but, in the real world there are agents that are faulty or malicious in nature, meaning, generals may vote sub-optimally, sending a message to some and the opposite message to others, creating ”two sides” where half may attack and the other half retreat.

This concept of Byzantine fault is important due to the fact that the robotic swarms may have problems related to swarm coordination methods. Strobel[SCD] presents a proof-of-concept method that uses blockchain smart contract technology in order to improve security of the robotic swarm in order to improve the stability of the swarm coordination mechanisms and expel Byzantine members from the swarm. This concept is also studied for its performance in decision making regarding the presence or absence of Byzantine robots.

Danilov[DRKA18] presents an agent model for operation on a kind of trading marked named *robonomics*[LKK+]. It is focused on agent-based systems, where the behaviour is described as ”nondeterministic finite state automata”, presenting a Model Checking verification technique in order to detect and filter malfunctioning agents. The validation technique can be implemented on a consensus protocol or part of a blockchain decentralized application. As a real live test, a prototype implementation of Duckietown with moving robots is provided, following a set of instructions related to movement.

In Ferrer *et. al.*[FRHP18], RoboChain (not RobotChain!) is presented. It’s a learning framework that attempts to fight the problems with privacy issues related to using personal information using blockchain technology, sharing data and machine learning models, allowing multiple robotic

units to work at different places, sharing their data and their knowledge. It uses machine learning technologies such as tensorflow lite, in embedded devices such as low-cost robotic units. Since this approach assumes a situation where the participants are private entities, not being available to the public, a level of trust between said parties is considered and as such, it isn't considered the presence of a "malicious entity", but it still provides a way to verify the integrity of the interactions and learning in the chain.

Hasan *et. al.*[HDRS18] present a blockchain framework for a secure ride-sharing service between autonomous vehicles and passengers, using blockchain as a communication mechanism that is dependable and trustworthy.

Lei[Lbk17] presents a mean to automatically extract pattern rules for Complex Event Processing Systems, where this kind of system is used to processes a multitude of data inputs to detect events that may point to other kind of complex events.

An example of this system may be the one used in a car, where the data inputs are the speed, tire pressure and seat sensors. An example of an event detected by this system is the lowering of tire pressure, indicating a leak, or, a sudden reduction of pressure that may indicate a tire blowout. Other complex events may be the sudden reduction of pressure in a tire, accompanied by a drop in vehicle speed with the addition of the sensor in the driver's seat no longer registering the driver's presence, which may indicate an accident. In order to understand and extrapolate these complex events, this system needs a set of rules. The method presented uses association analysis, where the higher the correlation between two or more items corresponds to a higher value between them. Time consumption over the proposed method and the traditional method is presented as results, with the proposed method having a lower time consumption over the number of inputs.

Snatkin[ASMAE] presents a system that collects in real time event information in a workshop floor, in a way that may be used efficiently for decision making, not just for floor management but for administration as well, allowing trend analysis, estimations and fault detection in order to improve efficiency. It starts by presenting the drawbacks of a Production Monitoring System in a Small or Medium Enterprise, where this PMS is a sub-part of a Manufacturing Execution System, where this system serves as a linking point between the factory floor and the offices in an attempt to solve ERP (Enterprise Resource Planning) system integration problems, meaning, MES is the system that provides the raw data from the shop floor for the rest of the ERP system, and, the ERP system provides the interface for the administration to decide upon the future of the company. It provides a case study in both the Tallinn University of Technology and at a private company, with the used sensors for the machines, and what those sensors are detecting.

2.2 Companies and Whitepapers

The SKYFchain[SKY18] platform attempts to provide a solution for an unmanned logistics system, providing robotic hardware for air, sea and land transportation.

The concept of this platform is to "integrate information of all market participants and organize the execution of smart-contracts for transactions". This platform will be tested with the drone already created, and will attempt in the following years to add to its ecosystem other ground and water based drones.

It will be based on two building blocks:

- The SKYFchain blockchain itself

- The Etherum SKYF tokens

And a third component that is the synchronization between the two blocks. In the beginning, this blockchain will be centralised, but, as it grows, with more participants, it will become decentralized.
Kambria[Kam18] is a project by Kambria International that will attempt to create an innovation & collaboration platform where the objective is improve development speed and adoption for robotic technologies. It does this by preventing the ”reinvention of the wheel” and allowing users to share their knowledge, be it code or schematics and also allowing companies to tap on the collective developer knowledge. OhmnLabs, also owned by the founders of Kambria, are providing a ”non-exclusive royalty-free license of necessary components of its Ohmni robot, consisting of a robotics repository, high-level behavioural library, and modular components, as the basis for the Kambria Platform”.
The developers are given an incentive with the integration of blockchain and crypto-economics, since it provides economic incentives to contribute to the platform. There is also the intent to ”punish” developers that ”defect”, by using a game theory technique named ”Grim Trigger” *Grim Trigger* is a trigger strategy for a repeatable game, where the most famous example is the prisoner’s dilemma, where there is player A and Player B, each with two options, either to Cooperate, or to Defect. Cooperation has a base value of 1 for both players, and defection has a base value of 2 for the defecting player and -1 for other player. With this trigger strategy, both players cooperate with each other up until one of them defects. Upon that defection, the player whose opponent defected would simply enter a ”defection loop”.
This architecture is going to be composed of five main parts:

- The Code Base - The modular essence of Kambria Platform, handling hardware and software, from mechanical designs to controllers for those designs.
- The Marketplace - The main engine for innovation, allowing entities to issue bounties in order to create contracts workable by the community.
- The Manufacturing Alliance - A partnership in order to improve the speed of robot fabrication by reducing bureaucratic process.
- Value Capture - Attempts to maintain sustainability by revaluating various critical value points.
- Legal Enforcement - Serves as a deterrent of freeloading and as a protection of the collective work of the platform.

Interactions in the platform will be facilitated by KAT and Kambria Karma, the first being a ERC20 token, allowing access, rewards and incentives for every stakeholders and the Karma is a kind of point system for evaluating useful work provided and contributions.
In the Bitcoin whitepaper[Nak08], a peer-to-peer online payment method is presented, where payments can be made directly without usage of a financial middleman, removing trust issues related to either participant or the financial entity. It does this by using digital signatures and peer-to-peer networking to impede double-spending, where the same ”coins” or ”tokens” are used to pay two different transactions, and this problem is solved using time stamped transactions and hashing them into a chain of hashed proof-of-work, creating a ledger that cannot be modified without recreating the proof-of-work. Due to the fact that most CPU power pool is ”loyal” to the concept, as in, the majority of the CPU processing power is collaborating instead

of attacking the network, attacks on the network that involve forging transactions are hard. Other advantage of this network is the simplicity in regarding its structure.
In Tezos[Goo08] a self-amending crypto-ledger implemented in *OCaml* is presented. Instead of using a genesis block or hash, it starts with a seed protocol, where this protocol can be amended in order to replicate other blockchains. The main feature of this blockchain is the fact that implements a protocol that can adapt itself by transforming a Context object. These amendments work over cycles, these cycles take about three months and are suggested by a submission to the chain, and stakeholders may vote for these amendments. These amendments, if accepted, are first inserted into a *"testnet"*, and after that, a second confirming vote is made. If the second vote is successful, it is integrated into the the main protocol. This technology is further explained on chapter 3, since it is the technological basis of this dissertation.
The whitepaper by Chorus[LV18] presents a blockchain operating as a network of vehicles under 5G networks. This rises feasibility concerns regarding transaction processing and smart contract usage. It uses Tezos network as a basis due to its Turing complete smart contract and built-in formal verification of the programming language, allowing a higher security to the blockchain and its participants. This allows focus in the creation of the mobile ad hoc blockchains that allow disconnections from the main network and allowing sub networks to preform transactions. It also presents smart contracts ad a "Orchestration and choreography protocols that facilitate, verify and enact with computing means a negotiated agreement between consenting parties" It presents the possible concept of vehicles self ownership, exchanging services, like transportation for parking space or battery charging. It presents a set of goals for this system to be successful, such as scalability, cost efficiency and automation. In the end it is also presented a prototype architecture for the chorus network
The whitepaper from Lonshakov[LKK+] presents a model for a kind of trading marked regarding robot economics, referred as *robonomics*. This trade model would allow to remove the middleman between the consumer and a robot, allowing a robot to "own itself", to pay its own expenses and maintenance, or pay for the resources need to develop its own work. With the aid of crypto-currency, in this case, Etherum, a robot is able to own money, and with smart contracts, it is possible to have an economical meaningful transaction between entities, allowing robots to execute programs specified in the smart contract in exchange of for token. This whitepaper presents the platform where these robots would work, how this concept is being developed, the robonomics token, the applications for the platform, the description for the robonomics work and the scalability of this system.

2.3 News Articles

This article[Agg18] refers the idea of applying blockchain in robotic swarms as a way to increase security, due to the possible existence of faulty or malicious robots contained in the swarm. It is achieved with public key cryptography.
In this article[Her18] the idea of using blockchain technology in order to improve security in swarm based robotic systems is also referred. It is considered useful in the security field but also in the decision making process. This is due to the fact that the blockchain provides a way of distributed decision making, by allowing votes for decisions faced by the swarm and by allowing some level of behaviour differentiation, where the blockchain can change control algorithms in order to have a better chance to overcome challenges. Finally some negative points related to the blockchain usage on robot swarms are mentioned: the fact that the current blockchain

takes 10 minutes in order to confirm a transaction and the possible overburden of each robot in the swarm if it has to have a copy of the entire transaction ledger.

2.4 Conclusions

This chapter presents work that is related to this dissertation's contributions, be it related to blockchains or monitoring systems in factories. It also presented both the academic research view as the enterprise view of robotic blockchains, their applications and counterpoints regarding the application of blockchains in the robotics field. The research done allows the understanding of the work published in the area, the problems that affect the technologies and how to solve them. This may hint at problems that we can face and also hint on the solution to challenges that we may face.

An example of a problem faced is the Input/Output referred in sections 5.3 & 5.4, where, in short, a hard drive storage device is unable to support the data throughput required by the Tezos network registering robotic logs. In Danilov[DRKA18], the ARIA project is mentioned. This project, the Autonomous Intelligent Robot Agent, is a project by the authors of the concept *robonomics*. During the mentioned First Symposium on Blockchain for Robotic Systems, in a conversation with Professor Aleksandr Kapitonov (researcher working in the ARIA project and an author of the robonomics[LKK+] concept), where it was mentioned a recommended system requirement for the ARIA project, due to the fact that their system had problems running in a hard drive storage device. Although different underlying technologies, the problem was of a similar nature with a similar solution approach, where in their case a solid state drive is considered a recomended system requirement, and in our case, a RAM based file system is used for the execution of RobotChain.

Chapter 3

The Tezos Blockchain

3.1 Introduction

Due to the usage of the Tezos blockchain technology, understanding the basis of the used technology is important. As such, a chapter describing and documenting the Tezos blockchain technology is considered important.
For the purpose of this document, the code and documentation that serves as a basis for this document is present in the link `https://gitlab.com/tezos/tezos/tree/mainnet` with the commit `079f5db94468eca575df273baf8c4d0fb6148aac`, present in the link `https://gitlab.com/tezos/tezos/tree/079f5db94468eca575df273baf8c4d0fb6148aac`.

3.2 Economy Concepts

A single coin in this network is a Tez, with the smallest unit being a cent, with two decimal cases, with the stock ticker XTZ. In addition, 10,000.00 is considered a roll initially, with the possibility of being changed as the network evolves. Rolls give eligibility to their owners to participate in baking (block construction) and endorsing (block validation) activities.
Double spending is the act of using the same token to pay two different transactions.

