

Sistemas Distribuídos

Modelos: Físico, de Arquitetura
e de Análise Fundamental

Abstrações

- Abstrações são familiares ao programador
 - Ajudam a simplificar um problema
 - Vários níveis
 - Os níveis superiores usam interfaces para as operações de níveis inferiores
- A distribuição tem implicações
 - Ausência de memória partilhada
 - Ausência de relógio global...
 - Concorrência
 - Falhas
 - Dificuldades de coordenação
- Modelo: especificação dos aspetos relevantes, um conjunto de pressupostos sobre o contexto e funcionamento de um sistema

Modelo Físico

- Modelo Físico
 - Representação da camada de hardware onde assenta todo o sistema distribuído
 - Identifica os diferentes computadores e outros dispositivos pertencentes ao sistema e o modo como estes estão interligados
 - Sem detalhes de tecnologia
 - Modelo físico mínimo
 - Conjunto (extensível) de computadores (os nós), interligados por rede
- Gerações de Modelos
 - Iniciais
 - redes locais, PCs e impressoras...
 - Escala global / Era da Web
 - desde 1990's; redes de redes; PCs e servidores (fixos)
 - Contemporâneo
 - nós podem ser notebooks o smartphones; mobilidade

Dificuldade e ameaças nos SD

- variados modos de utilização
 - distintas necessidades
 - capacidade de processamento
 - volume de informação
- grande variedade de ambientes
 - heterogeneidade de hardware, sistemas operativos, redes e respectivos protocolos...
- problemas internos
 - sincronização de relógios, falhas de hardware ou software, incoerência decorrente da concorrência
- problemas externos
 - ataques à integridade, confidencialidade, *denial of service*

Gerações de Sistemas Distribuídos

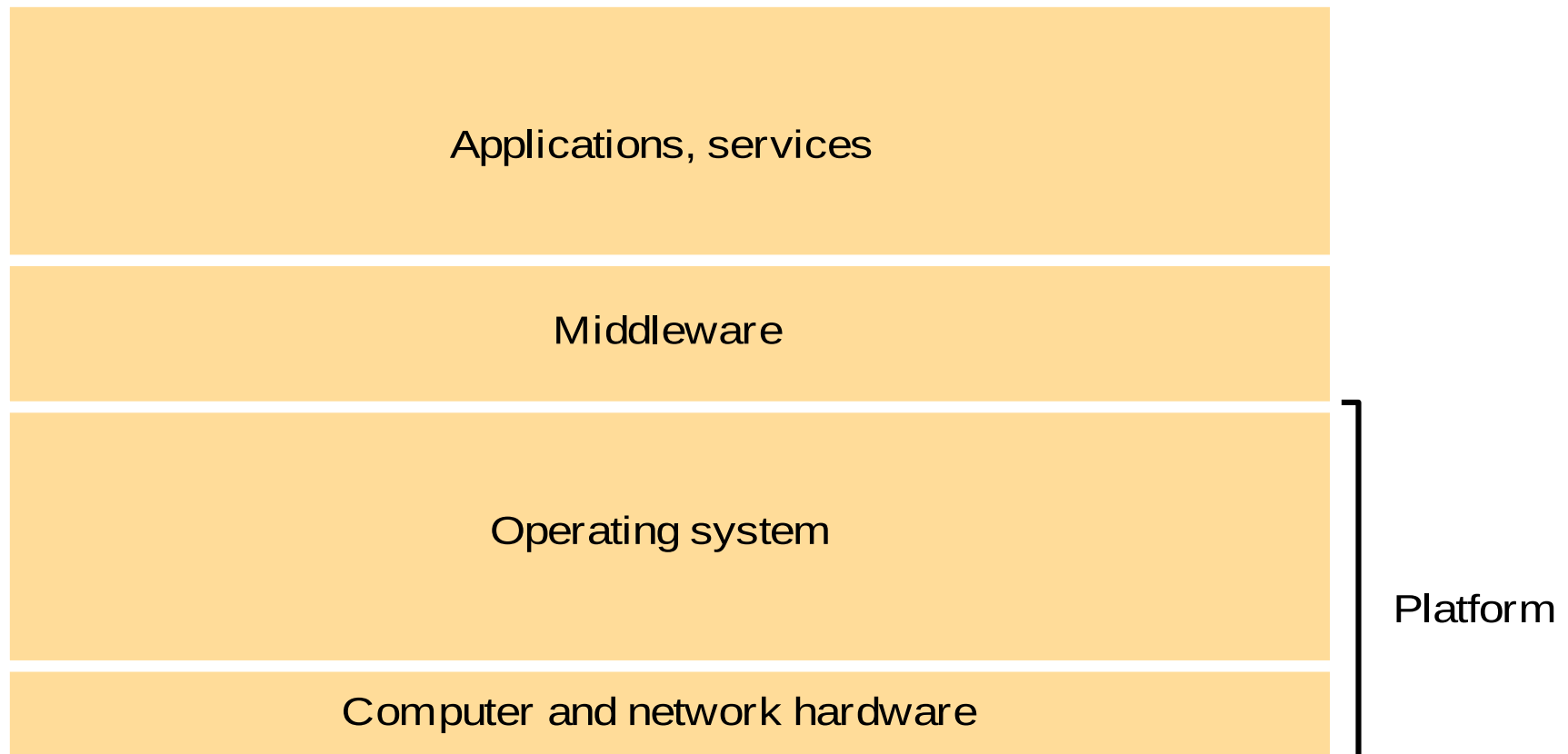
<i>Distributed systems:</i>	<i>Early</i>	<i>Internet-scale</i>	<i>Contemporary</i>
<i>Scale</i>	Small	Large	Ultra-large
<i>Heterogeneity</i>	Limited (typically relatively homogenous configurations)	Significant in terms of platforms, languages and middleware	Added dimensions introduced including radically different styles of architecture
<i>Openness</i>	Not a priority	Significant priority with range of standards introduced	Major research challenge with existing standards not yet able to embrace complex systems
<i>Quality of service</i>	In its infancy	Significant priority with range of services introduced	Major research challenge with existing services not yet able to embrace complex systems

Outros Modelos

- Modelos de Arquitetura
 - modelo *client-server*
 - *variantes*
 - modelo *peer process*
 - modelos em camadas *two-tier* e *three-tier*
- Modelos de Análise Fundamental
 - modelo de interação
 - modelo de falha
 - modelo de segurança

Camadas nos SD

cada camada fornece um conjunto de serviços à camada superior, escondendo os detalhes do nível abaixo



Categorias de Middleware

<i>Major categories:</i>	<i>Subcategory</i>	<i>Example systems</i>
<i>Distributed objects (Chapters 5, 8)</i>	Standard	RM-ODP
	Platform	CORBA
	Platform	Java RMI
<i>Distributed components (Chapter 8)</i>	Lightweight components	Fractal
	Lightweight components	OpenCOM
	Application servers	SUN EJB
	Application servers	CORBA Component Model
	Application servers	JBoss
<i>Publish-subscribe systems (Chapter 6)</i>	-	CORBA Event Service
	-	Scribe
	-	JMS
<i>Message queues (Chapter 6)</i>	-	Websphere MQ
	-	JMS
<i>Web services (Chapter 9)</i>	Web services	Apache Axis
	Grid services	The Globus Toolkit
<i>Peer-to-peer (Chapter 10)</i>	Routing overlays	Pastry
	Routing overlays	Tapestry
	Application-specific	Squirrel
	Application-specific	OceanStore
	Application-specific	Ivy
	Application-specific	Gnutella

Entidades em Comunicação, num SD

- Perspetiva orientada ao sistema
 - Processos
 - Threads
 - Eventualmente sensores e outros dispositivos; outros nós.
- Perspetiva orientada ao problema, a perspetiva da programação
 - Objetos
 - Componentes (CORBA, EJB)
 - Mecanismos parecidos aos objetos; oferecem uma abstração para implementar sistemas; baseados numa API; com algumas vantagens face ao manuseamento direto/convencional de objetos
 - Web Services
 - Outro paradigma para implementação de sistemas distribuídos, que recorre a protocolos e normas de internet, onde uma aplicação é identificada pelo URI

Paradigmas de Comunicação, entre entidades no SD

- **IPC**

- Comunicação com API de baixo nível
- Exemplos: comunicação baseada em sockets, com API do Soperativo

- **Invocação Remota**

- Execução remota de processo ou método
 - Protocolos Request-Reply
 - RPC
 - RMI

