SISTEMAS DISTRIBUÍDOS

Capítulo 2

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, 4th Edition, 2005

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

CONTEXTO

Camadas de software

- Especifica quais os componentes de software num processo
- Tem implicações na simplicidade da criação do software

Arquitetura de software

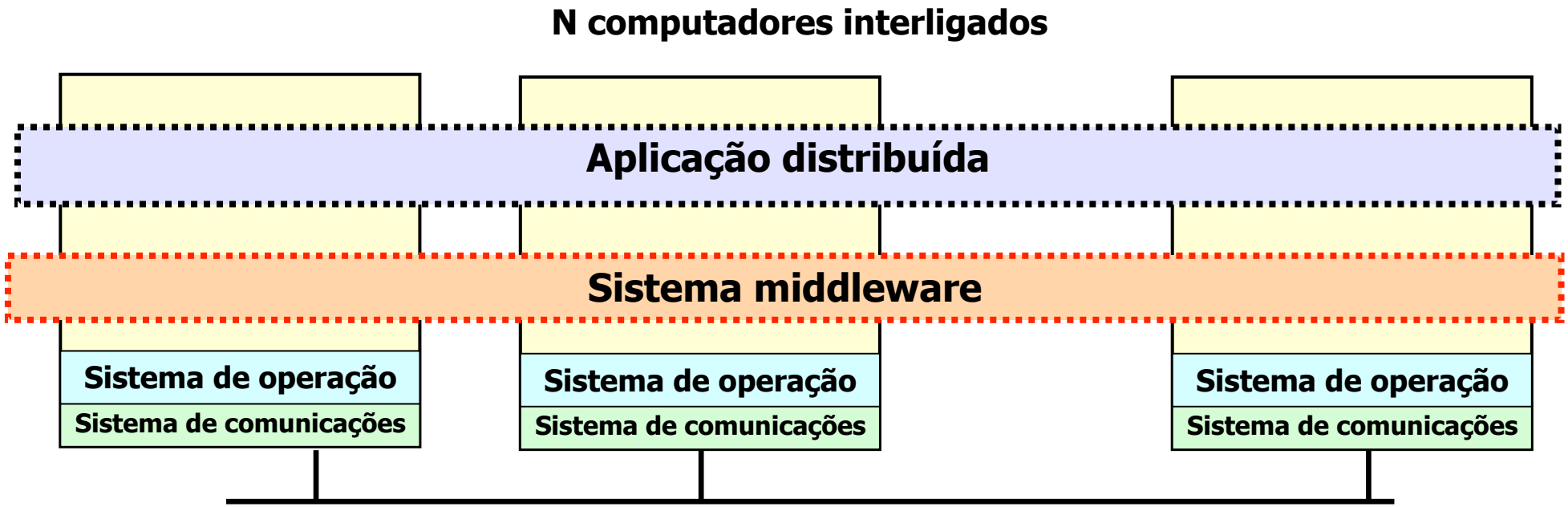
- Especifica como se organizam e quais as interações entre os processos a executar numa máquina

Arquitetura distribuída

- Especifica como se organizam e quais as interações entre os vários processos que compõem um sistema distribuído

- Tem implicações no desempenho, fiabilidade e segurança do sistema

CAMADAS DE SOFTWARE



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.)

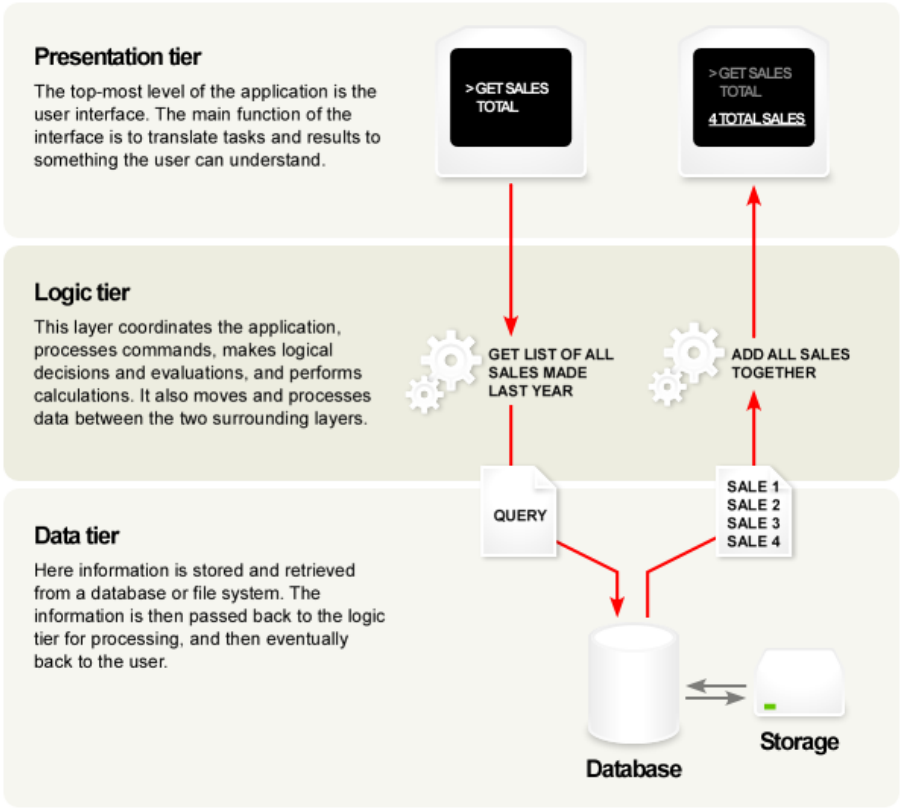
ARQUITECTURA DE SOFTWARE: MODELO *THREE-TIER*

Em aplicações de acesso a sistemas de informação

Arquitetura de três níveis (3-tier)