3.2.1 Gas

Gas is a measurement of operations and is used to calculate a fee for the various operations in the blockchain. In the Tezos network it serves as encouragement to write efficient programs. This gas is not just a fee for the transaction or smart contract itself, but also contains a fee per byte of data contained in the transaction. The latter was added as a deterrent to spam attacks.

3.3 Network Concepts

3.3.1 Node

This is the basic element of the network, that connects with every other element via peer-to-peer networking. It's initialization process is shown on Fig. 3.1. If synchronized, it also contains the latest information of the network, and as such, contains the needed information that the clients require in order to interface with the network.

3.3.2 Client

This is main the interface for interactions with the blockchain, and, due to the fact that Tezos has the ability to change over time, the commands present in the network are not integrated in the client, but are requested to the node used for interaction, normally a local node where

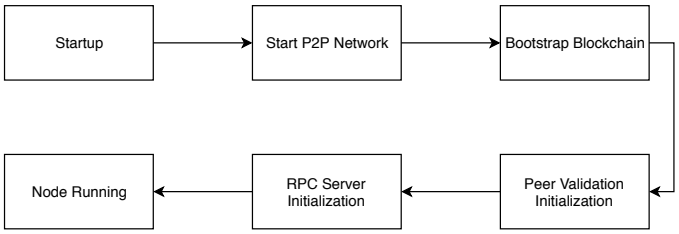

Figure 3.1: Node initialisation schematic.

the client is being run. It uses RPC to communicate with the mentioned node, providing JSON strings containing the information.

3.4 Protocol Genesis 000_Ps9mPmXa

The protocol that initializes the Genesis block is named *000_Ps9mPmXa*. This is a necessity to any blockchain network due to the fact that this first block needs to be hardcoded, since the following block requires a previous one to create its new hash. It contains the dictator public key hardcoded, accepting a single block signed with the mentioned key, with the only action to activate a new protocol chosen by the dictator, with a set of parameters. Similar to other blockchains, Tezos uses this protocol as a seed to initialize the blockchain. With this protocol concept, the Tezos blockchain is allowed to mutate, or amend itself, without forks.
It also contains a hardcoded public key regarding the Dictator account. The public key contained does not correspond to the key provided in the sandbox parameters. This key was later changed to a known public/secret key pair in order to separate the Tezos MainNet blockchain and the in-development RobotChain.

3.5 Protocol Demo

This protocol acts as a demo protocol that contains a minimal setup, with empty block baking, a sample error command and an echo service. This serves as a minimalist example of the concept of protocol that the Tezos platform uses.

3.6 MainNet Protocol 002_PsYLVpVv

Protocol *002_PsYLVpVv* was the first protocol version of the Tezos MainNet and was officially launched[Fou18] on September 17th, 2018, under the name ”Alpha” protocol. It was forcedly[Git] upgraded to Protocol 003_PsddFKi3 reaching block level 204762, on November 27th, 2018.

3.6.1 Context

The context is the current state of the blockchain. It is here where information such ”this account has XXXX Tez” or ”this smart contract has this information and code”, is contained. It is modified by operations contained in blocks.

3.6.2 Operation

An operation is a change of the context. It can be a transaction between accounts, a smart contract call, it can be an endorsement. These operations are included in blocks and the context is modified upon block inclusion. Since a block can have multiple operations, these are enacted in batches.

3.6.3 Block Creation

Tezos uses proof-of-stake as a consensus method, where the creator of the next block is chosen by randomly selecting a member of the network, with preference over wealthy or older members. In the Tezos case, it chooses the next block creator by randomly selecting a roll owner. This consensus method contrasts with proof-of-work, where a block creator is chosen via a CPU work time, like solving a computational puzzle or other CPU calculations. Proof-of-stake is more power efficient due to not needing to execute CPU intensive tasks for the chance to create a block. This allows more nodes to join the network and make it impervious against a 51% attack. A 51% attack is an attack on blockchain technologies where a malicious entity attains at least 51% of the block creation chance, in the Tezos case, owning at least 51% of the currency, which would not be feasible.

3.6.4 Baker

Baking is the act of creating, signing and publishing a new block onto the blockchain. A block is considered baked, by an entity operating on the blockchain named baker. With Tezos using proof-of-stake, the process of choosing a baker is achieved by randomly selecting a set of rolls, allowing the roll owner to bake the next block, where the first may be allowed to bake for the next minute, the next for two minutes, and so on.

3.6.5 Endorser

Endorsing is the act that verifies the blocks that are made, confirming that the block is in fact correctly built and is not doubly baked, preventing a double spending attack on the network. The entity that endorses a block is named an endorser and is selected via the same process used to select the baker. Endorsers receive improved rewards the lowest the time interval between the current block and the previous one. Due to the fact that in order to bake, there is a amount of Tez frozen as a security deposit, in occasions where double baking is achieved, said fee is forfeit. Rewards for both endorser and baker are only paid if the block is included in the blockchain.

3.6.6 Block

A block is collection of operations specific to the protocol running and contain metadata that is protocol agnostic. This metadata includes information as the block predecessor, block level or a timestamp of the block creation.
The block structure is as follows:

- protocol - the protocol hash that currently is running.
- chain_id: chain identifier

- hash: the hash of the current block.
- header:
  - level: the value concerning current block depth inside the blockchain.
  - proto: number concerning the number of changes of protocol since genesis.
  - predecessor: the block predecessor to the current block.
  - timestamp: the timestamp regarding date when the block was baked.
  - validation_pass: ”number of validation passes (also number of lists of lists of operations)”[Res19]
  - operations_hash: ”the root hash of a merkle tree of a list of root hashes of merkle tree for various sets of operations in the block”[Res19]
  - fitness: ”a sequence of sequences of unsigned bytes, ordered by length and then lexicographically. It represents the claimed fitness of the chain ending in this block.”[Res19]
  - context: ”Hash of the state of the context after application of this block. Useful for light clients.”[Res19]
  - priority: the priority for the baker in the list of available bakers.
  - proof_of_nonce: ”a nonce used to pass a low-difficulty proof-of-work for the block, as a spam prevention measure.”[Res19]
  - signature: ”a digital signature of the shell and protocol headers (excluding the signature itself).”[Res19]
- metadata:
  - protocol: the current protocol.
  - next_protocol: the next protocol that will be activated.
  - test_chain_status:
    - status: Status of the test_chain, if it is running or not.
  - max_operations_ttl:
  - max_operation_data_length:
  - max_block_header_length: [List]
  - baker: hash of the baker that created this block.
  - level
    - level The current block level.
    - level_position:
    - cycle: the current cycle.
    - cycle_position:
    - voting_period:
    - voting_period_position:
    - expected_commitment
  - voting_period_kind:
  - nonce_hash: Hash of the proof of nonce.
  - consumed_gas: Gas consumed by this operation.

- deactivated:
- balance_updates:
- operations: list of operations on the block:
    * kind: the kind of the operation made.
    * source: hash belonging to the transaction sender.
    * fee: the fee for the transaction processing.
    * counter: Number of times
    * gas_limit: Gas limit of the operation.
    * storage_limit: Storage limit of the operation
    * amount: the amount of Tez transferred.
    * destination: hash belonging to the transaction receiver.

Listing A.1 presents a block in JSON without any modification. Some of the fields seem not to be used on this specific protocol version.

3.7 Tezos Updates

3.7.1 Protocol 003_PsddFKi3

This is the first update to the Tezos Alpha protocol, forcedly activated on block level 204762, on November 27th, 2018. It includes new features and fixes to the protocol.
Several fixes were introduced:

- Disallowing spam account creation by requiring to burn a fee to create both regular and implicit accounts.
- An error of nonce revelation that compromised block building is now correctly handled.
- Correct count of the rolls for voting purposes.

It also introduces the RPC interfaces that allow status queries related to ballot system status. These are:

- Sum of ballots cast so far during a voting period.
- Ballots cast so far during a voting period.
- Current period kind: proposal, testing_vote, testing, promotion_vote.
- Current expected quorum.
- List of proposals with number of supporters.
- Current proposal under evaluation.
- List of delegates with their voting weight, in number of rolls.

3.7.2 History Mode

As of February 4th, 2019, Tezos has a new feature[Nom19] called the history mode. This new feature allows a node to select how it keeps its past data, with three different settings to chose from. These modes rely on the checkpoint feature present in the Tezos blockchain, where these checkpoints act as a regular interval anchor of consensus.
These modes are the archive mode, where the node keeps everything, which corresponds to the normal Tezos working mode before the update, the full mode, that is the new default mode, where the node stores all data like the archive mode, but erases information that can be regenerated from previous checkpoints. The last mode is the rolling mode, where the nodes only keep the latest checkpoint, deleting information previous to the checkpoint. These modes are a configuration parameter of a node in the Tezos blockchain.
As stated by the developers, this new feature has restrictions regarding mode changing. In short, it allows any mode to downsize, allowing archive mode to change to full mode or to rolling mode, and allows full to change to rolling mode, since it is only deleting data. Node upsizing is disallowed, such as upgrading from rolling node to full or archive mode, or from full mode to archive mode, since this situation requires rebuilding deleted information.
The initialization of the nodes now has two ways of bootstrapping, regardless of the activated mode, it can use peer-to-peer to connect to other peers and download the information or using the new snapshot feature, consisting on a file import/export. The snapshot import does the same checks as synchronizing from other nodes, recomputing and checking all the hashes encountered in the snapshot file, effectively doing the same work as the peer-to-peer synchronization but without the network cost.
Although the import feature validates the consistence of the imported file, it does not validate the veracity of the information contained, meaning, that the information imported may be valid, but it may not correlate to the running Tezos network being bootstrapped, be it MainNet or other network. Manual verification of the imported information, such as checking if the imported hash is contained in the Tezos main network is needed. These export snapshots must be created from full or archive nodes.

3.8 Conclusions

This chapter presents the Tezos blockchain technology, since it is the basis of the RobotChain project. The basis of the technology is explained, such as the block creation method and economical terms that are used throughout. With the knowledge obtained while understanding the Tezos blockchain, the modifications present on the next chapter were done.

Chapter 4

RobotChain

4.1 Introduction

This chapter will present the various changes done to the Tezos blockchain technology in order to create RobotChain, presenting the various concepts that were altered and how those concepts were altered. These concepts were altered in the presented chapter order. Unless stated otherwise, the basis for these modifications is Protocol 002_PsYLVpVv, and through the rest of this document, this modified protocol is referred to as ”RobotChain”.

4.2 Addition of a Description Field to the RobotChain Protocol

The first addition to the Tezos code was a description field for the transactions in the network, with this description field being a string needed as a parameter of a transaction. This was done as a way to understand the underling code, regarding how transactions work, how the block is constructed, how is the performance currently and how may the performance be improved to the required constraints imposed by robotic log registration. In order to enact this change, several files belonging to the two available protocols were changed.

In the genesis protocol, the dictator key, the key that is used to activate new protocols, needed to be changed to a known private-public key set, in this case, the bootstrap1 key pair. This allows our own activation of the network, allowing the creation of a private Tezos blockchain network, with our own protocol and protocol parameters. This was done by changing the hardcoded dictator key present in the file *data.ml* in the genesis protocol. This change was done instead of using the *set_pubkey* method in order to not be limited to the sandbox mode available. The sandbox mode allows modification of the network and the underlying parameters as was needed, but with the limitation of disallowing networking, effectively only allowing same host networked nodes.

