SISTEMAS DISTRIBUÍDOS

Capítulo 2

Arquiteturas e Modelos de Sistemas Distribuídos

... No capítulo anterior (aula 1 e 2)

Definição de Sistema Distribuído

Características fundamentais dum sistema distribuído

Desafios na concepção de sistemas distribuídos

- Heterogeneidade, Escala, Transparência, Segurança, Resiliência, ...
- Qualidade de Serviço

SISTEMAS DISTRIBUÍDOS

Capítulo 2 – Aula3

Arquiteturas e Modelos de Sistemas Distribuídos

NOTA PRÉVIA

A estrutura da apresentação é semelhante e utiliza algumas das figuras do livro de base do curso

G. Coulouris, J. Dollimore and T. Kindberg,
Distributed Systems - Concepts and Design,
Addison-Wesley, 5th Edition, 2011

Organização do capítulo

Modelos arquiteturais

- Arquitetura/camadas de software
- Cliente/servidor, peer-to-peer, variantes

Modelos fundamentais – usados para descrever propriedades parciais, comuns a todas as arquiteturas

- Modelo de interação
- Modelo de falhas
- Modelo de segurança

CONTEXTOS - ARQUITETURA

Camadas de software

Reparte a complexidade de um sistema, em várias camadas, com interfaces bem definidas entre si. Cada camada pode usar os serviços da camada abaixo, sem conhecimento dos detalhes de implementação.

Arquitetura (distribuída) multinível/camada

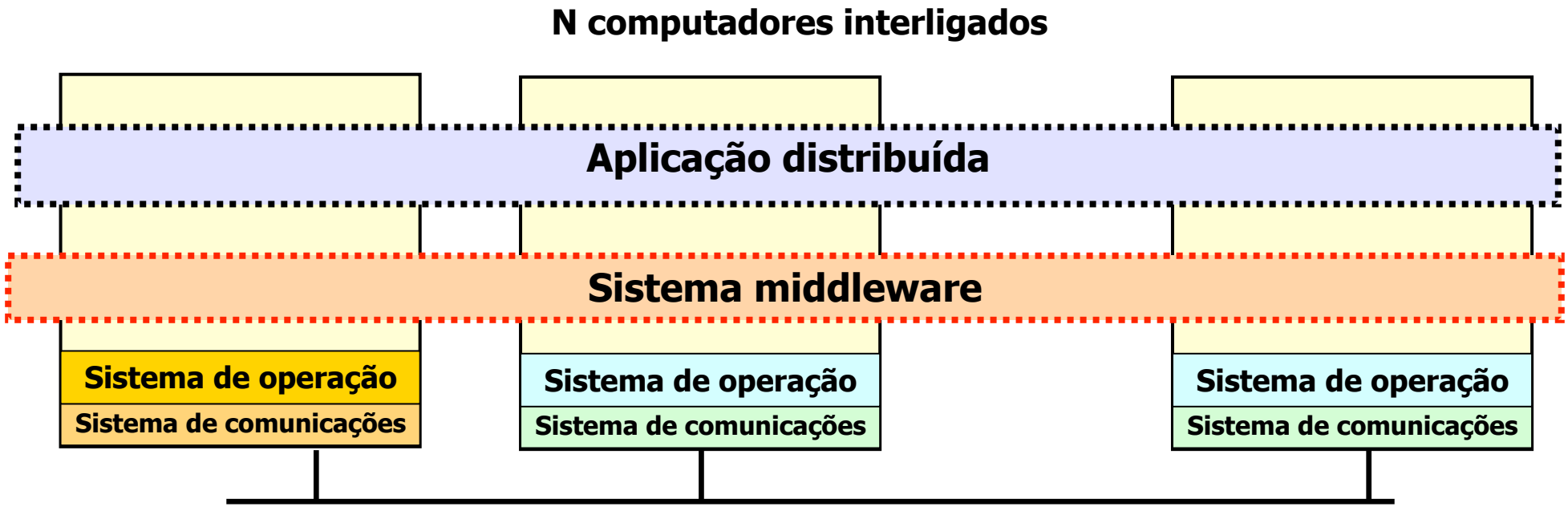
as camadas do sistema são atribuídas a processos/máquinas diferentes

Arquitetura distribuída

Especifica como se organizam e quais as interações entre os vários **componentes** de **um sistema distribuído**

Em todos os casos há implicações no desempenho, fiabilidade e segurança do sistema

CAMADAS DE SOFTWARE - MIDDLEWARE



O *middleware* fornece uma **interface homogénea** e **serviços mais complexos** que os disponibilizados pelo sistema de operação

Limitações: algumas funcionalidades apenas podem ser implementadas de forma eficaz com o conhecimento da semântica da aplicação, pelo que fornecer essa funcionalidade no sistema de *middleware* seria contraproducente (correção de erros, segurança, etc.)

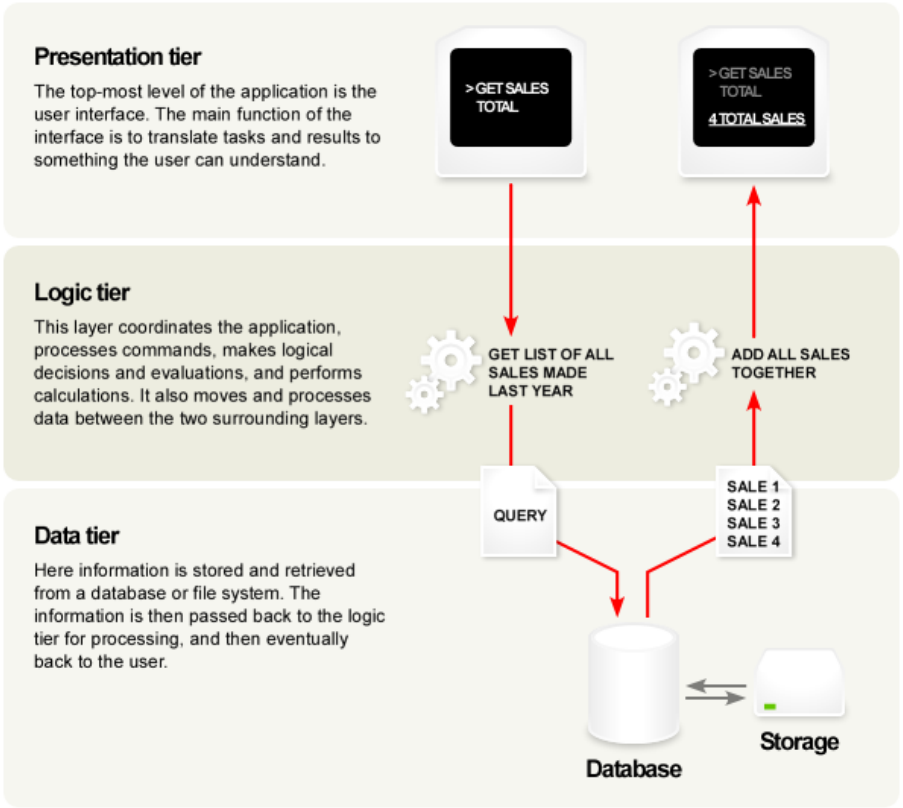
ARQUITECTURA EM CAMADAS: MODELO *THREE-TIER*

Em aplicações de acesso a sistemas de informação/web commerce, etc.

Arquitetura de três níveis (3-tier)

- Apresentação
- (Lógica) Aplicação
- (Armazenamento) Dados

Padrão arquitectural base típico: cliente/servidor



Arquitectura Distribuída

Arquitetura do sistema

Organização de um sistema (complexo) em componentes **mais simples** com funcionalidades/responsabilidades próprias

Arquitetura do sistema distribuído

Define os componentes, o que fazem, onde estão e como interagem entre si.

Terá implicações em diversas propriedades do sistema: desempenho, fiabilidade e segurança do sistema

ARQUITECTURA DISTRIBUÍDA

Arquitetura de um sistema distribuído pode (e deve) ser determinada por diversos fatores:

Requisitos funcionais:

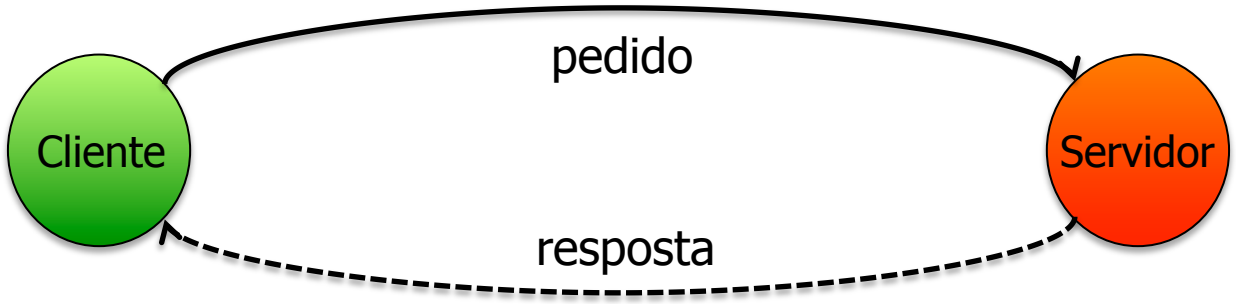
"tudo relacionado com o propósito direto (a função) do sistema"
ex. Lógica de negócio

Requisitos não funcionais:

- Desempenho (escalabilidade, latência), disponibilidade
- Custo (desenvolvimento, operação, manutenção)
- Segurança, confiabilidade

Arquitetura distribuída mais simples e muito comum?

CLIENTE/SERVIDOR



Sistema em que os processos podem ser divididos em dois tipos, de acordo com o seu modo de operação:

Cliente: programa que solicita pedidos a um processo servidor

Servidor: programa que executa operações solicitadas pelos clientes, enviando-lhes o respectivo resultado

CLIENTE/SERVIDOR: PROPRIEDADES

Arquitetura mais simples, muito comum e usada na prática...

Positivo

- Interação simples facilita implementação
- Segurança apenas tem de se concentrar no servidor

Negativo

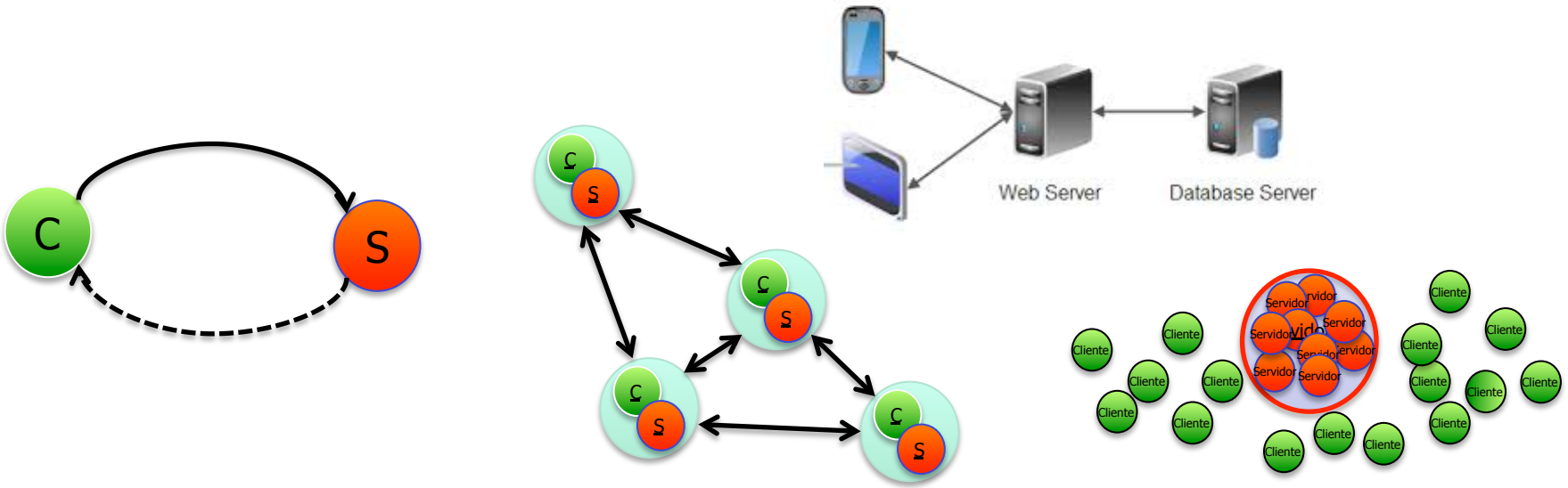
- Servidor é um ponto de falha único
- Não escala para além dum dado limite (servidor pode tornar-se ponto de contenção - *bottleneck*)

CLIENTE/SERVIDOR: COMPOSIÇÃO

Arquiteturas mais sofisticadas, com novas propriedades, podem ser obtidas por composição do modelo C/S base

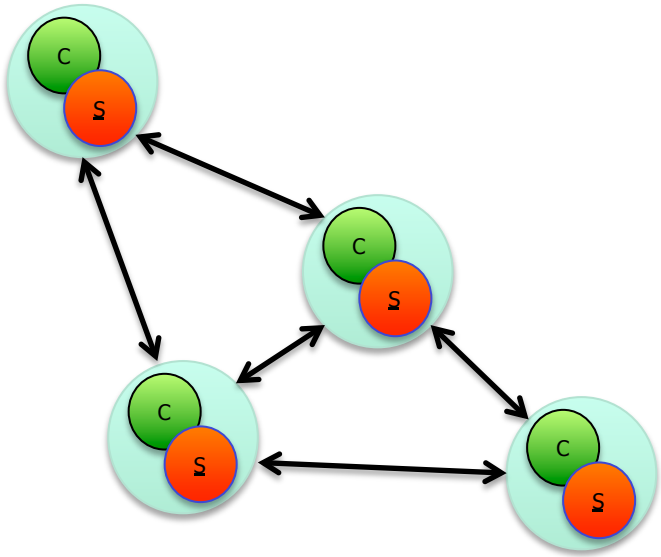
Servidor [particionado, replicado, geo-replicado],

P2P, 3-Tier, etc.



Modelo alternativo para lidar com limitações do modelo cliente/servidor

MODELO *PEER-TO-PEER (P2P)*



Todos os processos têm funcionalidades semelhantes

- Durante a sua operação podem assumir o papel de clientes e servidores do mesmo serviço em diferentes momentos

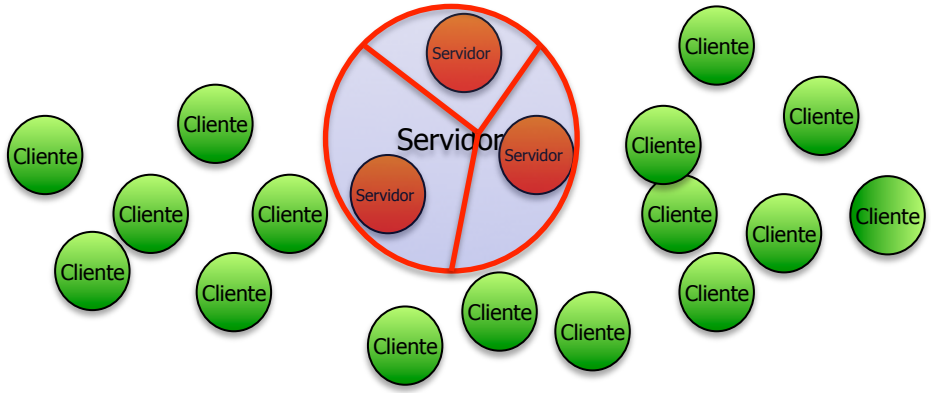
Exemplos: partilha de ficheiros, VoIP, edição colaborativa

MODELO *PEER-TO-PEER*: PROPRIEDADES

- Positivo
  - Não existe ponto único de falha
  - Melhor potencial de escalabilidade
  - Baixo custo de operação
- Negativo
  - Interação mais complexa (do que num sistema cliente/servidor) leva a implementações mais complexas
    - Operações de pesquisa são complexas
  - Maior número de computadores envolvidos pode colocar questões relativas a heterogeneidade e segurança
- Apropriado para ambientes em que todos os participantes querem cooperar para fornecer uma dado serviço
  - Capacidade agregada >> capacidade individual

Variantes do modelo cliente/servidor para lidar com limitações de escalabilidade e tolerância a falhas