- Apresentação
- Aplicação
- Dados

Arquitetura típica: cliente/ servidor



ARQUITECTURA DO SISTEMA

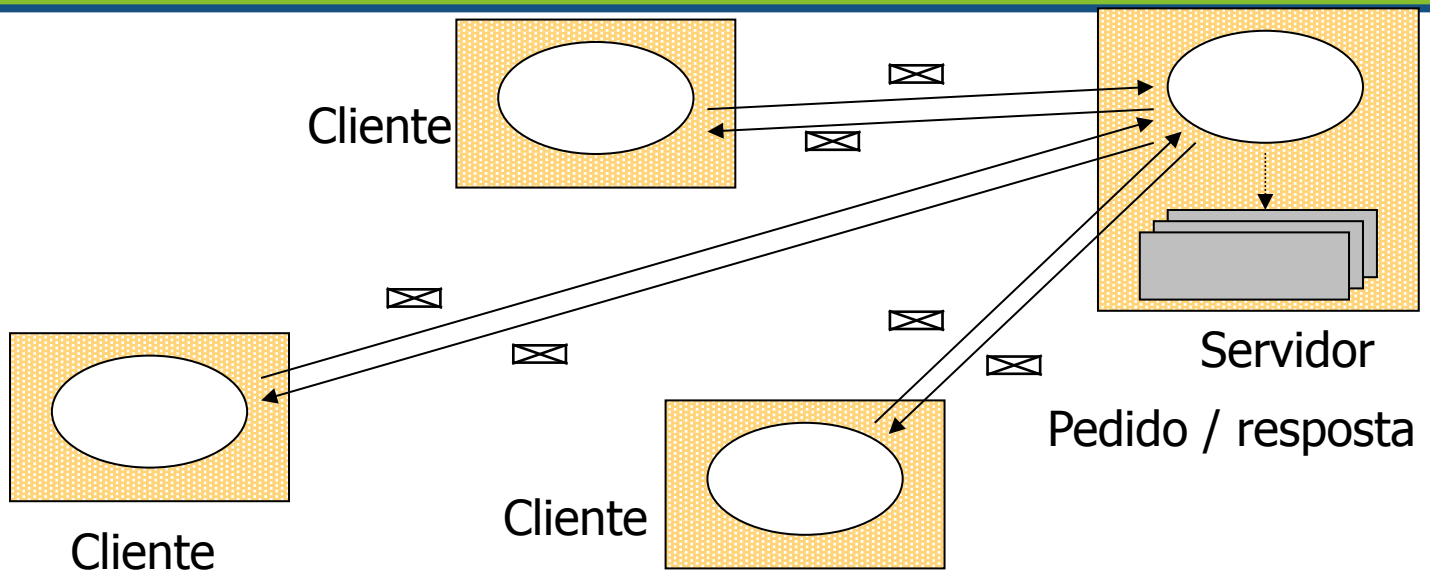
Arquitetura do sistema

- Define os componentes do sistema distribuído, identificando as suas funções de forma simples e abstracta
- Identifica a localização dos componentes numa rede
- Identifica as inter-relações entre os componentes (incluindo os padrões de comunicação)

A arquitetura tem implicações no desempenho, fiabilidade e segurança do sistema

Arquitetura mais comum e usada na prática

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 o respectiva resultado

CLIENTE/SERVIDOR: PROPRIEDADES

Arquitetura mais comum e usado na prática

Positivo

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

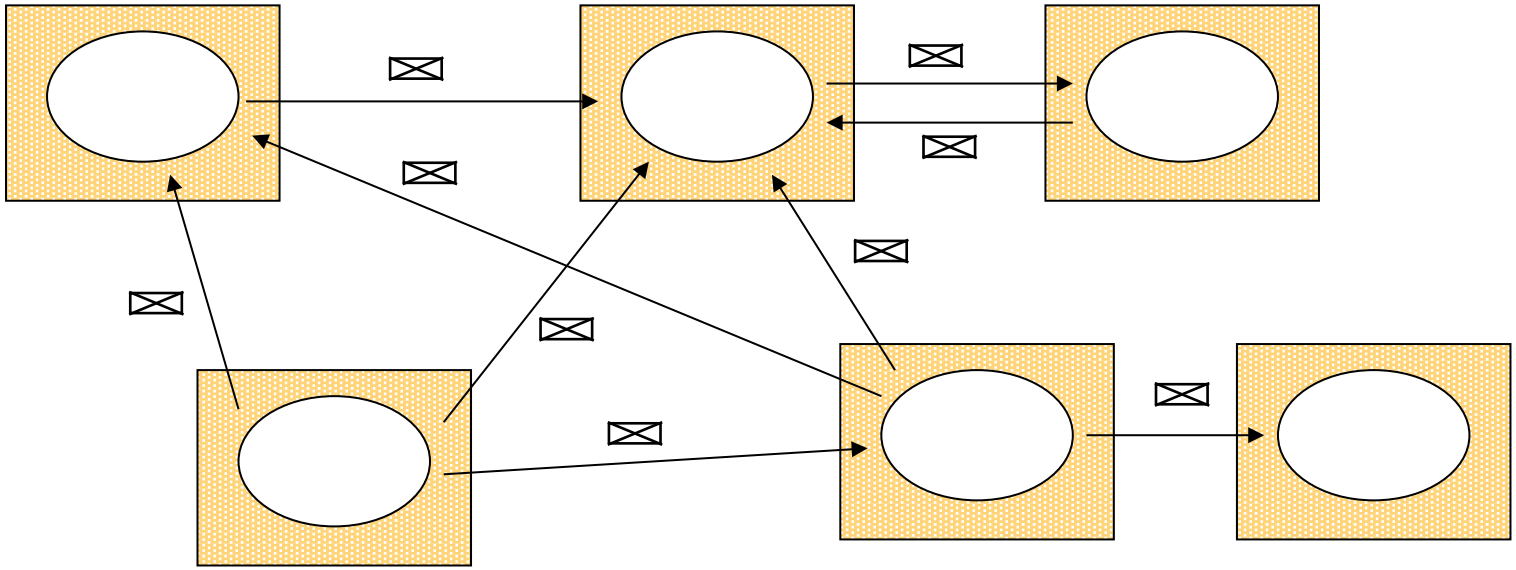
Negativo

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

Variantes podem eliminar/diminuir problemas

Modelo alternativo para lidar com limitações do modelo cliente/servidor

MODELO *PEER-TO-PEER*



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

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 replicado
  - Existem vários servidores idênticos (i.e. capazes de responder aos mesmo pedidos)
  - Positivo
    - Permite distribuir a carga, melhorando o desempenho (potencialmente)
    - Não existe um ponto de falha único
  - Negativo
    - Manter estado do servidor coerente em todas as réplicas
    - Recuperar da falha parcial de um servidor

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 (ex. DNS)
    - Servidor redirige cliente para outro servidor (iterativo)
    - Servidor invoca pedido noutro servidor (recursivo)
  - 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

VARIANTES DO MODELO CLIENTE/SERVIDOR: CLIENTE

- Cliente leve (*thin client*)/servidor
  - O cliente apenas inclui um interface (gráfico) para executar operações no servidor (ex.: browser)
  - Positivo:
    - Cliente pode ser muito simples
  - Negativo
    - Maior peso no servidor

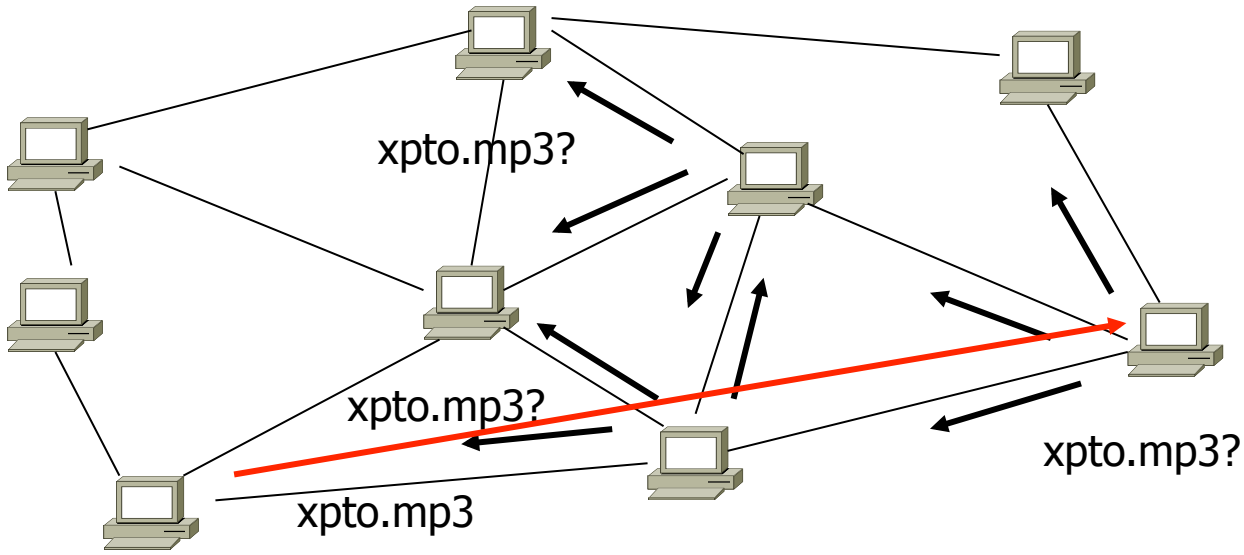
VARIANTES DO MODELO CLIENTE/SERVIDOR: CLIENTE

- Cliente completo (estendido)/servidor
  - O cliente executa localmente algumas operações que seriam executadas pelo servidor
  - Positivo:
    - Permite funcionar quando não é possível contactar o servidor (recorrendo a *caching*)
    - Permite diminuir a carga do servidor e melhorar o desempenho
  - Negativo:
    - Implementação do cliente mais complexa
    - Necessário tratar da coerência dos dados entre o cliente e o servidor