In the original MainNet protocol, protocol 002_PsYLVpVv, the concept of transaction was also changed, with the addition of the ”transaction_description” parameter to the transaction. Programmatically, the transaction object has a new field, the *transaction_description*, encoded as a string. The various methods that work with transactions, including the command line parsing, were modified in order to contain and use the new parameter.

To do this, changes to the client command parser were done, in order to expect a new parameter. Further changes were done in the client in order to show the transaction description in the transaction receipt shown after making a transaction.

This string was considered a starting point to the RobotChain protocol, since this description parameter could be used to store robot information.

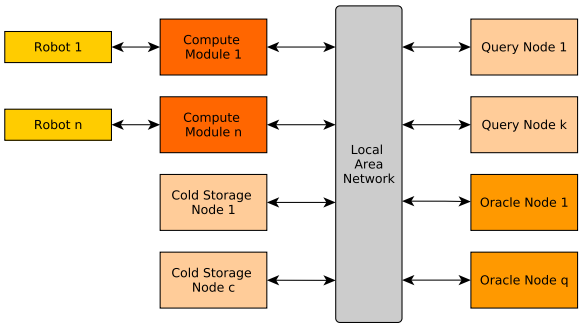

Figure 4.1: Overview of the RobotChain. The compute modules are devices that serve as interface between robots and RobotChain, Cold Storage nodes will save all the segments of the blockchain, Query nodes allow queries to the blockchain and Oracles are external entities that interact with the blockchain through smart contracts. One Cold Storage will also serve as the Genesis Node.

```
transfer <qty> from <src> to <dst> with <dsc> [--fee <amount>]
    [-D --dry-run] [-G --gas-limit <amount>] [-S --storage-limit <amount>]
    [-C --counter <counter>] [--arg <data>] [-q --no-print-source]
    [--minimal-fees <amount>] [--minimal-nanotez-per-byte <amount>]
    [--minimal-nanotez-per-gas-unit <amount>] [--force-low-fee]
    [--fee-cap <amount>] [--burn-cap <amount>]
  Transfer tokens / call a smart contract.
  <qty>: amount taken from source in ꜩ
  <src>: name of the source contract
  <dst>: name/literal of the destination contract
  <dsc>: description
  --fee <amount>: fee in ꜩ to pay to the baker
  -D --dry-run: don't inject the operation, just display it
  -G --gas-limit <amount>: Set the gas limit of the transaction instead of letting the client decide based on a simulation
  -S --storage-limit <amount>: Set the storage limit of the transaction instead of letting the client decide based on a simulation
  -C --counter <counter>: Set the counter to be used by the transaction
  --arg <data>: argument passed to the contract's script, if needed
  -q --no-print-source: don't print the source code
  --minimal-fees <amount>: exclude operations with fees lower than this threshold (in tez)
  --minimal-nanotez-per-byte <amount>: exclude operations with fees per byte lower than this threshold (in nanotez)
  --minimal-nanotez-per-gas-unit <amount>: exclude operations with fees per gas lower than this threshold (in nanotez)
  --force-low-fee: Don't check that the fee is lower than the estimated default value
  --fee-cap <amount>: Set the fee cap
  --burn-cap <amount>: Set the burn cap
```

Figure 4.2: Screenshot of the command *transfer* after the changes presented in section 4.2. The shown modification is the addition of *"with <dsc>"*, where the *with* is a non-variable keyword and the *"<dsc>"* the parameter itself.

4.3 Modification of the Gas Interpretation

Due to the fact that this project will use smart contracts to store information from each robot, changes have been made to the underling foundation of the blockchain, in this case, the gas (see section 3.2.1). The change produced here simply removed the concept of gas, allowing unlimited size of smart contracts, their storage, unlimited size of blocks, etc. Although this may be helpful for RobotChain, this change may produce unknown side effects.

This modification was achieved by modification of the costs of the several types contemplated by this gas interface, such as the value of an integer, or the value of the size of a string. Since the gas is a protocol addition, it is only present on the protocol that is activated on the blockchain. The file `gas_limit_repr.ml` of the protocol library contains the methods that calculate the cost. The most basic methods, methods that are called upon but do not call other methods that calculate these values are: *alloc_cost*, *step_cost*, *read_bytes_cost* and *write_bytes_cost*. By changing what these methods return to zero, such as `write_bytes_cost` or `step_cost` methods, the resulting calculations all return zero, effectively allowing `unlimited gas`.

```
->tezos-client transfer 1 from bootstrap1 to bootstrap2 with "Transaction Description"
```

Figure 4.3: Example of the process of doing a transaction, transferring 1 tz from bootstrap1 to bootstrap2, with the description *"Transaction Description"*. The symbol -> is the terminal command prompt.

```
Node is bootstrapped, ready for injecting operations.
Estimated gas: none
Estimated storage: no bytes added
Operation successfully injected in the node.
Operation hash is 'oof91nrSBf8DgnqHb2rmRNSMnNa9bYdg7P8Pzmp77U9G3Xq6s5P'
Waiting for the operation to be included...
Operation found in block: BL6xWBYmtL3ZN85JmCFM4BL4CQth27cCrn8DjnRTGxXh5Vscaen (pass: 3, offset: 0)
This sequence of operations was run:
  Manager signed operations:
    From: tz1KqTpEZ7Yob7QbPE4Hy4Wo8fHG8LhKxZSx
    Fee to the baker: ꜩ0.000277
    Expected counter: 1
    Gas limit: 0
    Storage limit: 0 bytes
    Balance updates:
      tz1KqTpEZ7Yob7QbPE4Hy4Wo8fHG8LhKxZSx ........... -ꜩ0.000277
      fees(tz1gjaF81ZRRvdzjobyfVNsAeSC6PScjfQwN,0) ... +ꜩ0.000277
    Transaction:
      Amount: ꜩ1
      From: tz1KqTpEZ7Yob7QbPE4Hy4Wo8fHG8LhKxZSx
      To: tz1gjaF81ZRRvdzjobyfVNsAeSC6PScjfQwN
           Desc: Transaction Description
      This transaction was successfully applied
      Consumed gas: 0
      Transaction Description: Transaction Description
      Balance updates:
        tz1KqTpEZ7Yob7QbPE4Hy4Wo8fHG8LhKxZSx ... -ꜩ1
        tz1gjaF81ZRRvdzjobyfVNsAeSC6PScjfQwN ... +ꜩ1

The operation has only been included 0 blocks ago.
We recommend to wait more.
Use command
  tezos-client wait for oof91nrSBf8DgnqHb2rmRNSMnNa9bYdg7P8Pzmp77U9G3Xq6s5P to be included --confirmations 30 --branch BKjTKwC7hWX7zFMMRiUXXMBM8CitwnhxoLfNnxyaAMQGz3T5E8j
and/or an external block explorer.
```

Figure 4.4: Screenshot of the receipt generated after the transaction.

4.4 Allow Multiple Operations per User in the Same Block

Another limitation found was that the blocks only accepted one entry per user per block. This meant that only one entry per robot was allowed in a block. This situation did not help achieving a high data throughput to the blockchain, since a robot was limited to one smart contract call or transaction per block.

In order to increase operation speed, this limit was removed by using the error associated with this situation, a contract counter checker, that checks if the source account of the transaction has a correct value, if its not set in either the past or the future. By not doing this verification, the blockchain allows multiple transactions between the same source and destination with the same token value. This may lead to a double spending situation, but said situation may, in the case of RobotChain, be disregarded since the token has no actual value or use. This contract verification was located in `contract_storage.ml` file in the protocol source code, meaning this is a modification that only affects the RobotChain protocol.

4.5 Time-Segmentation

Unlike the previous modifications, the time-segmentation is a modification of the architecture of the network, meaning that these changes work with any protocol that Tezos network run. With this in consideration, changes to the inherent blockchain network were made by differentiating nodes by roles assigned to them, as presented in Fig. 4.1.

4.5.1 Network Structure

Fig. 4.1 presents the RobotChain network structure in a schematic way. Each robot is connected to a computation module, and this connection is bidirectional in order to receive information from the robot to feed the blockchain and allow the blockchain, via smart contracts, to change the robot behaviour. The use of computational modules serves to ensure a uniform input into

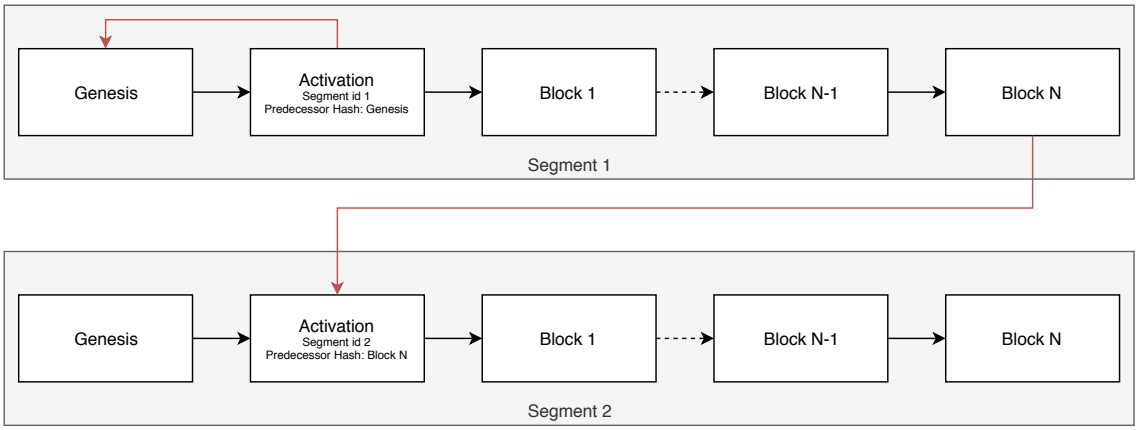

Figure 4.5: Visual representation of the second approach of the time-segmentation process. The red arrows represent the use of hashes to allow inter-segment connectivity.

the blockchain, as different robots may need different connection interfaces. It also ensures that the robots are not negatively affected with additional software running, that could cause degraded performance or other unforeseen consequences. Query nodes are nodes connected to the blockchain that allow human operators to query the blockchain for information. These are important to understand possible production line bottlenecks, or improve management understanding of the factory without directly interfacing with the robotic units. Oracle nodes are external entities that interact with the blockchain by the means of smart contracts. Cold storage nodes are nodes that save the entirety of the blockchain.

Due to the fact that our proposal uses compute devices instead of running its code directly on the robot in order to prevent robot performance loss, the embedded compute devices are limited regarding data storage, as such, it is unfeasible to maintain a copy of the entirety of the blockchain on each device in the long term. As such, other nodes can be added to the network with the sole propose of cold storage.

4.5.2 The Idea

Due to the fact that embedded devices have storage limitations, concerns arose regarding longevity of the blockchains with these compute devices. As such, a mean of dividing a blockchain while maintaining its immutability was needed. With Tezos blockchain working in a timely manner, where blocks are created at a set time interval, an idea was devised as time-segmentation, where the division of the blockchain in segments, or smaller parts is done at specific time intervals.

Although the proposed time-segmentation is applied to a Tezos blockchain network running RobotChain, the modifications made are protocol agnostic, meaning that it can run under any protocol that Tezos supports, with slight modifications to the genesis block regarding protocol activation. This solution is presented to solve the storage limitation of the compute devices and the fact that the older blocks are not entirely relevant to the continuous processing of the robot in a factory floor or the current state of the blockchain, and by having this time delimited blockchain, a definite limit of size can be achieved and as such, this feature allows lower storage devices to operate in the blockchain, with the possible addition of running the blockchain inside a RAMDisk.

