Capítulo II – Modelos de Programação Distribuída (parte 3)

From: Coulouris, Dollimore and Kindberg
Distributed Systems: Concepts and Design
Edition 3, © Addison-Wesley 2001

From: Cardoso, Jorge, "Programação de Sistemas Distribuídos em Java", FCA, 2008.

Paula Prata,
Departamento de Informática da UBI
http://www.di.ubi.pt/~pprata

1 – Modelos de comunicação por mensagens

2 – Exemplo: Comunicação por mensagens através de Sockets (em Java)

3 - Modelos Arquiteturais

4 - Modelos Fundamentais

Interação

Falhas

Segurança

Modelos fundamentais

Os sistemas distribuídos podem ainda ser analisados segundo 3 aspectos transversais a todos os sistemas:

. Modelo de Interação (ou de sincronismo)

. Modelo de Falhas (ou Avarias)

. Modelo de Segurança

a) Modelo de interação

Interação é a ação (comunicação e sincronização) entre as partes para realizar um qualquer trabalho.

É afetada por dois aspetos:

1 . Performance dos canais de comunicação;

2 . Inexistência de um tempo global;

1 . Performance dos canais de comunicação

1.a) Latência

Intervalo de tempo que medeia entre o início da transmissão de uma mensagem por um processo e o início da sua recepção pelo outro processo.

Depende de:

- Tempo requerido pelo sistema operativo em ambos os lados da comunicação
- Demora no acesso aos recursos da rede
- Demora (“delay”) de transmissão pela rede

1 . Performance dos canais de comunicação

1.b) Largura de banda (“bandwidth”)

Total de informação que pode ser transmitida pela rede num dado intervalo de tempo;

1.c) Jitter

Variação no tempo necessário para enviar grupos de mensagens consecutivos constituintes de uma informação transmitida de um ponto para outro na rede.

(importante na transmissão de som e imagem)

2. Inexistência de um tempo global

Cada computador tem um relógio "clock" interno.

=> Cada relógio tem um "**drift**" (um desvio) do tempo de referência

=> Os "drifts" de dois relógios distintos são também distintos

(o que significa que entre eles o tempo será sempre divergente)

- Uma solução passa por obter o tempo fornecido por GPS – Global Positioning System, e enviar aos participantes do sistema distribuído.

Problema: Delays no envio dessa mensagem!!!

4 - Modelos Fundamentais

Duas variantes no modelo de interacção:

1 – Sistemas distribuídos síncronos

Sistemas onde podem existir limites máximos de tempo conhecidos para:

- Tempos de execução dos processos,

- Atrasos na comunicação,

- Variações no tempo (*relógio*) de referência.

1 – Sistemas distribuídos síncronos (cont.)

Se:

- o tempo necessário para executar cada passo de um processo tem um limite inferior e um limite superior conhecidos

- cada mensagem transmitida por um canal é recebida dentro de um limite de tempo conhecido

- cada processo tem um relógio cujo desvio máximo para o tempo de referência é conhecido

Podem definir-se "timeouts" para detectar falhas.

Dificuldade em encontrar os limites para os tempos, mais difícil ainda, provar a sua correcção

2 – Sistemas distribuídos assíncronos

Não possui limites para:

- Tempo de execução dos processos – cada passo de execução pode levar um tempo arbitrariamente longo

- Tempo de transmissão de mensagens - uma mensagem pode chegar rapidamente ou demorar dias

- O desvio para o tempo de referência pode ser qualquer

2 – Sistemas distribuídos assíncronos (cont.)

Exemplo de um sistema assíncrono: Internet;

Como lidar com longos tempos de espera:

- O sistema pode avisar o utilizador que o tempo de espera pode ser longo e solicitar uma alternativa;
- O sistema pode dar oportunidade ao utilizador para fazer outras coisas;

...

O modelo de interacção e o problema da ordenação de eventos

Por vezes é importante conhecer a ordem pela qual ocorreu um conjunto de eventos.