VARIANTES DO MODELO CLIENTE/SERVIDOR: SERVIDOR



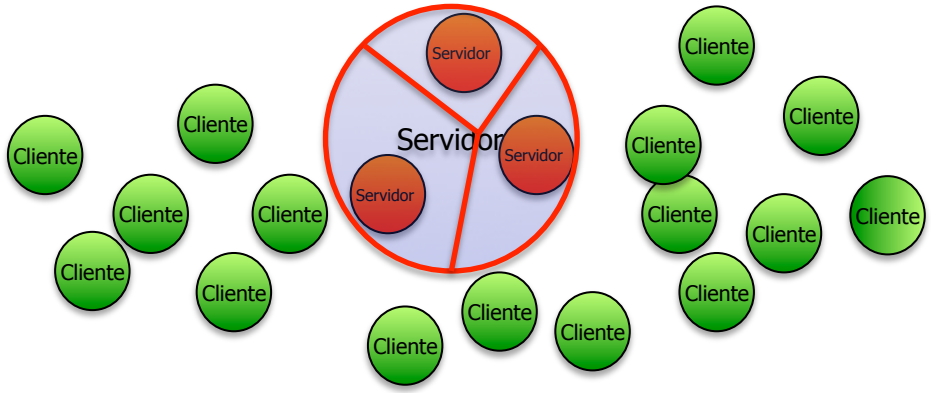
Cliente/servidor particionado

Existem vários servidores com a mesma interface, cada um capaz de responder **a uma parte** dos pedidos

Servidor redirige cliente para outro servidor (iterativo)
Servidor invoca pedido noutro servidor (recursivo)

exemplo: DNS

VARIANTES DO MODELO CLIENTE/SERVIDOR: SERVIDOR



Cliente/servidor particionado

Existem vários servidores com a mesma interface, cada um capaz de responder **a uma parte** dos pedidos

Servidor redirige cliente para outro servidor (iterativo)
Servidor invoca pedido noutro servidor (recursivo)

exemplo: DNS

VARIANTES DO MODELO CLIENTE/SERVIDOR: SERVIDOR

Cliente/servidor particionado

Positivo

- Permite distribuir a carga, melhorando o desempenho (potencialmente)
- Não existe um ponto de falha único

Negativo

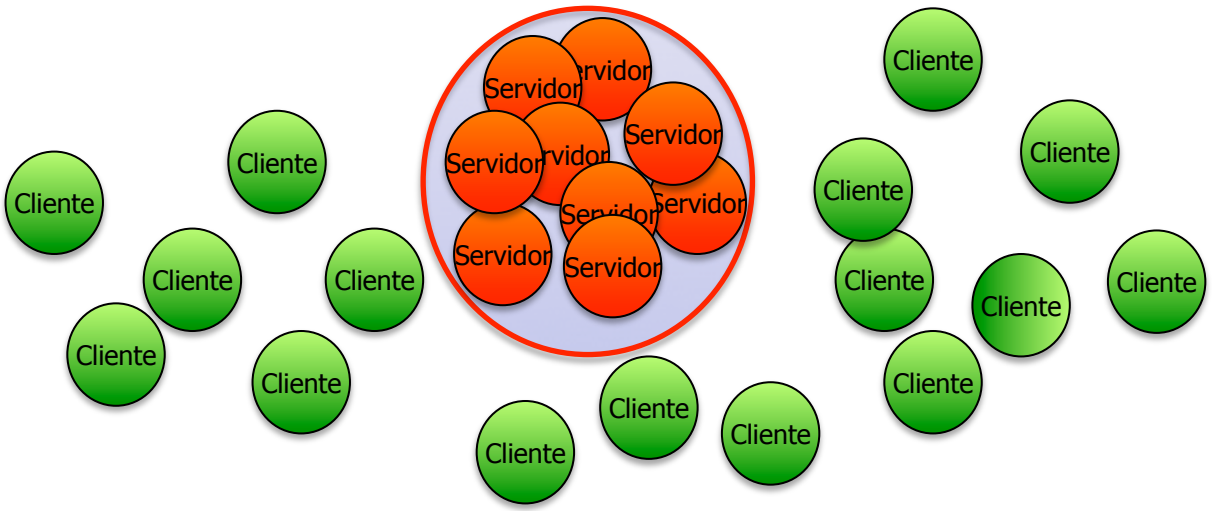
- Falha de um servidor impede acesso aos dados presentes nesse servidor
- Difícil de aplicar em alguns modelos de dados

Sendo: *w* : nº de escritas; *r* : nº de leituras; n : nº de partições
Cada partição recebe, em média: w / n + r / n pedidos

VARIANTES DO MODELO CLIENTE/SERVIDOR: SERVIDOR

Cliente/servidor replicado

Existem vários servidores idênticos (i.e. capazes de responder aos mesmo pedidos)



VARIANTES DO MODELO CLIENTE/SERVIDOR: SERVIDOR

Cliente/servidor replicado

Positivo

- Redundância - não existe um ponto de falha único
- Permite distribuir a carga, melhorando o desempenho (potencialmente)

Negativo

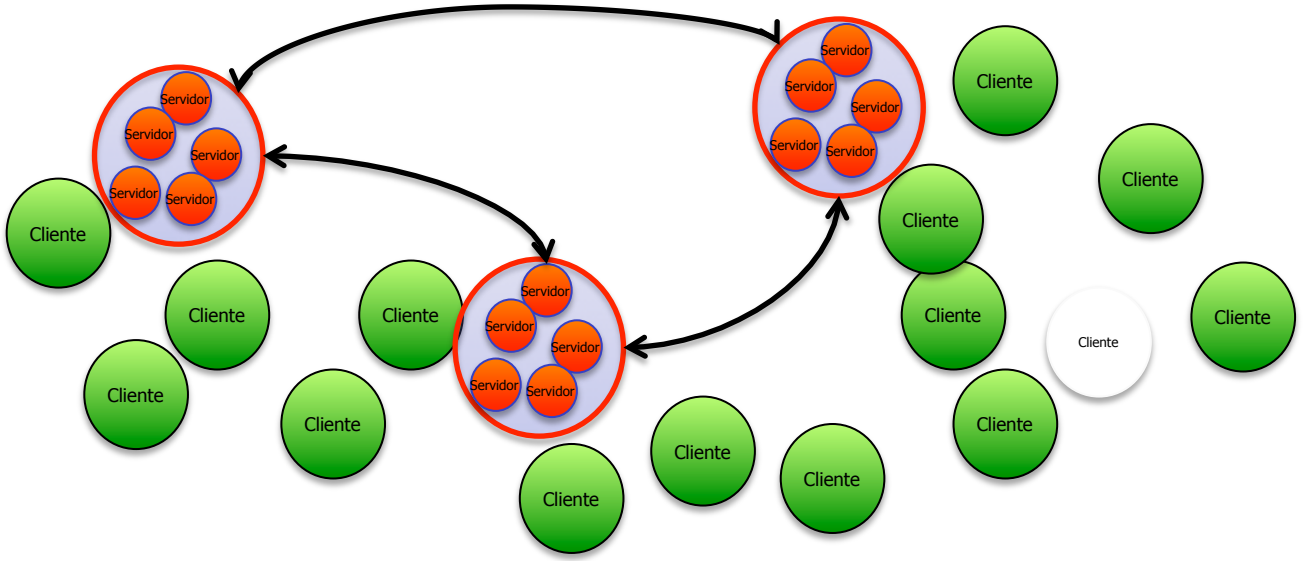
- Coordenação - manter estado do servidor coerente em todas as réplicas
- Recuperar da falha parcial de um servidor

Sendo: *w* : nº de escritas; *r* : nº de leituras; n : nº de réplicas
Se todas as réplicas receberem todos os pedidos de escrita, cada réplica recebe: w + r / n pedidos

VARIANTES DO MODELO CLIENTE/SERVIDOR: SERVIDOR

Cliente/servidor geo-replicado

Servidor replicado, com réplicas distribuídas geograficamente



VARIANTES DO MODELO CLIENTE/SERVIDOR: SERVIDOR

Cliente/servidor replicado

Positivo

Proximidade aos clientes melhora a qualidade de serviço (latência)

Redundância acrescida – as falhas das réplicas são (ainda) mais independentes

Negativo

Coordenação mais dispendiosa – maior separação física das réplicas traduz-se na utilização de canais com latência significativa

VARIANTES DO MODELO CLIENTE/SERVIDOR: CLIENTE

Cliente leve (*thin client*)/servidor

O cliente apenas inclui uma interface (gráfica) para executar operações no servidor (ex.: browser)

Positivo:

- Cliente pode ser muito simples

Negativo

- Maior peso no servidor
- Impacto na interatividade (latência)

VARIANTES DO MODELO CLIENTE/SERVIDOR: CLIENTE

Cliente completo (estendido)/servidor

O cliente executa localmente algumas operações que seriam executadas pelo servidor

Usado na Web em vários níveis: browsers fazem cache de páginas, imagens, etc; HTML 5.0 permite às aplicações Web guardar dados localmente – e.g. suporte offline no Google Docs, Gmail

VARIANTES DO MODELO CLIENTE/SERVIDOR: CLIENTE

Cliente completo (estendido)/servidor

Positivo:

- Permite funcionar *offline,* quando não é possível contactar o servidor (recorrendo a *caching*)
- Permite diminuir a carga do servidor e melhorar o desempenho e a interatividade

Negativo:

- Implementação do cliente mais complexa
- Necessário tratar da coerência dos dados entre o cliente e o servidor

SISTEMAS DISTRIBUÍDOS

Capítulo 2

Arquiteturas e Modelos de Sistemas Distribuídos

... NA AULA 3

Arquitetura de um sistema distribuído

Estabelece a sua organização em componentes mais simples com funcionalidades/responsabilidades próprias

Define: quais os componentes, o que fazem, onde estão e como interagem entre si

Desenhada com base num conjunto de requisitos (funcionais e não funcionais)

Obtida por composição do modelo C/S

P2P, C/S[particionado, replicado, georeplicado], etc.

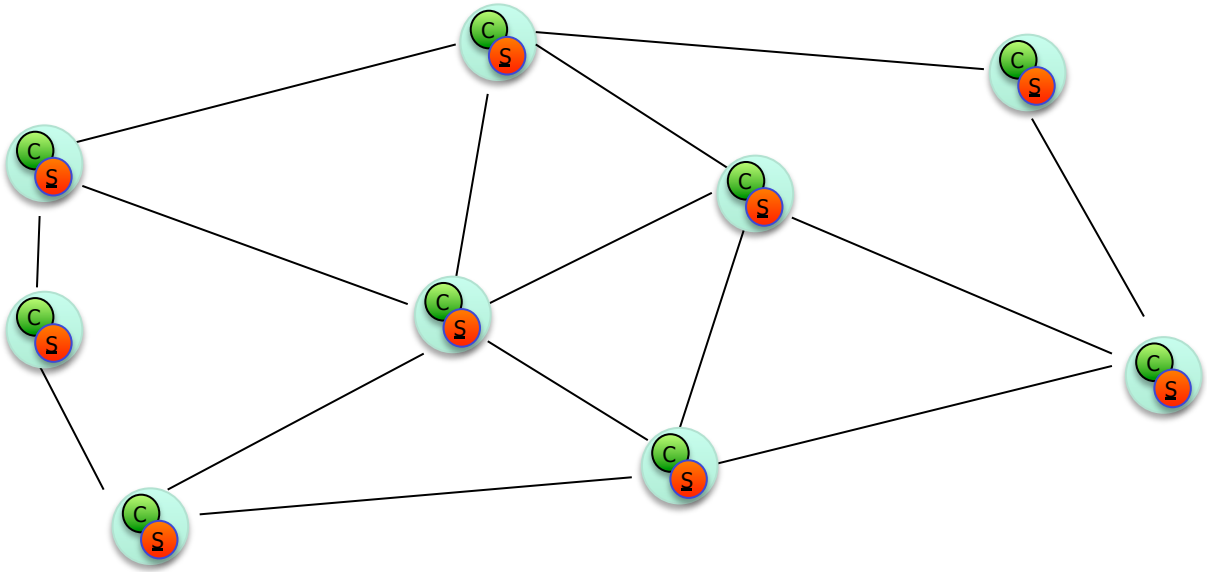
SISTEMAS DISTRIBUÍDOS

Capítulo 2 – aula 4

Arquiteturas e Modelos de Sistemas Distribuídos

Variantes do modelo P2P com diferentes propriedades

VARIANTES DO MODELO *PEER-TO-PEER*

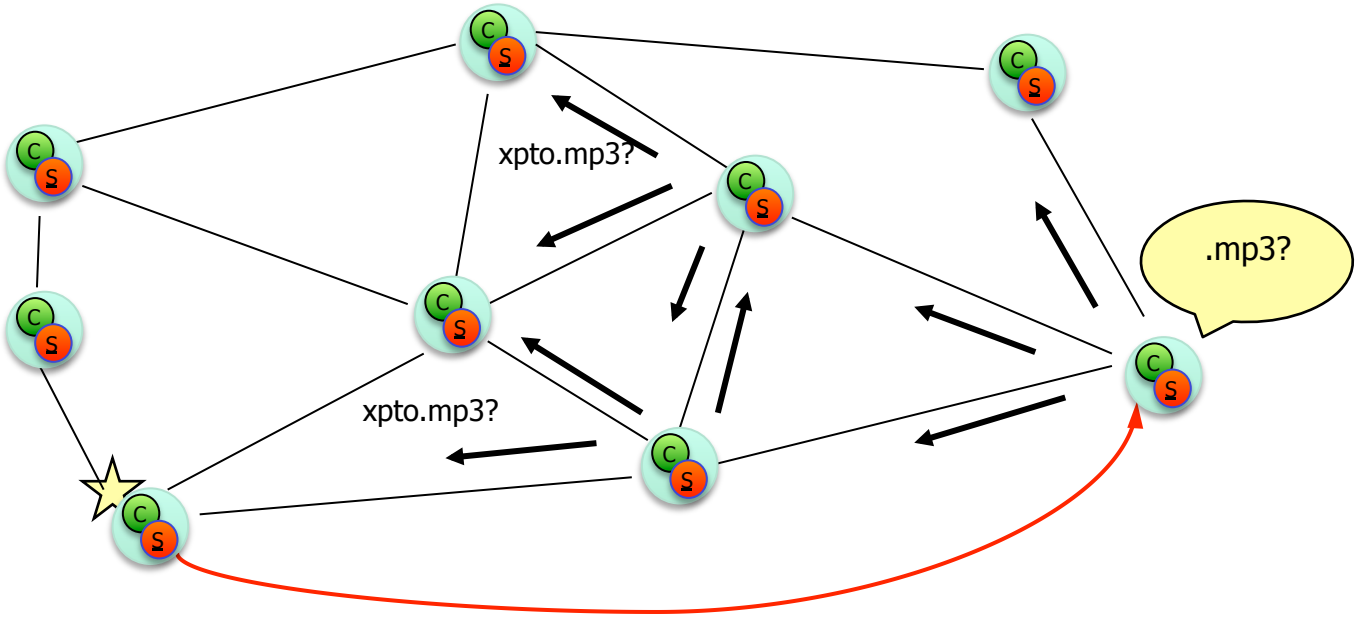


Sistema P2P não estruturado

As ligações entre os membros são formadas de forma **não-determinista**

E.g. quando se junta à rede, um membro escolhe para vizinhos um pequeno conjunto de contactos (os contactos podem variar durante a execução do sistema e de execução para execução)

VARIANTES DO MODELO *PEER-TO-PEER*



Sistema P2P não estruturado

Positivo:

Simplicidade de construir, robusto ao dinamismo da rede (churn – entrada e saída de nós participantes)

Negativo:

Produz topologias que podem ser difíceis de explorar de forma eficiente
eg., dificuldade em indexar informação / pesquisa pesada (frequentemente por inundação)

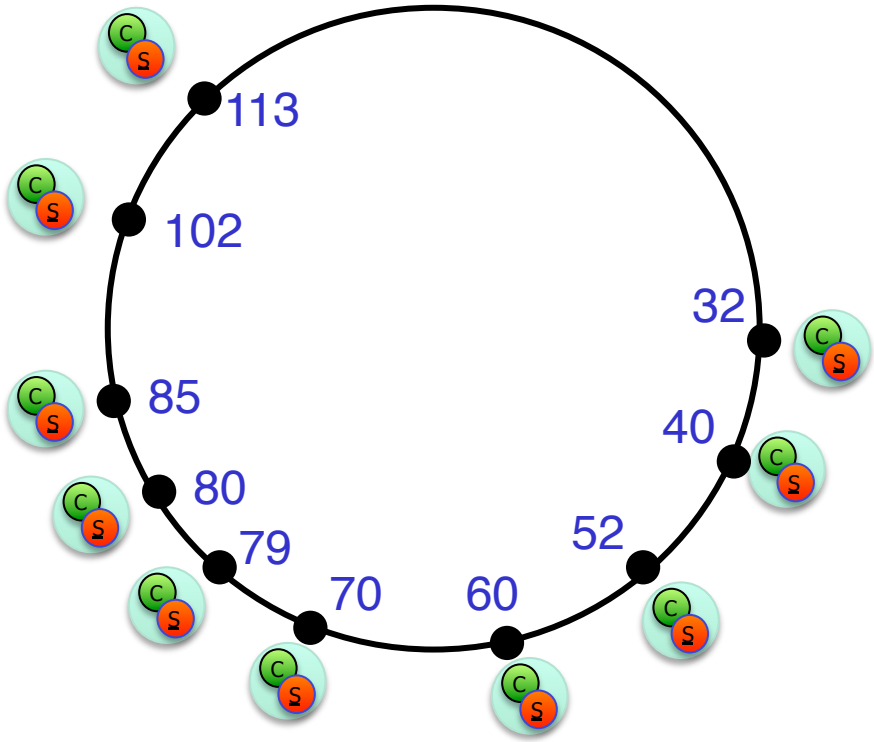
VARIANTES DO MODELO *PEER-TO-PEER*

Sistemas *P2P* estruturados

Os membros do sistema comunicam de acordo com uma organização definida de forma determinista com base num **endereço lógico**

A topologia é induzida por uma relação (matemática) entre os endereços lógicos.