    E.g.: alguns sistemas aplicam o princípio à Web: Google Gears

Variantes do modelo peer-to-peer têm diferentes propriedades

VARIANTES DO MODELO *PEER-TO-PEER*



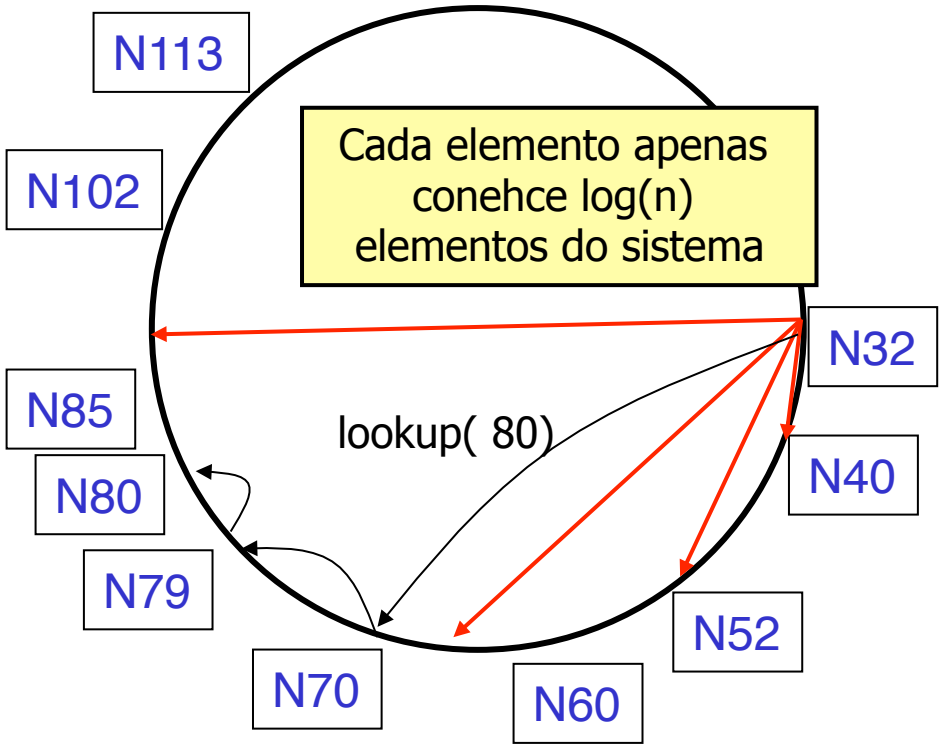
Sistema peer-to-peer não estruturado

- As ligações entre os membros são formadas de forma não-determinista
  - E.g. quando se junta à rede, um membro escolhe um conjunto de contactos (os contactos podem variar durante a execução do sistema)
- Positivo:
  - Simplicidade
- Negativo:
  - Pesquisa pesada (geralmente por inundação)
  - Latência/escalabilidade depende da árvore formada

VARIANTES DO MODELO *PEER-TO-PEER*

Sistemas *peer-to-peer* estruturados

- Os membros do sistema comunicam de acordo com uma organização definida de forma determinista
  - E.g. DHT permitem pesquisar valores conhecidos
    - Nó responsável por uma dada chave depende do identificador do nó
- Positivo
  - Boa latência/escalabilidade - O(log n)
- Negativo
  - Maior complexidade



DHTs: CARACTERÍSTICAS INTERESSANTES

Identificar informação usando hash seguro – hash(info)

- Quais as implicações?

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

- E.g. cada nó tem associado identificador único gerado aleatoriamente, ficando responsável por manter informação cujo identificador é mais próximo do seu identificador
- Implicações? Pesquisa? Distribuição da informação? Replicação da informação?

Usar identificadores para criar um sistema de encaminhamento overlay, em que qualquer percurso tem, no máximo, log(n) nós

Combinando os dois modelos pode-se ter o melhor dos dois

COMBINAÇÃO DOS MODELOS C/S E P2P

Cliente + peer-to-peer

- Num sistema peer-to-peer, podem existir elementos que disponibilizam o serviço a outros processos (clientes) que não pertencem ao sistema peer-to-peer

Propriedades:

- Permite a uma máquina aceder a um serviço disponibilizado por um sistema peer-to-peer
- Permite limitar o número de processos que fazem parte do sistema peer-to-peer

COMBINAÇÃO DOS MODELOS C/S E P2P

Cliente/servidor + peer-to-peer

O serviço disponibilizado por um sistema pode ser dividido em várias funcionalidades, sendo umas fornecidas por um sistema cliente/servidor e outras por um sistema peer-to-peer.
Neste caso é comum o sistema cliente/servidor servir como serviço de diretório.

E.g. BitTorrent

Propriedades:

Permite combinar as vantagens de ambos os sistemas

BitTorrent (resumido e simplificado)