Ex.lo

Sejam os utilizadores X,Y, Z e A que trocam mails para marcar uma reunião:

. X envia uma mensagem, com o assunto: Meeting, para Y, Z e A

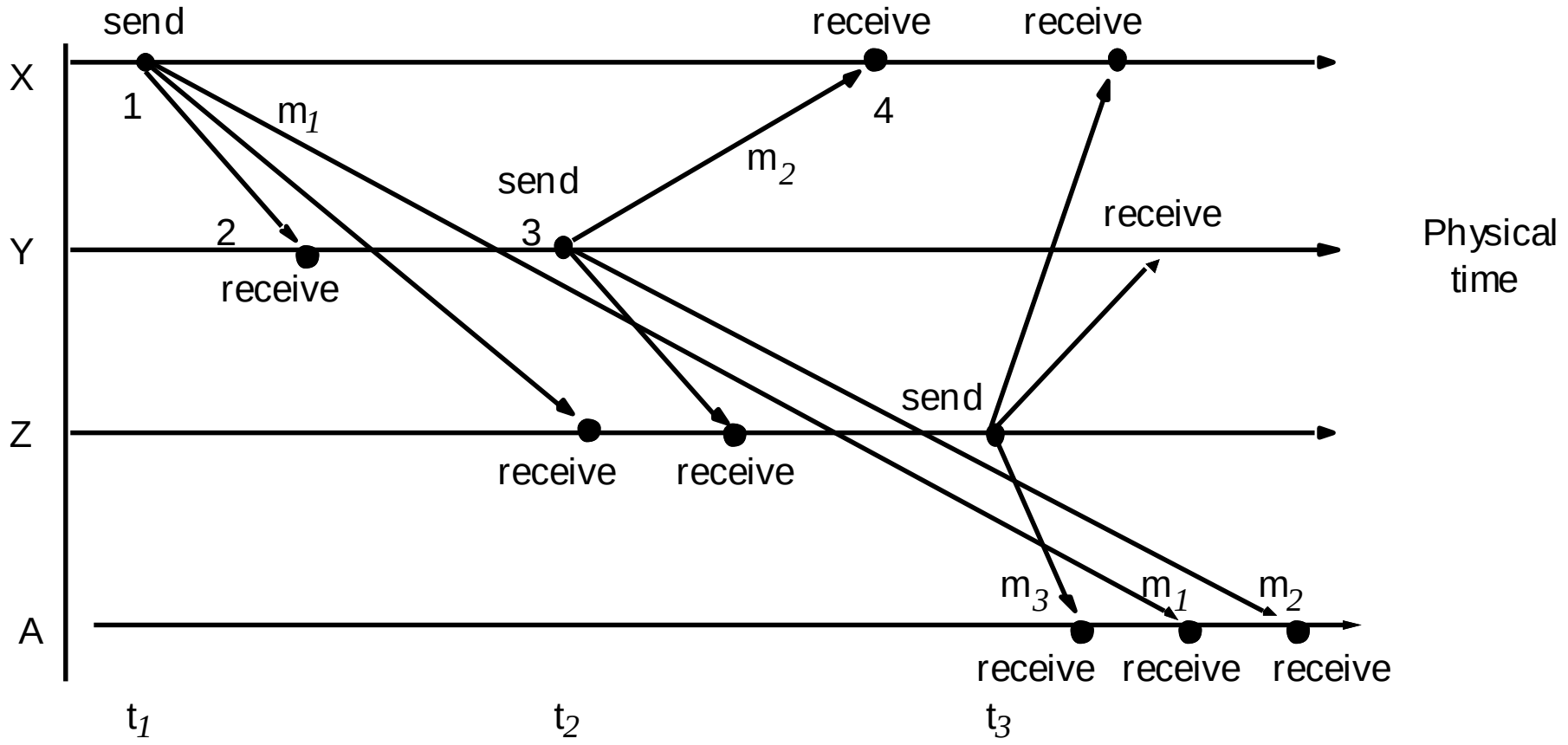
. Y e Z respondem para os outros com o assunto: Re: Meeting

O modelo de interacção e o problema da ordenação de eventos (cont.)

Uma vez que não há limites no tempo de comunicação, as mensagens podem ser entregues de tal forma que o utilizador A receba as mensagens pela ordem:

De:	**Subject:**
Z	Re:Meeting
X	Meeting
Y	Re:Meeting

O modelo de interacção e o problema da ordenação de eventos (cont.)



O modelo de interacção e o problema da ordenação de eventos (cont.)