- **Comunicação Indireta**

- Comunicação em grupo (um para vários; *multicast*)
- *Publish-subscribe* (um para muitos)
- *Message Queues* (um para muitos, mas ponto a ponto; fila entre produtor e consumidor)
- *Distributed shared memory* (DSM): abstração para partilha de dados entre processos que não partilham a mesma memória física

Arquitetura do Sistema

- O sucesso com que se satisfazem as necessidades atuais e futuras depende da arquitetura do sistema
- Preocupações:
 - Escalabilidade
 - Fiabilidade
 - Capacidade de Manutenção

Capacidade de Manutenção (*Maintainability*)

- Capacidade de implantar sistema e acompanhar o seu funcionamento (*Operability*)
- Simplicidade
 - Evitar complexidade do sistema, para que a resolução de problemas não encontre obstáculos
- Evolucionabilidade (*Evolvability*)
 - Facilitar alterações ou atualizações futuras
 - Não tomar decisões que limitem ou condicionem o trabalho futuro

Aspetos a considerar

no Desenho de Arquiteturas Distribuídas

- Desempenho
 - responsiveness: a acessibilidade às respostas
 - throughput: taxa de trabalho computacional por unidade de tempo
 - balanceamento (para resolver situações de carga)
- Qualidade do Serviço (QoS)
 - Depende de aspetos não funcionais, como a fiabilidade e o desempenho
- Cache e Replicação
 - cache: server proxy e também do lado do cliente
- Segurança e tolerância a falhas

Arquitetura do Sistema

- Modelo de Arquitetura (*Architectural Model*) de um SD:
 - Descreve o sistema em termos das tarefas computacionais e comunicação desempenhadas por cada elemento
 - **Elemento**: computador; ou conjunto de computadores interligados
- Definir a disposição dos componentes individuais de um sistema numa rede de computadores e o modo como interagem entre si
 1. simplifica e abstrai o papel de cada componente individual
 2. considera a disposição dos componentes pela rede
 - identificar distribuição de dados e locais onde se requer grande quantidade de trabalho/processamento (*workload/carga*)
 3. relações entre os componentes
 - do ponto de vista do papel funcional que desempenham
 - comunicações que estabelecem

Arquitetura do Sistema

- Simplificação inicial do papel de cada componente:
 - **classificar** os processos como:
 - server
 - client
 - peer processes: processos que cooperam e comunicam de forma simétrica para realizar uma tarefa (usualmente de interesse comum)
 - esta classificação facilita a percepção:
 - responsabilidades dos componentes
 - locais de *workload* (carga)
 - ajustes para alcançar fiabilidade e eficiência

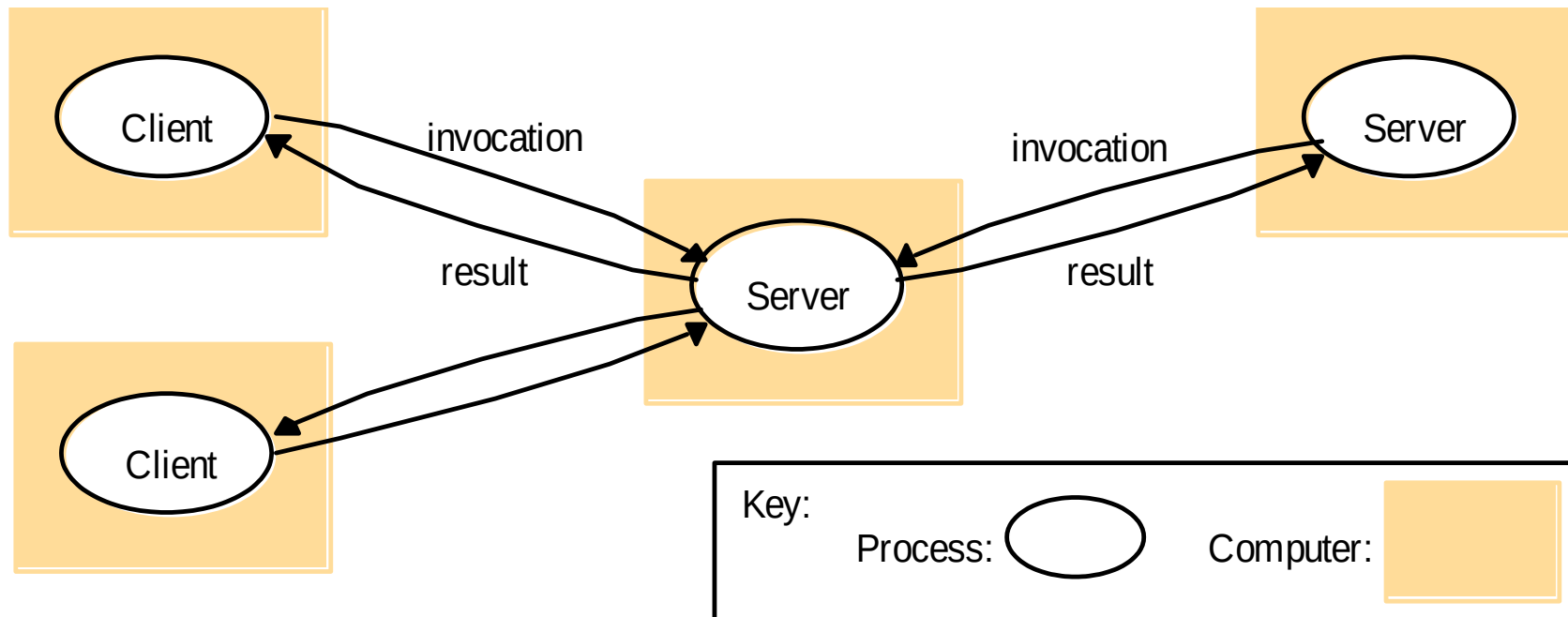
Modelos na Arquitetura de SD

- cliente-servidor
- variantes do modelo cliente-servidor
 - divisão ou replicação dos dados em sistemas de múltiplos servidores
 - caching de dados por clientes ou *proxy servers*
 - código móvel e agentes móveis
- *peer processes*
- *two-tier* e *three-tier*

Modelo Cliente-Servidor

Clientes invocam um servidor

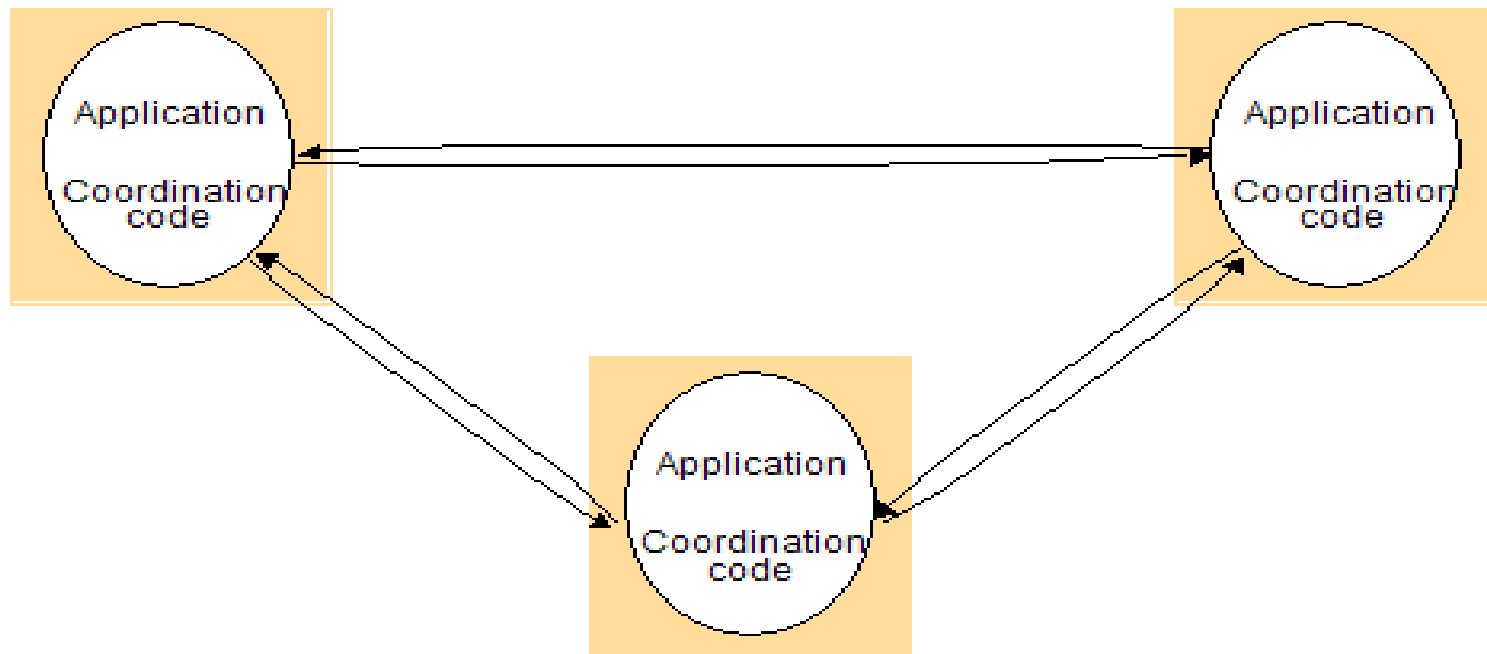
-o servidor pode também assumir o papel de cliente para ligar-se a outro servidor