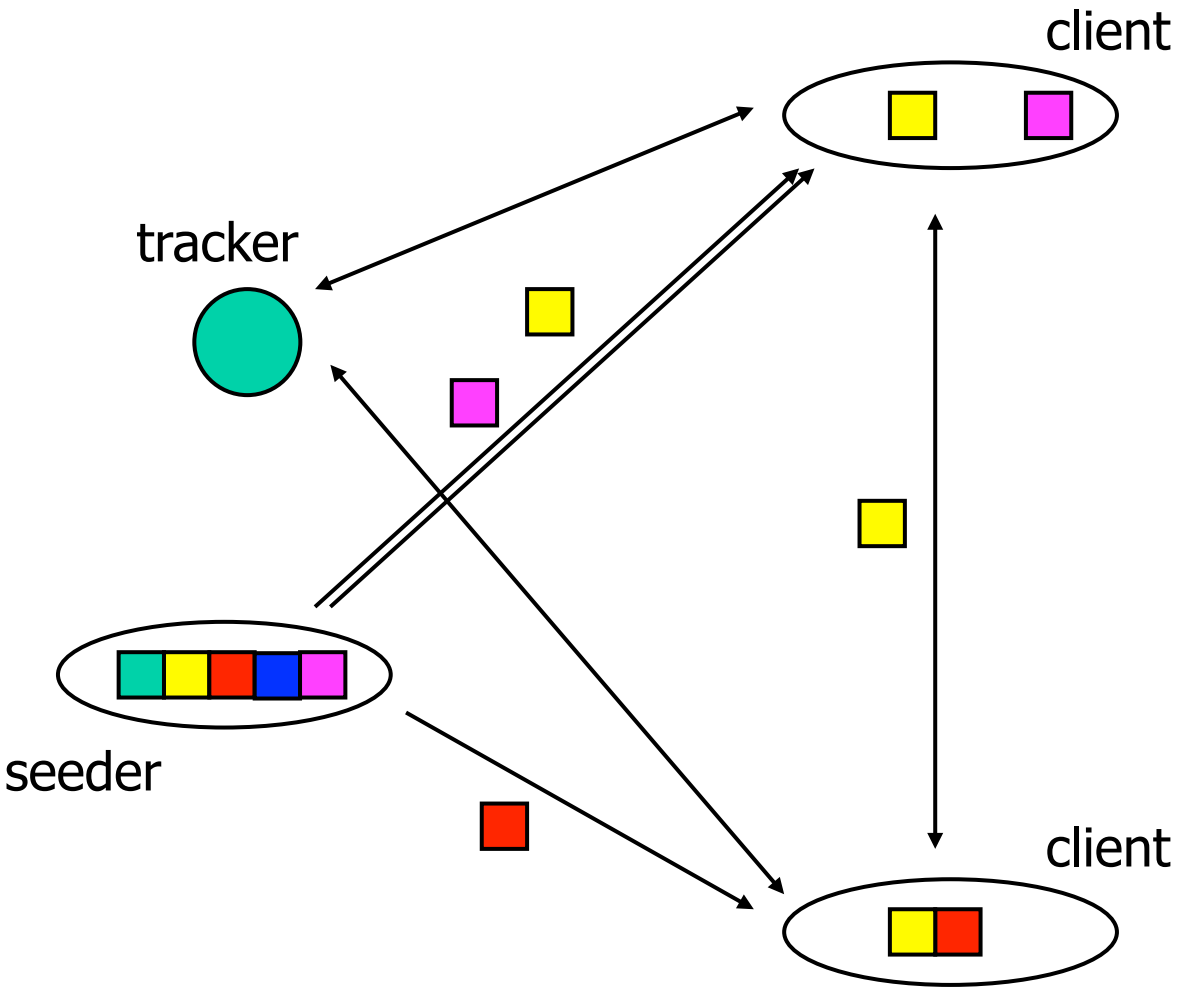
.torrent: contém informação sobre ficheiro a ser partilhado, incluindo: url do tracker, hash seguro de cada bloco do ficheiro (SHA-1)

Arquitetura:

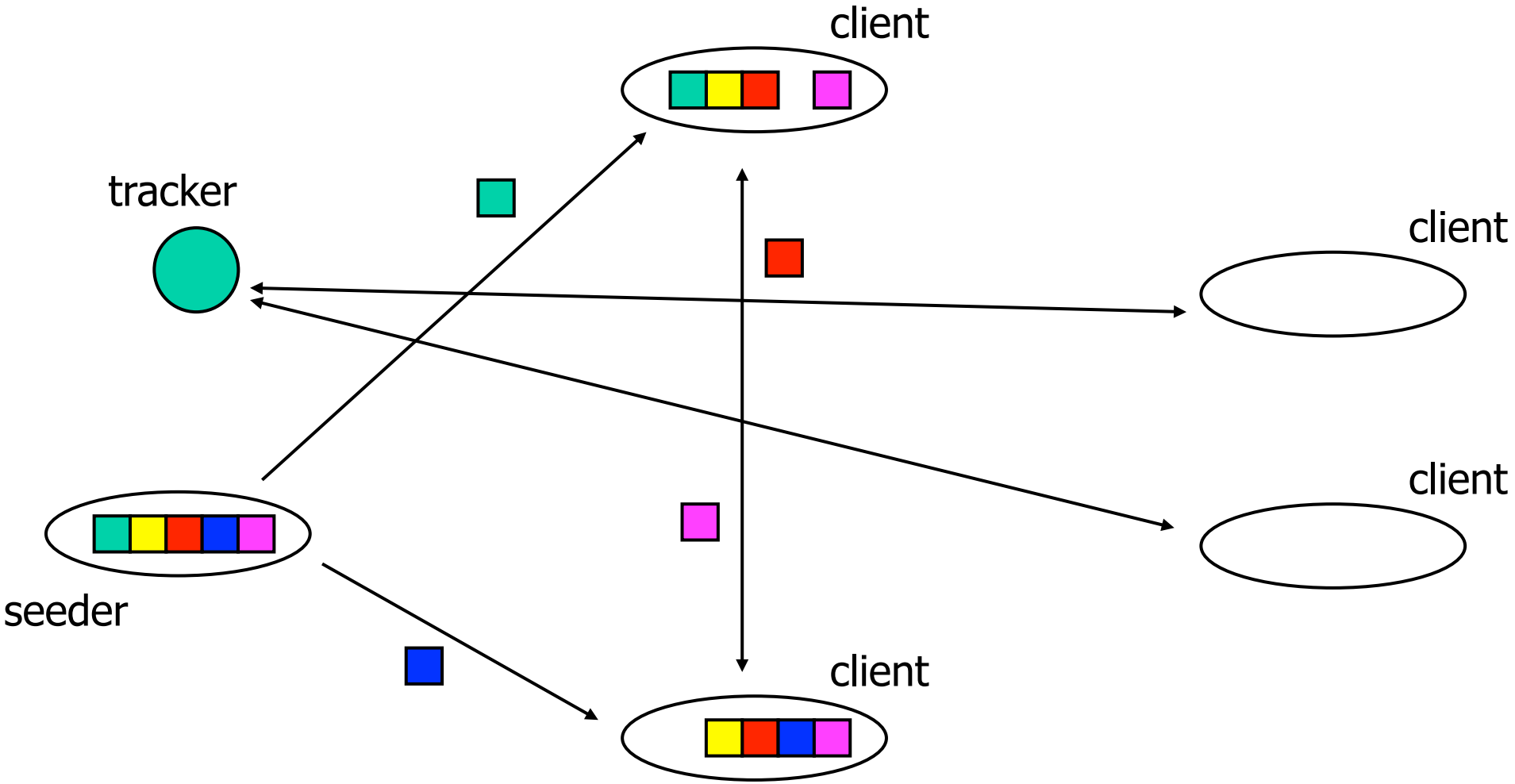
- Cliente/servidor
  - Tracker: servidor centralizado mantém informação sobre quais os peers que estão a partilhar/descarregar o ficheiro
  - Peers: acedem ao tracker para obter lista de outros peers a descarregar o ficheiros
- Peer-to-peer
  - Peers comunicam entre si para trocar blocos do ficheiro
    - 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

    - Para que serve cada uma destas propriedades?