Solução proposta por Lamport [1978]

Criar um tempo lógico para marcar a sequência de eventos e determinar a ordem correcta em que eles aparecem no tempo.

Ver capitulo 10 (Coulouris) ...

b) O modelo de Avarias (Failure Model)

Uma avaria é qualquer alteração do comportamento do sistema em relação ao esperado (i.é, em relação à sua especificação).

Define de que maneira as avarias podem ocorrer

. Podem atingir os processos ou os canais de comunicação

Tipos de Avarias:

i) Avarias por omissão;

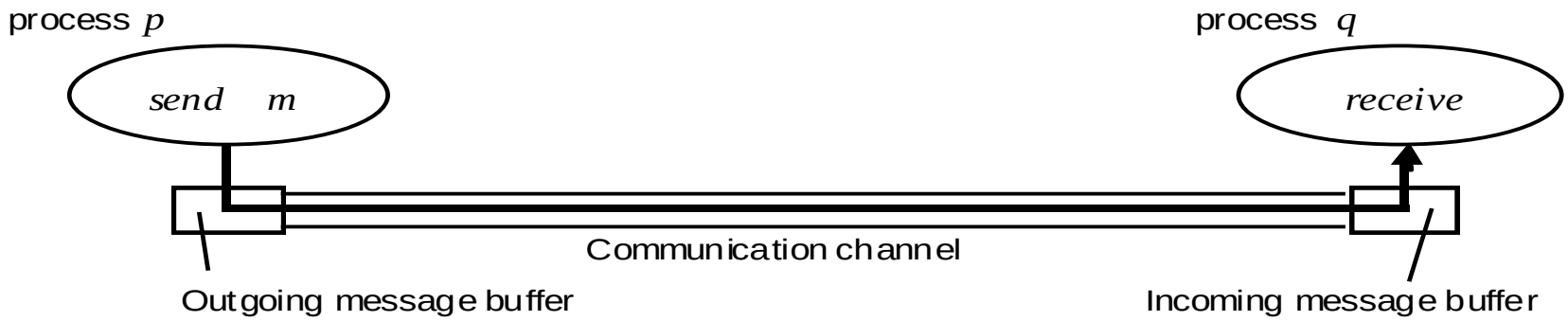
ii) Avarias arbitrárias;

iii) Avarias em tempo;

i) Avarias por omissão

- quando um processo "deixa de funcionar" em algum ponto do sistema distribuído;

- quando o canal de comunicação falha;



. Tipos de Avarias por omissão

No processo:

Fail-stop – o processo bloqueou (crashed) e esse facto pôde ser detectado por outros processos.

Crash – o processo aparentemente bloqueou mas não é possível garantir que

apenas deixou de responder por estar muito lento, ou porque

as mensagens que enviou não chegaram.

. Tipos de Avarias por omissão (cont.)

Na comunicação

Omission – uma mensagem colocada no buffer de emissão nunca chega ao buffer de recepção
(falta de espaço no buffer, perda de pacotes na rede, …).

Send-omission – uma mensagem perde-se entre o emissor e o buffer de emissão

Receive-omission – uma mensagem perde-se entre o buffer de recepção e o receptor

ii) Avarias arbitrárias (ou Bizantinas)

Qualquer tipo de erro pode acontecer:

- **Nos processos:**
  - . O processo não responde;
  - . O estado do processo é corrompido;
  - . Responde de forma errada;
  - . Responde fora de tempo.

. Avarias arbitrárias (cont.)

- **Nos canais de comunicação:**

  . mensagens corrompidas;

  . mensagens não entregues;

  . mensagens duplicadas;

  . mensagens inexistentes são entregues;

(São raras de ocorrer nos canais de comunicação porque o software de comunicação protege as mensagens com somas de verificação ("checksums"), números de sequenciamento, etc)

iii) Avarias em tempo

- Ocorrem quando o tempo limite para um evento ocorrer é ultrapassado;

- Em sistemas eminentemente síncronos é um indicativo seguro de falha;

Importantes em sistemas de tempo real.
(por ex.lo: sistemas de controlo, sistemas multimédia, ...)

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send,* but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

4 - Modelos Fundamentais

O modelo de falhas é uma descrição do tipo de falhas que podem ocorrer e de como o sistema se deve comportar perante essas falhas.