Modelo *Peer Processes*

Aplicação baseada em *peer processes*

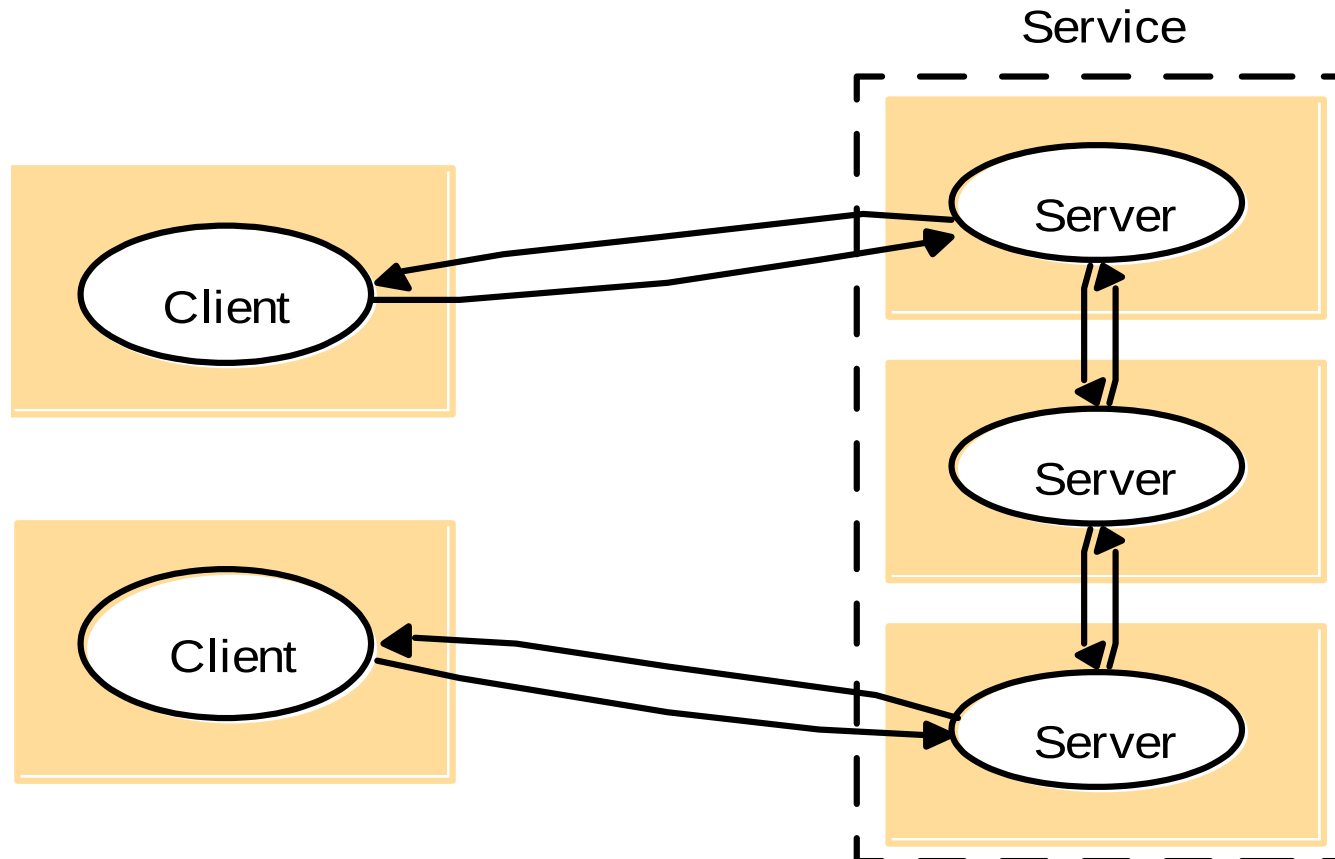
- arquitetura em que os processos (nós) desempenham papéis idênticos, sem uma separação prévia entre clientes e servidores
- pontualmente, os processos poder-se-ão comportar como clientes ou como servidores
- a interação varia em função das necessidades



Variante do Modelo Cliente-Servidor: múltiplos servidores

Serviço prestado por múltiplos servidores

- os servidores podem correr em diferentes computadores
- cada servidor pode lidar com uma partição dos dados ou uma réplica dos dados

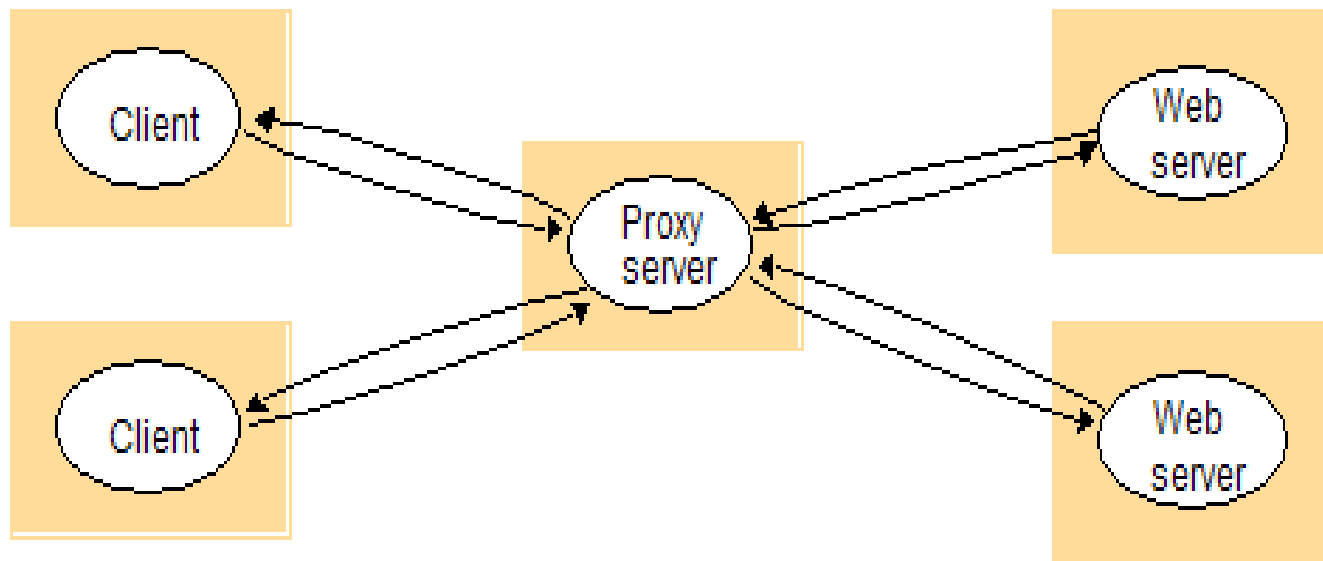


Variante do Modelo Cliente-Servidor: proxy e cache

Cache: uma réplica com fragmento de dados acedidos recentemente que é mantida

- na aplicação cliente
- num servidor (Proxy Server), quando a réplica dos dados é disponibilizada a vários clientes