Existem topologias para todos os gostos...



VARIANTES DO MODELO *PEER-TO-PEER*

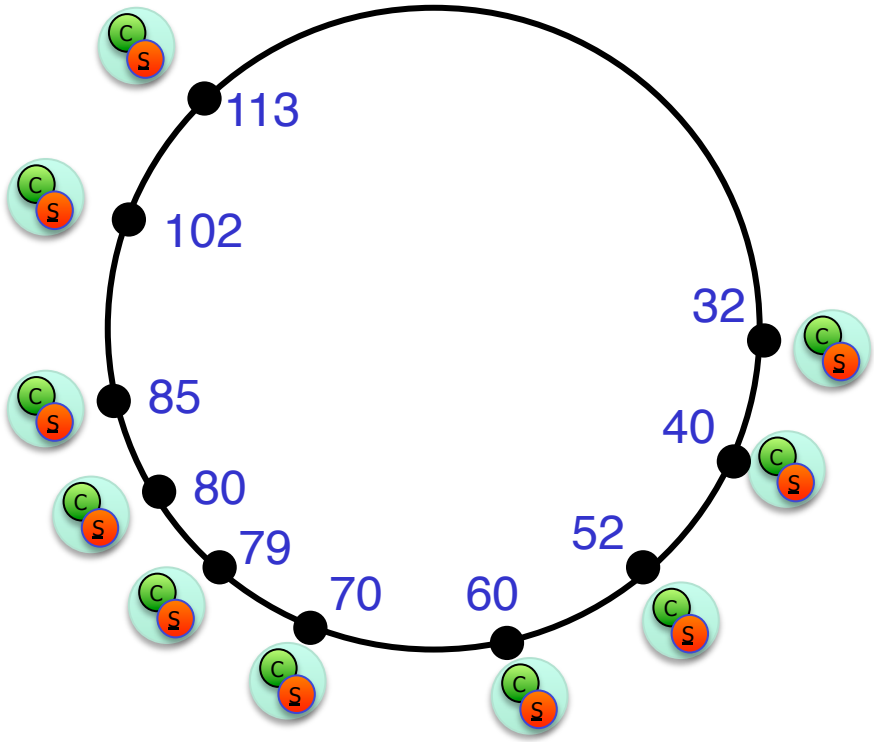
Sistemas *P2P* estruturados

Positivo

- Boa latência/escalabilidade - Conhecem-se topologias com encaminhamento/pesquisa com custo ***O(log n)***, com ***n*** nós no sistema, por exemplo.

Negativo

- Maior complexidade
- É necessário gastar recursos para manter a topologia correta face às entradas e saídas dos nós (*churn*)

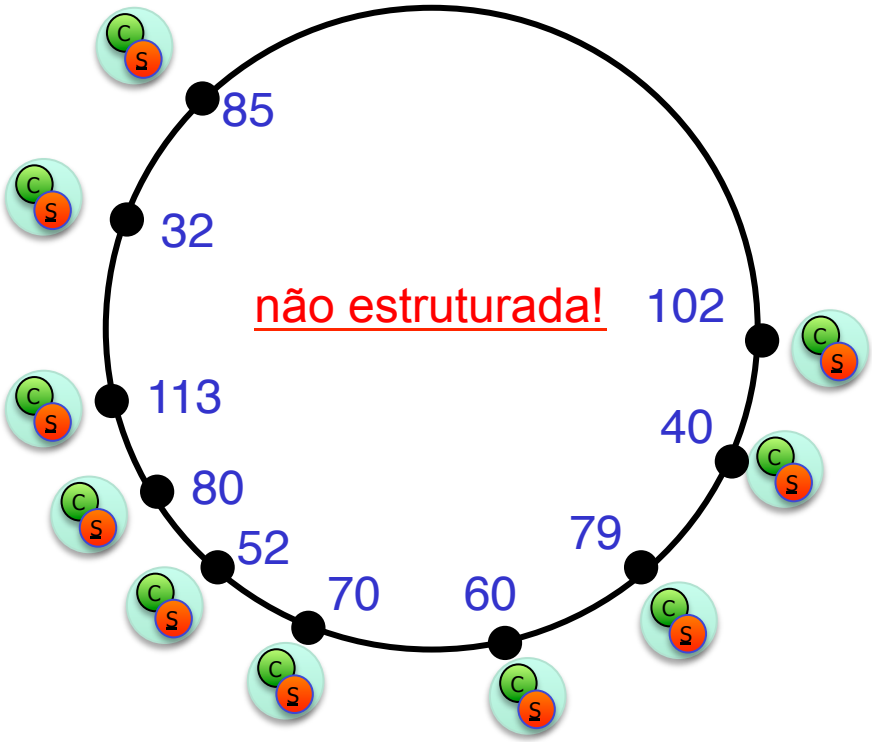


VARIANTES DO MODELO *PEER-TO-PEER*

Sistemas *P2P* estruturados

Importante: A topologia (as relações de vizinhança) dos nós têm que ser induzidas pelos identificadores lógicos

Os nós podem estar organizados num anel, por exemplo, e não formar um sistema P2P estruturado...



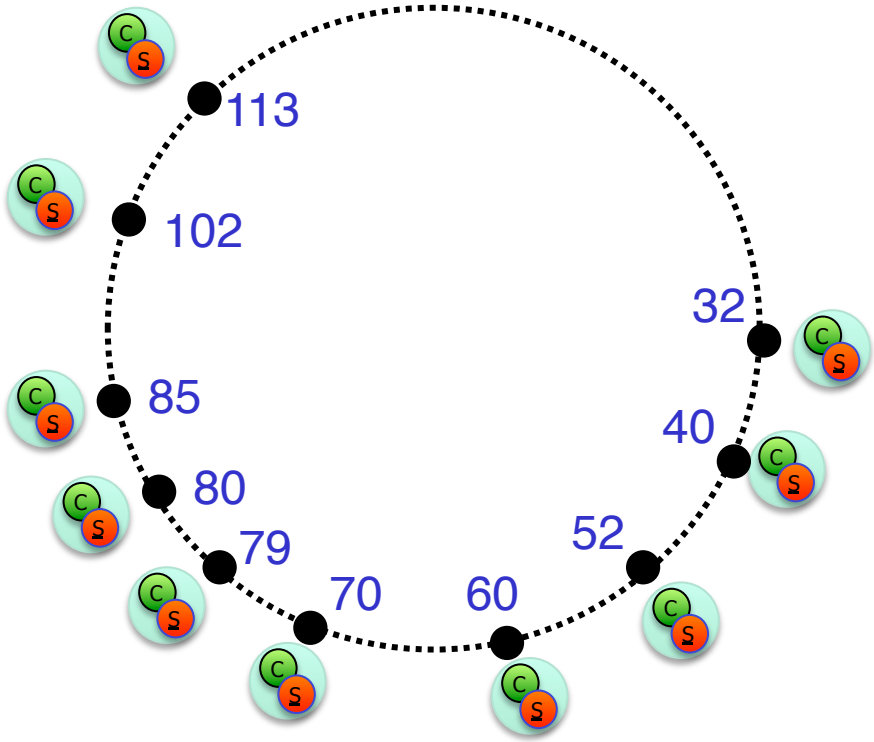
Analogia: árvore binária vs. árvore binária de pesquisa

quanto custa pesquisar um valor no primeiro caso vs. no segundo?

VARIANTES DO MODELO *PEER-TO-PEER*

Sistemas *P2P* estruturados

Encaminhamento e pesquisas por identificador O(log N)?

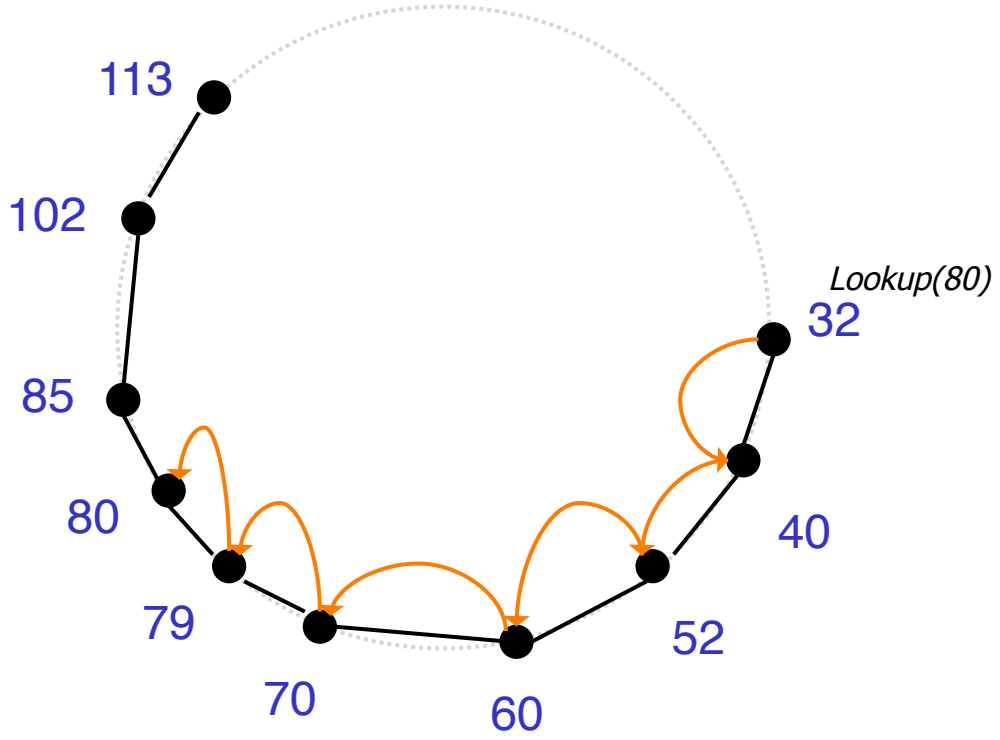


VARIANTES DO MODELO *PEER-TO-PEER*

Sistemas *P2P* estruturados

Encaminhamento e pesquisas por identificador O(log N)?

Seguir o sucessor não é solução; tem custo O(N)

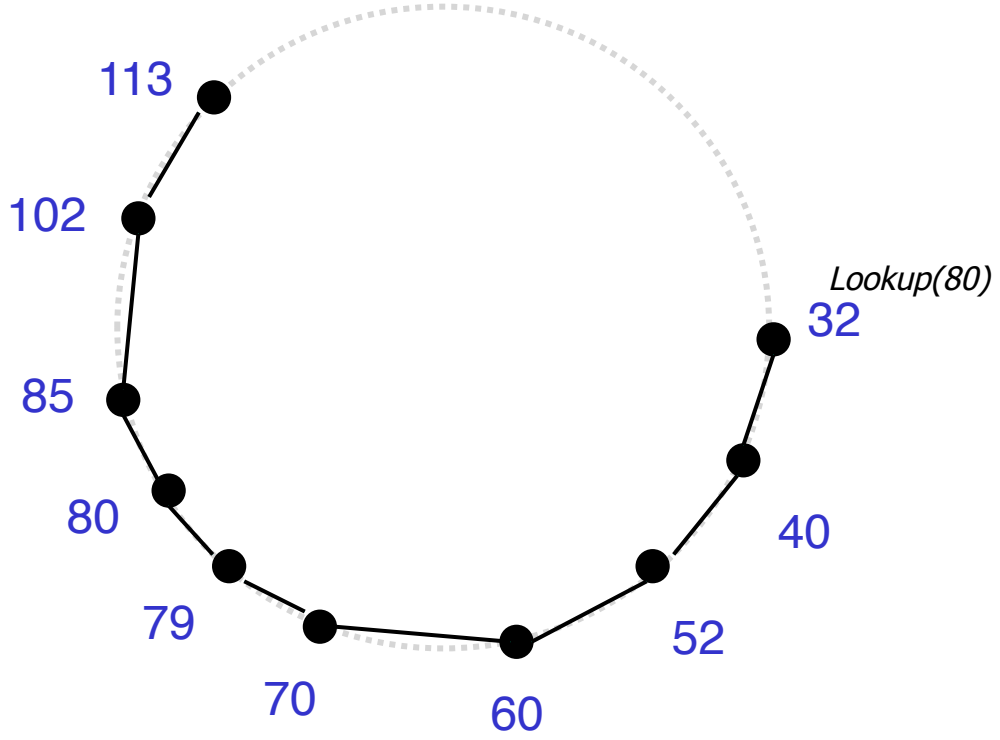


VARIANTES DO MODELO *PEER-TO-PEER*

Sistemas *P2P* estruturados

Encaminhamento e pesquisas por identificador O(log N)?

Em cada passo é necessário reduzir o espaço de pesquisa pela metade...



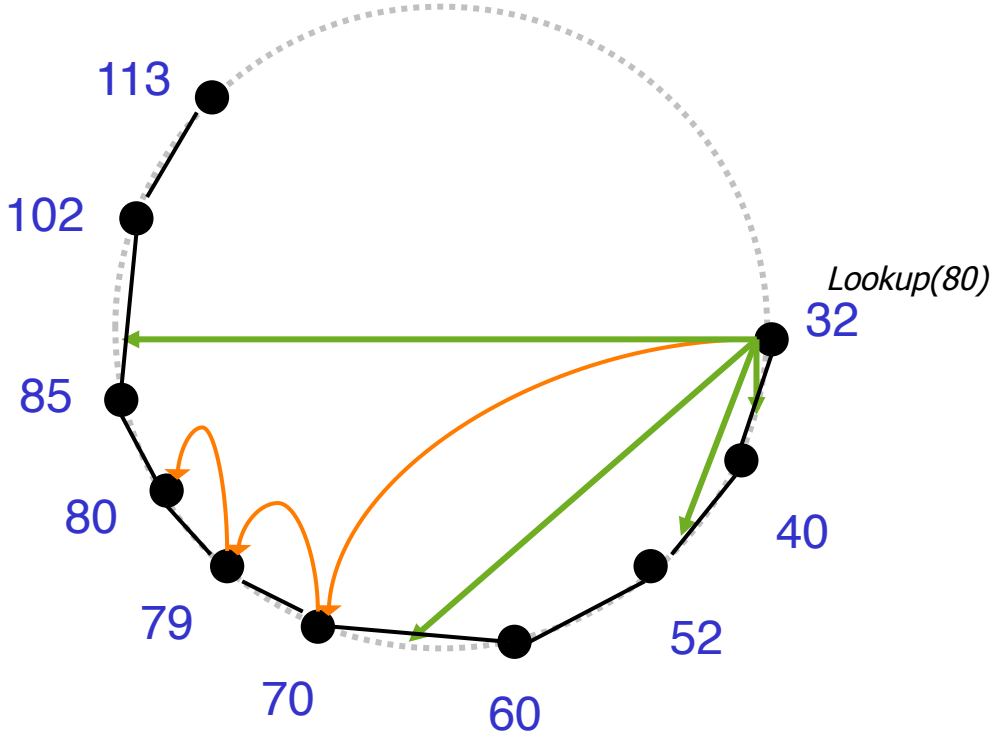
VARIANTES DO MODELO *PEER-TO-PEER*

Sistemas *P2P* estruturados

Encaminhamento e pesquisas por identificador O(log N)?

Em cada passo é necessário reduzir o espaço de pesquisa pela metade...

Ideia: cada nó liga-se a nós noutros pontos da topologia P2P e usa essas ligações como atalhos: ex: O(log N) vizinhos



VARIANTES DO MODELO *PEER-TO-PEER*

Sistemas *P2P* estruturados

<Simulação Sistema Chord, circa 2001>

Características:

Identificadores de 128 bits

Tabelas de vizinhança com O(log N) peers

Encaminhamento e Pesquisa por identificador em O(log N) passos

VARIANTES DO MODELO *PEER-TO-PEER*

Sistemas *P2P* estruturados

permitem implementar tabelas de dispersão distribuídas, i.e., DHTs (Distributed Hash Tables)

- *lookup*(chave) -> IP
- *get*(IP, chave) -> valor
- *put*(IP, chave, valor)

lookup(key) com custo O(log N), as outras operações têm custo constante (independente de N)

DHTs: CARACTERÍSTICAS

Identifica-se a informação usando uma função de *hash*

identificador = hash(info)

Cada nó fica responsável por um conjunto de identificadores (de forma determinista)

- e.g. cada nó usa um identificador único, gerado aleatoriamente, ficando responsável por manter a informação com os identificadores mais próximos do seu

Aspetos importantes...