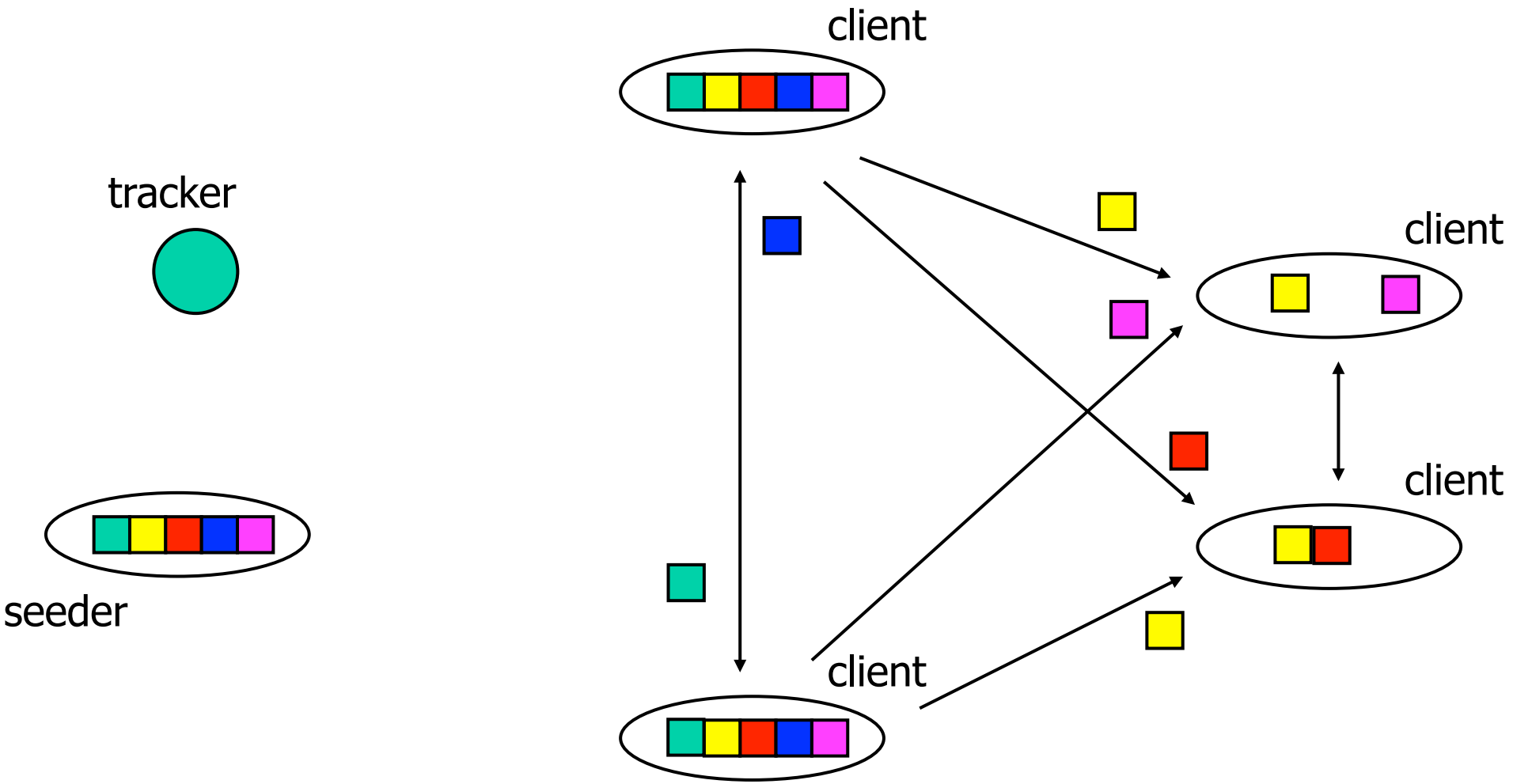
BITTORRENT



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 armazenada nos nós com identificadores mais próximos

Nota: baseado no sistema Kademlia (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 podia ser envolvido na comunicação peer-to-peer para contornar clientes bloqueados por *firewalls*

AINDA OUTROS EXEMPLOS...

Sistemas peer-to-peer hierárquicos

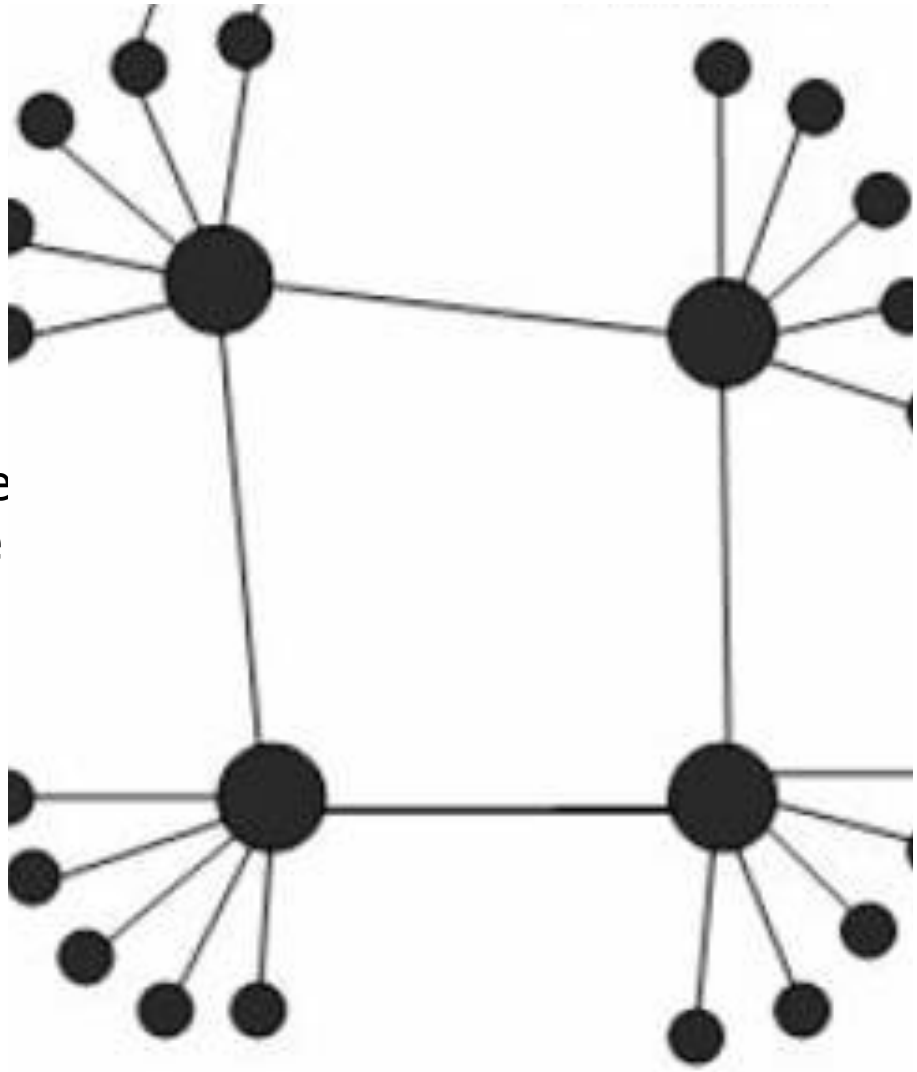
- Subconjunto de super-nós que se agrupam como num sistema peer-to-peer
- Nós ligam-se a um super-nó

...

KaZaA / Skype

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 conhe
  - (após se ter ligado, atualiza lista de



Pesquisa:

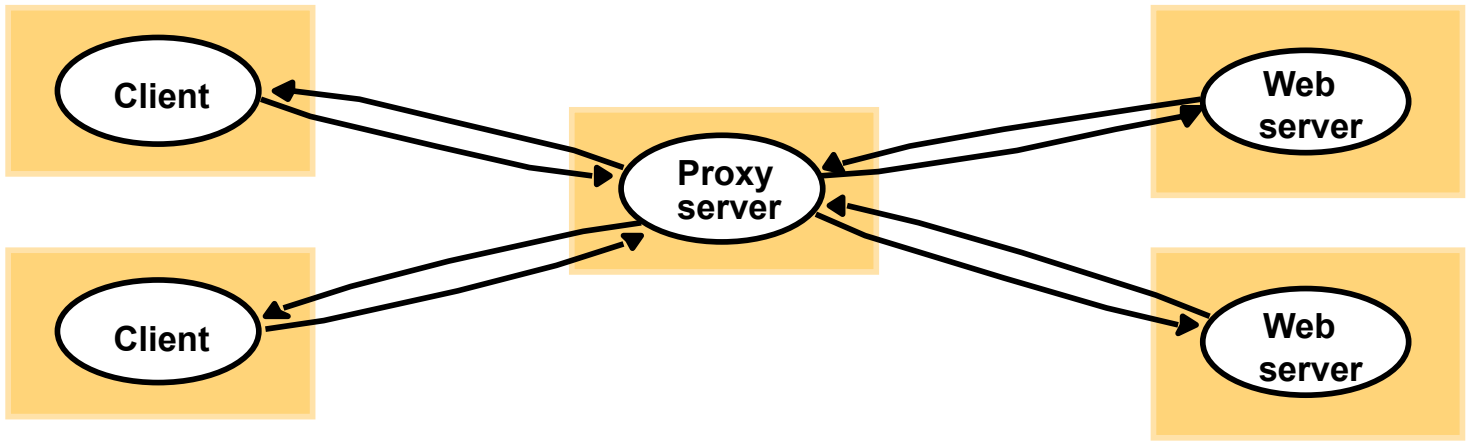
- ON contacta o seu SN
- SNs propagam pedidos entre si