- Ao projetar o sistema, permite prever comportamentos adversos e definir mecanismos de deteção e de recuperação ou mitigação.

- O tipo de falha que se assume vai influenciar mecanismos de:

  Deteção de falhas (ex: timeouts, heartbeats)

  Tolerância a falhas (ex: replicação, verificação do resultado)

  Recuperação de falhas(ex. roolback, logs, checkpoints)

Problema dos generais de Bizâncio:

Um grupo de generais bizantinos está a cercar uma cidade inimiga.
Cada general comanda uma parte do exército e está posicionado em diferentes locais. Para conquistar a cidade, eles precisam coordenar os ataques simultaneamente.

No entanto:
Comunicação é limitada: Os generais só podem comunicar enviando mensageiros entre si;

Risco de traição: Alguns generais podem ser traidores e deliberadamente enviar mensagens falsas para sabotar o plano.

Objetivo: Todos os generais leais precisam concordar sobre se vão atacar ou recuar, mas qualquer decisão tomada deve ser unânime entre os generais leais.

- O problema dos generais bizantinos é uma metáfora que destaca os desafios de alcançar consenso em sistemas distribuídos, onde a presença de participantes maliciosos ou que podem falhar complica a coordenação e a tomada de decisão.

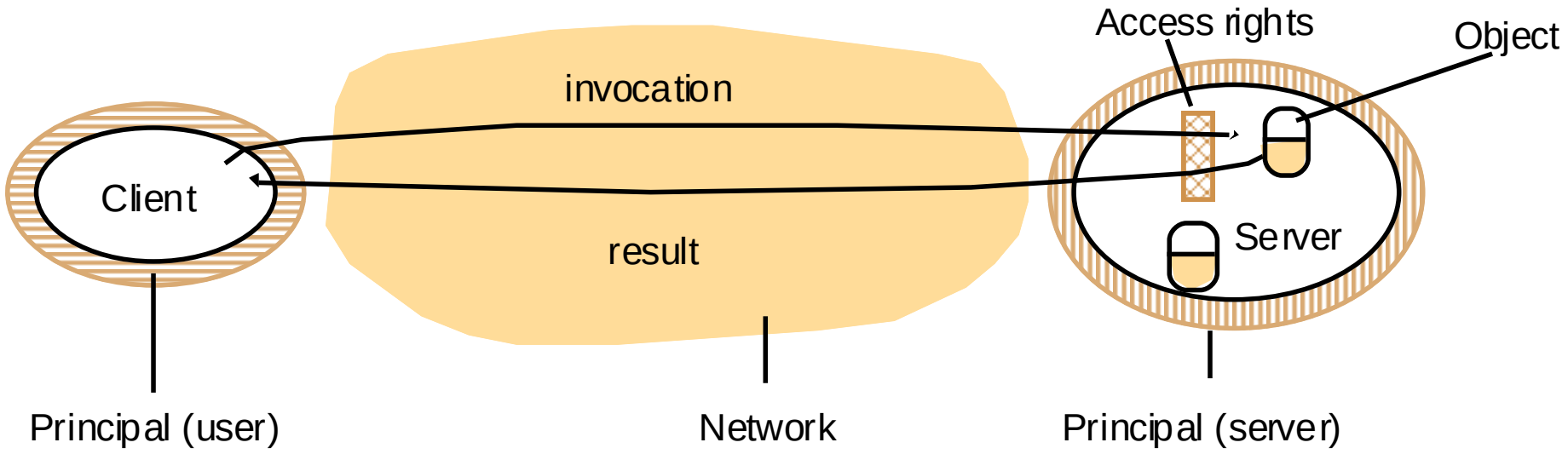
*L. Lamport; R. Shostak; M. Pease (1982). "The Byzantine Generals Problem". ACM Transactions on Programming Languages and Systems. **4** (1): 382–401.*

- Assumir falhas de fail-stop, permite usar algoritmos mais simples.
- Assumir falhas bizantinas, implica algoritmos mais robustos e mais redundância (para tolerar **f** falhas bizantinas são necessárias 3***f**+1 réplicas).

c) O modelo de Segurança

Proteção das entidades do sistema, processo/utilizador (“principal”)

Direitos de acesso especificam **que entidades** podem aceder, e **de que forma**, **a que recursos**.