RAMDisk is a block of the Random Access Memory(RAM), that a program like the Linux Kernel, is treating as a storage device equal to a regular hard drive. This RAMDisk has speed advantage

over other storage devices as shown in section 5.4.1.

4.5.3 Implementation

The time-segmentation initially considered two approaches. In the first approach, on the N-th block, the blockchain creates a copy of its current state, creating a backup of the segment, then it removes information unneeded of the existent blocks, removing the information regarding transactions from the previous segments, and inserts the new block onto the blockchain, starting the new segment. Cold storage nodes would keep all the various segments and the compute devices would only keep the latest segment.
Node synchronization problems appeared with this approach and the fact that we could not assert that the occupied storage had a maximum value, deemed it unreliable and development was focused on the second approach.
This first approach is similar to the Full/Rolling mode of the Tezos History update, with the difference that this approach failed due to deleting needed protocol information, where the Tezos History update simply solves that problem by re-saving the protocol on each checkpoint. This approach was not pursued further due to the successful initial tests of the second approach.
The second approach was devised and works as follows: on the network, three types of nodes are now introduced, the genesis node, the cold storage nodes and the compute device nodes, where the genesis is meant to serve as a bootstrap point for other nodes, protocol activation on its first block generated, allowing a single node with dictator key, and smart contract initialisation on a second block generated for mass smart contract initialization by initializing all at the same time. The cold storage type is meant to store all the segments, to retrieve older segments as needed and aid the bootstrap process, and the compute devices nodes are the uniform interface between the blockchain and the robots, converting robot output into uniform input for the blockchain.
The main idea is to create linked sub-blockchains, referred as segment or segments, that allow compute devices with lower storage capability the ability to keep only the latest segment instead of the entire blockchain, while maintaining the non-modification of the chain itself. The non-modification of the chain is ensured with the first block of the new segment, a modified protocol activation that takes into account the current segment identifier which is an integer and previous segment hash (the hash of the last block of the previous segment).
The first segment, segment 1, starts the network as a regular blockchain, with the activation block receiving as parameters the segment id 1 and the original genesis block as the predecessor hash. Then, on its N-th block, the blockchain does a sequence of steps in order to terminate the current segment and create the new one. Firstly it increments the segment ID on the node's configuration file, then shuts down the validation and database parts of the blockchain, leaving the peer-to-peer interface online. By shutting down both validation and database parts, data is flushed onto the storage devices. By leaving the peer-to-peer interface online, there is no need to re-initialise the peer-to-peer interface or to rediscover peers.
The state and validation are then reactivated with the updated configuration file, creating a new segment from scratch. The genesis node then activates the protocol, receiving as parameter the current segment ID and the predecessor segment hash, the hash from the last block of the previous segment, with the other nodes receiving this activation and resuming normal operations.
Considering the first block the activation block referred, the second block is then used to initialise smart contracts present on the previous segment and other features needed. Fig. 4.5

presents this process in a visual way.
Taking into consideration a new compute device node that joins the blockchain, it will only synchronize the latest segment, the one running on the network currently. In the case that the segmentation process happens before the bootstrap finishes, the node receives a reboot signal sent by the genesis node that instructs the node to create the new segment as described previously.
In the case that the new node is a cold storage node, the node will synchronize previous segments and keep the previous segments saved instead of deleting them.
Since the initial results of the time-segmentation version, presented in 5.5, did not correlate with the initial results from the protocol 4 with the History Mode additions, time-segmentation was implemented on the protocol 4, leading to the results in section 5.6.
Both approaches are similar to the new Tezos History Mode, without using the checkpoint system, where the regular nodes would work as the rolling mode presented in section 3.7.2, and the cold storage nodes would work as the full mode.
Several differences are still existent such as node initialisation, where the node has to synchronize from the genesis block up to the current information, or use the snapshot feature introduced. In the case of factory fast pace environment, creating snapshots and waiting for the new elements to catch-up in order to start operations may not be feasible.
The final segmentation approach (the second one), has the advantage of only needing to synchronize the latest segment, without the need to request older segment information. Only the genesis node requires previous segment hash in order to activate the protocol in a way that maintains the unmodification advantage of a blockchain solution. This advantage counterpoints the need of synchronizing the entirety of the blockchain data from a snapshot generated from a full/archive node.
Results regarding this feature are presented in two sections 5.5 corresponding to the two versions with protocol 1 and 5.6 corresponds to the version with protocol 4.

4.6 Conclusions

This chapter presents RobotChain, the modifications made to the Tezos blockchain that allow the registering of robotic events with the requirement of the high throughput without constraints to the data size generated. It also presents a time-segmentation technique that allows cheaper compute devices to operate on the blockchain. The modifications are expected to run in a contained environment, that, while needing a level of security, does not need the same level of security as the regular world wide public blockchain network, allowing compromises of security, such as allowing double spending, or other economical attacks that have no effect, in order to allow a higher data throughput. Economical attacks have no effect on this consortium blockchain due to the fact that the token is not meant to have economical value, having the possibility to be removed on further improvements of the network. These modifications were tested regarding their performance and the results are presented in the next chapter.

Chapter 5

Experiments

5.1 Introduction

This chapter presents the various experiments done to the Tezos blockchain and the operating system running the Tezos blockchain. The first experiment, the ”Sandbox Experiment” is named due to the fact that it was ran using the sandbox environment. This environment was used as an entry point to the understanding of the Tezos Blockchain. The second experiment, ”CPU Performance Experiments”, regards a study focused on the CPU performance of the Tezos Blockchain. After the results of this experiment, the third set of experiments, ”I/O Performance Experiment” presents a study of the Input/Output (I/O) performance of the various available storage devices, the operating system using them, the network interface, and the Tezos Blockchain operating on the storage devices.
Lastly, results regarding the last implemented feature, the Time-Segmentation, are presented for both protocols tested, protocol 1 and protocol 4. Both protocols were tested due to the fact that the feature was initially developed for protocol 1 and then updated to protocol 4.

5.2 Sandbox Experiment

5.2.1 Experimental Setup

For usage of the Tezos blockchain, experiments were made in order to understand the possible changes that are needed in order to use this system in the desired context.
Since RobotChain uses a private blockchain, contained inside a factory, unable to be accessed by outside means, the preliminary experiments were made using a Virtual Machine, using the provided sandbox configurations. The sandbox configurations were used since this was the first contact with running the Tezos blockchain. This virtual machine consists of 4 threads of an i7 CPU 960 @ 3.20GHz, leaving the other 4 threads for the host system, 8 GB of RAM, 20 GB of hard drive and it is virtualized a with KVM Hypervisor.
For performance measurement of this sandbox mode, several metrics were chosen, such as: time per transaction, number of crashed nodes and blocked transactions. These metrics were chosen due to the importance of transaction speed, since the blockchain needs to handle the throughput of the robots, either the time it takes for each transaction and if the transaction actually occurs. Stability of the network is also considered important.

5.2.2 Trial Description

We conducted experiments with different ”blocks per cycle” parameter, where the default value is 8, and the alternative tested value was 4, with a different number of nodes, (5, 10, 50 and 100) and each experiment consisted of five trials. For the 50 and 100 nodes, the experiments are considered stress tests to the network rather than stability or functionality tests, due to the

fact that we are considering 50 or 100 programs running at the same time, under only 4 CPU threads, a setting which is not meant to be used in a real world scenario.
Each trial was ran in the following manner. N node instances are initialized, with connections from each node to all the other nodes. These instances are created with the provided sandbox scripts, with the modification of the original 9 node limit to a maximum of 100 nodes.
The node initialization process is described on Fig. 3.1, where the bootstrap is the act of loading the information present locally and synchronize with the other nodes. The RPC Server present in the node is the interface used by the client processes to act in the blockchain.
Following the node initialization, the sandbox provided protocol parameters are used on the network, five bootstrap accounts are loaded, each with credit of 4000000 *tez*, where bootstrap one and two are used for transfers, and bootstrap three, four and five are used as bakers. Bootstrap 5 is also nominated as a Delegate.
Rather than using the provided sandbox scripts for the baker and endorser nodes, two different choices for running the nodes were tested. They were either as standalone clients, where they bake or endorse *ad eternum*, or a request is made, via the regular client, to bake or endorse a block. The latter approach was selected in order to have a finer control of the resources used by the virtual machine.
As such, a random node is selected and accounts three to five, in parallel, bake and endorse a block. Also, in parallel to this, transfer requests are made, on random nodes and of random amount of Tez, from account one to account two and account two to account one. The time that each transaction took is saved for statistical purposes. A timeout of 15 seconds for each transaction was defined. If the 15 seconds are elapsed without the transaction being successful, it is considered a failure, which is also accounted for in the statistics.

5.2.3 Results and Discussion

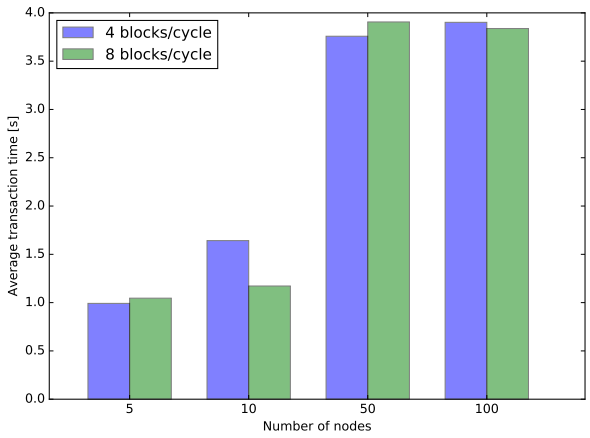

Figure 5.1: Average transaction time as a function of the number of nodes.

Table 5.1 present the transactions statistics retrieved from examining the execution logs, and Figure 5.1 contains the average for transaction time as a function of the number of nodes, for both 4 blocks and 8 blocks per cycle.
The total transaction time is measured in seconds, without contemplating transactions that timed-out, and the average transaction time is the average number of seconds that a successful transaction took.

Table 5.1: Transaction statistics for the trials in the experiments with four blocks per cycle and eight blocks per cycle. Using a virtual machine with four threads allocated. Each experiment has represented the Total transaction time, Average transaction time and Timed-out transactions.

Number of Nodes	Blocks per cycle 4			Blocks per cycle 8		
	Tot. trans. time	Avg. trans. time	T/O trans.	Tot. trans. Time	Avg. trans. time	T/O trans.
5	1003.69	1.004±0.226	0	994.3	0.994±0.168	0
	985.77	0.986±0.195	0	990.5	0.990±0.177	0
	988.20	0.988±0.165	0	1141.8	1.142±0.596	0
	986.77	0.987±0.190	0	1120.8	1.121±0.636	0
	987.81	0.988±0.163	0	989.6	0.990±0.182	0
10	934.68	1.138±0.574	179	995.0	1.015±0.228	20
	1871.82	1.881±1.769	5	1127.8	1.129±0.897	1
	1653.63	1.655±1.445	1	1357.1	1.369±0.978	9
	1851.15	1.860±1.738	5	1222.6	1.225±0.662	2
	1635.62	1.672±1.538	22	638.7	1.124±0.411	432
50	2480.74	3.205±2.687	226	3322.6	3.918±3.074	152
	3408.44	3.904±3.087	127	3244.6	3.849±3.122	157
	3442.78	4.089±3.173	158	3203.0	3.901±3.281	179
	3027.17	3.516±2.855	139	2941.2	3.737±2.946	213
	3189.37	3.613±3.028	216	3306.1	4.122±3.497	198
100	1715.61	5.922±18.850	652	1713.8	3.328±2.710	485
	1286.90	5.148±3.653	750	594.4	2.727±2.278	782
	672.81	4.128±3.094	837	1666.6	3.894±3.240	572
	965.73	4.534±3.396	77	3020.1	4.786±3.520	369
	1083.11	4.247±2.920	745	2230.0	4.714±3.561	499

The provided sandbox scripts only allow connections inside the same host. This setup has the advantage of not having a network delay.