- Pesquisa?
- Distribuição da informação?
- Replicação da informação?

DHTs: CARACTERÍSTICAS

Identifica-se a informação usando uma função de *hash*

identificador = hash(info)

Cada nó fica responsável por um conjunto de identificadores (de forma determinista)

- e.g. cada nó usa um identificador único, gerado aleatoriamente, ficando responsável por manter a informação com os identificadores mais próximos do seu

Aspetos importantes...

- Pesquisa? – pesquisa por identificador deve ser eficiente e suportada por todos os nós.
- Distribuição da informação? – deve ser uniforme por todos os nós
- Replicação da informação? – a entrada e saída contínua de nós obriga a manter a informação replicada nos nós certos e em número adequado.

Combinando os dois modelos (P2P e C/S) pode-se ter o melhor dos dois

COMBINAÇÃO DOS MODELOS C/S E P2P

Cliente + (Serviço) P2P

Um sistema P2P pode disponibilizar um serviço a outros processos (clientes) que não pertencem ao sistema P2P

Propriedades:

Permite limitar o número de processos que fazem parte do sistema P2P

COMBINAÇÃO DOS MODELOS C/S E P2P

Cliente/servidor + P2P

- Serviço disponibilizado por um sistema pode ser dividido em várias funcionalidades, sendo umas fornecidas por um sistema cliente/ servidor e outras por um sistema P2P.

- O sistema cliente/servidor pode, por exemplo, servir como serviço de diretório, suportar pesquisas eficientes, etc.
  - e.g. BitTorrent

Propriedades:

Permite combinar as vantagens de ambos os sistemas

Terá contras?

Ter também as desvantagens de ambos...

BitTorrent-versão original (simplificado)

.torrent contém informação sobre ficheiro a ser partilhado, incluindo: url do tracker, hash seguro de cada bloco do ficheiro (SHA-1)

Arquitetura BitTorrent:

C/S

Tracker: servidor centralizado HTTP que serve o ficheiro .torrent e a lista dos *peers* que estão a descarregar o ficheiro

Peers: como clientes, acedem ao *tracker* para obter a lista de outros peers a descarregar o ficheiro

P2P

Peers comunicam entre si para trocar blocos do ficheiro

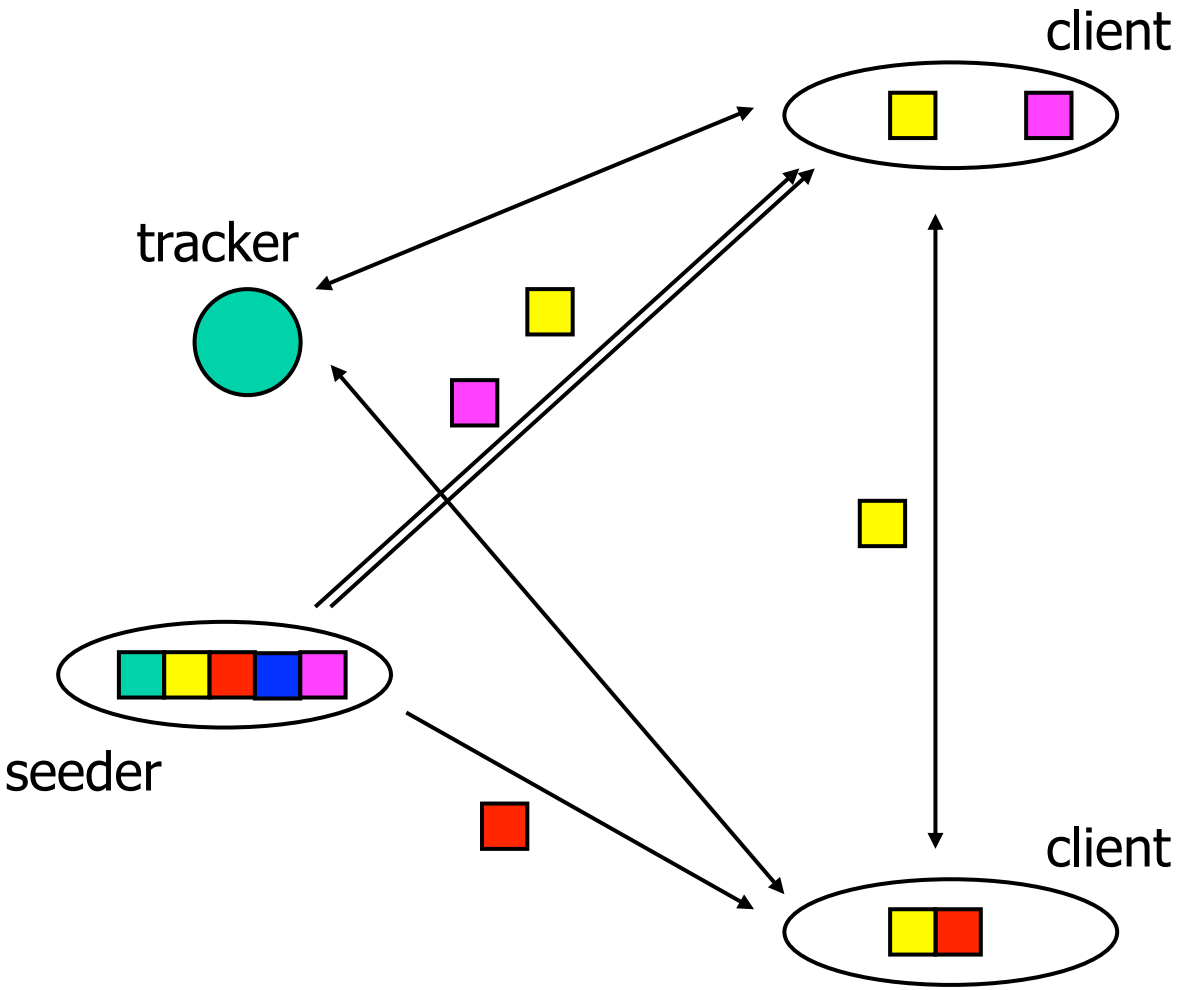
Toma-lá dá-cá (*tit-for-tat*): peer só envia bloco em troca de outro bloco

Peer envia alguns blocos a outros peers (sem contrapartida - aleatoriamente)

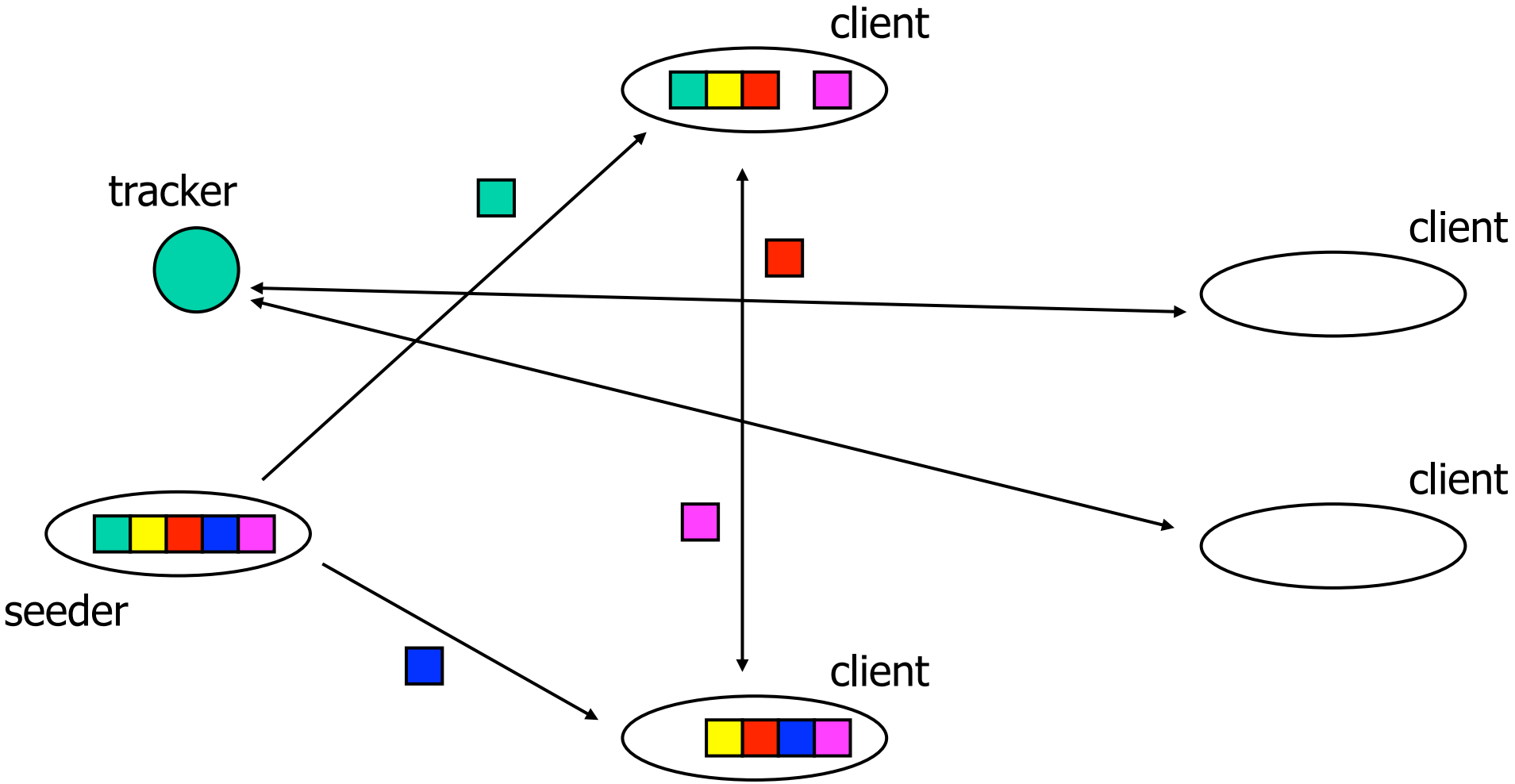
Peer descarrega blocos aleatoriamente

(Qual a motivação de cada uma destas propriedades?)

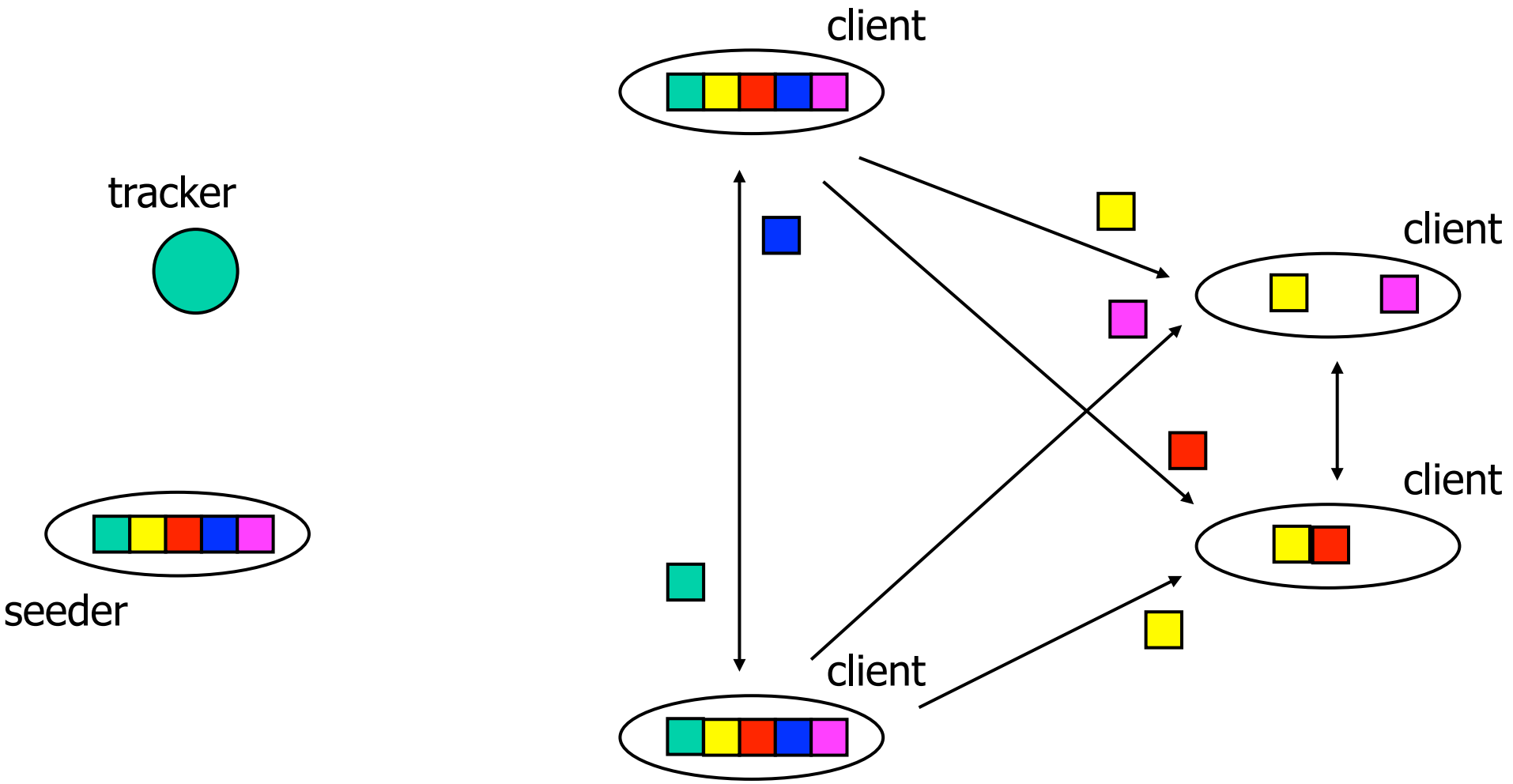
BitTorrent



BITTORRENT



BITTORRENT



BitTorrent DHT

BitTorrent tem a possibilidade de funcionar sem trackers

Neste caso, os peers formam uma DHT

- Cada nó gera um identificador único
- Cada *torrent* tem um identificador único

Informação sobre quais os nós que estão a partilhar/descarregar um ficheiro/torrent é armazenada nos nós com identificadores mais próximos ao identificador do torrent

Nota: baseado no sistema Kademlia (circa 2002)

eDonkey/eMule

Arquitectura:

- Cliente/servidor
  - Servidor: mantém informação sobre os ficheiros que cada peer tem
  - Peers: acedem ao servidor para fazer: (1) pesquisas; (2) obter informação sobre quais os peers que partilham cada ficheiro
- Peer-to-peer
  - Peers comunicam entre si para trocar blocos do ficheiro
    - Cada peer tem uma lista de outros peers que lhe fizeram pedidos. Esta lista é ordenada em função: do tempo de espera, dos créditos (função dos uploads recebidos)
    - Ordem dos pedidos tenta que aumente o número de fontes
  - Servidor pode ser envolvido na comunicação peer-to-peer para contornar clientes bloqueados por *firewalls*

AINDA OUTROS EXEMPLOS...

Sistemas peer-to-peer hierárquicos

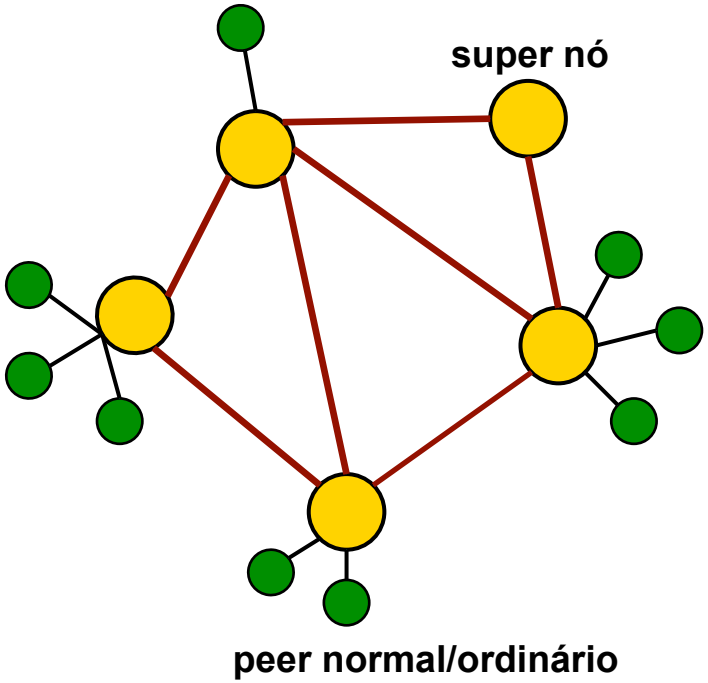
- Subconjunto de super-nós que se agrupam como num sistema peer-to-peer
- Nós ligam-se a um super-nó

...