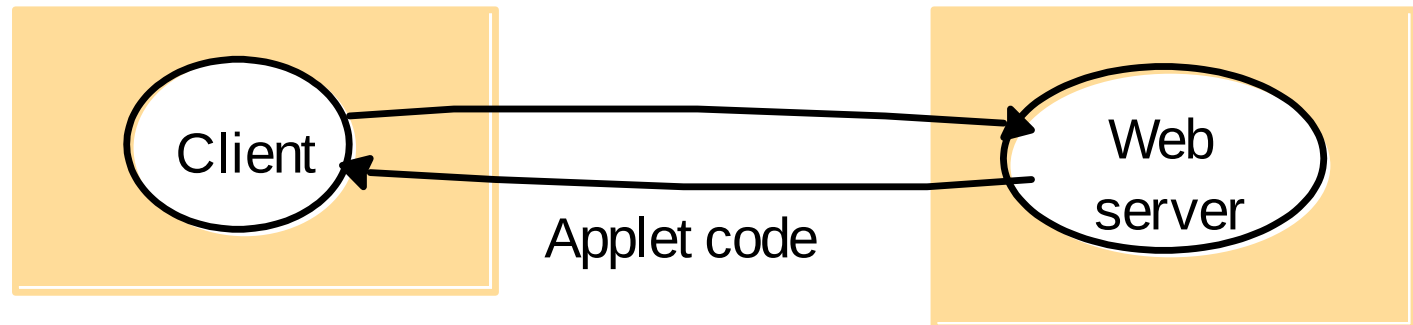
Cache: visa contribuir para a disponibilidade e desempenho do sistema



Variante do Modelo Cliente-Servidor: Código Móvel

• *Web applets*

a) client request results in the downloading of applet code



b) client interacts with the applet



Variante do Modelo Cliente-Servidor: Código Móvel

- Agentes móveis
 - programas em execução
 - incluem o código do programa e os dados
 - viajam na rede para realizar uma tarefa (para um utilizador ou um processo)
- *Network Computer*
 - SO e restante software obtido pela rede, a partir de um servidor remoto
 - execução local

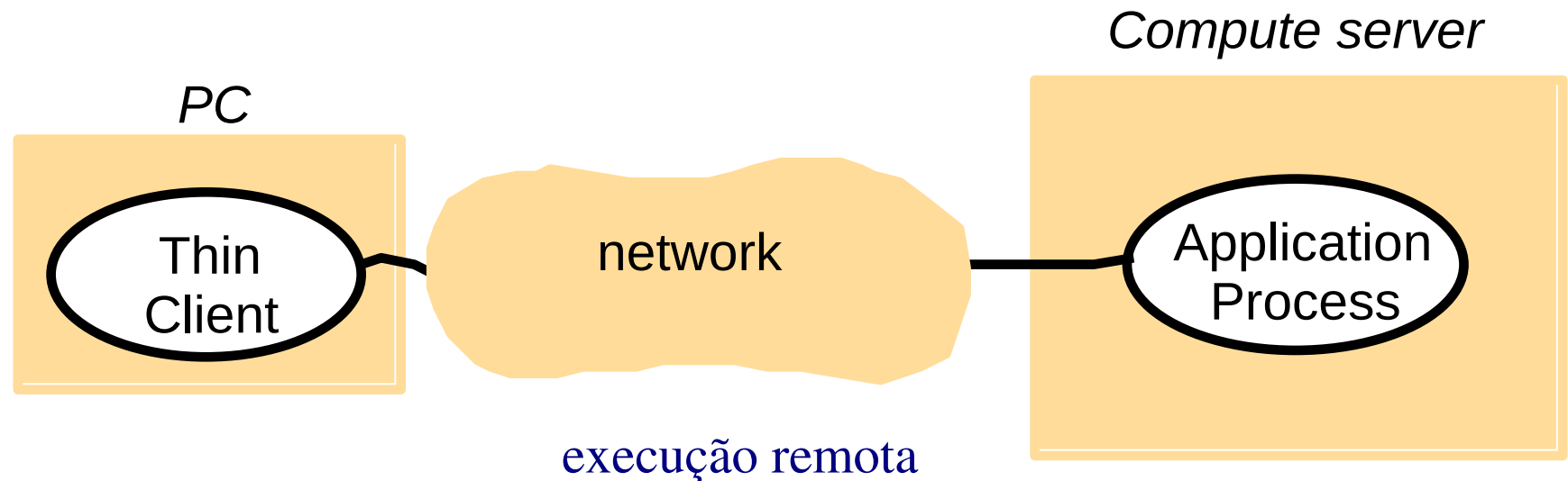
Variante do Modelo Cliente-Servidor: Código Móvel

- *Thin clients e compute servers*

- interface gráfica **local** (*thin client*) que controla a execução de aplicações num computador **remoto** (*compute server*)

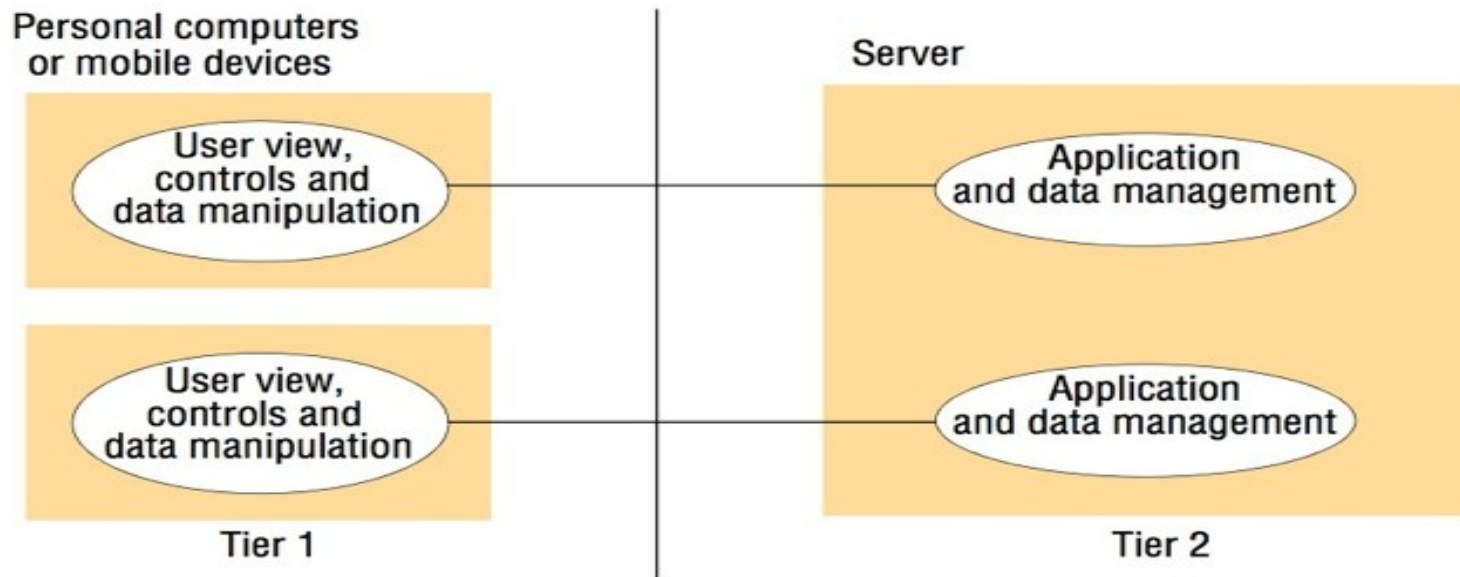
- Gestores gráficos

- X11 (clientes invocam operações no servidor via RPC)
- VNC: virtual network computer