Transferência de ficheiros:

- Directamente entre Ons

Na prática, tendem a existir mais componentes num sistemas 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
- 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
- Propriedades
  - Menor latência, filtragem, distribuição de carga, etc.

ARQUITECTURAS ORIENTADAS A SERVIÇOS – SERVICE-ORIENTED ARCHITECTURES (SOA)

Arquitetura em que os recursos são disponibilizados como serviços independentes que podem ser acedidos conhecendo apenas a interface e protocolo de acesso. Pressupõe-se que:

- Existe mecanismo de descoberta de serviços
- Serviços tendem a ser *stateless*
- Pode-se implementar usando qualquer tecnologia: Web Services, RPCs, RMI, Jini, etc.
  - Surge normalmente associado com Web Services (permitem a invocação remota usando protocolo sobre HTTP)

Propriedades desejáveis:

- Reutilização, modularidade, possível compor serviços mais complexos, descrição dos serviços

Desafios:

- Segurança, interoperablidade, estado nos serviços

MODELOS FUNDAMENTAIS

Modelo de interacção

Modelo de falhas

Modelo de segurança

Modelo de interacção (1)

- Tipo de interacção
  - Activa
    - Processo solicita execução de operação noutro processo
  - Reactiva
    - Evento no sistema desencadeia acção num processo
  - Indirecta
    - Processos comunicam através de um espaço partilhado

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

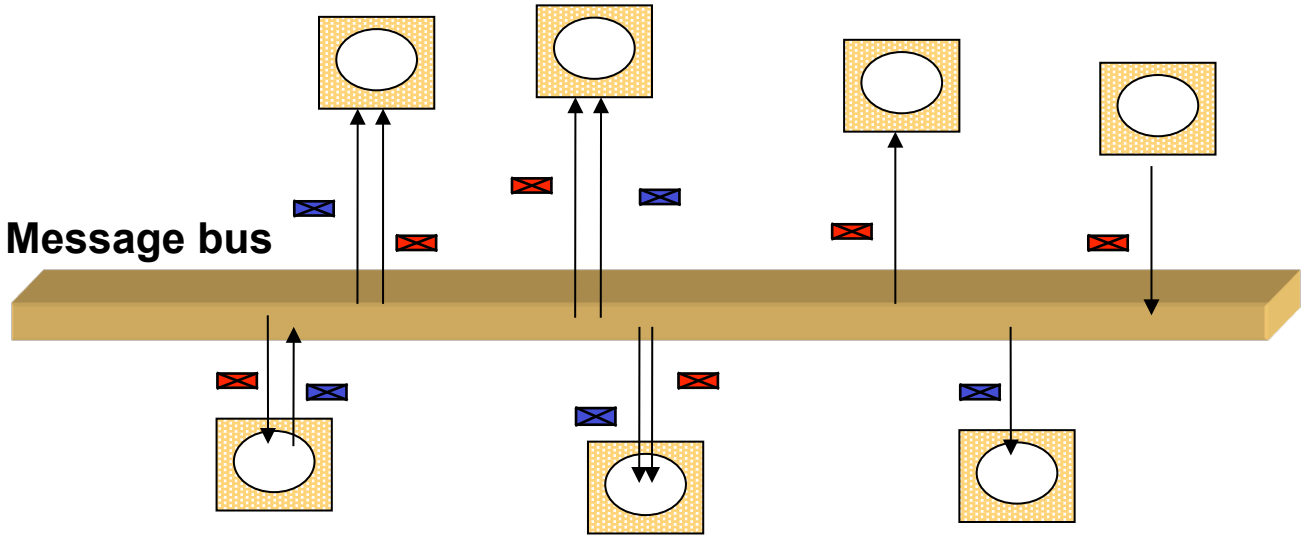
Como difundir informação sobre:

- Eventos dum jogo de futebol
- Cotações da bolsa

Propriedades

- 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 "publish/subscribe"

- Processo subscritor/consumidor subscreve o interesse num conjunto de eventos
- Processo produtor produz eventos

Sistema encarrega-se de propagar eventos produzidos para subscritores interessados

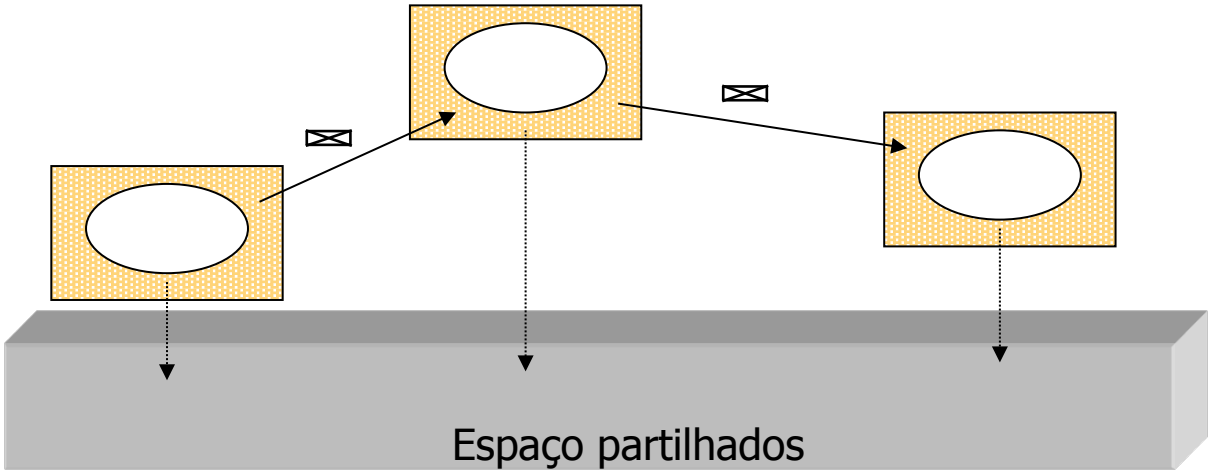
Modelo de interação indireta

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 informção 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 interacção

- Activa
  - Processo solicita execução de operação noutro processo
- Reactiva
  - Evento no sistema desencadeia acção num processo
- Indirecta
  - Processos comunicam através de um espaço partilhado

