SISTEMAS DISTRIBUÍDOS

João Leitão, Sérgio Duarte, Pedro Camponês

(baseado nos slides de Nuno Preguiça)

Aula 10: Capítulo 2

Arquiteturas e Modelos de Sistemas Distribuídos

NOTA PRÉVIA

A apresentação 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, 2005

NA ÚLTIMA AULA...

Modelos Arquiteturais de Sistemas Distribuídos

- Modelo Cliente-Servidor
- Arquitetura em Três Camadas
- Peer-to-Peer

Variantes do Modelo Cliente-Servidor

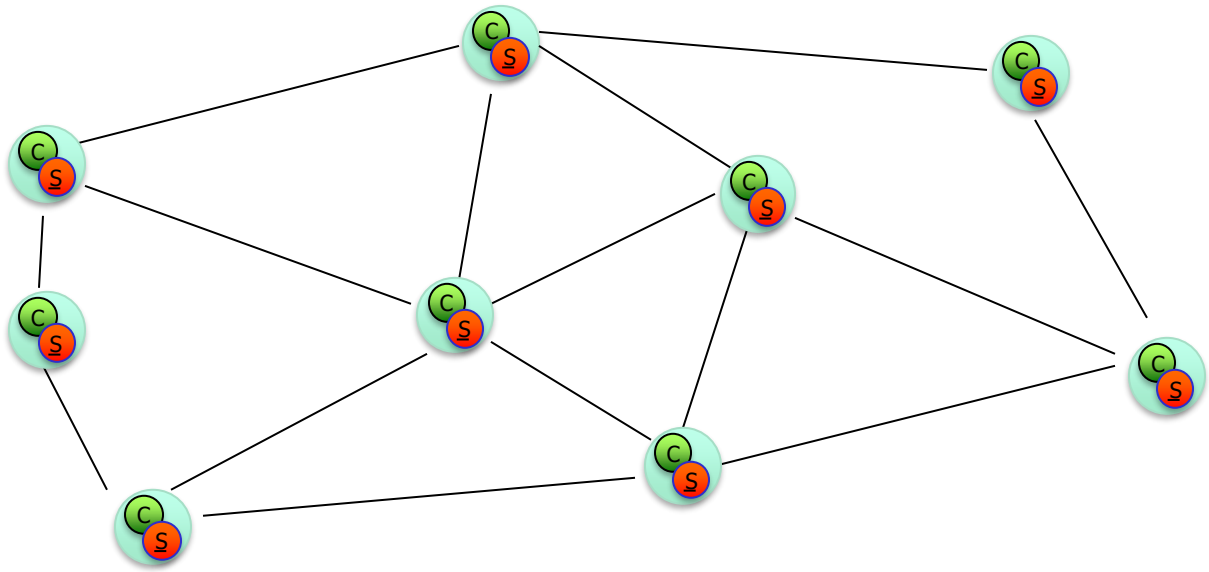
- Servidor: Particionado, Replicado, Geo-Replicado
- Cliente: Leve (Thin) e Completo (Extended)

NA AULA DE HOJE

Variantes do Modelo Arquitetural Peer-to-Peer

Variantes do modelo P2P com diferentes propriedades

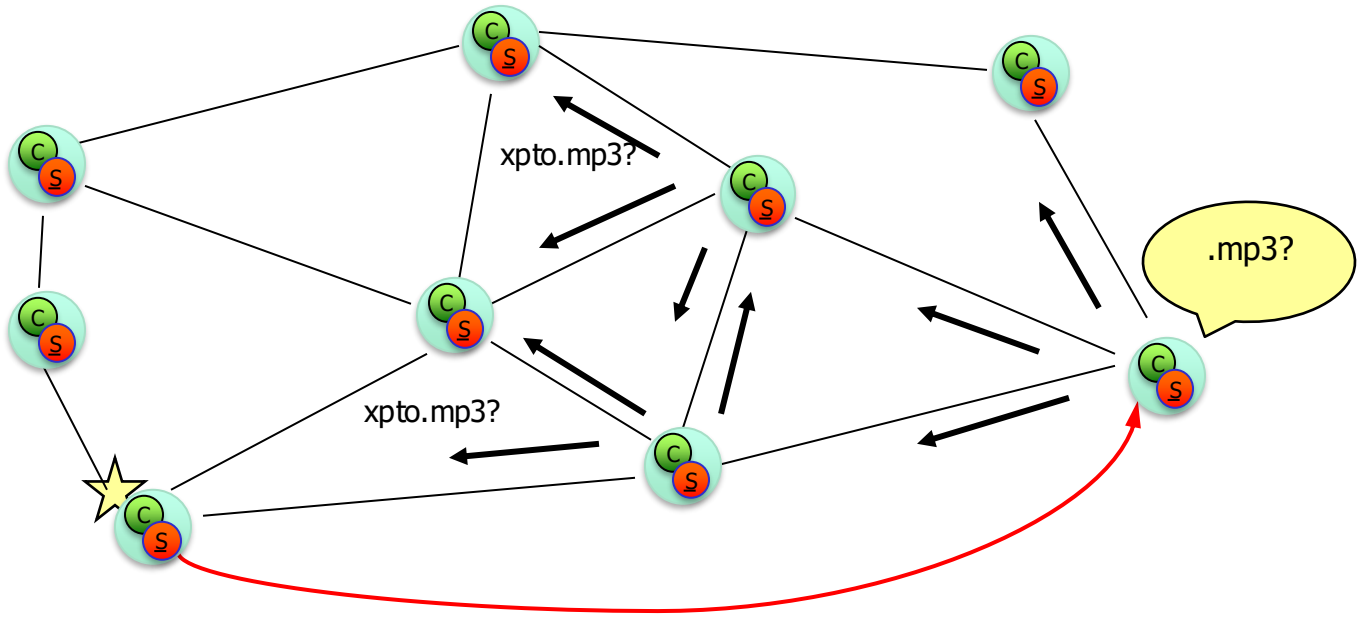
SISTEMA P2P NÃO ESTRUTURADO



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)