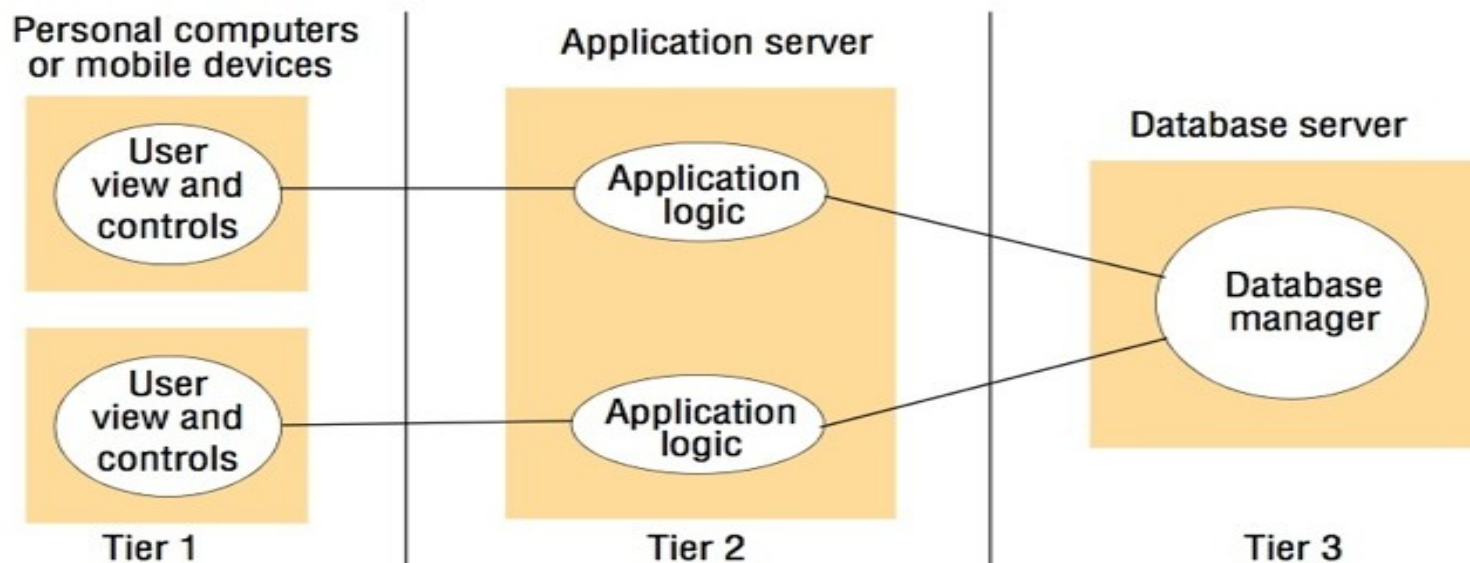


Aquiteturas multicamada: *Two-tier* e *Three-tier*

a)



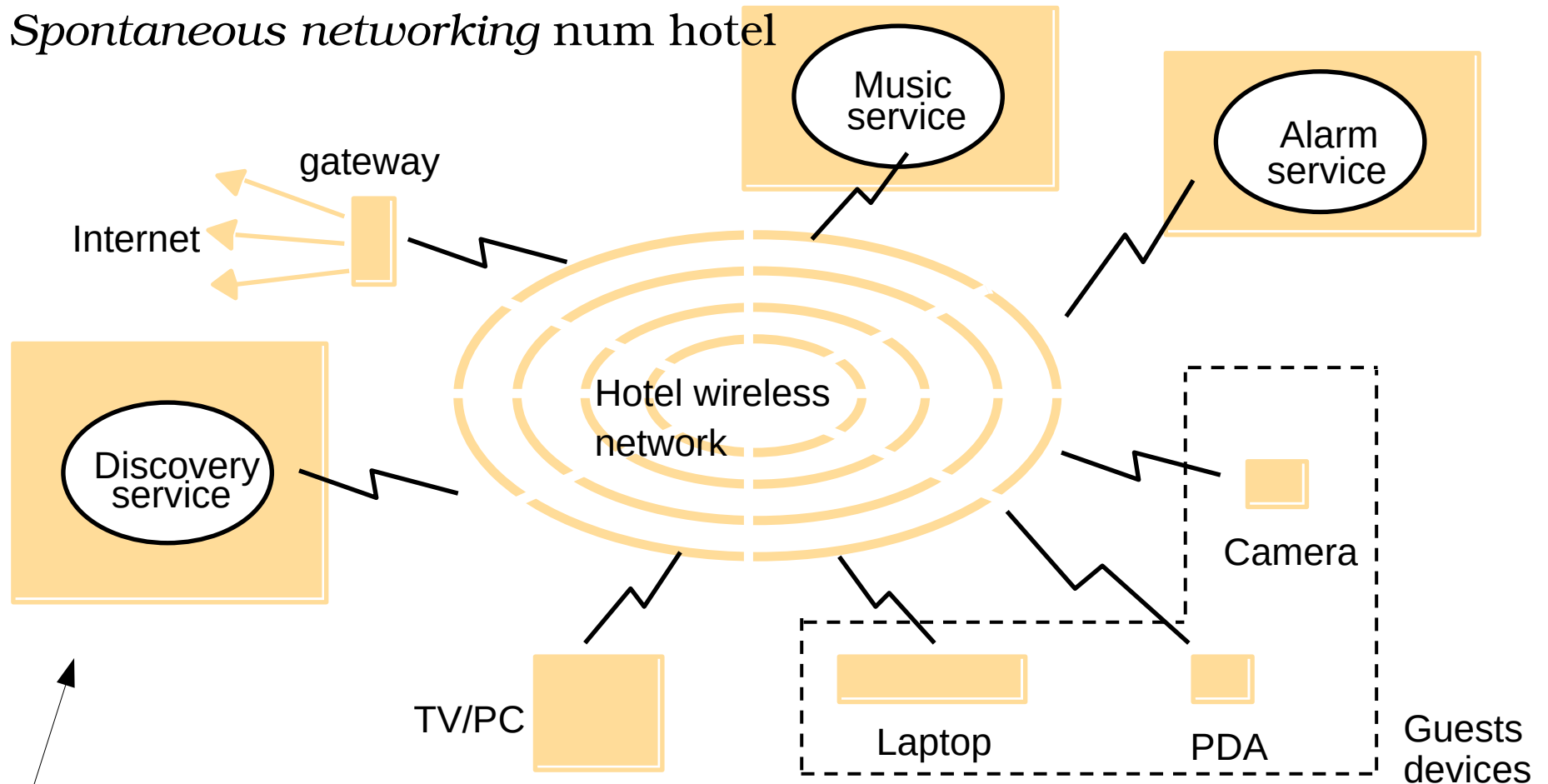
b)



Mobilidade e Redes de Computadores

Dispositivos móveis e *Spontaneous networking*

Spontaneous networking num hotel



livro

Spontaneous Networking

- *Spontaneous Networking*
 - A forma de interação automática entre dispositivos móveis e outros dispositivos inseridos numa rede ou ambiente distribuído
- Aspetos chave
 - Facilidade de associação a uma rede local, de preferência sem fios
 - Conectividade limitada
 - Dificuldades inerentes à mobilidade
 - Capacidade reduzida dos equipamentos terminais
 - Aspectos de segurança e privacidade
 - Facilidade de integração com **serviços** locais
 - *Discovery Services*
 - *Registration Service + Lookup Service*

Interfaces e Objetos

- As funções disponíveis para interação com um processo são especificadas em **Interfaces**
- cliente-servidor (simples)
 - interface fixa
- modelos Orientados por Objetos (CORBA, RMI)
 - objetos podem variar: novas funcionalidades
 - interface dinâmica (tipicamente)

Modelos de Análise Fundamental (*Fundamental Models*)

- **Propósito**
 - realçar aspetos de desenho, dificuldades e ameaças a considerar no desenvolvimento de SD, de modo que estes possam desempenhar a sua tarefa de modo correto, fiável, eficiente e seguro
- Descrição formal de aspetos
 - comuns a todos os modelos de arquitetura
 - que influenciam a sua fiabilidade
 - a nível de processos, rede e recursos

Modelos de Análise Fundamental

- Modelos de análise a diferentes níveis:
 - **Interação:** análise de aspetos relacionados com o desempenho e a dificuldade de estabelecer limites temporais num SD, associados à comunicação e coordenação entre processos
 - **Falhas:** especificação exata das falhas que poderão surgir em processos, dispositivos ou canais de comunicação/rede
 - **Segurança:** análise das possíveis ameaças a processos e canais de comunicação/rede
 - ataques internos
 - ataques externos

Modelo de Interação

- **Algoritmo**
 - sequência de passos a executar para completar uma tarefa
- **Algoritmo Distribuído**
 - sequência de passos a executar **por cada um dos processos** do sistema, incluindo a **transmissão de mensagens** (de dados e de coordenação) entre os mesmos, para completar uma tarefa

Desempenho geral do sistema

- Quando a carga (*load*) aumenta
 - Recursos constantes: como evolui o desempenho?
 - Desempenho constante: quantos mais recursos são necessários?
- Arquiteturas gerais
 - 1) *Batch processing systems*
 - *throughput*
 - 2) *Online systems* (sistemas que respondem na hora / síncronos)
 - Tempo de resposta
 - Envolve: tempo de serviço + latência