For both 4 and 8 blocks per cycle, when using only 5 nodes, there are no timed out transactions. These start to appear for experiments with 10 or more nodes. Note that the sandbox original setup parameters were limited to 9 nodes, and as such, we are running these experiments outside of the original specification.

From the presented results we conclude that the change in the blocks per cycle from 8 to 4 did not produce any noticeable performance change. The justification for this is that the actions performed at the end of each cycle, namely the baker credit pay-out and backer staking, do not alter the overall statistics if done every 4 or 8 blocks.

As the number of nodes increase, we see a larger number of timed out transactions. As an example, running 100 nodes, 3 bakers and 3 endorsers, results in the total of 106 processes, and these processes are being handled by only 4 threads, meaning, each thread is processing 26 processes each, which may justify the observed decreased success rate. Similar analysis can be carried out for the other cases.

According to Fig. 5.1, in the 5 node experiments we had a 1 second average transaction time which is still far from the desired transaction speed for our use case.

We did not observe any noticeable performance change with variation in the ”block per cycle” parameter, due to the fact that this parameter does not produce significant work in the nodes.

Table 5.2: Average transaction time with the 6 tested protocol parameters, with the parameter ”time per block” being 1,2,3,5,10,60 respectively, with the left referring the hard drive results and in the right the Tmpfs.

	Hard Drive		Tmpfs	
Time Between Blocks	Average Transaction Time	Total Transactions	Average Transaction Time	Total Transactions
1	8.338±168.790	3	1.046±2.387	4963
2	6.841±154.000	1	1.004±0.161	5000
3	8.160±167.454	5	0.933±2.400	4117
5	8.251±168.678	2	1.004±0.163	5000
10	9.720±183.339	2	1.004±0.164	5000
60	8.204±165.791	6	1.004±0.092	5000

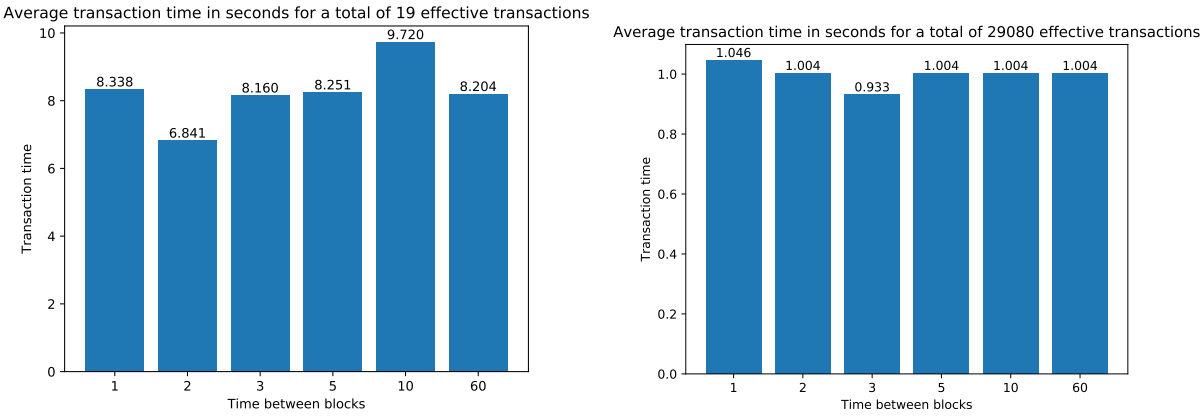

Figure 5.2: Average transaction time with the 6 tested protocol parameters, with the parameter ”time per block” being 1,2,3,5,10,60 respectively, with the left referring the hard drive results and in the right the Tmpfs. Note the different ”Transaction time” scales.

5.3 CPU Performance Experiments

Further experiments were made with the first addition to the network, the new *transaction_description*” parameter, as presented in section 4.2. These results revolved around the *time_between_blocks* parameter in the *protocol_parameters.json* provided to activate the RobotChain protocol. The variation chosen to this value was, ”1, 2, 3, 5, 10, 60”, providing 6 different ”protocols”, numbered from 1 to 6, with 6 being the original value, and each experiment was ran 5 times. Two storage devices were tested, the tmpfs and the hard drive disk.

This experiment was ran in a similar manner to the previous one, where, 11 random nodes are executed, with one being a genesis node where protocol activation happens, and the other 10 nodes connect to every other node, including the genesis node. A random node is selected for each account, from one to five, that in parallel, bake and endorse a block. Also, in parallel to this, transfer requests are made, on random nodes and of random amount of Tez, from a random account to other random account. A timeout of 3600 seconds for each transaction was defined. If the 3600 seconds are elapsed without the transaction happening, it is considered a failure and the process is terminated, preventing a deadlock situation.

Table 5.2 presents the results for this experiment, for both storage devices tested, the hard drive and the Tmpfs, with Fig.5.2 presenting an easier to read view of the tabulated results.

As seen in the left graph of Fig.5.2, the application while running in the hard drive could only inject 19 transactions in total for the 6 variations of the ”time_between_blocks” parameter, in

a total of 30000 transactions attempted. The total number comes from the 6 tested protocols times the 5 executions times 1000 transactions per execution for a total of 30000 transactions. It may appear strange the total transactions for the experiment is 19, while having an average transaction time well over 3600 seconds. What actually happens is a situation where, after one or two failed transactions, which processes were forcedly terminated (with Linux command kill -9 <process id>), the Tezos Client would present a error, like the one presented on listing A.2 and immediately terminate. This error is attributed to the storage medium used, that could not withstand the pressure from the multiple programs attempting to write information, meaning, 5 bakers, 5 endorsers, 11 nodes and 5 clients for a total of 26 processes that would struggle to write their information on the hard drive. This situation prompted to repeat the test in other storage medium, in this case, the Tmpfs file system. The solid state drive present was not used for this experiment due to be the drive running the operating system.
The Tmpfs did perform in an expected manner, reaching a number close to the 30000 transactions. Unlike the experiment ran in the hard drive, the experiment running in the Tmpfs did not enter in an ”error loop” and with the faster storage medium, a higher ammount of transactions were executed.
Tables 5.3 and 5.4 present the various priority values for the produced blocks, where this priority value is the baker’s baking right rank. This value can be also understood as the number of bakers that did not bake the block inside the baking right time-frame, where priority 0 would correspond to a block that was baked by the highest priority baker, priority 1 is baked by the second highest priority baker and so on. The results presented in these tables confirm the average transaction time, with the blocks created in the Tmpfs having a lower priority value, without considering the outlier of the experiment with Time Between blocks = 60, where the majority of the blocks were baked with priority 0 and a lower range of priority values, with the higher a range from 0 to 2, comparing with the HDD, where the priority values were distributed in a larger range from 0 to 47 priority value. It is important to repeat that, a high priority value, such as 40, as an example, means that, the block had to pass by 40 bakers in order to be created by the 41th. The worst case, 47, means that at the very least, 46 times 60, 2820 seconds were lost to create that block, or rather, that no baker could create the block in their own time frame before that. This alludes to a hardware difficulty. These results, as mentioned, were considered rather strange and further investigation of the input/output process was done, as presented in the next section.

5.4 I/O Performance Experiments

With the results presented on the previous section and the finding of the issue present on `https://tinyurl.com/yxldzxqr`[1] , where a Tezos developer acknowledges an issue that results in the ”heavy write pressure on the harddrive during the synchronisation of the chain.”[sic], a difference on the performance between storage devices was found, and further tests were focused on this subject. Before testing Tezos input/output performance, tests regarding the storage performance of the available interfaces was needed, in order to know the normal function of the storage devices without and with Tezos.
This experiment is split into three sub-experiments, the first one testing the various storage devices, the second one testing the network interfaces and the third one testing the Input/Output

[1] https://gitlab.com/tezos/tezos/issues/380?fbclid=IwAR3BGcW-MiCtG4W5d41XpfFgUoQj68SvJylfSC-vk-5TLNsO15pjQEgML7k

Table 5.3: Priority block value for each of the variations of the parameter ”Time between blocks”, in the experiment on Hard Drive storage device. The priority of a block corresponds to the baker’s baking right rank in the priority list. The Quantity is the Quantity of blocks of a given priority, the Sum is the sum of blocks of the given priority and previous priorities, and the percent is the percentage of blocks of the given priority.

Time between blocks = 1				Time between blocks = 2				Time between blocks = 3			
Priority	Quantity	Sum	Percent	Priority	Quantity	Sum	Percent	Priority	Quantity	Sum	Percent
0	2128	2128	37.68	0	3549	3549	57.99	0	5187	5187	67.86
1	1596	3724	28.26	1	1330	4879	21.73	1	1489	6676	19.48
2	628	4352	11.12	2	605	5484	9.89	2	541	7217	7.08
3	440	4792	7.79	3	296	5780	4.84	3	226	7443	2.96
4	313	5105	5.54	4	144	5924	2.35	4	76	7519	0.99
5	169	5274	2.99	5	61	5985	1.00	5	57	7576	0.75
6	117	5391	2.07	6	53	6038	0.87	6	19	7595	0.25
7	94	5485	1.66	7	30	6068	0.49	7	15	7610	0.20
8	41	5526	0.73	8	17	6085	0.28	8	10	7620	0.13
9-36	121	5647	2.14	9-20	35	6120	0.57	9-23	24	7644	0.31
Time between blocks = 5				Time between blocks = 10				Time between blocks = 60			
Priority	Quantity	Sum	Percent	Priority	Quantity	Sum	Percent	Priority	Quantity	Sum	Percent
0	4928	4928	80.16	0	8560	8560	94.70	0	1490	1490	21.16
1	907	5835	14.75	1	372	8932	4.12	1	1130	2620	16.05
2	170	6005	2.77	2	58	8990	0.64	2	835	3455	11.86
3	58	6063	0.94	3	25	9015	0.28	3	700	4155	9.94
4	33	6096	0.54	4	7	9022	0.08	4	592	4747	8.41
5	20	6116	0.33	5	8	9030	0.09	5	467	5214	6.63
6	14	6130	0.23	6	1	9031	0.01	6	320	5534	4.54
7	4	6134	0.07	7	5	9036	0.06	7	309	5843	4.39
8	4	6138	0.07	8	1	9037	0.01	8	253	6096	3.59
9-20	10	6148	0.16	9	2	9039	0.02	9-47	945	7041	13.42

of Tezos.

5.4.1 Storage speed evaluation

This experiment explores the storage speed of the various types of storage available in order to understand the different speed of the different filesystems and interfaces, without concerning the Tezos Blockchain itself. A smaller experiment like this allows to understand the limitations of the system where Tezos will operate.