SISTEMA P2P NÃO ESTRUTURADO



Positivo:

Simplicidade de construir, robusto ao dinamismo da rede (churn – entrada e saída de nós participantes), adequado para disseminar informação (*application-level Broadcast*)

Negativo:

Produz topologias que podem ser difíceis de explorar de forma eficiente

e.g., dificuldade em indexar informação / pesquisa pesada (frequentemente por inundação)

SISTEMA P2P NÃO ESTRUTURADO

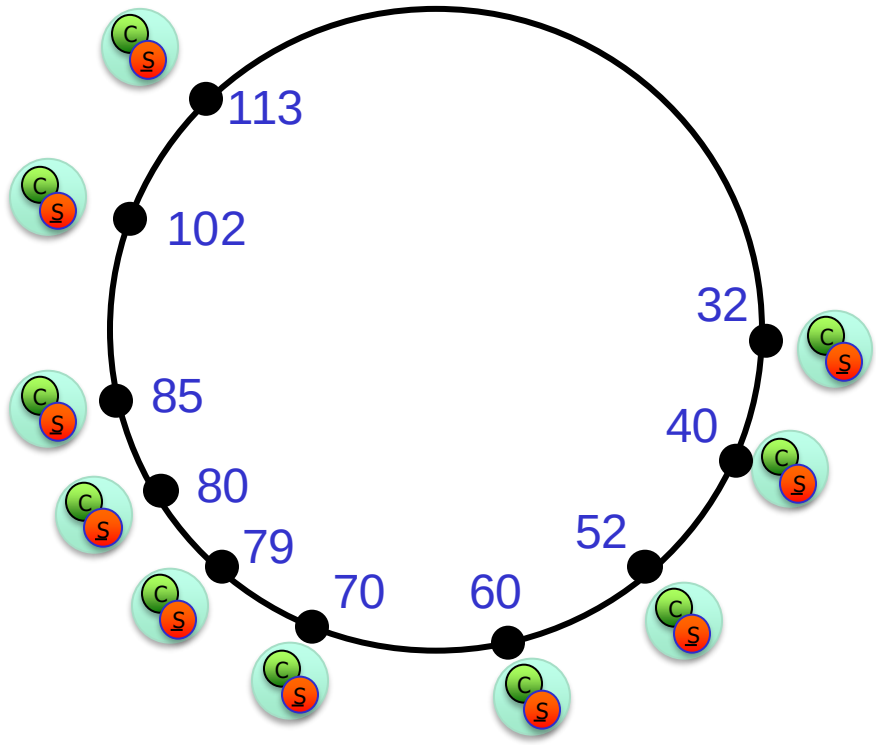
Como é que os nós no Sistema mantêm os seus vizinhos ao longo do tempo?

Várias soluções:

- Trocas Periódicas – os nós trocam informação sobre os seus vizinhos com outros vizinhos periodicamente (e.g., Cyclon, circa 2005)
- Gestão Reativa – os nós apenas trocam de vizinhos quando necessário e.g., um nó entra ou sai do sisteam (e.g., Scamp, circa 2001)
- Híbrido - Combina as duas alternativas acima (e.g., HyParView, circa 2007)

SISTEMAS *P2P* ESTRUTURADOS

Os membros do sistema organizam-se (e comunicam) de acordo com uma estratégia pré-definida de forma determinista com base num **endereço lógico**



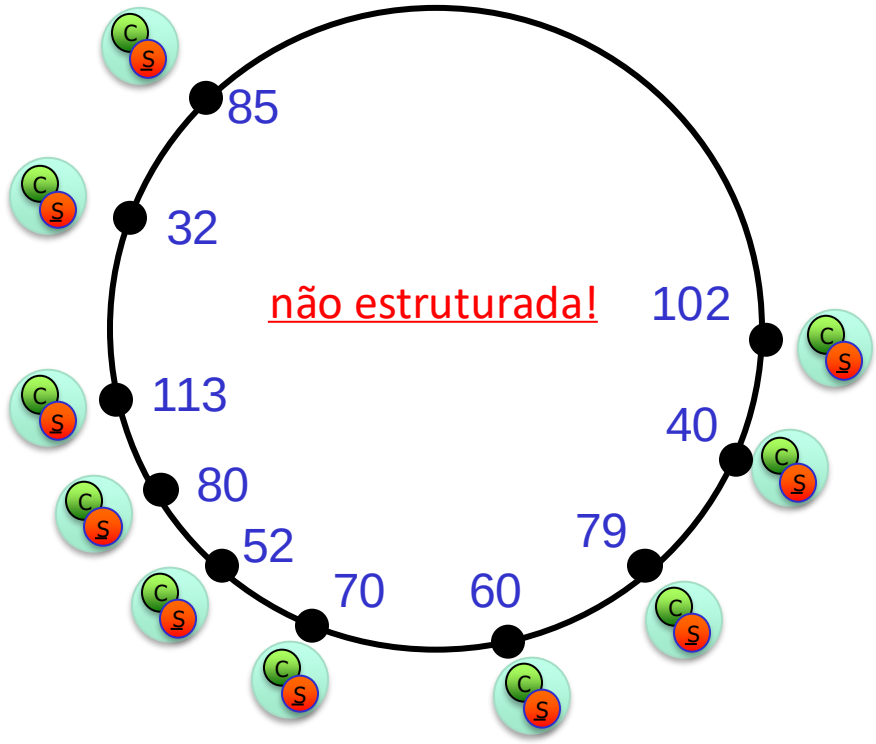
A topologia é induzida por uma relação (matemática) entre os endereços lógicos.