Modelo de Interação

- Fatores que influenciam a interação entre processos em SD:
 1. Desempenho dos canais de comunicação
 - latência
 - largura de banda
 - *Jitter*
 - variação no tempo necessário para o envio/entrega de fragmentos de dados de uma mesma mensagem.
Exemplo: *Streaming multimedia, > jitter > distorção < qualidade*
 2. Relógios e Eventos Temporais
 - *clock drift rate*
 - taxa de desvio do tempo face a uma referência correta

Modelo de Interação: Variantes

- **SD Síncronos**: existem limites para:
 - tempo de execução de cada passo de um processo
 - tempo até à recepção de uma mensagem enviada
 - *clock drift rate* (conhecido) em cada máquina
- **SD Assíncronos**: não há limite definido ou garantias para:
 - velocidade de execução de um processo
 - tempo de transmissão de uma mensagem, pode ter atraso (*delay*)
 - *clock drift rate*: a taxa é arbitrária
- exemplo de SD assíncrono: **Internet**

Modelo de Interação: Consistência de repositórios e a Ordenação Cronológica de Eventos

- Sistema linearizável
 - Ordenação total das operações: é sempre possível determinar a sequência cronológica entre quaisquer operações
 - É não compatível com um ambiente concorrente normal
- Sistema com ordenação causal
 - Ordenação parcial (não total)
 - Duas operações são consideradas concorrentes se não se pode dizer que uma aconteceu antes da outra
 - Dois eventos podem ordenar-se se tiverem uma relação causal (“acontece primeiro que”)

Modelo de Interação:

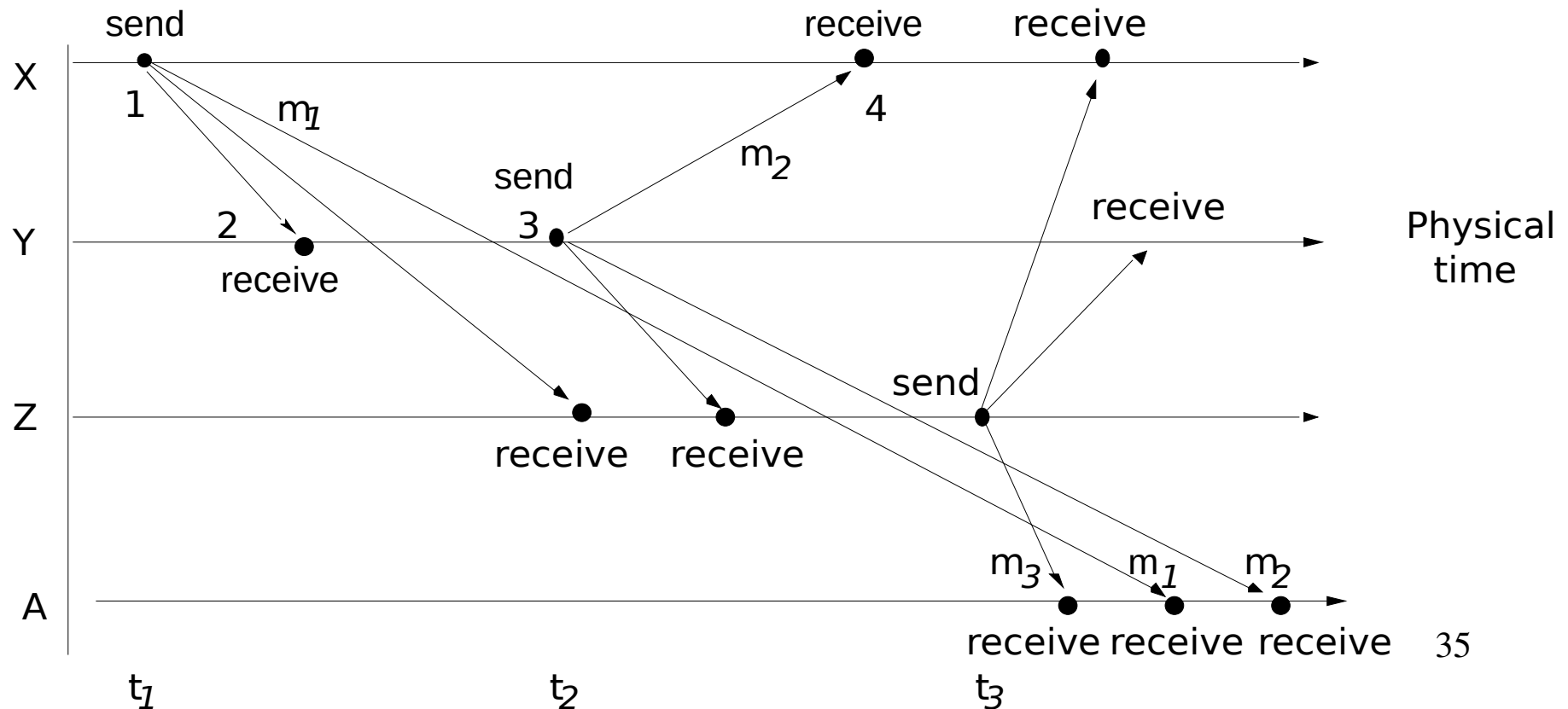
Ordenação Cronológica de Eventos

Princípio do **Tempo Lógico**

regras para ordenação (ordenação causal ou outra)

E-mail: Y envia m2 depois de receber m1; X envia m1 antes de Y receber m1

... portanto o que concluir em A, sobre a ordenação de m1 e m3?

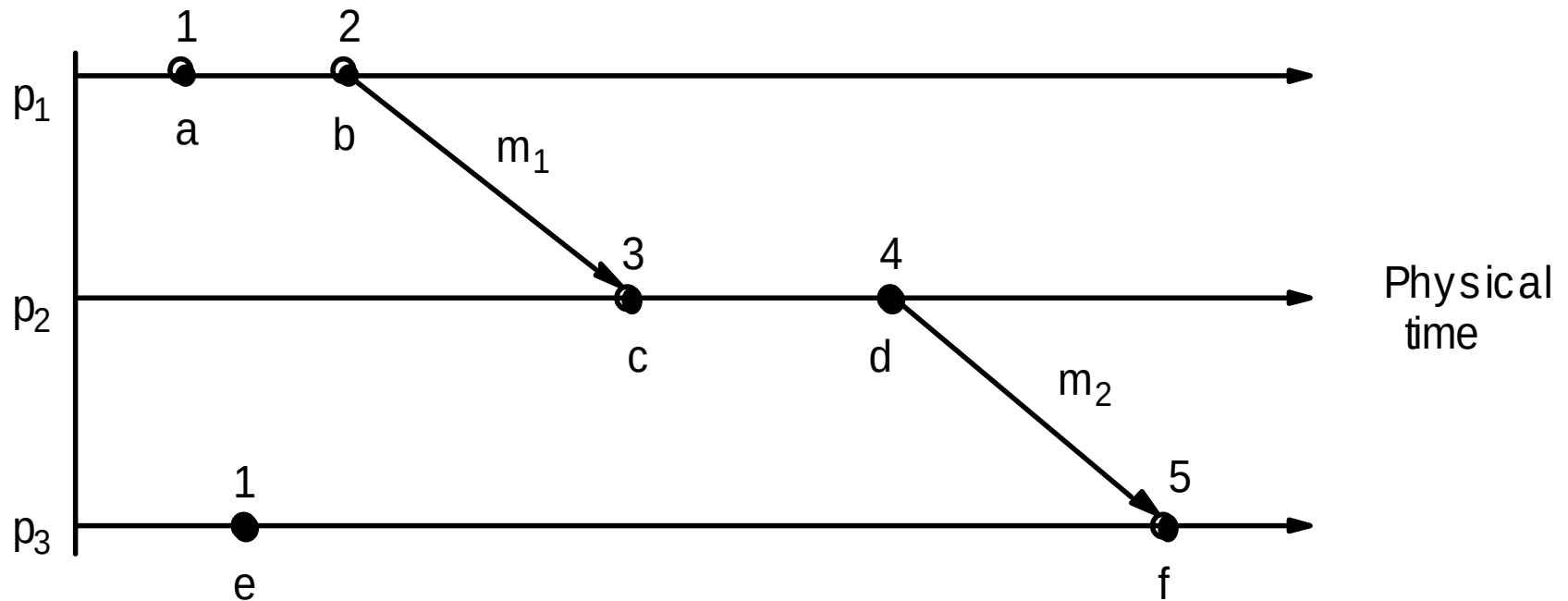


Modelo de Interação: ordenação lógica por Lamport

Lamport

- dados dois eventos, determinar a ordenação dos mesmos (independente do relógio)
- atribuir um número