Ex. Que entidade pode executar que operações num dado objeto

- O servidor é responsável por verificar:

- a identidade de quem (entidade) fez o pedido;

- se essa entidade tem direitos de acesso para realizar a operação pretendida.

- O cliente deverá verificar

a identidade de quem lhe enviou a resposta, para ver se a resposta veio da entidade esperada.

Que ameaças?

Supondo que existe um processo inimigo (adversário) capaz de:

- enviar qualquer mensagem para qualquer processo

- interceptar (ler/copiar) qualquer mensagem trocada entre 2 processos

Classificação das ameaças:

i) aos processos
ii) à comunicação
iii) negação de serviço

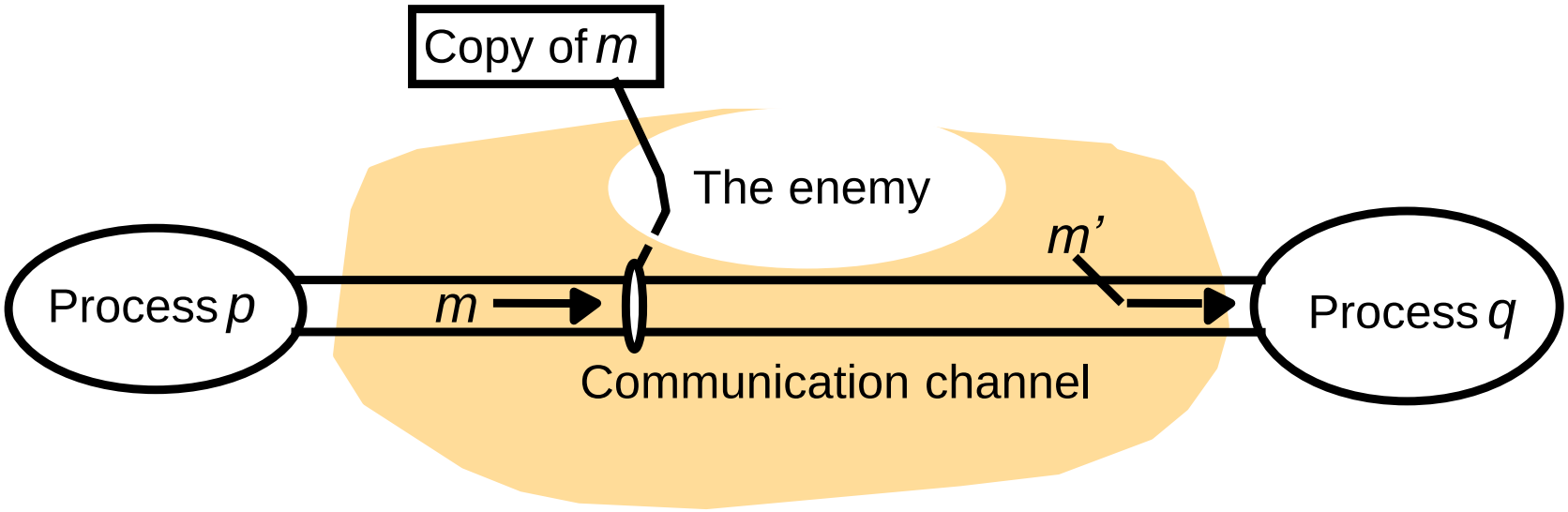
i) Ataques a processos

- Ao projectar um servidor, ter consciência de que:

  - Os protocolos de rede não oferecem protecção para que o servidor saiba a identidade do emissor

    (IP inclui o endereço do computador origem da mensagem mas um processo inimigo pode forjar esse endereço)

  - Um cliente também não dispõe de métodos para validar as respostas de um servidor



Em ambos os casos um processo inimigo pode fazer-se passar pela entidade (cliente /servidor) e enviar a mensagem solicitada

ii) Ataques a canais de comunicação

- Um processo inimigo pode copiar, alterar ou injectar mensagens na rede

A comunicação pode ser violada por processos que observam a rede à procura de mensagens significativas