Using the tool DD, a file with size 134217728 bytes, is written with differing block sizes (512b, 1K, 2K, 4K, 8K, 16K, 32K, 64K, 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M, 64M) in order to understand how each storage device behaves in writing speed. It tests the RAM-based file systems, RAM File System (ramfs), Tmpfs (tmpfs) and regular storage devices Solid State Drive (SSD) and Hard Drive Disk (HDD). Both SSD and HDD are using the ext4 filesystem. Ramfs and Tmpfs are similar, both work with RAM as main storage area, with the difference that ramfs

Table 5.4: Priority block value for each of the variations of the parameter "Time between blocks", in the experiment on Tmpfs storage device. The priority of a block corresponds to the baker's baking right rank in the priority list. The Quantity is the Quantity of blocks of a given priority, the Sum is the sum of blocks of the given priority and previous priorities, and the percent is the percentage of blocks of the given priority.

Time between blocks = 1				Time between blocks = 2				Time between blocks = 3			
Priority	Quantity	Sum	Percent	Priority	Quantity	Sum	Percent	Priority	Quantity	Sum	Percent
0	2930	2930	55.66	0	5048	5048	99.92	0	4694	4694	99.91
1	2332	5262	44.30	1	2	5050	0.04	1	4	4698	0.09
2	0	5262	0.00	2	2	5052	0.04	2	0	4698	0.00
3	0	5262	0.00	3	0	5052	0.00	3	0	4698	0.00
4	0	5262	0.00	4	0	5052	0.00	4	0	4698	0.00
5	2	5264	0.04	5	0	5052	0.00	5	0	4698	0.00
6	0	5264	0.00	6	0	5052	0.00	6	0	4698	0.00
7	0	5264	0.00	7	0	5052	0.00	7	0	4698	0.00
8	0	5264	0.00	8	0	5052	0.00	8	0	4698	0.00
9	0	5264	0.00	9	0	5052	0.00	9	0	4698	0.00
Time between blocks = 5				Time between blocks = 10				Time between blocks = 60			
Priority	Quantity	Sum	Percent	Priority	Quantity	Sum	Percent	Priority	Quantity	Sum	Percent
0	5050	5050	99.96	0	5051	5051	100.00	0	1021	1021	20.22
1	2	5052	0.04	1	0	5051	0.00	1	785	1806	15.54
2	0	5052	0.00	2	0	5051	0.00	2	588	2394	11.64
3	0	5052	0.00	3	0	5051	0.00	3	537	2931	10.63
4	0	5052	0.00	4	0	5051	0.00	4	427	3358	8.46
5	0	5052	0.00	5	0	5051	0.00	5	323	3681	6.40
6	0	5052	0.00	6	0	5051	0.00	6	251	3932	4.97
7	0	5052	0.00	7	0	5051	0.00	7	214	4146	4.24
8	0	5052	0.00	8	0	5051	0.00	8	200	4346	3.96
9	0	5052	0.00	9	0	5051	0.00	9-38	704	5050	13.94

does not use swap memory and has no storage limitations, meaning that it may occupy all the RAM the system has available, regardless if any limit was defined, and will not swap allocated memory, where Tmpfs will use swap memory and respects defined storage limits. From figure 5.3, it is possible to see that the HDD is rather slow in comparison to the SSD or to the virtual storage devices. This test only references the speed factor, not other advantages that HDD's may have in comparison to other storage devices. Both virtual storage devices have at least 3 times the speed of the SSD device, as was expected. These virtual storage devices have the speed advantage over the regular storage devices, but, they have the main disadvantage of, in a case of power loss, they will lose the information contained, and as such, they need to backup their own information to a regular storage device.

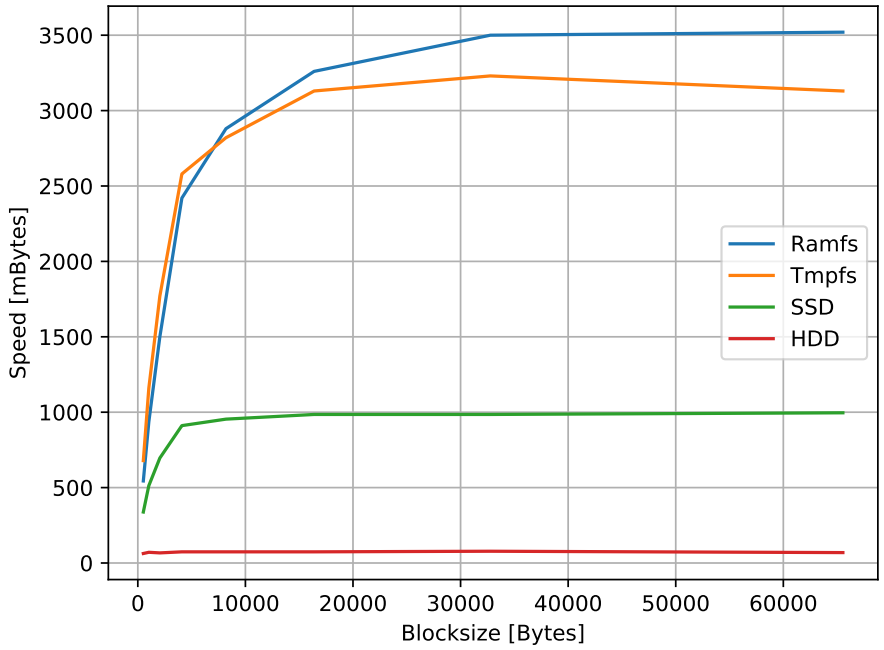

Figure 5.3: Storage speed on various storage devices. Ramfs corresponds to RAM FileSystem, Tmpfs corresponds to Tmpfs, a Ramfs derivate that uses swap memory. SSD correspons to a Samsung SSD 970 EVO 500GB, and the HDD is a Seagate Barracuda ST3000DM007-1WY10G, where both SSD and HDD using ext4 filesystem.

Table 5.5: Data transfer between two computers, using iperf3, with left table using PC A as server and PC B as client, and the right table using PC B as server and PC A as client, running for a total of 10 seconds.

	PC A to PC B	
Experiment	Transfer [mBytes]	Bitrate [MBits\s]
1	112.0±0.000	941.3±0.675
2	112.0±0.000	941.3±1.252
3	111.9±0.316	940.9±2.132
4	111.9±0.316	940.3±4.111
5	112.0±0.000	941.7±0.949

	PC B to PC A	
Experiment	Transfer [mBytes]	Bitrate [MBits\s]
1	112.0±0.000	941.4±0.516
2	112.0±0.000	941.6±0.516
3	112.0±0.000	941.5±0.527
4	112.0±0.000	941.5±0.527
5	112.0±0.000	941.5±0.527

5.4.2 Network Speed Evaluation

This experiment explores the networking segment of the systems, without the usage of Tezos Blockchain. This and the previous experiment allow to set a baseline between the system running and the Tezos Blockchain own performance.

This experiment setup was done using two computers on their own private network containing only a switch device, firstly with one computer set as a server, and the other as a client, and the same exact experiment with the roles switched. Using the program iperf3, the results of the Table 5.5 show that the used network interfaces are almost reaching 1 Gigabit Bitrate, where bitrate is the rate at bits are transferred between two computers at a given time. The tested setup has a approximately 94% of the gigabit speed, since a gigabit bitrate is 1000 megabits per second.

5.4.3 Tezos I/O Evaluation

Considering the results of experiments 5.4.1 & 5.4.2, an operating system baseline was created and it is possible to understand how Tezos blockchain is using the resources provided by the operating system.
Since the work on this project was constantly evolving, this experiment will contain the changes stated in 4.2 & 4.3.
This experiment was ran on a single computer running *iotop* for monitoring input/output of a single running node, from its genesis, activation and the injection of 5000 transactions, with no block limit. The time between blocks parameter was set as 1. It was ran 5 times for both HDD and SSD storage devices. Tmpfs was not tested in order to prevent RAM shortage, with consideration to the added monitoring tools.
A single node was used to test a close to life-like situation, where a computer runs a node, a baker and a client. The network was not tested in this experiment in order to remove a possible I/O pressure from the network interface.

Table 5.6: IOtop results and transaction statistics for the 5 executions of the experiment in the HDD storage device, where TpB corresponds to Transactions per Block.

Total Seconds	Avg. Kbytes\s	Total Kbytes Written	Total Blocks	Avg. TpB	Total Transactions	Total Blocks with Transactions
381	211.626±239.163	77476	78	64.766±3.304	4987	77
309	262.020±279.143	77248	78	64.935±4.615	5000	77
328	249.577±272.114	78384	79	64.935±3.690	5000	77
334	245.987±257.660	78816	79	64.935±4.435	5000	77
315	258.880±274.072	78248	78	64.935±4.937	5000	77

Table 5.7: IOtop results and transaction statistics for the 5 executions of the experiment in the SSD storage device, where TpB corresponds to Transactions per Block.

Total Seconds	Avg. Kbytes\s	Total Kbytes Written	Total Blocks	Avg. TpB	Total Transactions	Total Blocks with Transactions
154	658.098±487.614	94700	97	54.945±4.151	5000	91
154	662.685±575.371	95356	96	54.348±5.128	5000	92
155	663.500±568.258	96136	96	54.348±7.409	5000	92
155	659.694±468.207	95580	96	54.945±5.216	5000	91
154	659.028±566.674	95492	96	54.945±4.600	5000	91

The results presented on tables 5.6 & 5.7 both support the previous experiments regarding the various storage devices and performance, where the HDD device has a weaker performance in relation to the other storage devices, and as such, it is not feasible to run RobotChain with an HDD, due to the fact that the HDD does not support the high throughput of a robot injecting thousands of operations per second. As such, this situation lead to the time-segmentation technique presented in 4.5.1, in order to use the Tmpfs with minimal RAM intact.

5.5 Time-Segmentation with Protocol 1

This experiment is based on the time-segmentation feature described in 4.5.1, and includes every other feature or modification described in chapter 4, running the RobotChain protocol. It

considers the time-segmentation with two initial approaches as described in subsection 4.5.3. Tests were made to evaluate the storage capability of this solution, comparing the unsegmented

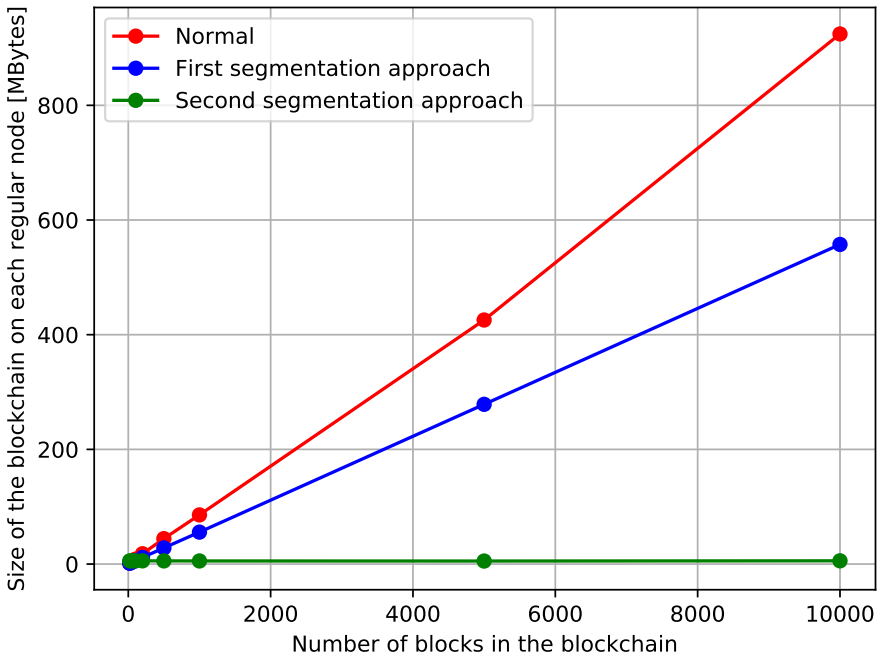

Figure 5.4: Storage size on each regular node (not cold storage), for three versions of the blockchain, as a function of the number of blocks. For the unsegmented blockchain (normal), every node contains the complete blockchain.

blockchain and the two approaches to build the segmented blockchain, where both approaches segment the blockchain at every 10 blocks. The tests were ran for 20, 50, 100, 200, 500, 1000, 5000, 10000 total blocks. Random transactions were injected into the network, with an orchestrator program running up to 32 transaction clients running at the same time. These clients inject a transaction between random accounts, with the value of 1 XTZ, with a transaction description composed by the current transaction counter of the orchestrator concatenated to a random alphanumeric string size 1000. The storage test results are provided in Fig. 5.4.
Although the first segmentation approach is considered an improvement over the regular non-segmented network, it has the problem of not having a limit value for the necessary normal node storage, unlike the second approach, which has a average size of 5386 Kilobytes per segment. With the previous experimental results, the ability of setting a hard limit on the storage size of the blockchain, it opens the way to use the discussed virtual storage systems, allowing a higher information throughput to the blockchain. It also aids with bootstrap due to the fact that a new node won't need to obtain every segment from the start, needing only the latest one to work.