(cada processo p_1, \dots, p_n , tem o seu relógio físico)



Modelo de Interação: ordenação lógica por Lamport

Lamport

- cada processo inicia com um contador, a zero
- cada processo incrementa o seu contador se envia uma mensagem, ou se recebe uma ação
- cada envio leva um timestamp (o contador)
- cada mensagem recebida conduz a um ajuste no contador de destino:
$$\text{localCounter} = \max(\text{localCounter}, \text{messageTS}) + 1$$

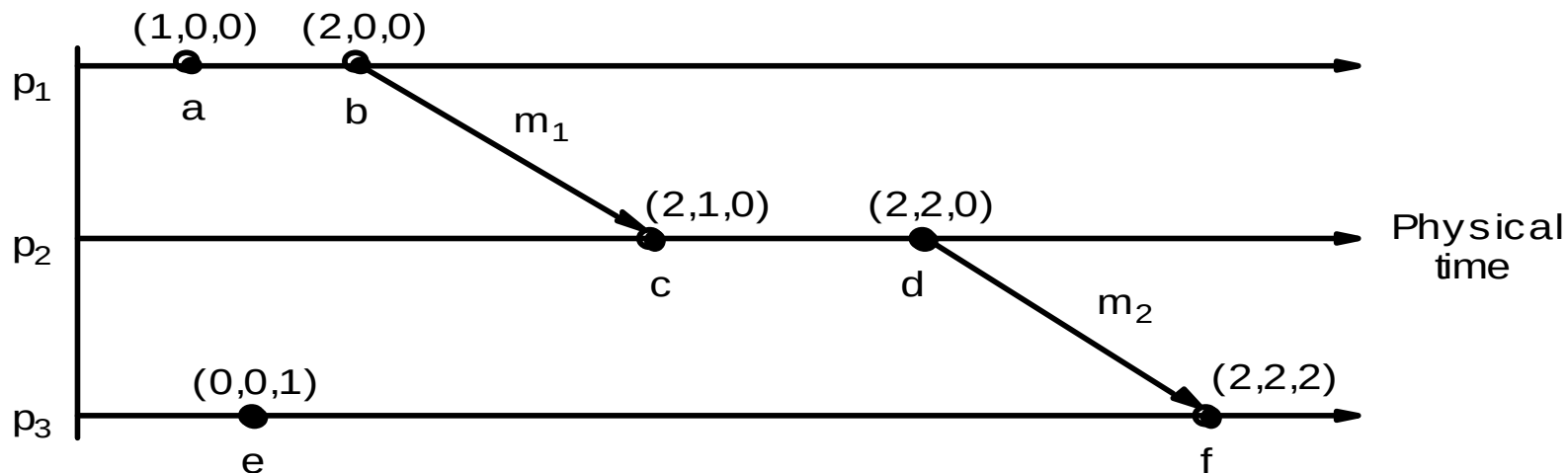
Ordenação Lógica de Lamport

- no mesmo processo: a antes de b implica $\text{TS}(a) < \text{TS}(b)$
- se p1 envia m a p2, *send(m)* precede *receive(m)*
- transitividade: se a precede b e b precede c, então a precede c
- em geral: $\text{TS}(a) < \text{TS}(b)$ não implica necessariamente que a ocorre antes de b
Ver atrás, caso de máquinas diferentes, para p1 e p3:
 - neste caso, o timestamp lógico: $\text{TS}_{p1}(a) < \text{TS}_{p3}(e)$
sse $\text{TS}_{p1} < \text{TS}_{p3}$ ou $(\text{TS}_{p1} = \text{TS}_{p3} \text{ e } p1 < p3)$

Modelo de Interação: ordenação de eventos com *Vector Timestamps* (1989 a 1991)

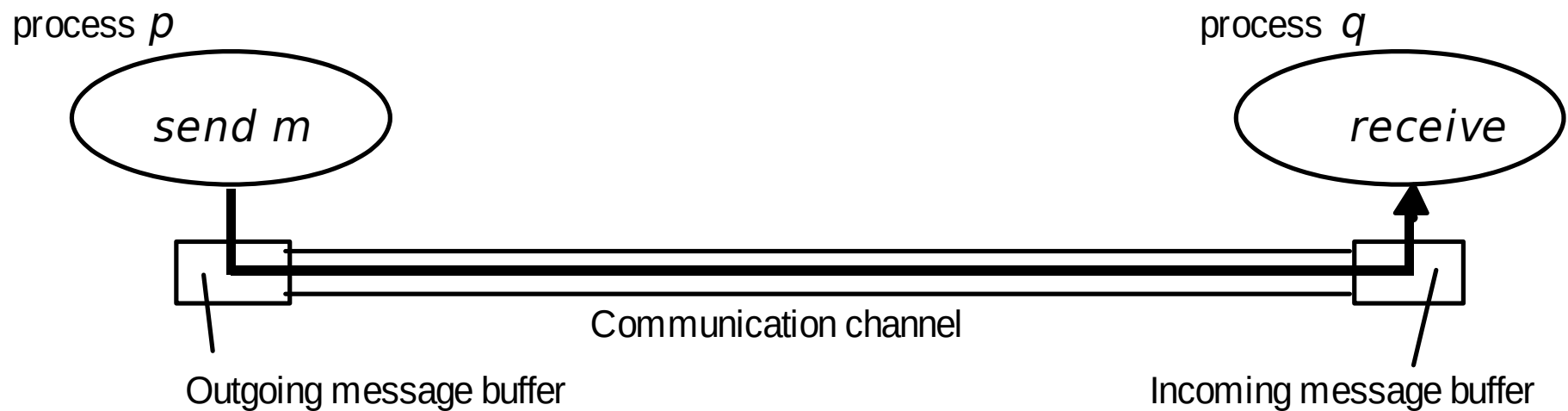
Cada processo do sistema de n (regista o que se passa com os demais)

- tem um vector de timestamps com um valor p/ processo (1.. n)
- valores iniciam a 0; cada processo k incrementa o seu valor $v[k]$ num envio, ou quando executa uma instrução; e anexa o vetor T à mensagem enviada; o recetor ajusta: $V[j] = \max(V[j], T[j])$, para $j = 1..n$
- um $V < V'$ se $V[j] < V'[j]$ **para todos os j**
eventos “paralelos” (exemplo e e c) se
não temos $TS(c) < TS(e)$ nem $TS(e) < TS(c)$



Modelo de Falhas

- Enumerar as **formas em que o sistema pode falhar**, facilitando a compreensão dos efeitos de cada falha



Modelo de Falhas

- Falhas de Omissão (*omission failures*)
 - quando o processo ou o canal falham no desempenho da tarefa que lhes cabe
 - ***falhas no processo (process omission failures)***
 - por crash efetivo ou por lentidão na resposta
 - ***falhas no canal de comunicação (communication omission failures)***
 - *dropping messages* (canal falha o transporte entre os buffers)
 - *send-omission*
 - *receive-omission*
- Falhas Arbitrárias (bizantinas)
 - pior cenário
 - falhas diversas... não determinísticas
- Falhas Temporais

Modelo de Falhas

Falhas por Omissão e Arbitrárias

<i>Tipo de Falha</i>	<i>Afecta</i>	<i>Descrição</i>
Fail-stop	Processo	Process halts and remains halted. Other processes <u>may detect this state</u> (Sistemas Síncronos onde há mais garantias).
Crash	Processo	Process halts and remains halted. Other processes <u>may not be able to detect</u> this state.
Omissão	Canal	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Processo	A process completes <i>asend</i> , but the message is not put in its outgoing message buffer.
Receive-omission	Processo	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrária (Bizantina)	Processo ou canal	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.

Modelo de Falhas

Falhas Temporais

sistemas síncronos: estes erros levam à não entrega de respostas (que teriam de chegar no intervalo determinado)

<i>Tipo de Falha</i>	<i>Afecta</i>	<i>Descrição</i>
Relógio	Processo	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Processo	Process exceeds the bounds on the interval between two steps.
Performance	Canal	A message's transmission takes longer than the stated bound.