Conteúdo da interacção

- Informação/dados
- Código

Conteúdo da interacção: dados

Processos trocam dados

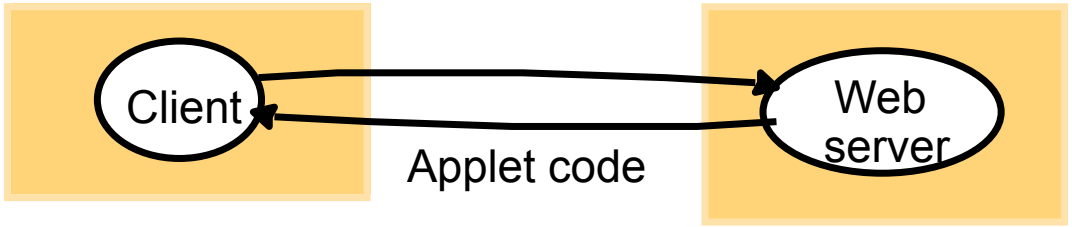
- Pedido (operação a invocar + parâmetros + inf. utilizador + ...)
- Resposta (resultado de operação + ...)
- Evento (num sistema de eventos)

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

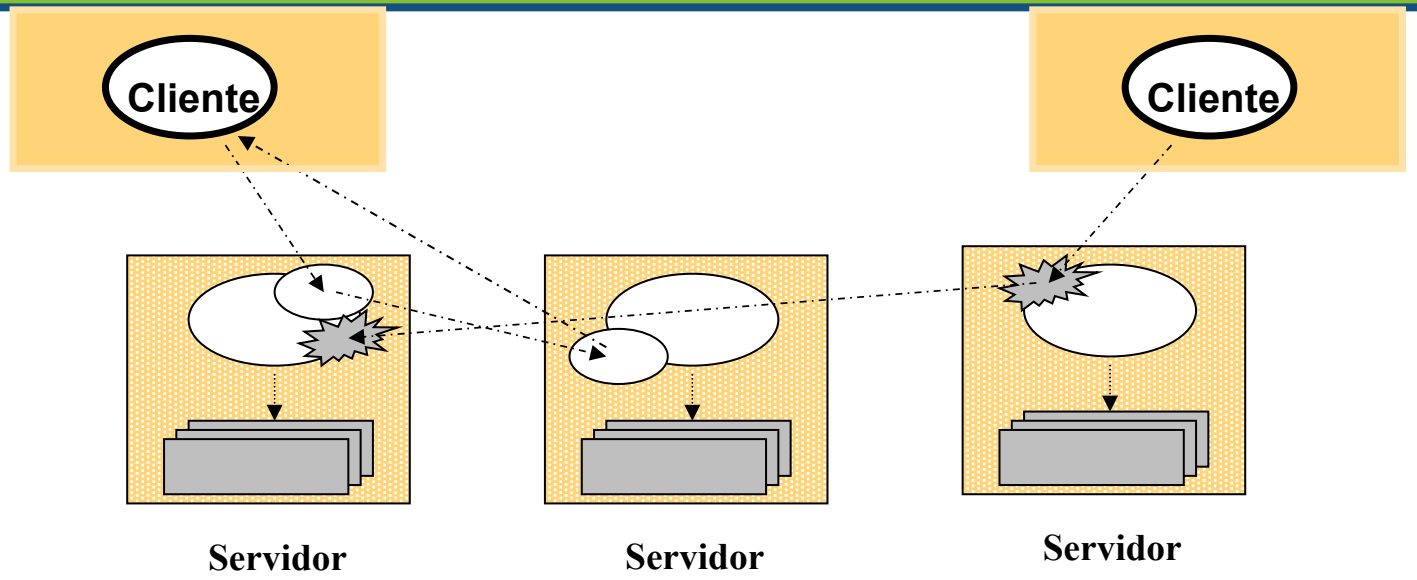
A execução do código no cliente pode:

- Melhorar o desempenho
- Ser usado para implementar funcionalidade adicional

Segurança

- Necessário proteger o cliente do código executado

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

O 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 incorrectos. 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

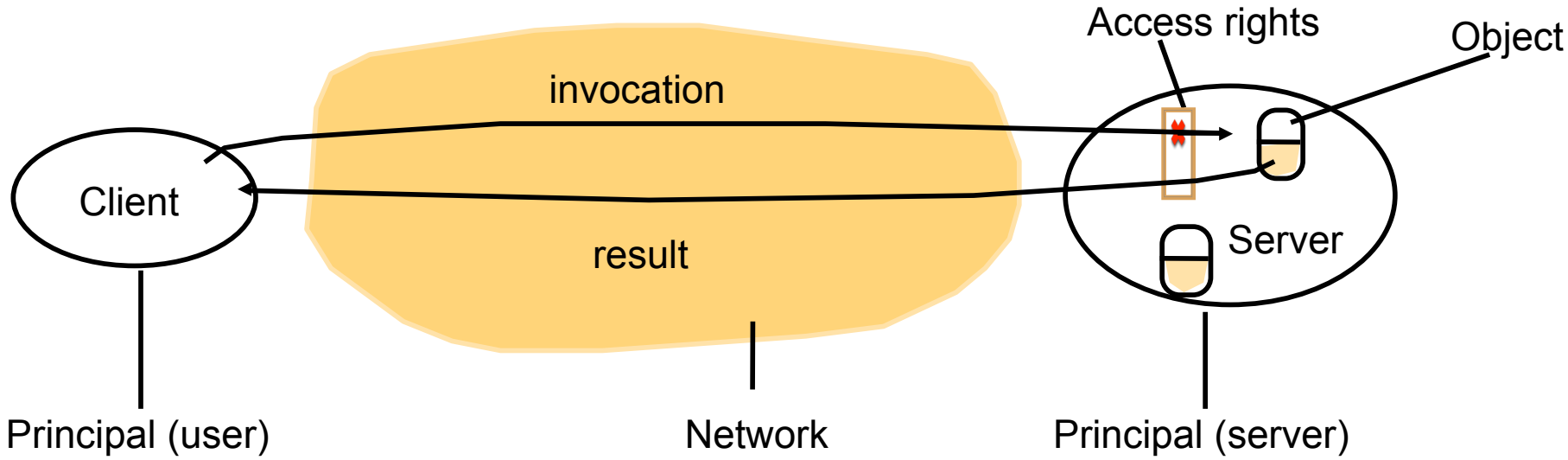
Erros transientes: ocorrem instantaneamente, ficam reparados imediatamente após terem ocorrido (ex.: perda de mensagem)