KaZaA / Skype (original)

Arquitetura:

- **Super-nós** (SN): nós com maior capacidade tornam-se super-nós
  - Cada SN tem 60-150 nós ligados
  - Cada SN liga-se a outros 30-50 SN
- **Peers** (ordinary node – ON)
  - ON liga-se a um dos SNs que conhece (após se ter ligado, atualiza lista de SNs)



Pesquisa:

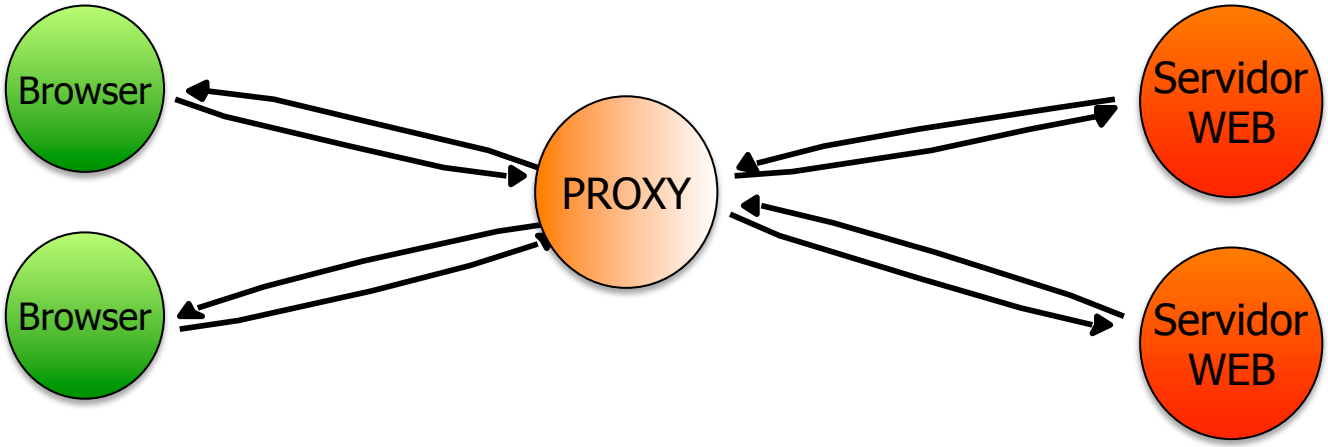
- ON contacta o seu SN
- SNs propagam pedidos entre si

Transferência de ficheiros:

- Directamente entre **peers**

Na prática, tendem a existir mais componentes num sistema distribuído

NOÇÃO DE PROXY DE UM SERVIÇO



Proxy de um serviço

- Processo que fornece um serviço recorrendo a um servidor (desse serviço) para executar o serviço

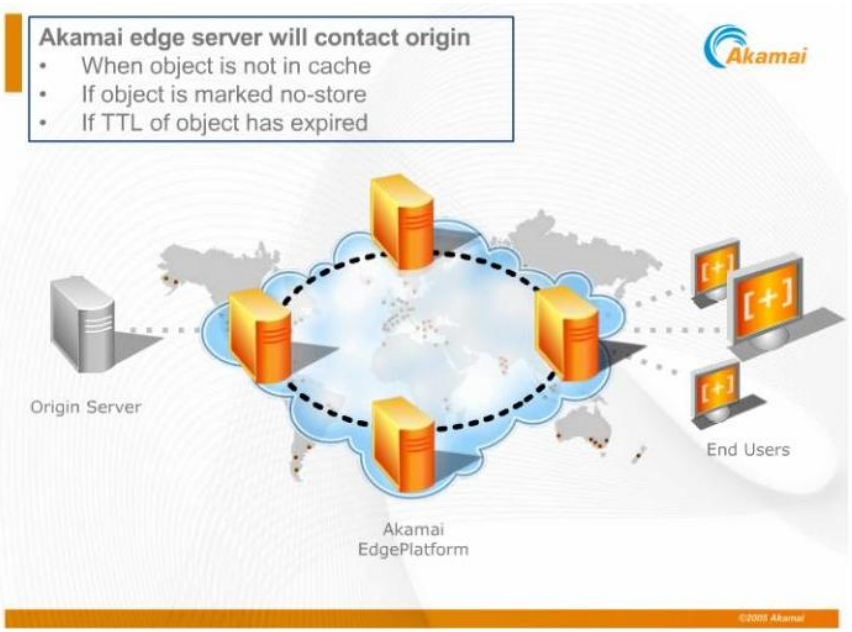
Utilizações possíveis

- Intermediário simples (apenas encaminha pedidos e respostas)
- Intermediário complexo (*gateway*)
  - Transformação dos pedidos
- Serviço adicional através do *caching* das respostas
  - Diminuição do tempo de resposta (latência inferior para o proxy)
  - Diminuição da carga do servidor
  - Mascarar falhas do servidor / desconexão

EDGE-SERVER

Sistemas edge-server

- Existem servidores colocados nos ISP para responder a pedidos
  - e.g., content-distribution networks (CDNs)
- Propriedades
  - Menor latência, filtragem, distribuição de carga, etc.



src: https://clickmotive.files.wordpress.com/2009/11/akamaiedgeplatform.jpg

SISTEMAS DISTRIBUÍDOS

Capítulo 2 – aula 5

Arquiteturas e Modelos de Sistemas Distribuídos

MODELOS FUNDAMENTAIS

Modelos fundamentais de um sistema distribuído

Permitem estabelecer quais as premissas que existentes a respeito de aspetos chave.

Permitem avaliar de forma objetiva as propriedades e garantias do sistema resultante.

Modelo de interação

Modelo de falhas

Modelo de segurança

MODELO DE INTERACÇÃO (1)

Modelo de interação

Num sistema distribuído, a coordenação que permite as várias componentes operar como um todo, requer comunicação por troca de mensagens.

- Modelo de iteração, carateriza as interações que ocorrem entre os componentes do sistema distribuído
  - define os padrões de comunicação: no tempo e espaço
  - o grau de acoplamento entre componentes: no tempo e espaço
  - as características e garantias oferecidas pelos canais de comunicação

Modelo de interacção (1)

Tipo de interação

Ativa

Processo solicita execução de operação noutro processo (de forma explícita)

Reativa

Evento no sistema desencadeia ação num processo

Indireta

Processos comunicam através de um espaço partilhado

Modelo reactivo (publish/subscribe – event-based system): motivação

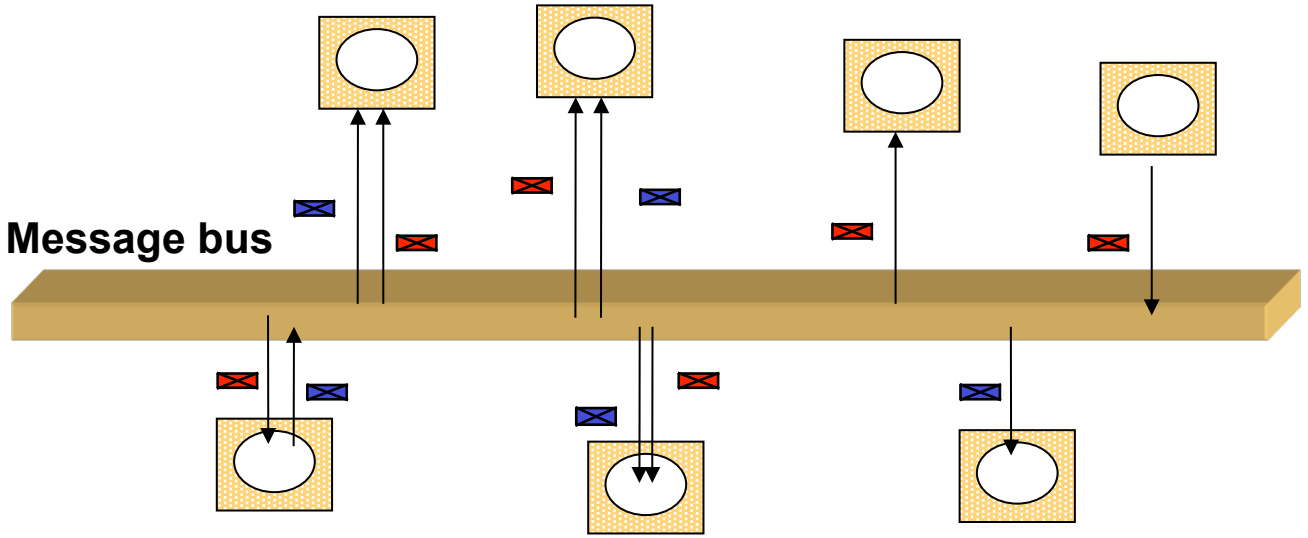
Como difundir informação sobre:

- Eventos dum jogo de futebol
- Cotações da bolsa

Propriedades

- Permite desacoplar o emissor, dos recetores(no espaço e no tempo)
- Eventos são produzidos de forma independente dos consumidores
- Todos os consumidores consomem todos os eventos
- Consumidores podem variar

Modelo reactivo (publish/subscribe – event-based system)



Modelo editor/assinante / "publish/subscribe" / pub/sub

- Processo subscritor/consumidor manifesta o interesse num conjunto de eventos (tópicos ou com base num filtro sobre o conteúdo)
- Processo produtor produz eventos

Sistema encarrega-se de propagar eventos produzidos para subscritores interessados

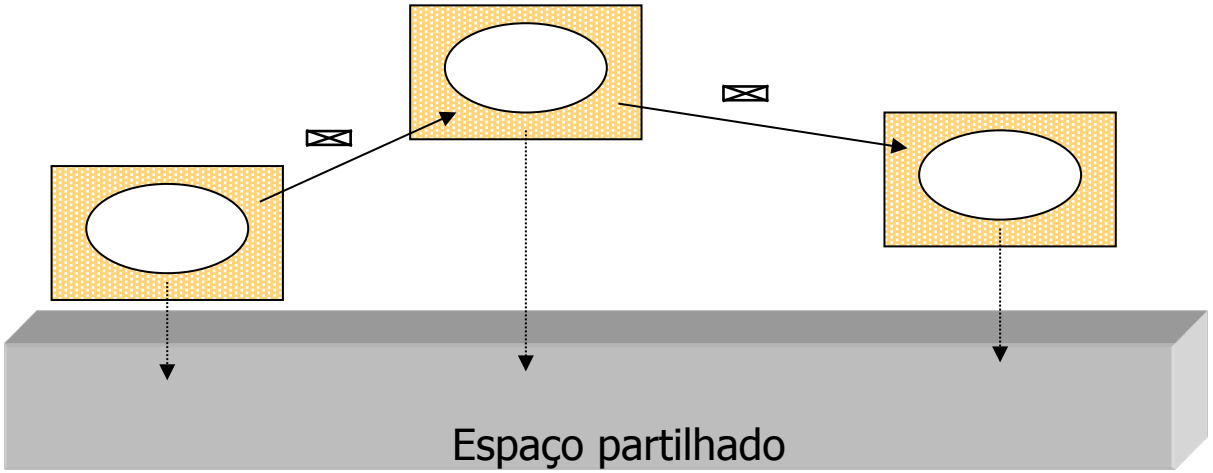
MODELO DE INTERAÇÃO INDIRETA

Como processar informação sobre sinais recebidos do espaço (SETI@home) ?

Propriedades

- Informação produzida independemente do seu consumo
- Cada unidade de informação é consumida por apenas um consumidor
- Consumidores podem variar e consomem informação quando o desejam

MODELO DE INTERAÇÃO INDIRETA



Processos comunicam e coordenam-se através dum espaço de "memória partilhada distribuída"

- Processos escrevem e leem dados no espaço de dados partilhada
- Espaço de dados partilhado pode ser base de dados, espaço de tuplos, sistema de message-queue, etc.

Modelo de interacção (2)

- Tipo de interação
  - Ativa
    - Processo solicita execução de operação noutro processo
  - Reativa
    - Evento no sistema desencadeia ação num processo
  - Indireta
    - Processos comunicam através de um espaço partilhado
- Conteúdo da interação
  - Informação/dados
  - Código

Conteúdo da interacção: dados

Processos trocam dados

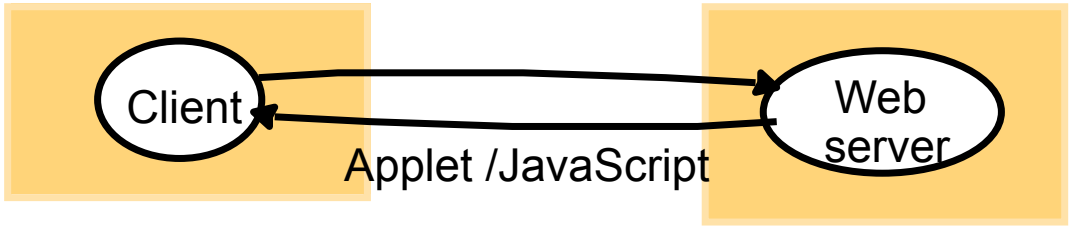
- Pedido (operação a invocar + parâmetros + inf. utilizador + ...)
- Resposta (resultado de operação + ...)
- Evento (num sistema de eventos)

Propriedades

- Parceiros devem conhecer formato das mensagens
- Parceiros devem saber processar mensagens (operações)

CONTEÚDO DA INTERACÇÃO: CÓDIGO MÓVEL – CLIENTE/SERVIDOR

a) client request results in the downloading of applet code



b) client interacts with the applet

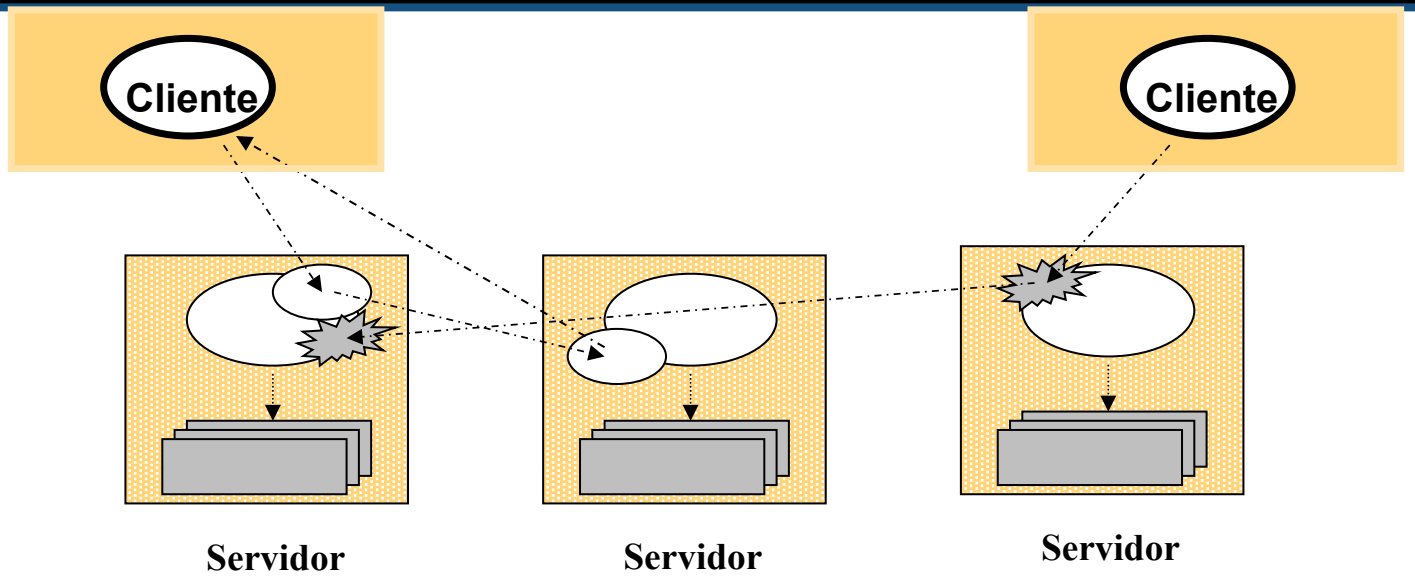
A execução do código no cliente pode:

- Melhorar o desempenho
- Ser usado para implementar funcionalidade adicional

Segurança

- Necessário proteger o cliente do código executado (sandboxing)

CONTEÚDO DA INTERACÇÃO: CÓDIGO MÓVEL – AGENTES



Ideia: agentes navegam entre servidores, executando cada parte no servidor em que é mais apropriado

- Exemplo: ?

Problemas

- Proteger informação do agente do ambiente de execução (impossível?)
- Desenhar sistema de forma que recursos usados sejam menores do que outra arquitetura

MODELO DE FALHAS

Num sistema distribuído, tanto os processos (e computadores) como os canais de comunicação podem falhar

Não é possível conceber componentes sem falhas, apenas se pode diminuir a probabilidade de as mesmas ocorrerem

- modelo de falhas consiste na definição rigorosa de quais os erros ou avarias, assim como das falhas que podem ter lugar nas diferentes componentes.
- o modelo de falhas abrange ainda a indicação rigorosa do comportamento global do sistema na presença dos diferentes tipos de falhas.