Existem topologias para todos os gostos...

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...

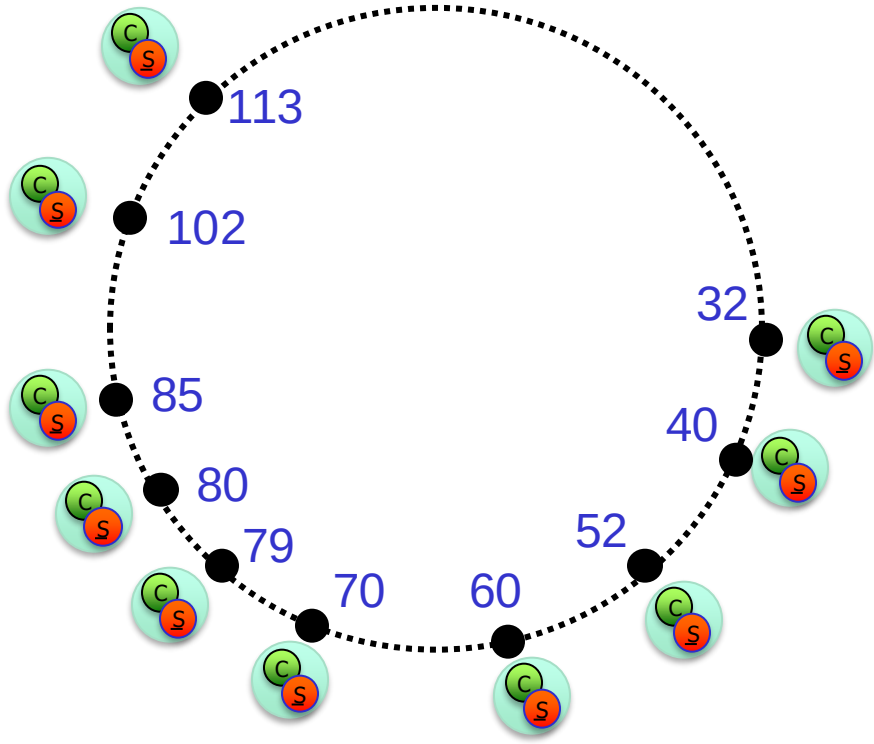


Analogia: árvore binária vs. árvore binária de pesquisa

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

SISTEMAS *P2P* ESTRUTURADOS

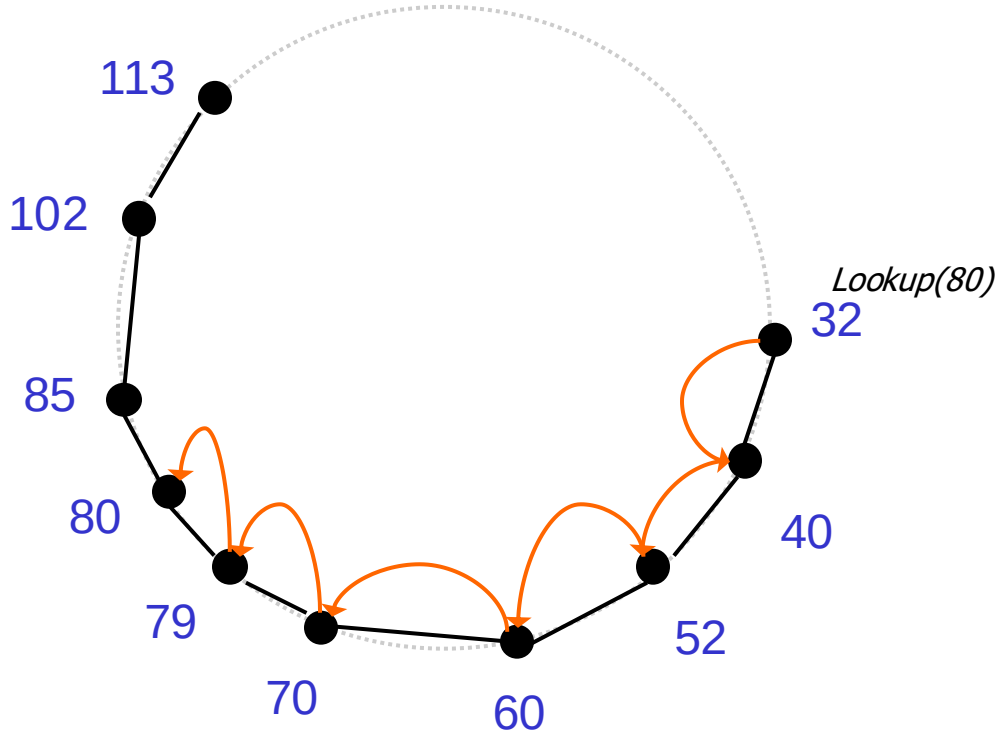
Encaminhamento e pesquisas por identificador O(log N)?



SISTEMAS *P2P* ESTRUTURADOS

Encaminhamento e pesquisas por identificador O(log N)?