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

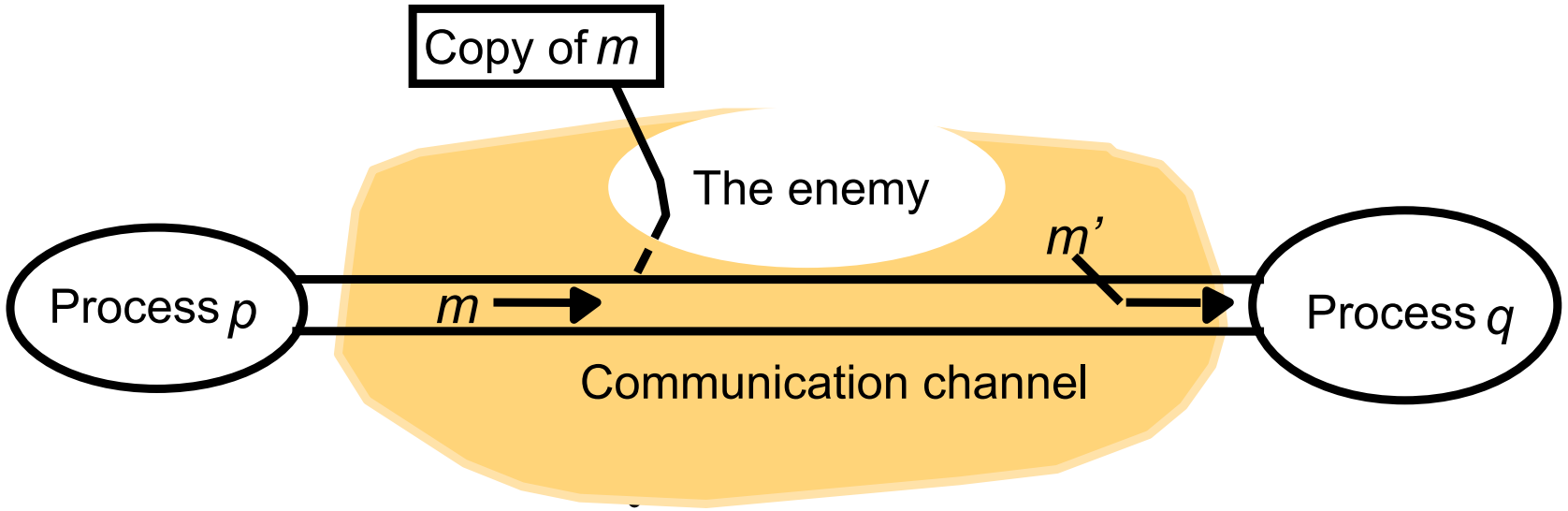
PRINCIPAIS, OBJECTOS, CONTROLO DE ACESSO E CANAIS



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)

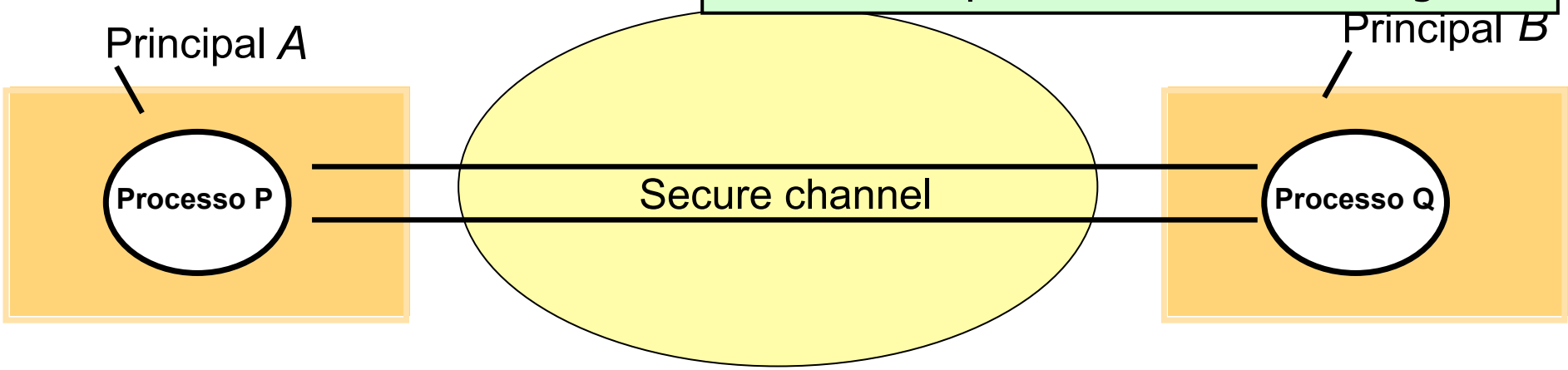


Para modelar ameaças de segurança, assume-se que o inimigo tem grande capacidade e pode:

- 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.

QUESTÃO (1)

Suponha que pretende desenvolver um sistema tipo Facebook, em que é mantido um conjunto de mensagens, e para cada mensagem um conjunto de comentários associados. Suponha que apenas podem ser efectuadas duas operações – criar mensagem e adicionar comentário a uma mensagem. Os dados do sistema devem ser mantidos replicados em vários centros de dados distribuídos geograficamente pelo mundo (cada centro de dados tem todos os dados do sistema). Suponha por simplicidade que cada centro de dados tem apenas um computador.

Os utilizadores executam as suas operações contactando um centro de dados (em principio, aquele geograficamente mais próximo).

1. Uma arquitetura alternativa para o sistema seria manter a informação de cada utilizador em apenas um centro de dados, usando uma estratégia de cliente/ servidor particionado. Discuta as vantagens e desvantagens desta aproximação face à aproximação indicada.
2. Um centro de dados é composto por um conjunto de computadores. Para manter a informação das mensagens e respectivas respostas, seria interessante usar um arquitetura tipo DHT? Se sim, explique como organizaria a informação. Se não, explique as razões.

QUESTÃO (2)

Suponha que pretende desenvolver um sistema tipo Google Docs, que permite manter um conjunto de documentos – ficheiros de texto, folhas de cálculo, apresentações, etc. Vários utilizadores podem editar no seu computador um mesmo documento simultaneamente usando um browser de Internet, observando imediatamente as operações dos outros utilizadores. Adicionalmente, o sistema permite observar em que posição do documento o outro utilizador tem o seu cursor.

1. O sistema Google Docs é baseado numa arquitetura cliente/servidor, em que os clientes enviam as suas operações para o servidor que as propaga para os outros clientes. Apresente vantagens desta aproximação face a uma solução peer-to-peer em que os clientes propagavam diretamente entre si as operações efectuadas (as quais podem explicar a solução do Google Docs).

QUESTÃO (3)

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.

MASCARAR FALHAS DE COMPONENTES

Para compensar os problemas levantados pelas falhas usam-se técnicas para as mascarar. Desta forma é possível confinar os seus efeitos sobre o sistema.

As falhas de omissão podem ser mascaradas por

- replicação ou repetição

  Exemplo: se uma mensagem não chegou dentro de um certo período – o que se detecta por um timeout – então pode-se emiti-la novamente. Outra hipótese é duplicar o canal, enviar mais do que uma cópia em paralelo e filtrar as mensagens duplicadas.

As falhas arbitrárias podem ser difíceis de mascarar. Pode-se tentar transformá-las em falhas por omissão

- Exemplo: um CRC numa mensagem permite transformar uma falha bizantina do canal numa falha por omissão

O mesmo tipo de técnicas usam-se muitas vezes nos componentes hardware do tipo discos, memórias, etc... mesmo nos sistemas centralizados.

FALHAS TEMPORAIS

As falhas temporais são difíceis ou impossíveis de mascarar.

Normalmente apenas os sistemas de tempo real se preocupam com este tipo de falha.

Classe de falhas	Afecta	Descrição
Relógio	Processo	O relógio do processo tem um desvio superior ao permitido em relação ao tempo real
Desempenho	Processo	O processo leva mais tempo do que o estipulado a executar uma operação
Desempenho	Canal	A mensagem levou mais tempo do que o previsto

DEFINIÇÃO

Num sistema existem entidades que do ponto de vista da segurança têm identidade própria, direitos e deveres – essas entidades podem ser utilizadores, processos, etc. Utiliza-se o termo *principal* para as designar.

> Em Inglês, no contexto legal, um principal é alguém que está a ser julgado pelos seus actos.

Mais definições

Adequação temporal ou pontualidade. Em sistemas de tempo real é a garantia de que o sistema é capaz de obedecer a constrangimentos temporais, isto é, a capacidade que o sistema tem de garantir limites para o tempo que as diferentes acções levam a executar (e.g. Propagação de uma mensagem, execução de um passo de computação, etc).