ALGUMAS DEFINIÇÕES

Fault tolerance - tolerância a falhas. Propriedade de um sistema distribuído que lhe permite **recuperar** da existência de falhas **sem introduzir comportamentos incorretos**.

Um sistema deste tipo pode **mascarar as falhas e continuar a operar**, ou **parar** e voltar a operar mais tarde, de forma coerente, após reparação da falha.

ALGUMAS DEFINIÇÕES

Availability – disponibilidade. Mede a fracção de tempo em que um serviço está a operar correctamente, isto é, de acordo com a sua especificação.

- Para um sistema ser altamente disponível (highly available) deve combinar
- um reduzido número de falhas com um curto período de recuperação das falhas (durante o qual não está disponível).

Reliability - fiabilidade. Mede o tempo desde um instante inicial até à primeira falha, i.e., o tempo que um sistema funciona correctamente sem falhas.

- Um sistema que falha com grande frequência e recupere rapidamente tem
- baixa fiabilidade, mas alta disponibilidade.

CLASSIFICAÇÃO DOS SISTEMAS

Classe	Disponibilidade	Indisponibilidade (por ano – máximo)	Exemplos
1	90,0%	36d 12h	Personal clients, experimental systems
2	99,0%	87h 36min	entry-level business systems
3	99,9%	8h 46min	top Internet Service Providers, mainstream business systems
4	99,99%	52min 33s	high-end business systems, data centers
5	99,999% (alta disponibilidade)	5min 15s	carrier-grade telephony; health systems; banking
6	99,9999%	31,5 seg	military defense systems

TIPOS DE FALHAS DOS COMPONENTES

Uma falha por omissão dá-se quando um processo ou um canal de comunicação falha a execução de uma acção que devia executar

Exemplo: uma mensagem que devia chegar não chegou, processo falha (crash)

Uma falha temporal dá-se quando um evento que se devia produzir num determinado período de tempo ocorreu mais tarde – normalmente em sistemas de tempo real

Exemplo: limite temporal para a propagação de uma mensagem, execução dum passo de computação, etc.

Uma falha arbitrária ou bizantina dá-se quando se produziu algo não previsto

Exemplo: chegou uma mensagem corrompida, um atacante produziu uma mensagem não esperada.

Para lidar com estas falhas é necessário garantir que elas não levam a que outros componentes passem a estados incorrectos

TIPOS DE ERRO/FALHA: DURAÇÃO

Permanentes: mantêm-se enquanto não forem reparadas (ex: computador avaria)

- Mais fáceis de detectar
- Mais difíceis de reparar

Temporárias: ocorrem durante um intervalo de tempo limitado, geralmente por influência externa

- Mais difíceis de reproduzir, detectar
- Mais fáceis de reparar

Erros transientes: ocorrem instantaneamente, ficam reparados imediatamente após terem ocorrido (ex.: perda de mensagem)

SEGURANÇA NUM SISTEMA DISTRIBUÍDO (MODELO DE SEGURANÇA)

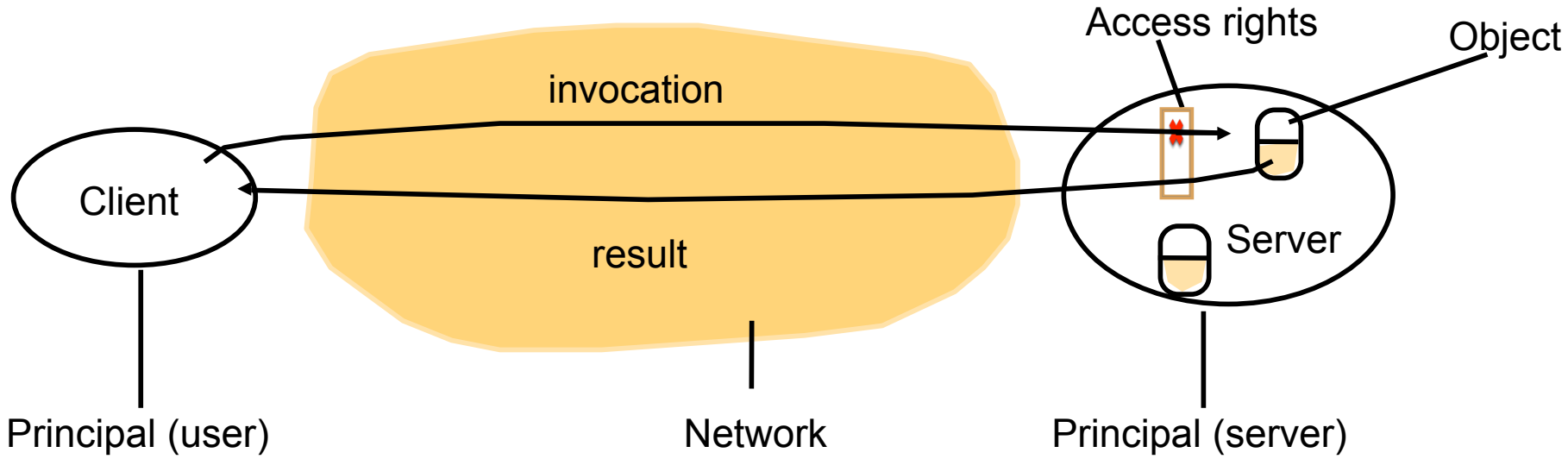
A informação gerida por um sistema distribuído tem valor para os utilizadores

O modelo de segurança consiste em definir quais as ameaças das quais um sistema se consegue defender

A segurança de um sistema não é uma garantia absoluta. Tratam-se de medidas para gerir o risco de o sistema ser comprometido.

Como? Aumentando o custo do ataque face ao valor associado ao sistema.

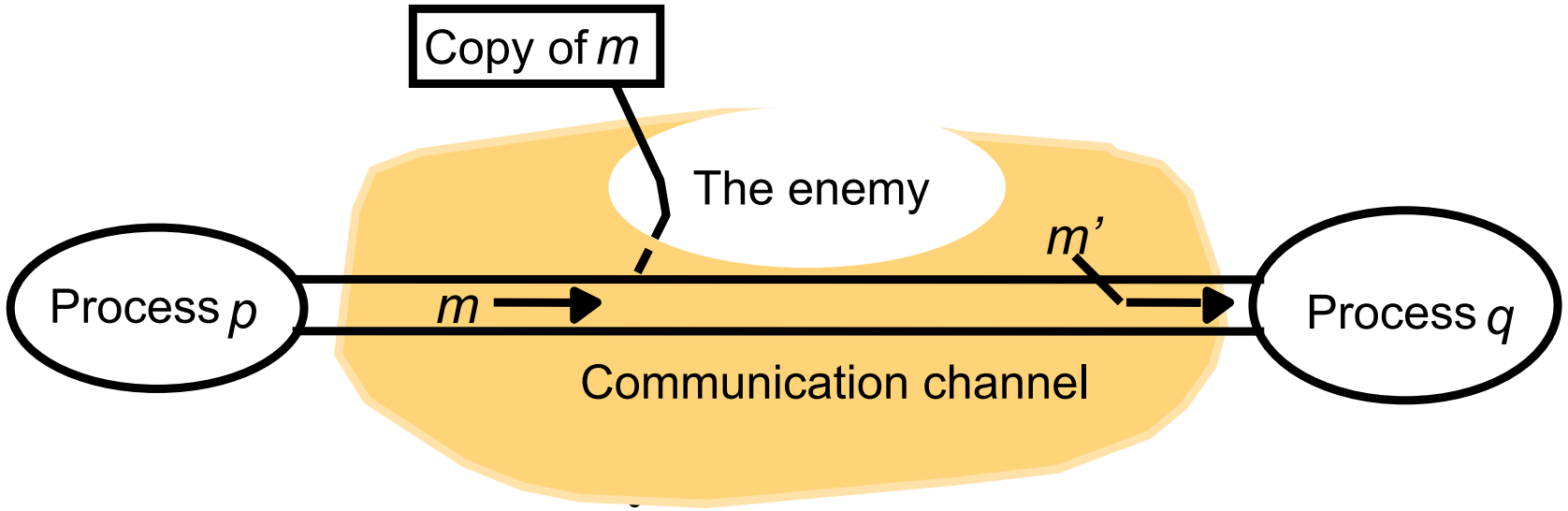
PRINCIPAIS, OBJECTOS, CONTROLO DE ACESSO E CANAIS



A segurança num sistema distribuído passa por:

- Autenticar os principais
- Verificar direitos de acesso aos objectos – controlo de acessos
- Utilizar canais seguros

AMEAÇAS BÁSICAS AOS CANAIS DE COMUNICAÇÃO (E CONSEQUENTEMENTE AOS PROCESSOS)

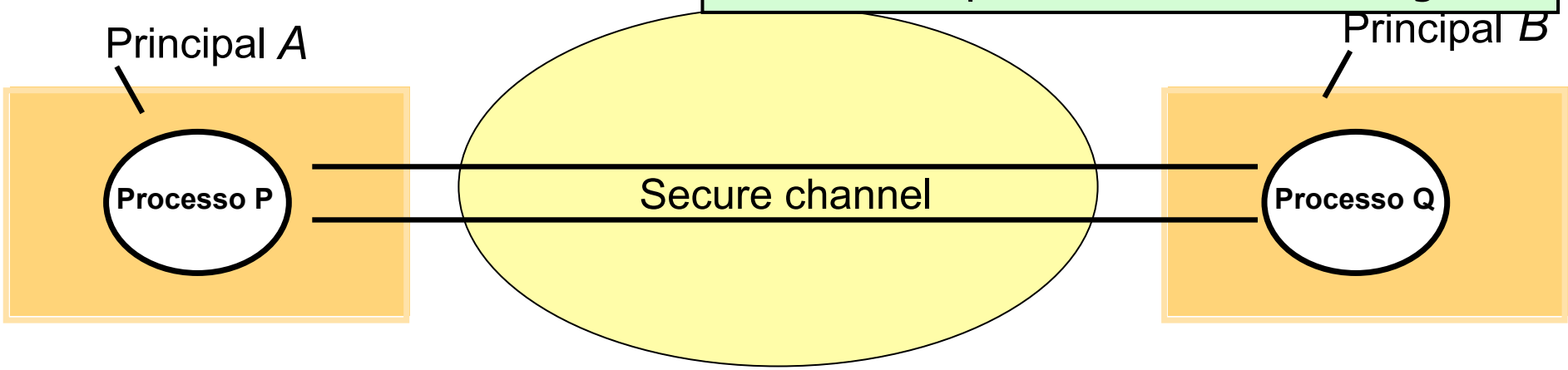


Para modelar ameaças de segurança, assume-se que o inimigo tem grande capacidade e pode:

- Enviar mensagens para qualquer processo
  - Forjar endereço das mensagens, fazendo-se passar por outro processo
- Ler, copiar, remover e reenviar mensagens que passam no canal
  - Impedir interacção, repetir interacção, etc.

CANAIS SEGUROS

Existem técnicas criptográficas (com partilha de segredos) que podem ser usadas na autenticação de parceiros e cifra de informação:
Permitem implementar um canal seguro



Num canal seguro os interlocutores (A e B) estão autenticados

O inimigo não pode copiar, alterar ou introduzir mensagens

O inimigo não pode fazer *replaying* de mensagens

O inimigo não pode reordenar as mensagens

OUTRAS AMEAÇAS

Denial of service: ataque em que o inimigo interfere (impede) actividade dos utilizadores autorizados

Código móvel: problemas de segurança para os processos que recebem e devem executar código móvel

Outras???

PARA SABER MAIS

George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair,
Distributed Systems – Concepts and Design,
Addison-Wesley, 5th Edition, 2011

Capítulo 2.

AULA ANTERIOR

Exemplos de arquiteturas P2P, Híbridas:

Chord, DHTs, BitTorrent

Edge-servers

Noção de Proxy de um serviço

MODELOS FUNDAMENTAIS

Modelos fundamentais de um sistema distribuído

Permitem avaliar de forma objetiva as propriedades e garantias do sistema distribuído.

Porque estabelecem as premissas na base do seu funcionamento.

Modelo de interação

Modelo de falhas

Modelo de segurança

MODELO DE INTERACÇÃO (1)

Modelo de interação

A coordenação das componentes de um sistema distribuído, a fim de operarem como um todo, requer comunicação por troca de mensagens.

O **modelo de interação** carateriza as interações que ocorrem entre os componentes do sistema distribuído, segundo:

os padrões de comunicação: no tempo e no espaço
o grau de acoplamento entre componentes: no tempo e no espaço
as características e as garantias oferecidas pelos canais de comunicação

Modelo de interacção (1)

Tipos de interação

Ativa

Processo solicita execução de operação noutro processo (de forma explícita)

Reativa

Evento no sistema desencadeia ação num processo (reage ao evento)

Indireta

Processos comunicam através de um espaço partilhado

(ie., observam os efeitos das outras componentes no espaço partilhado)

Interacão Ativa

Processo solicita execução de uma operação noutro processo (de forma explícita)

Na base de sistemas RPC (remote procedure call) e RMI (Remote Method Invocation)

Grande variedade de implementações, incluindo WebServices REST e SOAP, mas também Java RMI, .NET Remoting, CORBA, etc.

Estudado no Capítulo 4.

Modelo reactivo (publish/subscribe – event-based system): motivação

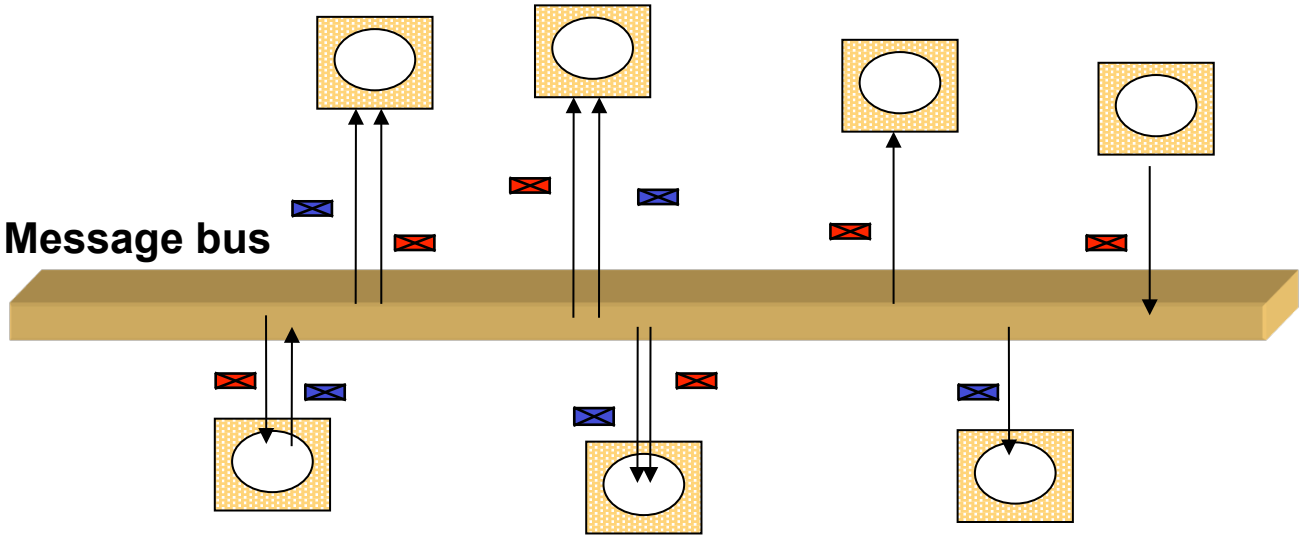
Como difundir informação sobre:

- Eventos dum jogo ou campeonato de futebol
- Cotações da bolsa

Propriedades

- O(s) emissor(es) e os receptores podem não coexistir no tempo; podem não se conhecer ou comunicar diretamente (desacoplados, no tempo e no espaço)
- Eventos são produzidos de forma independente dos consumidores
- Os consumidores consomem todos os eventos, ou só alguns (filtragem)
- Consumidores podem variar no tempo

MODELO REACTIVO (PUBLISH/SUBSCRIBE – EVENT-BASED SYSTEM)



Modelo editor/assinante ; "publish/subscribe" ; pub/sub

- Processo assinante manifesta o interesse num conjunto de eventos (tópicos ou com base num filtro sobre o conteúdo)
- Processo produtor produz eventos

O sistema encarrega-se de propagar eventos produzidos para os assinantes interessados

Modelo de interação indireta

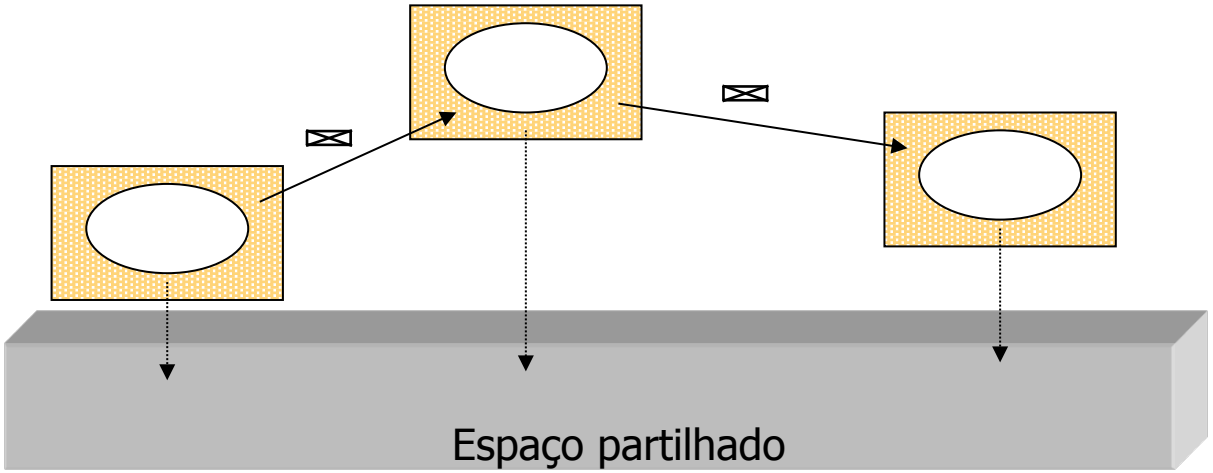
Como processar informação sobre sinais recebidos do espaço (SETI@home) ?