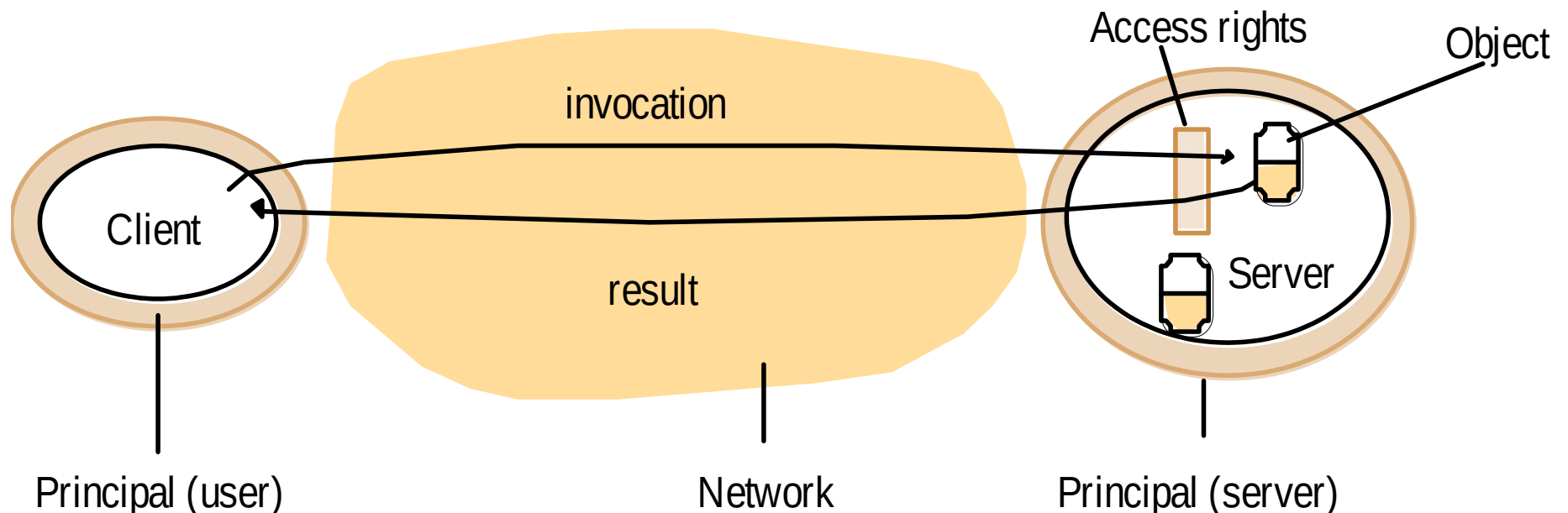
Num sistema **assíncrono** estes fenómenos acarretam lentidão mas não correspondem necessariamente a Falhas Temporais, pois não há imposições temporais rígidas.

Modelo de Segurança

- Identifica possíveis ameaças num sistema distribuído (aberto)
 - ameaças a processos (clientes, servidores)
 - identidade do interlocutor remoto
 - legitimidade daquele para aceder ao recurso do processo
 - ameaças a canais de comunicação
 - introdução de mensagens forjadas
 - adulteração do conteúdo de mensagens em trânsito
- Visa:
 - garantir segurança de objetos, processos e dos canais de comunicação

Modelo de Segurança

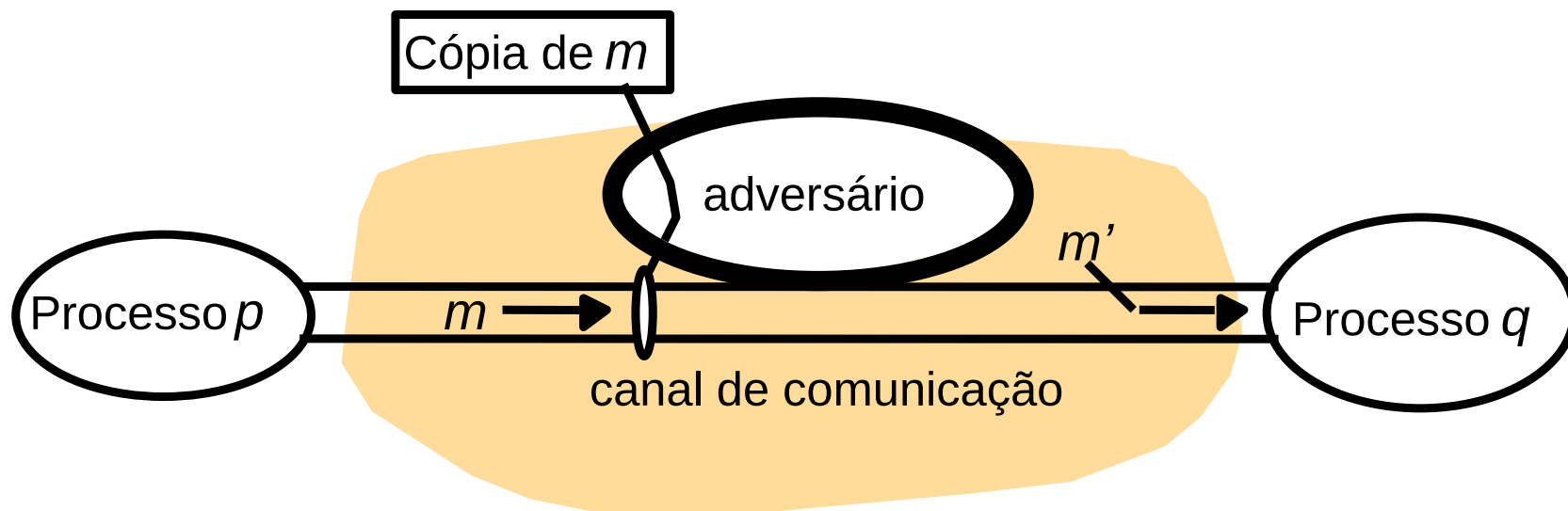
- proteger os objetos
 - *principal*: uma entidade envolvida, utilizador ou processo
 - direitos de acesso
 - especificar Quem pode fazer o Quê sobre Que Objetos



Modelo de Segurança

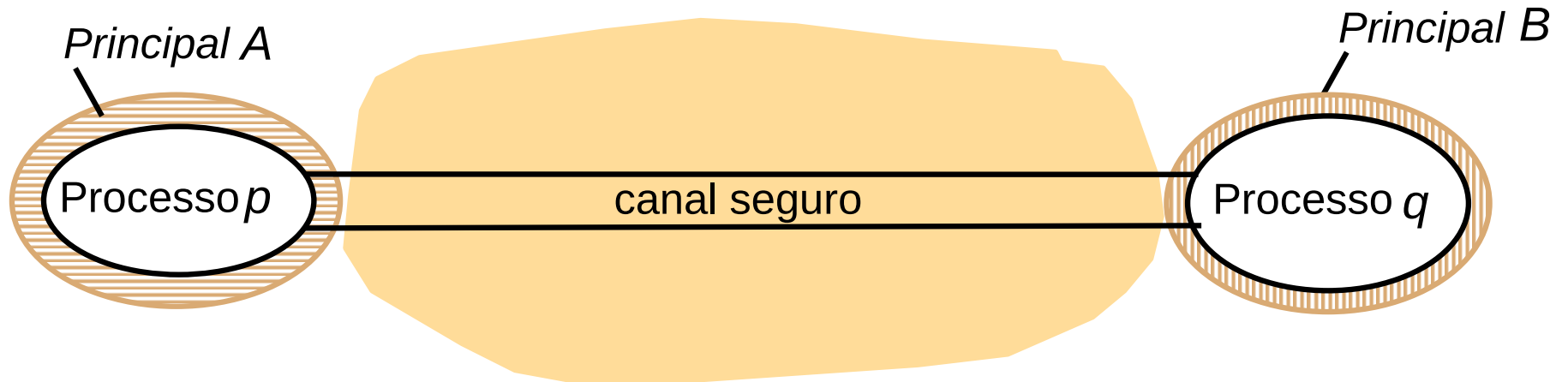
Proteger os canais de comunicação contra adversários

- canais podem ser alvo de ataques externos por parte de utilizadores hostis (adversários/oponentes/atacantes)



Modelo de Segurança

- Superar ameaças à segurança
 - criptografia
 - autenticação
 - canais seguros (exemplos: stunnel, ssh, ssl...)



Modelo de Segurança

- Outros ataques possíveis
 - congestionamento / *denial of service* (DoS)
 - *mobile code*
- Utilização de Modelos de Segurança
 - Repercussões na eficiência
 - Custos diretos e indiretos...
 - É preciso um compromisso entre
 - os ganhos
 - e o impacto que esses mecanismos acarretam...