5.6 Time-Segmentation with Protocol 4

5.6.1 Regular Node Storage

After testing the original protocol, and with the results presented in previous subsection, it was decided to test the new history mode. Initial testing was thought to be done by comparing the RobotChain protocol with the latest Tezos MainNet with the new features. With the initial results presented by the latest Tezos MainNet, this idea was scrapped due to the fact that the

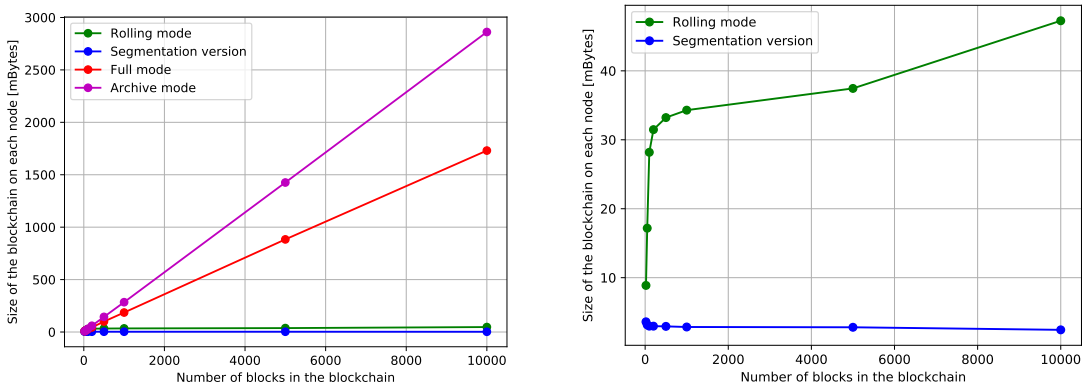

Figure 5.5: Storage size on each compute device node (not cold storage), for four versions of the blockchain, as a function of the number of blocks. The segmented blockchain running mode is archive. The left sub-figure presents the results for the four tests, while the right sub-figure presents only the results for the rolling mode and the segmentation version, for improved comparison.

rolling release storage size results showing that the full mode had significant higher storage compared to the RobotChain protocol.

As such, this protocol with the new history mode features was modified in the same manner as the RobotChain protocol, leading to a RobotChain Version 4.

Tests were made to evaluate the storage requirements of second approach mentioned in subsection 4.5.3, comparing the unsegmented blockchain, running the three versions (archive, full, rolling) and the approach configured in full mode, to build the segmented blockchain, creating a new segment every 10 blocks. The tests were ran for 20, 50, 100, 200, 500, 1000, 5000, 10000 total blocks. Random transactions were injected into the network, with an orchestrator program running up to 32 transaction clients running at the same time. These clients inject a transaction between random accounts, with the value of 1 XTZ, with a transaction description composed by the current transaction counter of the orchestrator concatenated to a random alphanumeric string size 1000. The storage test results are provided in figure 5.5.

The presented approach has an average size of 2282 Kilobytes per segment, having a definite hard limit for the maximum storage occupied by each segment, similar to the results presented on the previous experiment.

5.6.2 Cold Storage Node Storage

Additional tests were made to understand how the segmentation affected the storage capabilities of both the compute device nodes and the cold storage nodes, and how the network would grow with the various segments, considering the repeated addition of the genesis and activation blocks to each segment. The tests were ran for 20, 50, 100, 250 blocks per segment with a total of 1000 blocks per experiment. As with the previous experiment, random transactions were injected into the network, with an orchestrator program running up to 32 transaction clients running at the same time. These clients inject a transaction between random accounts, with the value of 1 XTZ, with a transaction description composed by the current transaction counter of the orchestrator concatenated to a random alphanumeric string size 1000. The storage results are presented in Fig. 5.6.

The segmentation approach has a smaller running storage footprint, with a cold storage space occupied similar to the archive node. Depending on the value for the number of segments per

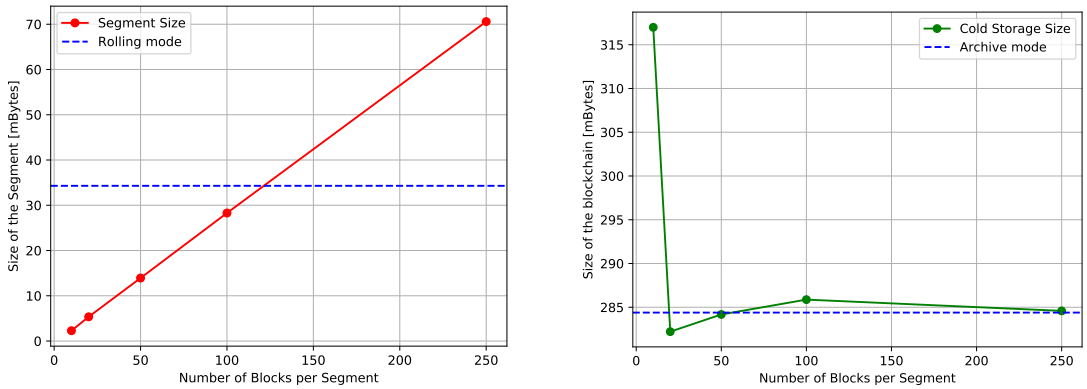

Figure 5.6: The left sub-figure presents the compute device storage size as a function of the number of blocks per segment for a total of 1000 blocks, compared with the corresponding rolling mode storage size of the Tezos blockchain. The right sub-figure presents the cold storage size as a number of blocks per segment for a total of 1000 blocks, compared with the archive mode of the Tezos blockchain. Results for the full mode are not presented since they are not comparable to any of the node types on our approach, regarding storage capabilities.

block, the resulting segment storage space occupied can be smaller than the rolling mode. This time-segmentation feature has the benefit of providing a definite hard limit for the segments, with a cold storage space occupied similar to a node running archive mode, taking into consideration the increased number of blocks with the added genesis and activation block. This would allow low cost nodes running the blockchain, with an increase of information throughput by allowing running inside a Tmpfs or other virtual filesystem. With the updated protocol, segmentation has the advantage of not needing to bootstrap the entire network and just needing the latest segment, with the added possibility of the availability of the snapshot import feature. Finally it is important to state that both the cold storage node and the compute device node are running in archive mode and that Tezos rolling mode does not guarantee a fixed maximum node memory size, and the memory requirements slowly grow as can be seen in Fig.5.5 (right). This invalidates the Tezos rolling mode as a solution to the limited capacity of the said nodes, since the rolling mode would eventually exhaust the available memory and the network would stop working.

5.7 Conclusions

This chapter presented the various experimental results for the main contributions of this dissertation. Several results have been presented and several infrastructure decisions regarding the RobotChain were made.

Concerning the improvements, (sections 4.2 to 4.4), they aid with the data insertion on a transaction, data throughput by allowing multiple transactions from one robot unto a single block and the removal of data storage limitations, latter important to smart contract execution since it allows a higher data storage.

Regarding the I/O situation, with the time-segmentation feature addition, Tmpfs was chosen to be used as the main runtime storage device sided with other storage device such as a hard drive or sdcard in the compute devices for storage, since tmpfs is a volatile memory. Tmpfs was also chosen versus ramfs due to the fact that the Tmpfs can be programmatically limited in its storage capacity size and has the capability to use swap memory, where ramfs does not allow to

define that storage capacity size and does not use swap memory, effectively, in extreme cases, ramfs would simply crash the system due to occupying the entirety of the system's memory. Considering the time-segmentation solution, with the mentioned Tmpfs storage device, with a segment size value of 100, it is considered sufficient for the execution of the blockchain without affecting the underlying operating system or other processes running. This is important in the compute device since it will need to run the RobotChain along with other software, namely an interface between the robot and the blockchain itself.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this dissertation a new blockchain tailored to registering robotic events in closed environments, such as factories, called RobotChain, is presented with the addition of a time-segmentation technique applicable to any blockchain. RobotChain is not just useful for guaranteeing an immutable event ledger, that can be used for deciding which robot made particular actions in conflicting situations, but also allows for performance monitoring and even robot tuning by taking advantage of smart contracts.

RobotChain as been used for controlling and monitoring robots: in [LA19] an application of the log registering with abnormal behavior detection is presented. It is followed by [LAP19], where smart-contracts store image analytics of detected supply information in order to adjust robot velocity to the input provided. An improvement of the previous work is presented in [LA], showing a smart-contract monitoring capabilities of a robot's workspace.

We further improved the initial proposal with the introduction of a time-segmentation mode to solve the problems related to the small storage capacity of compute modules. This feature allows the use of cheap compute modules for the majority of network nodes (all but the cold storage ones) and makes the processing and connection of new nodes faster both by allowing the use of faster memory for storing the segment (such as RAM) and also because only the current segment is needed for syncing the new node with the network. The new history feature from Tezos, although similar to the proposed method, has differences and requirements that deemed it not compatible with the performance that RobotChain needs to achieve. These differences include the time needed to include a new node in the RobotChain, the information needed for the robots to operate and the capability of defining a storage space hard limit that allows deployment of the compute modules for an unlimited duration.

6.2 Future work

Regarding future work, the main concern is the stability of the network and the increase of the data throughput that is needed for receiving and recording the data from various robots. The creation of a uniform compute device that is able to be mass produced could also be investigated. Several features related to the time-segmentation solution are also meant to be implemented such as RPC interfaces for block retrieval for previous segments, allowing query nodes to access information not contained on the current segment or cold storage full synchronization by allowing new cold storage nodes to obtain previous segments from other cold storage nodes. As an alternative, the snapshot feature will also be investigated for the purposes of cold storage node bootstrap. We also intend to study the possibility of defining a different number of blocks per segment on a on node-to-node basis, that could be useful for accommodating nodes with different capabilities.

Furthermore, the removal of unneeded features, such as the token itself, is under investigation as a way to simplify the network. This would need a new way to select a baker as the current baker selection way depends on the token.