As antenas do programa SETI varrem os céus, escutando estrelas promissoras; os sinais recolhidos são processados offline. O processamento é feito de forma colaborativa.

Propriedades

- Informação produzida independemente do seu consumo
- Cada unidade de informação é consumida por apenas um consumidor
- Consumidores podem variar e consomem informação quando o desejam

MODELO DE INTERAÇÃO INDIRETA



Processos comunicam e coordenam-se através dum espaço de "memória partilhada distribuída"

- Processos escrevem e leem dados no espaço de dados partilhado
- Espaço de dados partilhado pode ser uma base de dados, um espaço de tuplos, ou um sistema de message-queue, etc.

Modelo de interacção (2)

- Tipo de interação
  - Ativa
    - Processo solicita execução de operação noutro processo
  - Reativa
    - Evento no sistema desencadeia ação num processo
  - Indireta
    - Processos comunicam através de um espaço partilhado
- Conteúdo da interação
  - Informação/dados
  - Código

Conteúdo da interacção: dados

Processos trocam dados

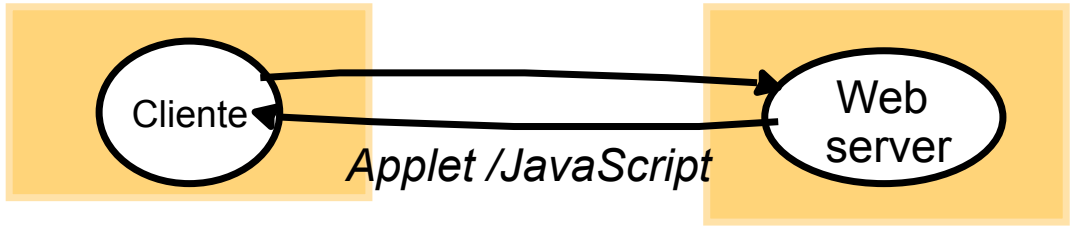
- Pedido (operação a invocar + parâmetros + inf. utilizador + ...)
- Resposta (resultado de operação + ...)
- Evento (num sistema de eventos)

Propriedades

- Parceiros devem conhecer formato das mensagens
- Parceiros devem saber processar mensagens (operações)

CONTEÚDO DA INTERACÇÃO: CÓDIGO MÓVEL – CLIENTE/SERVIDOR

a) Pedido do cliente resulta na transferência de código móvel



b) O cliente executa e interage com o código móvel

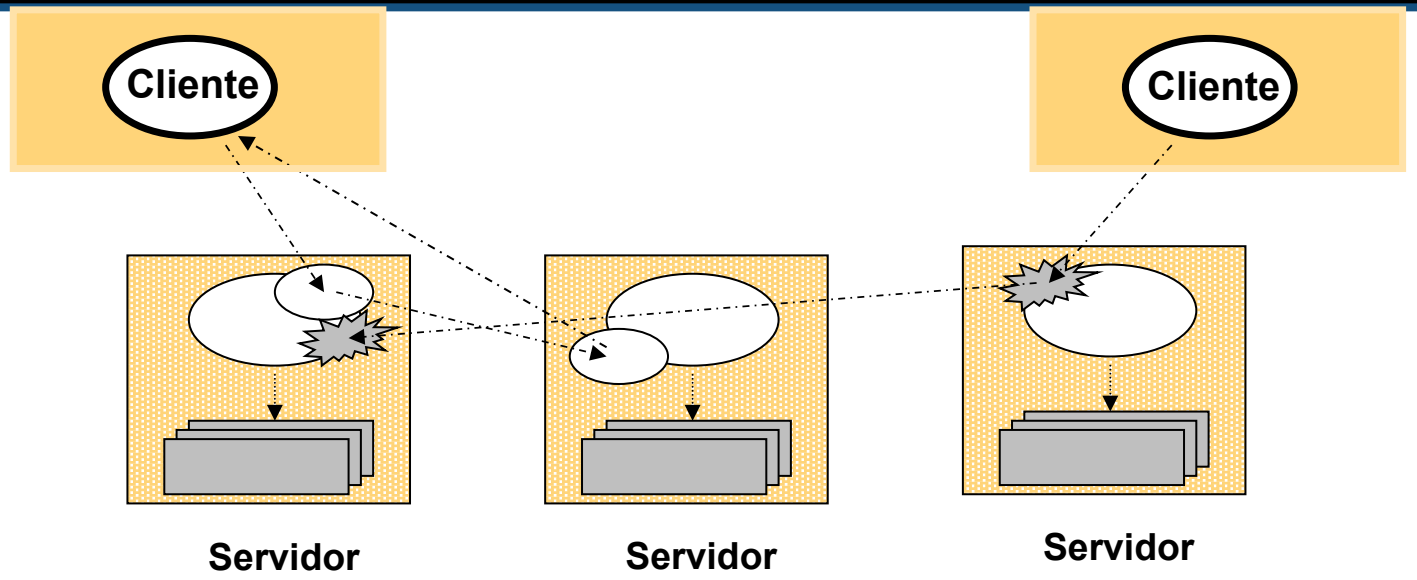
A execução do código no cliente pode:

- Melhorar o desempenho (reduzindo a latência)
- Ser usado para implementar funcionalidade adicional (e.g. usar sensores de um dispositivo móvel)

Segurança

- Fundamental proteger o cliente do código executado (*sandboxing*)

Conteúdo da interacção: código móvel – agentes



Ideia: agentes navegam entre servidores, executando cada parte da funcionalidade no servidor em que é mais apropriado

- Exemplo: ?

Problemas

- Proteger informação do agente do ambiente de execução (impossível?)
- Eficiência: poupar recursos face a uma arquitetura tradicional (pedido/ resposta)

MODELO DE FALHAS

Num sistema distribuído, tanto os processos (e computadores), como os canais de comunicação podem falhar

Não é possível conceber componentes sem falhas, apenas se pode diminuir a probabilidade de as mesmas ocorrerem

- modelo de falhas consiste na definição rigorosa de quais os erros ou avarias, assim como das falhas que podem ter lugar nas diferentes componentes.
- o modelo de falhas abrange ainda a indicação rigorosa do comportamento global do sistema na presença dos diferentes tipos de falhas.

ALGUMAS DEFINIÇÕES

Fault tolerance - tolerância a falhas**.** Propriedade de um sistema distribuído que lhe permite **recuperar** da existência de falhas **sem introduzir comportamentos incorretos**.

Um sistema deste tipo pode **mascarar as falhas e continuar a operar**, ou

parar e voltar a operar mais tarde, de forma coerente, após reparação da falha.

ALGUMAS DEFINIÇÕES

Availability – disponibilidade. Mede a fracção de tempo em que um serviço está a operar correctamente, isto é, de acordo com a sua especificação.

Para um ser altamente disponível (highly available) é necessário combinar um **reduzido número** de falhas, com um **curto período de recuperação** das falhas (durante o qual não está disponível)

Reliability - fiabilidade. Mede o tempo desde um instante inicial até à primeira falha, i.e., o tempo que um sistema funciona correctamente sem falhas.

Um sistema que falha com grande frequência e recupere rapidamente tem baixa fiabilidade, mas alta disponibilidade.

CLASSIFICAÇÃO DOS SISTEMAS

Classe	Disponibilidade	Indisponibilidade (max)		Exemplos
		Ano	Dia	
1	90,0% (one nine)	36d 12h	2,4h	Personal clients, experimental systems
2	99,0%	87h 36min	14,4 m	entry-level business systems
3	99,9%	8h 46min	1,44 m	top Internet Service Providers, mainstream business systems
4	99,99%	52min 33s	8,7 s	high-end business systems, data centers
5	99,999% (5 nines) (alta disponibilidade)	5min 15s	864 ms	carrier-grade telephony; health systems; banking
6	99,9999%	31,5 seg	86,4 ms	military defense systems

TIPOS DE FALHAS DOS COMPONENTES

Uma falha por omissão dá-se quando um processo ou um canal de comunicação falha a execução de uma ação que devia executar

Exemplo: uma mensagem que devia chegar não chegou, processo falha (crash)

Uma falha temporal dá-se quando um evento que se devia produzir num determinado período de tempo ocorreu mais tarde – normalmente em sistemas de tempo real

Exemplo: limite temporal para a propagação de uma mensagem, execução dum passo de computação, etc.

Uma falha arbitrária ou bizantina dá-se quando se produziu algo não previsto

Exemplo: chegou uma mensagem corrompida, um atacante produziu uma mensagem não esperada.

Para lidar com estas falhas é necessário evitar que outros componentes também passem a estados incorretos

TIPOS DE ERRO/FALHA: DURAÇÃO

Permanentes: mantêm-se enquanto não forem reparadas (ex: computador avaria)

- Mais fáceis de detectar
- Mais difíceis de reparar

Temporárias: ocorrem durante um intervalo de tempo limitado, geralmente por influência externa (ex., perda de conectividade)

- Mais difíceis de reproduzir, detectar
- Mais fáceis de reparar

Erros transientes: ocorrem instantaneamente, ficam reparados imediatamente após terem ocorrido (ex.: perda de mensagem)

SEGURANÇA NUM SISTEMA DISTRIBUÍDO (MODELO DE SEGURANÇA)

A informação gerida por um sistema distribuído e a sua funcionalidade têm **valor** para os utilizadores

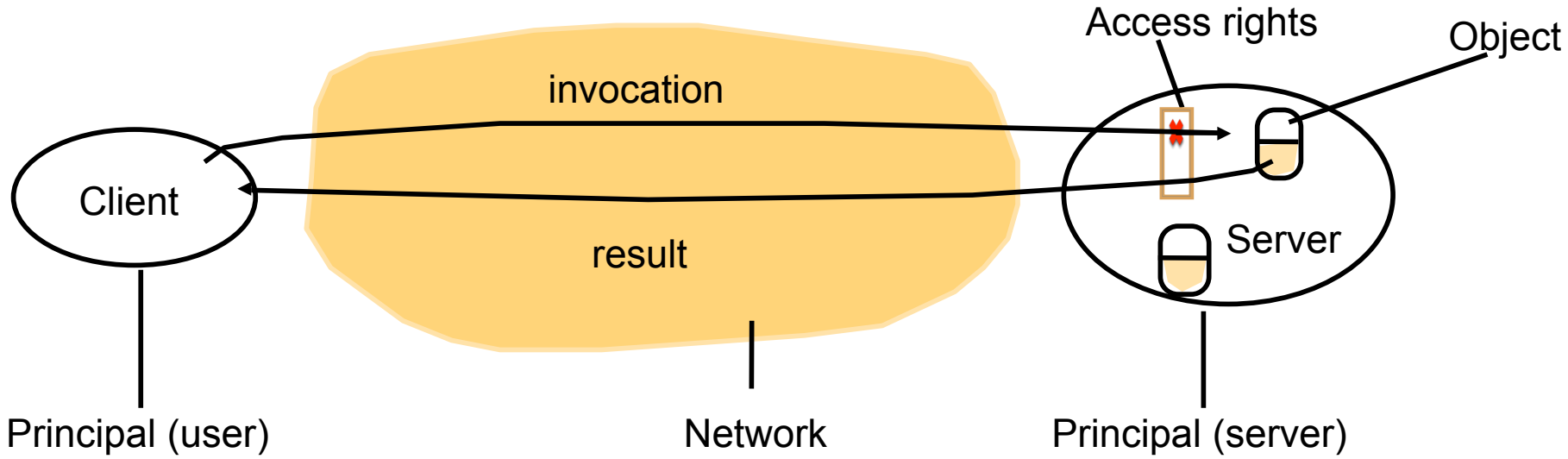
O modelo de segurança consiste em definir quais as ameaças das quais um sistema se consegue defender

A segurança de um sistema não é uma garantia absoluta. Tratam-se de medidas para gerir o risco de o sistema ser comprometido.

Como?

Aumentando **o custo do ataque** face ao valor associado ao sistema.

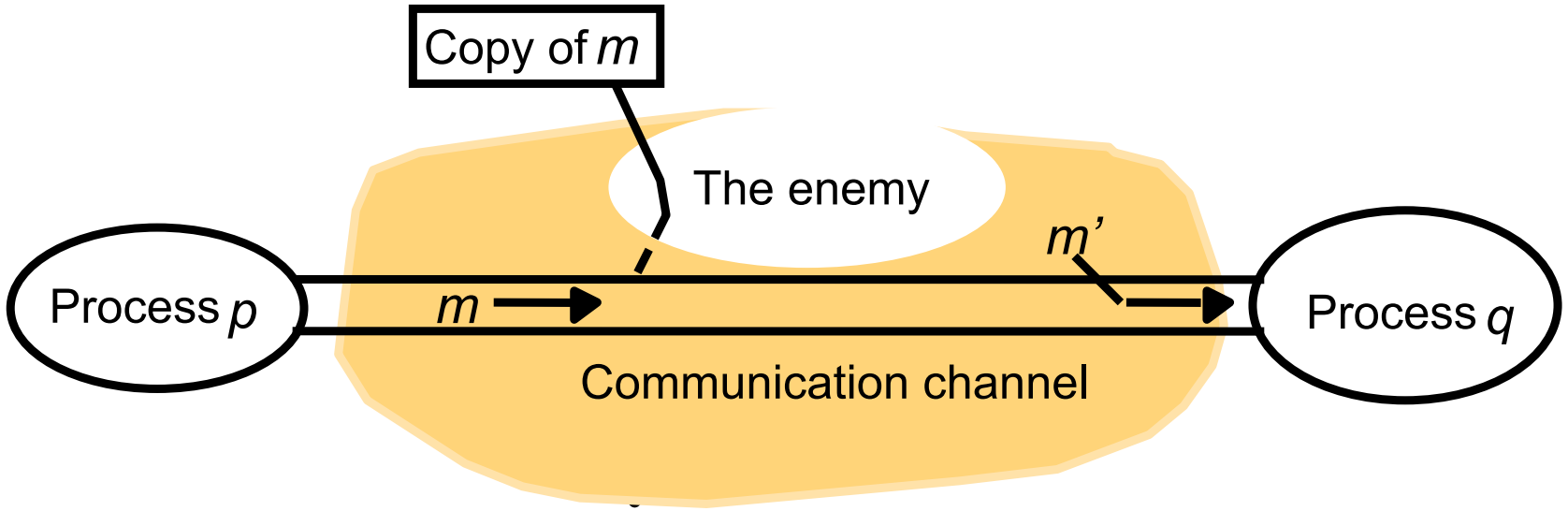
PRINCIPAIS, OBJECTOS, CONTROLO DE ACESSO E CANAIS



A segurança num sistema distribuído passa por:

- Autenticar os principais
- Verificar direitos de acesso aos objectos – controlo de acessos
- Utilizar canais seguros

AMEAÇAS BÁSICAS AOS CANAIS DE COMUNICAÇÃO (E CONSEQUENTEMENTE AOS PROCESSOS)

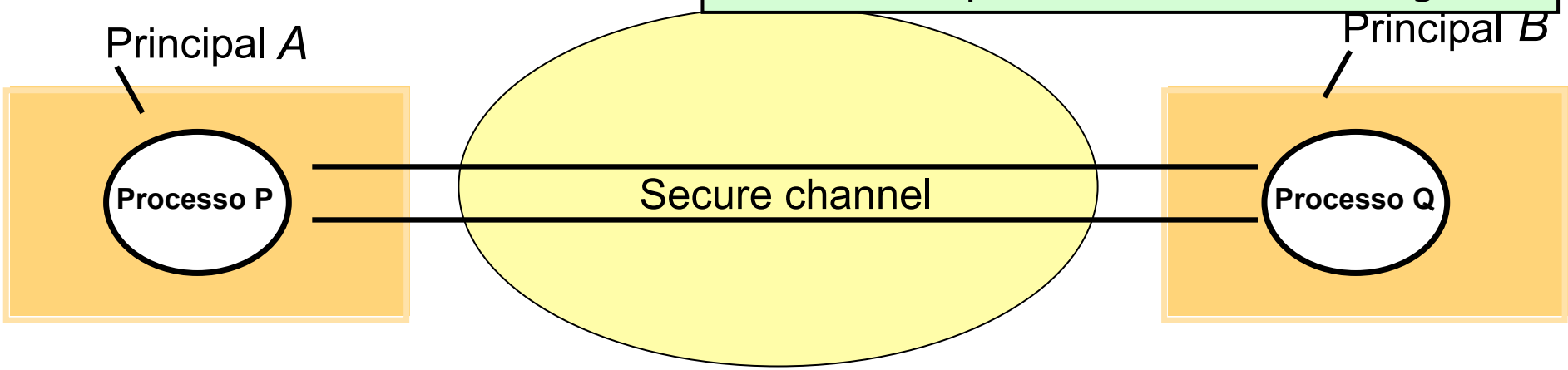


Para modelar ameaças de segurança, assume-se que o inimigo tem grande capacidade e pode:

- Enviar mensagens para qualquer processo
  - Forjar endereço das mensagens, fazendo-se passar por outro processo
- Ler, copiar, remover e reenviar mensagens que passam no canal
  - Impedir interação, repetir interação, etc.