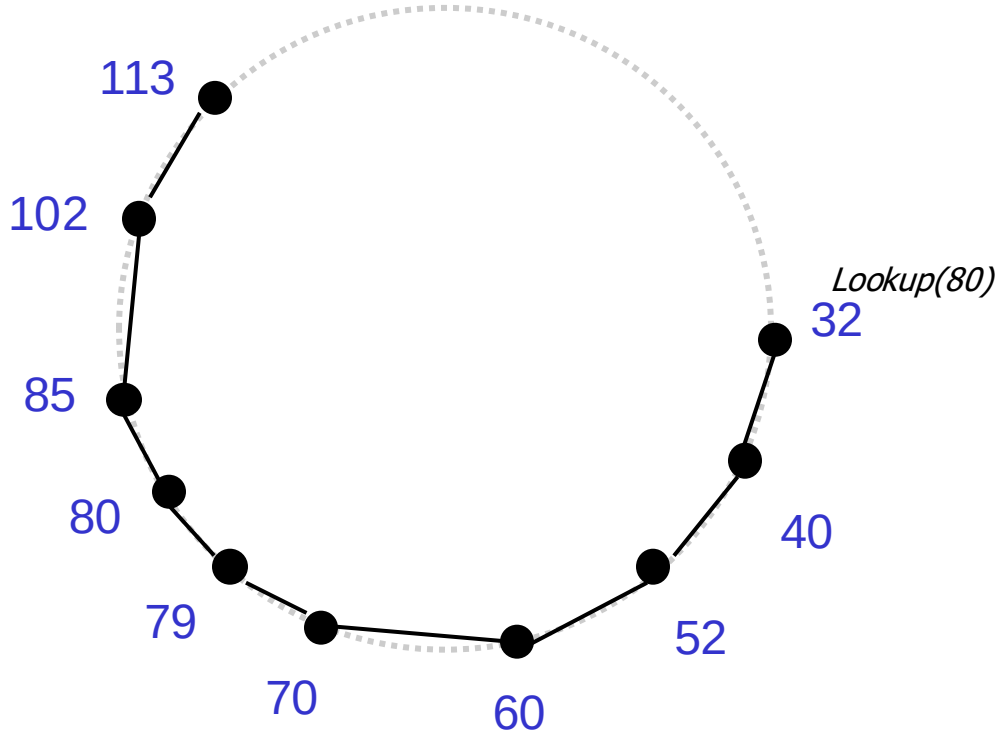
Seguir o sucessor não é solução; tem custo O(N)



SISTEMAS *P2P* ESTRUTURADOS

Encaminhamento e pesquisas por identificador O(log N)?

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

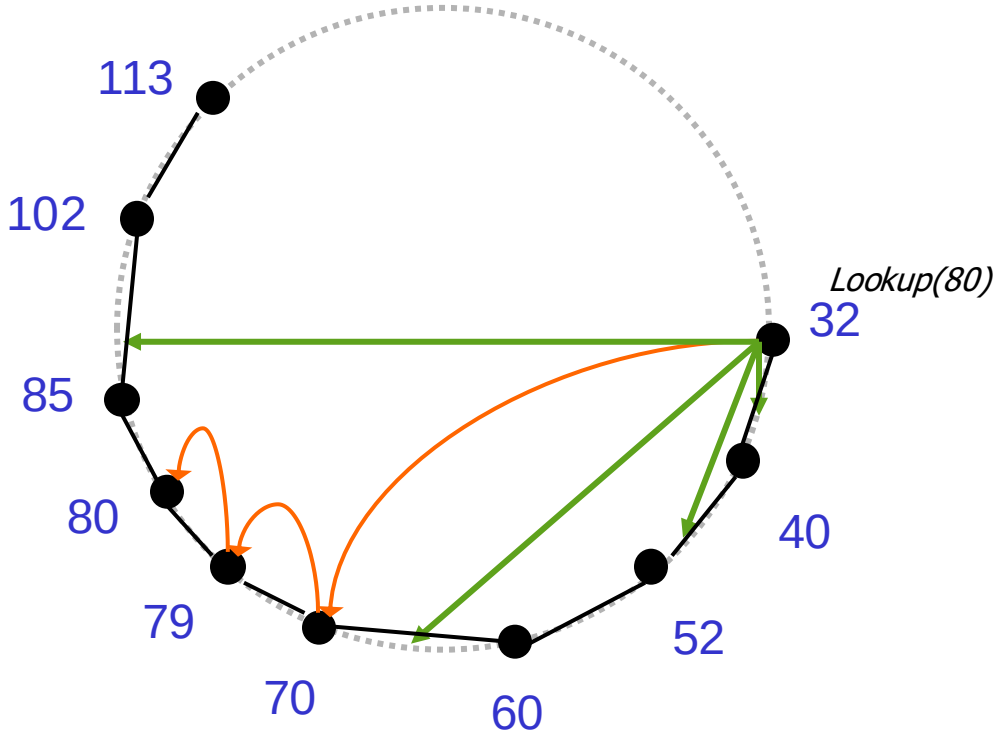


SISTEMAS *P2P* ESTRUTURADOS

Encaminhamento e pesquisas por identificador O(log N)?

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

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



SISTEMAS *P2P* ESTRUTURADOS

Exemplo canónico: 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

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM computer communication review*, *31*(4), 149-160.