Bibliography

[Agg18] Manuj Aggarwal. Blockchain in robotics — a sneak peek into the future [online]. March 2018. Available from: https://medium.com/@manuj.aggarwal/blockchain-in-robotics-a-sneak-peek-into-the-future-4e115ccf4931. 6

[ASMAE] Kristo Karjust Aleksei Snatkin, Jüri Majak, Tanel Aruväli, and Tanel Eiskop. Real time production monitoring system in sme. Available from: http://innomet.ttu.ee/daaam_publications/2012/snatkin.pdf. 4

[DRKA18] Konstantin Danilov, Ruslan Rezin, Alexander Kolotov, and Ilya Afanasyev. Towards blockchain-based robonomics: autonomous agents behavior validation. may 2018. Available from: http://arxiv.org/abs/1805.03241. 3, 7

[FA18] Miguel Fernandes and Luís A. Alexandre. Robotchain: Using Tezos Technology for Robot Event Management. 2018. Available from: https://www.ledgerjournal.org/ojs/index.php/ledger/article/view/175. 1

[FA19] Miguel Fernandes and Luís A. Alexandre. A time-segmented consortium blockchain for robotic event registration. *CoRR*, abs/1904.04306, 2019. Available from: http://arxiv.org/abs/1904.04306. 1

[Fer16] Eduardo Castelló Ferrer. The blockchain: a new framework for robotic swarm systems. aug 2016. Available from: http://arxiv.org/abs/1608.00695. 3

[Fou18] Tezos Foundation. Tezos launch on twitter [online]. September 2018. Available from: https://twitter.com/TezosFoundation/status/1041675135412043776. 10

[FRHP18] Eduardo Castelló Ferrer, Ognjen Rudovic, Thomas Hardjono, and Alex Pentland. RoboChain: A Secure Data-Sharing Framework for Human-Robot Interaction. feb 2018. Available from: http://arxiv.org/abs/1802.04480. 3

[Git] Tezos Gitlab. Protocol 3 description [online]. Available from: https://tezos.gitlab.io/mainnet/protocols/003_PsddFKi3.html. 10

[Goo08] L.M. Goodman. Tezos - a self-amending crypto-ledger. 2008. Available from: https://tezos.com/static/papers/white_paper.pdf. 1, 6

[HDRS18] M. G. M. Mehedi Hasan, A. Datta, M. Ashiqur Rahman, and H. Shahriar. Chained of things: A secure and dependable design of autonomous vehicle services. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 02, pages 498–503, July 2018. 4

[Her18] Alyssa Hertig. Bitcoin's next big software upgrade to feature new language for crypto keys [online]. September 2018. Available from: https://www.coindesk.com/bitcoins-next-big-software-upgrade-to-feature-new-language-for-crypto-keys/. 6

[Kam18] Kambria. Kambria - fueling the ai & robotics future [online]. 2018. Available from: https://kambria.io/. 5

[LA] V. Lopes and L. A. Alexandre. Robot Workspace Monitoring using a Blockchain-based 3D Vision Approach. *CVPR Workshop: Blockchain Meets Computer Vision & AI, Long Beach, CA, June 17, 2019*. 35

[LA19] V. Lopes and L. A. Alexandre. Detecting robotic anomalies using robotchain. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, April 2019. 35

[LAP19] Vasco Lopes, Luís A. Alexandre, and Nuno Pereira. Controlling Robots using Artificial Intelligence and a Consortium Blockchain. CoRR, abs/1903.00660. 2019. 35

[Lbk17] Qu Jing Lei, Li Shao bo, and Chen Jing kun. Online monitoring of manufacturing process based on autocep. *iJOE*, 13(6):22–34, 2017. Available from: `http://www.online-journals.org/index.php/i-joe/article/view/6812`. 4

[LKK+] Sergey Lonshakov, Aleksandr Krupenkin, Aleksandr Kapitonov, Evgeny Radchenko, and Alisher Khassanov andAleksandr Starostin. Robonomics: platform for integration of cyber physicalsystems into human economy. Available from: `https://robonomics.network/robonomics_white_paper_en.pdf`. 3, 6, 7

[LV18] Benjamin Leiding and William V. Vorobev. Enabling the vehicle economy using ablockchain-based value transaction layerprotocol for vehicular ad-hoc networks. Augst 2018. 6

[Nak08] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Www.Bitcoin.Org*, page 9, 2008. Available from: `https://bitcoin.org/bitcoin.pdf`. 5

[Nom19] Nomadic Labs. Introducing snapshots and history modes for the tezos node [online]. 2019. Available from: `https://blog.nomadic-labs.com/introducing-snapshots-and-history-modes-for-the-tezos-node.html`. 14

[Res19] Tezos Dev Resources. Proof-of-stake in tezos [online]. January 2019. Available from: `https://tezos.gitlab.io/tezos/whitedoc/proof_of_stake.html`. 12

[SCD] Volker Strobel, Eduardo Castello Ferrer, and Marco Dorigo. Managing Byzantine Robots via Blockchain Technology in a Swarm Robotics Collective Decision Making Scenario. Technical Report December 2017. 3

[SKY18] SKYF. Skyfchain [online]. 2018. SKYFchain is the first B2R (business-to-robots) blockchain based operating platform. Available from: `https://www.skyfchain.io/`. 4

Appendix A

Tezos Blockchain

A.1 Block

Listing A.1: Block Example


```
{
 "protocol": "PsYLVpVvgbLhAhoqAkMFUo6gudkJ9weNXhUYCiLDzcUpFpkk8Wt",
 "chain_id": "NetXdQprcVkpaWU",
 "hash": "BM2eVYPu3AJTxy8srqCnCGZG55wX2XsRn32PDLmZTwnQq5b7JUn",
 "header": {
  "level": 330,
  "proto": 1,
  "predecessor": "BM7LfoNRMeQ3LVqsBoq1JxicXhYkTw92Y9mxaAZMztnHayj7vp9",
  "timestamp": "2019-01-11T12:28:01Z",
  "validation_pass": 4,
  "operations_hash": "LLoabn6Y7qWTNZDQQxWReGVGB1qfoshZoSHT7eeMSfucSPwND8Bg8",
  "fitness": [
   "00",
   "000000000000014a"
  ],
  "context": "CoWSya6e5QC4kmCUnDPG3LrcDFKFzHKeo9ZoNq92pbUPXmhMiM4L",
  "priority": 0,
  "proof_of_work_nonce": "5a9359ee2f3d90ce",
  "signature": "sigptcJNiCCtk4bMjsQy77deVzWKvYuvwMHFnjY8sBindex4nAtenA1NEmRowozRUnDCgV5hMRdp6b1uyiZ6eYnJspbQzLLP"
 },
 "metadata": {
  "protocol": "PsYLVpVvgbLhAhoqAkMFUo6gudkJ9weNXhUYCiLDzcUpFpkk8Wt",
  "next_protocol": "PsYLVpVvgbLhAhoqAkMFUo6gudkJ9weNXhUYCiLDzcUpFpkk8Wt",
  "test_chain_status": {
   "status": "not_running"
  },
  "max_operations_ttl": 60,
  "max_operation_data_length": 16384,
  "max_block_header_length": 238,
  "max_operation_list_length": [
   {
    "max_size": 32768,
    "max_op": 32
   },
   {
    "max_size": 32768
   },
   {
    "max_size": 135168,
    "max_op": 132
   },
   {
    "max_size": 524288
   }
  ],
  "baker": "tz1faswCTDciRzE4oJ9jn2Vm2dvjeyA9fUzU",
  "level": {
   "level": 330,
   "level_position": 329,
   "cycle": 41,
   "cycle_position": 1,
   "voting_period": 0,
   "voting_period_position": 329,
   "expected_commitment": false
  },
  "voting_period_kind": "proposal",
  "nonce_hash": null,
  "consumed_gas": "100",
  "deactivated": [],
  "balance_updates": [
   {
    "kind": "contract",
    "contract": "tz1faswCTDciRzE4oJ9jn2Vm2dvjeyA9fUzU",
    "change": "-512000000"
   },
   {
    "kind": "freezer",
    "category": "deposits",
    "delegate": "tz1faswCTDciRzE4oJ9jn2Vm2dvjeyA9fUzU",
    "level": 41,
    "change": "512000000"
   },
```

```
      {
       "kind": "freezer",
       "category": "rewards",
       "delegate": "tz1faswCTDciRzE4oJ9jn2Vm2dvjeyA9fUzU",
       "level": 41,
       "change": "16000000"
      }
     ]
    },
    "operations": [
     [],
     [],
     [],
     [
      {
       "protocol": "PsYLVpVvgbLhAhoqAkMFUo6gudkJ9weNXhUYCiLDzcUpFpkk8Wt",
       "chain_id": "NetXdQprcVkpaWU",
       "hash": "onm875Gh5e4cACMx6NSPcufxrawCqW4UkmxfKVHkWDTdNEMN53Q",
       "branch": "BKj25VhsQRPwwypg37hExwoDK4tS4sgESvxWa17XM7YTWq7CoXv",
       "contents": [
        {
         "kind": "transaction",
         "source": "tz1KqTpEZ7Yob7QbPE4Hy4Wo8fHG8LhKxZSx",
         "fee": "50000",
         "counter": "51",
         "gas_limit": "200",
         "storage_limit": "0",
         "amount": "100000000",
         "destination": "tz1gjaF81ZRRvdzjobyfVNsAeSC6PScjfQwN",
         "metadata": {
          "balance_updates": [
           {
            "kind": "contract",
            "contract": "tz1KqTpEZ7Yob7QbPE4Hy4Wo8fHG8LhKxZSx",
            "change": "-50000"
           },
           {
            "kind": "freezer",
            "category": "fees",
            "delegate": "tz1faswCTDciRzE4oJ9jn2Vm2dvjeyA9fUzU",
            "level": 41,
            "change": "50000"
           }
          ],
          "operation_result": {
           "status": "applied",
           "balance_updates": [
            {
             "kind": "contract",
             "contract": "tz1KqTpEZ7Yob7QbPE4Hy4Wo8fHG8LhKxZSx",
             "change": "-100000000"
            },
            {
             "kind": "contract",
             "contract": "tz1gjaF81ZRRvdzjobyfVNsAeSC6PScjfQwN",
             "change": "100000000"
            }
           ],
           "consumed_gas": "100"
          }
         }
        }
       ],
       "signature": "sigjjiz6mvYn8P5z7afNae3oSExcEsHhg7rsEXCrMbJhuUsqf9ZuNZ8HQuepS7dddFSt4kCYSUjZED6yfTMEHAftsAoFY3Vx"
      }
     ]
    ]
   }
  }
```


A.2 Transactions

Listing A.2: Transaction Error

```
    Unregistred error:
  { "kind": "branch",
    "id": "proto.001-RobotChain.contract.counter_in_the_past",
    "contract": "tz1gjaF81ZRRvdzjobyfVNsAeSC6PScjfQwN", "expected": "2",
    "found": "1" }
Fatal error:
  transfer simulation failed
```

Glossary

Checkpoint	A point in the blockchain considered to be a anchor of the consensus in the real world at regular intervals.
tmpfs	Linux file system that is stored in RAM instead of a storage device.
Double Baking	The act of signing/baking two blocks at the height, a indicative of a double spending situation.
Double Spending	The act of using the same token for two or more different payments. Similar to counterfeit money.
RAMDisk	A block of RAM that emulates via a program, a disk drive like a Hard Drive Disk (HDD) or Solid State Drive (SSD). Used due to the fact that RAM is in general several times faster than other storage devices like HDD's, SSD's, tape drives, between others.