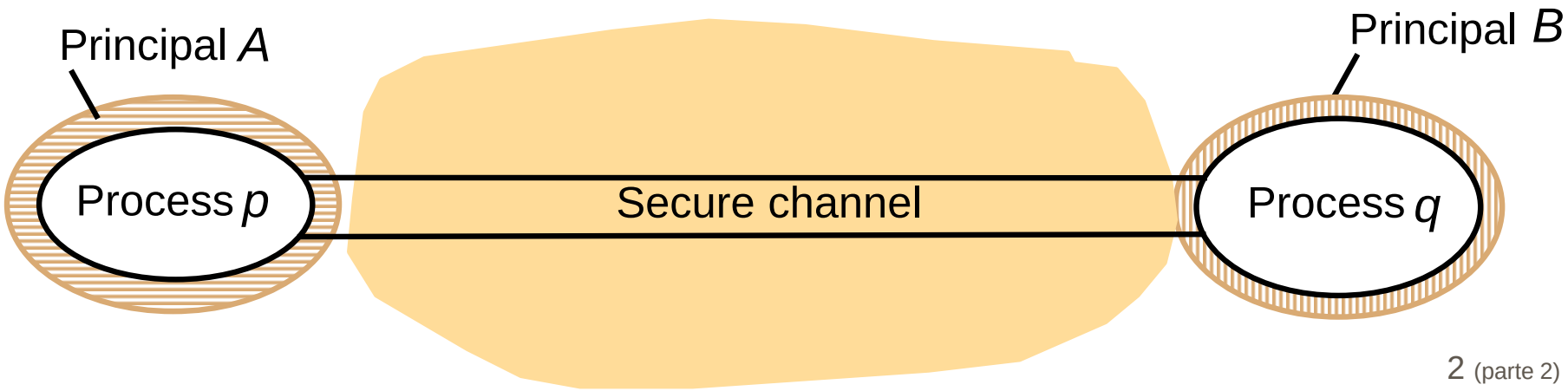
(essas mensagens podem posteriormente ser reveladas a terceiros)

iii) Negação de serviço

Um processo intruso **captura** uma mensagem de solicitação de serviço e **retransmite-a** inúmeras vezes ao destinatário,

fazendo-o executar sistematicamente o mesmo serviço e **ultrapassando** a sua capacidade de resposta

Como lidar com estas ameaças? - utilização de canais seguros



Definição de canal seguro:

Canal utilizado para comunicação entre dois processos com as seguintes características:

- Cada processo pode identificar com 100% de confiança a entidade responsável pela execução do outro processo;
- As mensagens que são transferidas de um processo para outro são garantidas do ponto de vista da integridade e da privacidade;
- As mensagens têm garantia de não repetibilidade ou reenvio por ordem distinta
  (cada mensagem inclui um tempo físico ou lógico);

Criptografia:

´Técnica de codificar o conteúdo de uma mensagem de forma a "esconder" o seu conteúdo.

É necessário que ambos os processos possuam a chave de codificação /descodificação

jtup f tfhsfep

Autenticação:

Incluir na mensagem uma porção (encriptada) que contenha informação suficiente para identificar a entidade e verificar os seus direitos de acesso

Principais componentes de um modelo de segurança:

- **Autenticação**

(Garantir que as entidades são quem dizem ser. Ex: senhas, certificados digitais, biometria)

- **Controlo de acesso**

(Definir o que cada entidade pode ou não fazer. Ex: listas d controlo de acesso)

- **Confidencialidade**

(Assegurar que os dados trocados não podem ser lidos por terceiros não autorizados. Ex: criptografia)

- **Integridade**

(Assegurar que os dados não são alterados ou corrompidos durante a comunicação ou armazenamento. Ex: usar assinaturas digitais, *hashes*)

- **Disponibilidade**

(o sistema deve funcionar corretamente mesmo na presença de ataques mal intencionados. Ex. proteção contra ataques de negação de serviço)

- **Auditoria e registo**

De ser possível rastrear ações e detetar comportamentos anómalos. Ex: sistemas de logging)

- **Não-repúdio**

(Garantir que uma autoridade não pode negar a autoria de uma ação)

Criar uma modelo de segurança

Analisar as principais ameaças:

– riscos envolvidos /possíveis consequências;

Fazer o balanço entre o custo de proteger o sistema e o **risco** que de facto as ameaças representam.