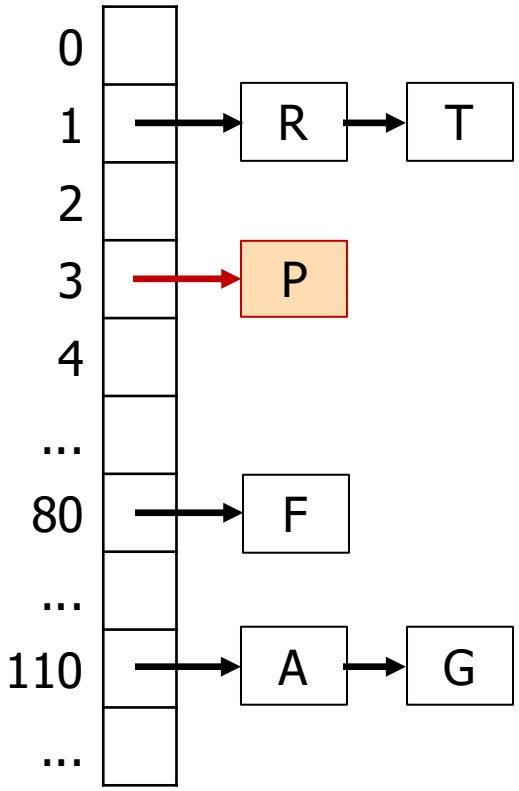
(Mais de 15000 citações segundo o Google scholar)

SISTEMAS *P2P* ESTRUTURADOS E DADOS

Um sistema P2P estruturado permite-nos descobrir um nó eficientemente (em log(n)).

Como é que podemos usar esta funcionalidade para guardar informação?

DISTRIBUTED HASH TABLE (DHT)

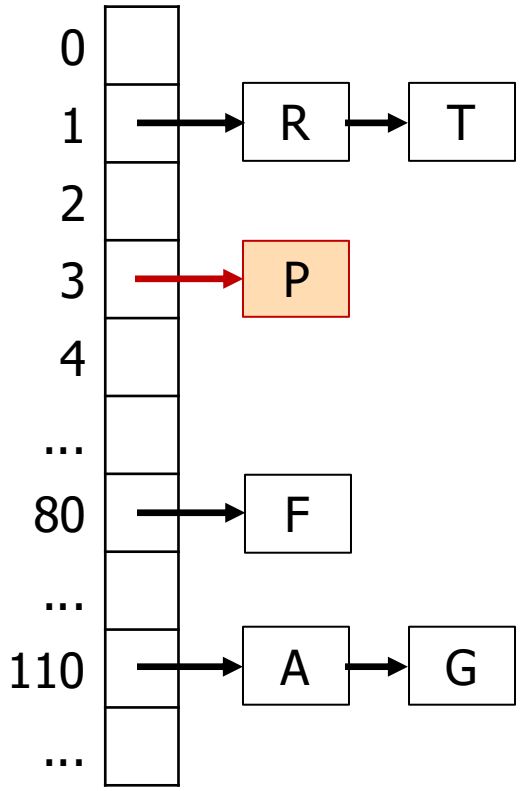


Numa **tabela de dispersão**, quando se quer guardar um valor, v, calcula-se o hash(v) e guarda-se o valor nessa posição.

E.g., para inserir P, calcula-se hash(P) = 3, e insere-se o valor na posição respetiva.

Nota: Em geral P é um par (chave,valor), sendo que neste caso só se faz hash(chave).

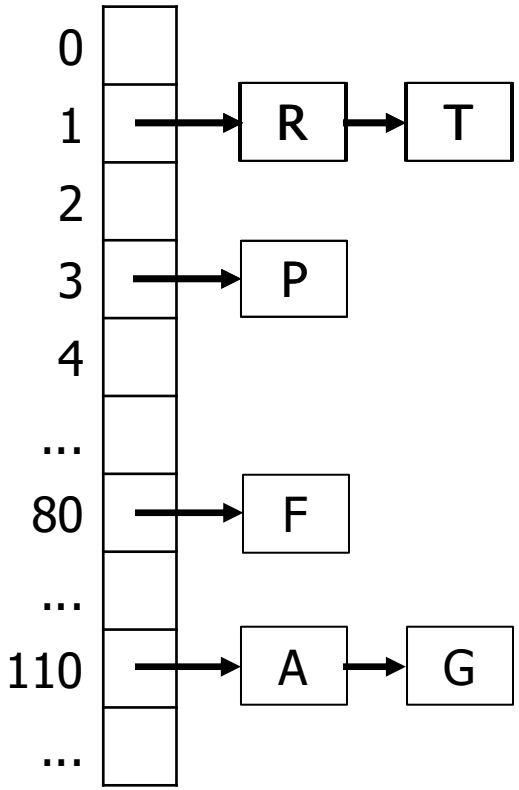
DISTRIBUTED HASH TABLE (DHT)