CANAIS SEGUROS

Existem técnicas criptográficas (com partilha de segredos) que podem ser usadas na autenticação de parceiros e cifra de informação:
Permitem implementar um canal seguro



Num canal seguro os interlocutores (A e B) estão autenticados

O inimigo não pode copiar, alterar ou introduzir mensagens

O inimigo não pode fazer *replaying* de mensagens

O inimigo não pode reordenar as mensagens

OUTRAS AMEAÇAS

Denial of service: ataque em que o inimigo interfere (impede) atividade dos utilizadores autorizados

Código móvel: problemas de segurança para os processos que recebem e devem executar código móvel

risco de infecção: virus, trojans, worms, etc.

Outras???

PARA SABER MAIS

George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair,
Distributed Systems – Concepts and Design,
Addison-Wesley, 5th Edition, 2011

Capítulo 2.

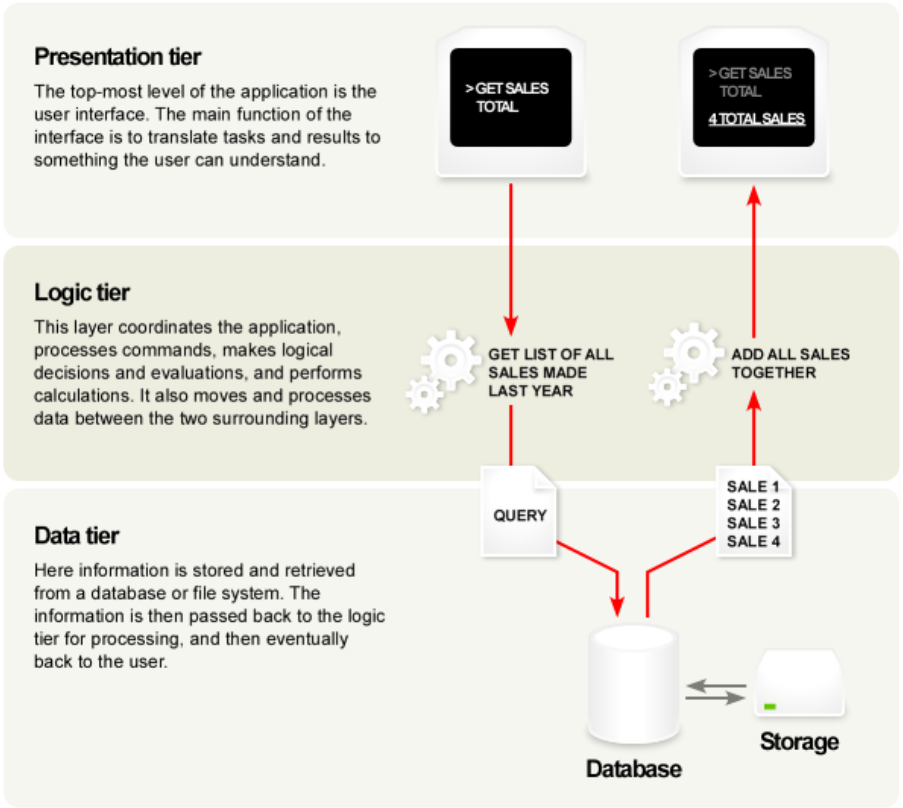
ARQUITECTURA EM CAMADAS: MODELO *THREE-TIER*

Em aplicações de acesso a sistemas de informação/web commerce, etc.

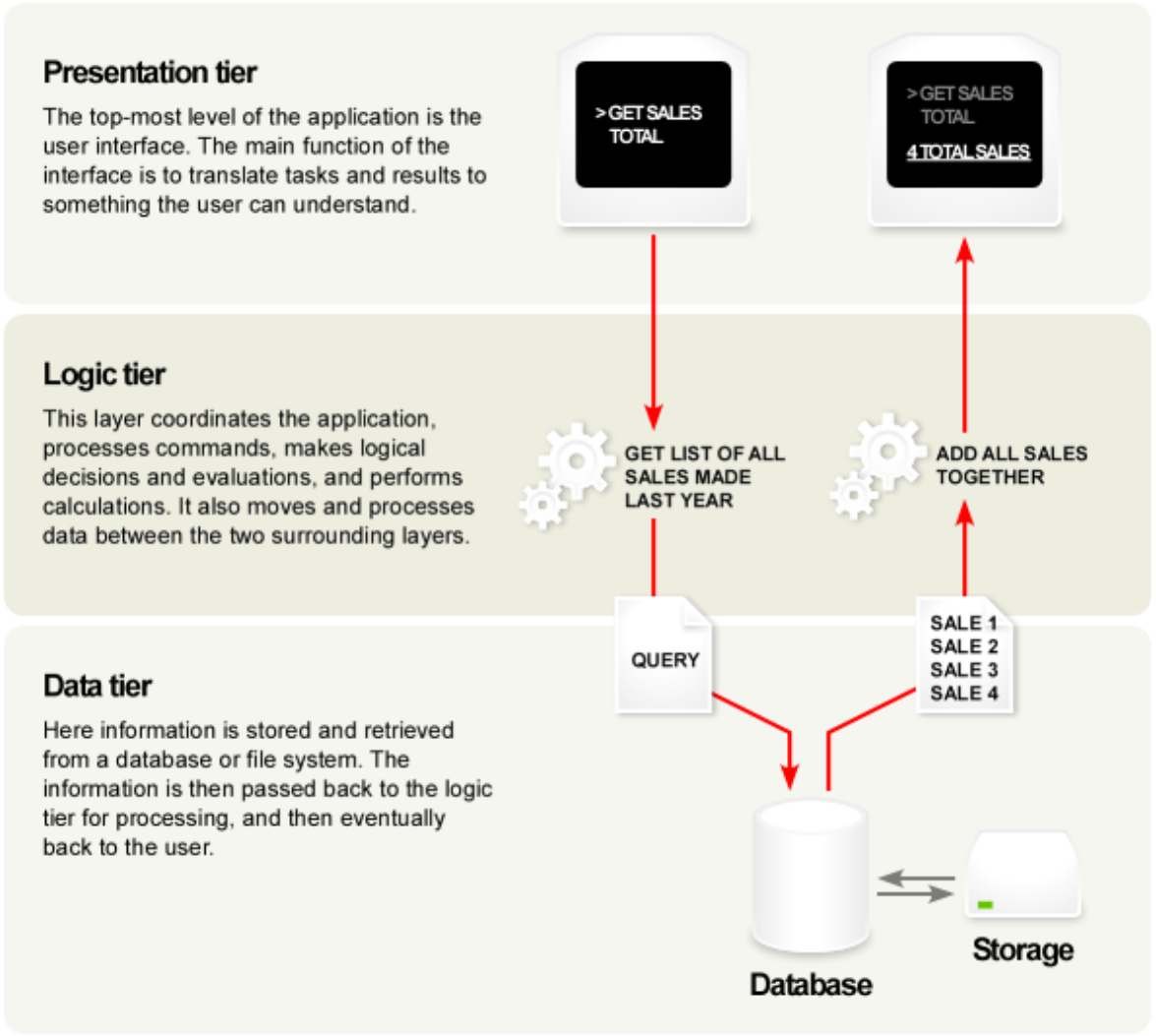
Arquitetura de três níveis (3-tier)

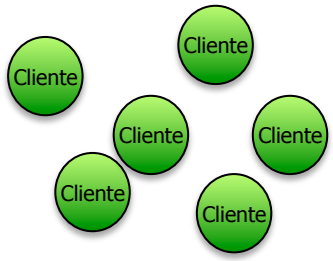
- Apresentação
- (Lógica) Aplicação
- (Armazenamento) Dados

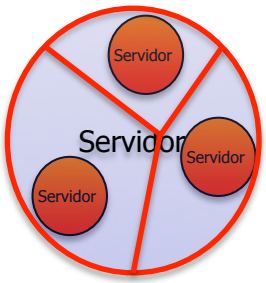
Arquitetura típica: cliente/ servidor

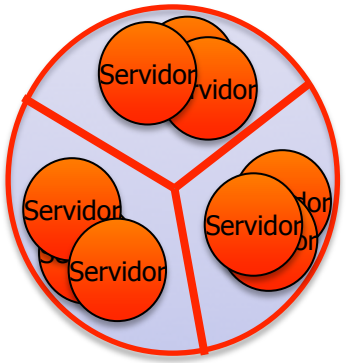


ARQUITECTURA EM CAMADAS: MODELO *THREE-TIER*

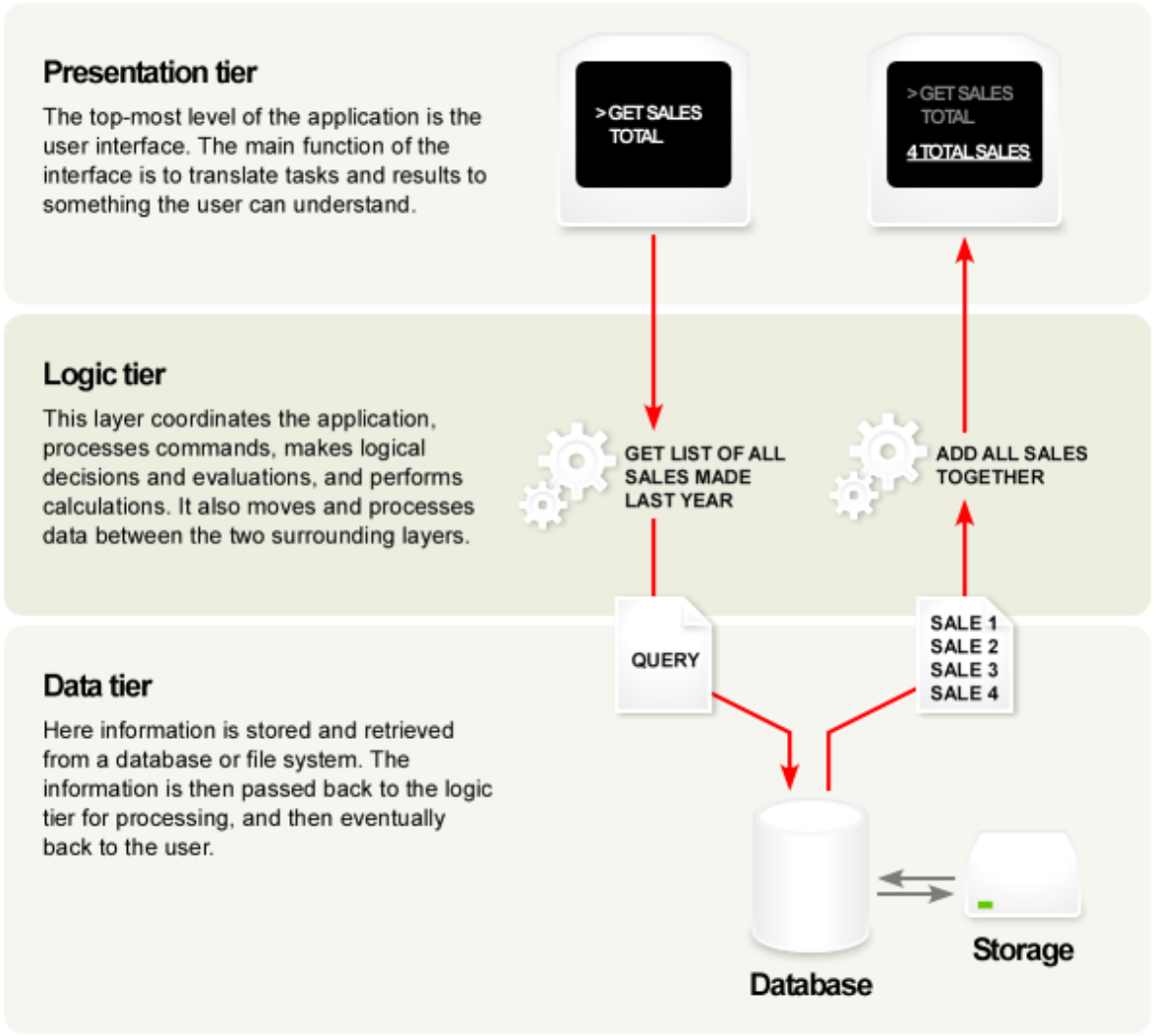


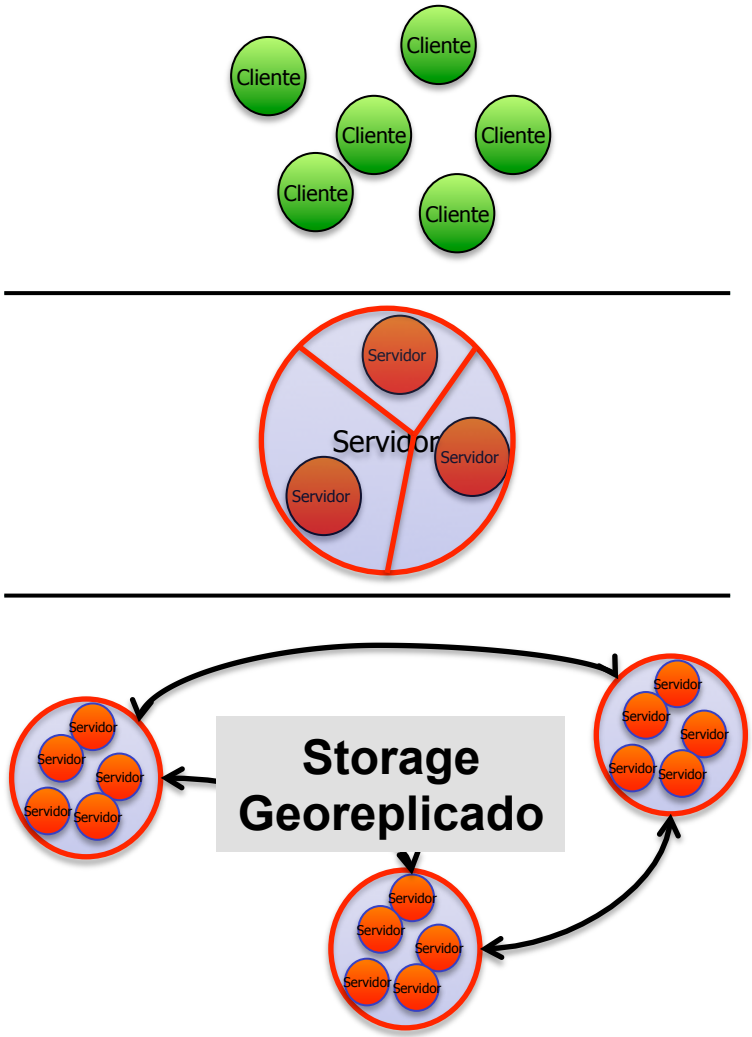






ARQUITECTURA EM CAMADAS: MODELO *THREE-TIER*





QUESTÃO...

Considere o sistema BitTorrent de partilha de ficheiros. O protocolo implementado pelo sistema torna-o apropriado para efetuar reprodução em tempo real dos ficheiros multimédia transferidos? Justifique.