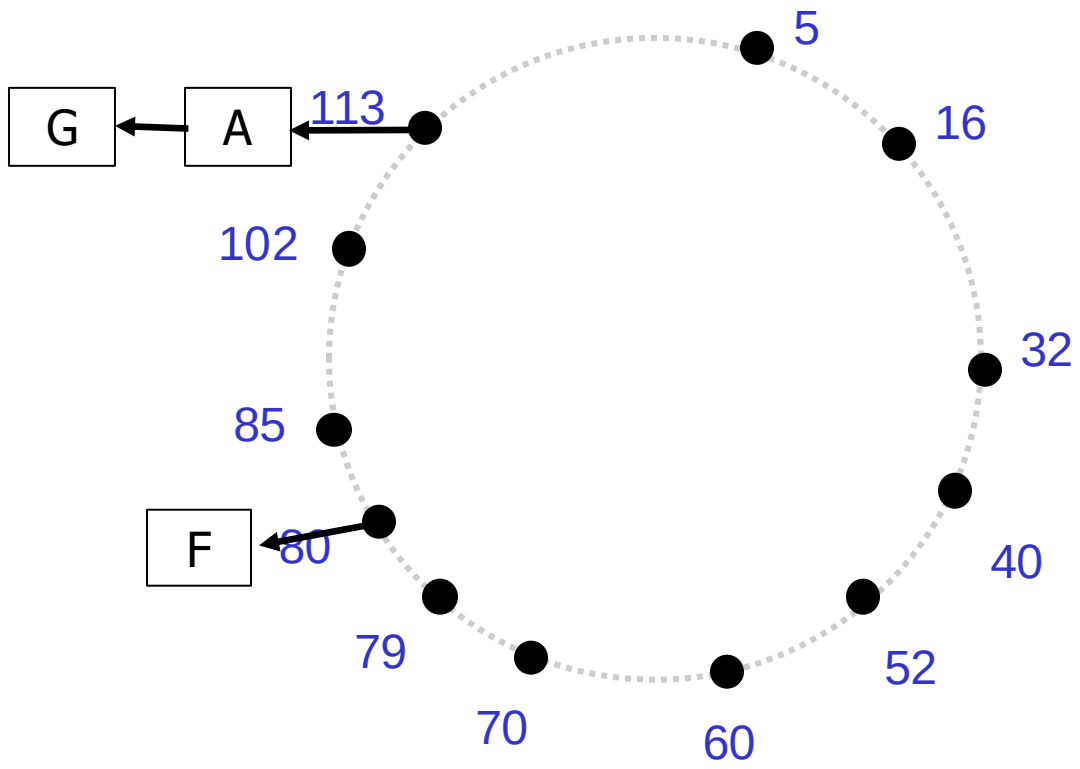


Numa **tabela de dispersão distribuída** (distributed hash table, DHT), vai-se usar a mesma ideia: vamos guardar ficheiros/blocos no servidor que tiver o identificador igual ao hash(chave).

DISTRIBUTED HASH TABLE (DHT)

Os dados que numa tabela de dispersão ficaria na posição n, na DHT ficam no nó com o identificador igual ou superior a n.



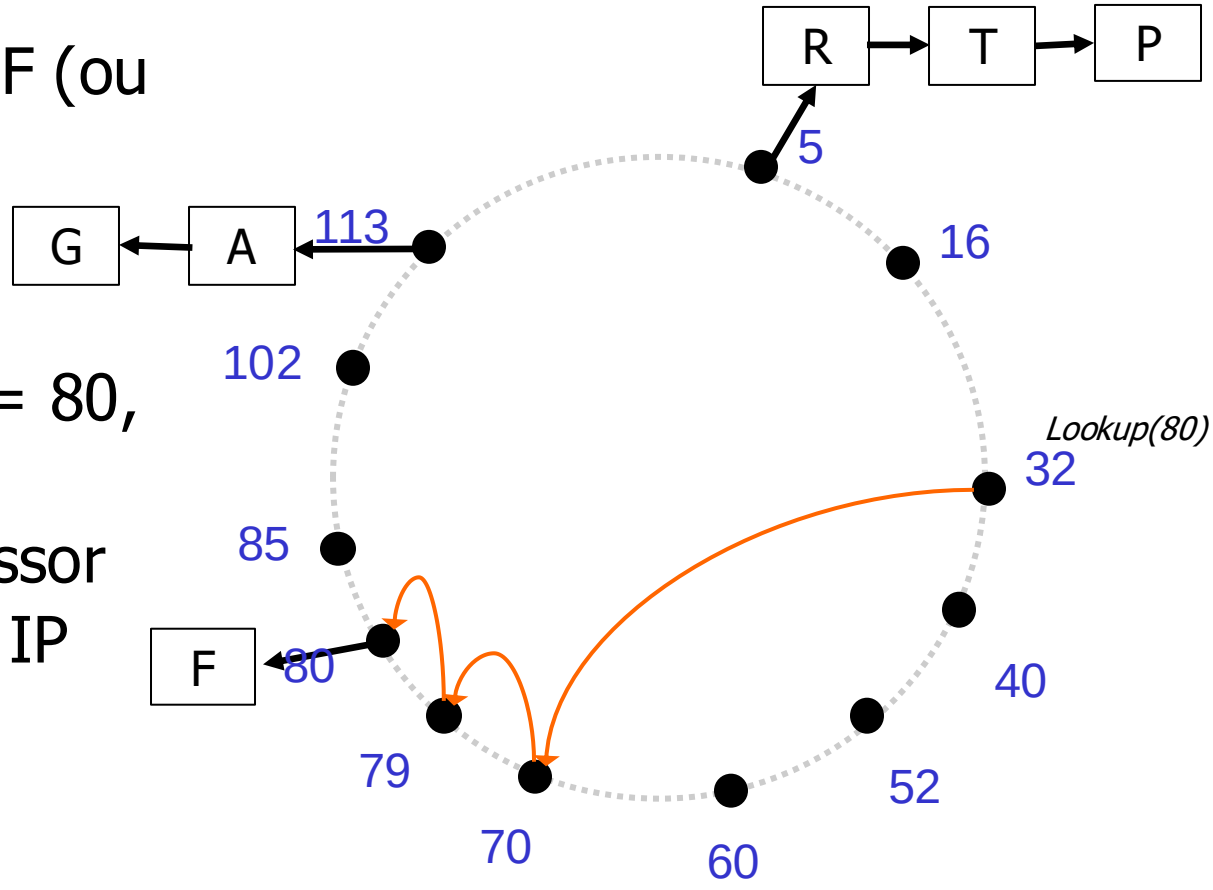


DISTRIBUTED HASH TABLE (DHT)

Para ler o ficheiro/bloco F (ou com chave F):

1. **lookup(F)->IP_{80}**

Lookup calcula hash(F) = 80, e usa sistema P2P para procurar nó 80 (ou sucessor de 80). Sistema devolve IP do nó 80.



DISTRIBUTED HASH TABLE (DHT)

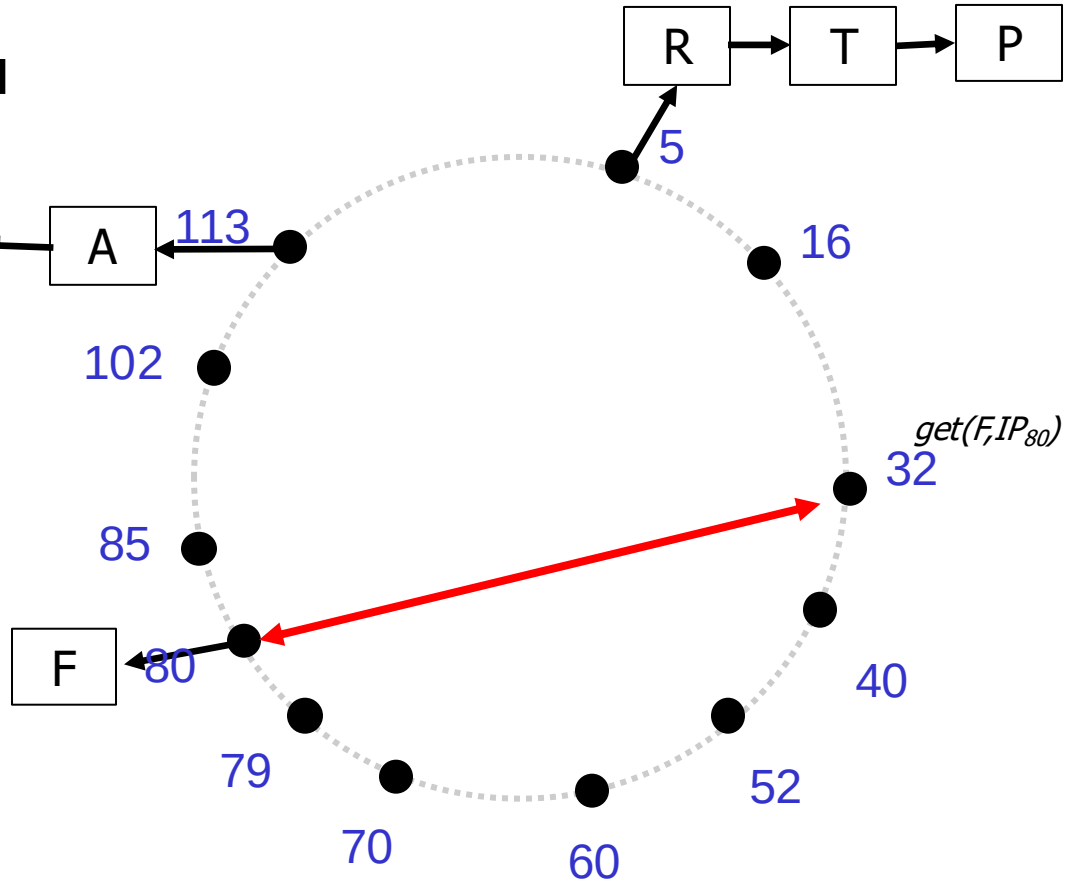
Para ler o ficheiro/bloco F (ou com chave F):

1. **lookup(F)->IP_{80}**

Lookup calcula hash(F) = 80, e usa sistema P2P para procurar nó 80 (ou sucessor de 80). Sistema devolve IP do nó 80.

2. **get(F,IP_{80})->valor**

Contacta-se diretamente o nó para obter os dados.

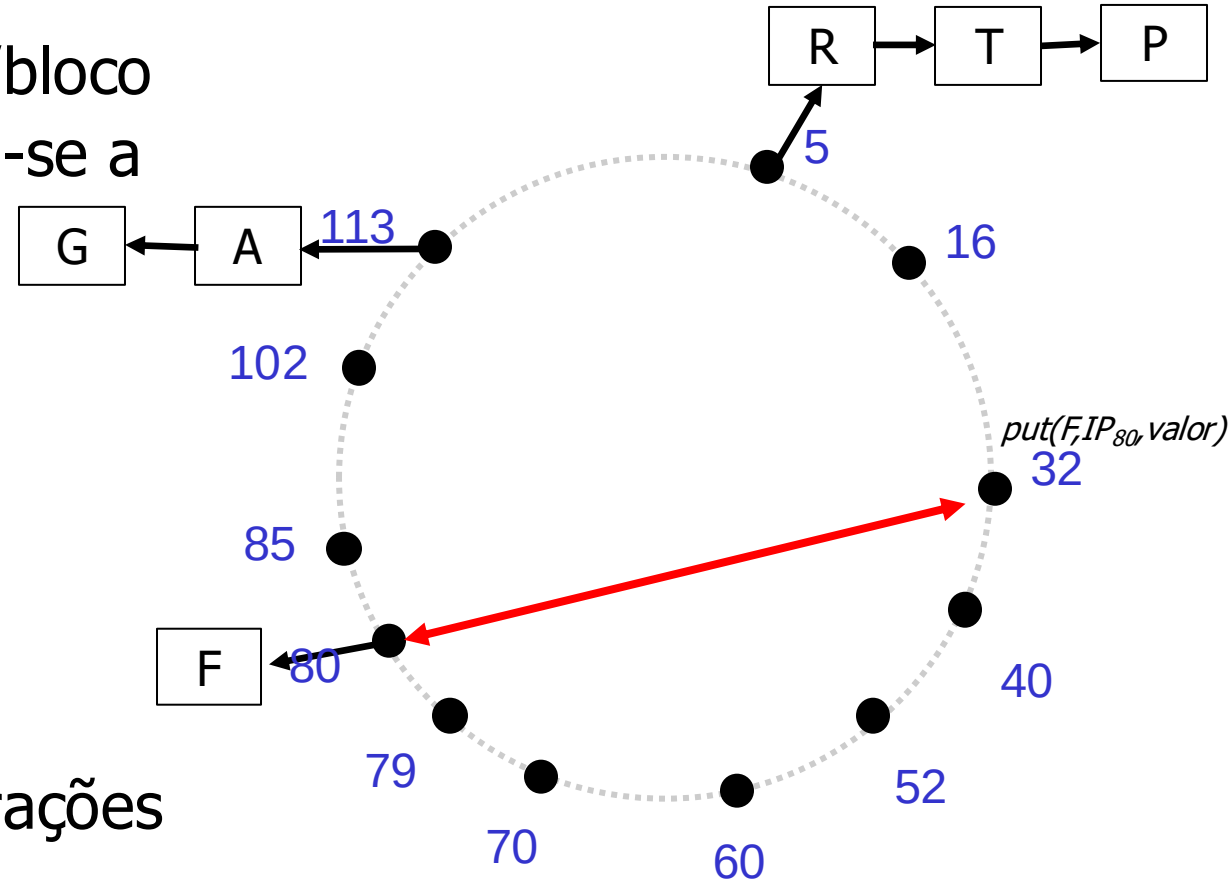


DISTRIBUTED HASH TABLE (DHT)

Para escrever o ficheiro/bloco F (ou com chave F), usa-se a mesma aproximação.

1. **lookup(F)->IP_{80}**
2. **put(F,IP_{80}, 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

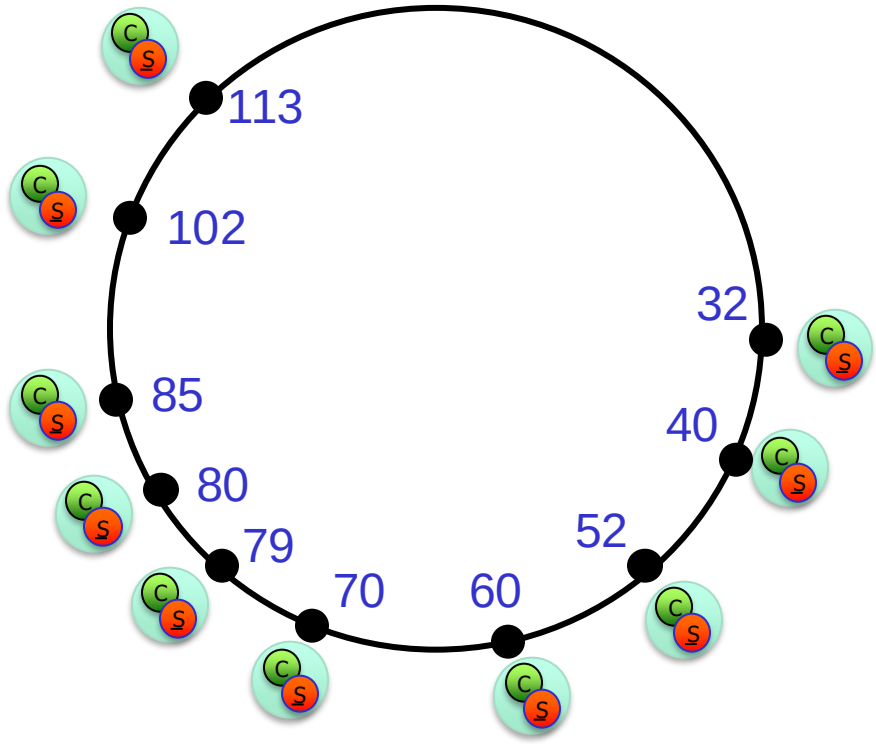
Aspetos importantes...

- 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.

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*)

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

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

UMA VARIANTE DE CLIENTE + (SERVIÇO) P2P

Sistemas peer-to-peer hierárquicos

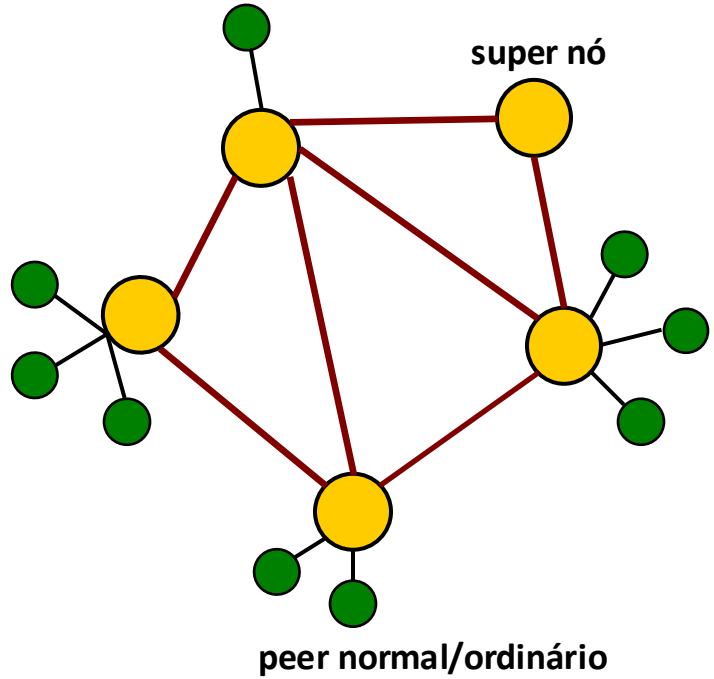
- Subconjunto de super-nós que se agrupam num sistema peer-to-peer
- Nós (regulares) ligam-se a um super-nó

...

GNUTELLA / EMULE / KAZAA

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:

- Diretamente entre **peers** (*firewalls* e *NAT boxes* podem ser um desafio)

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?
- Poderá ter também as desvantagens de ambos...

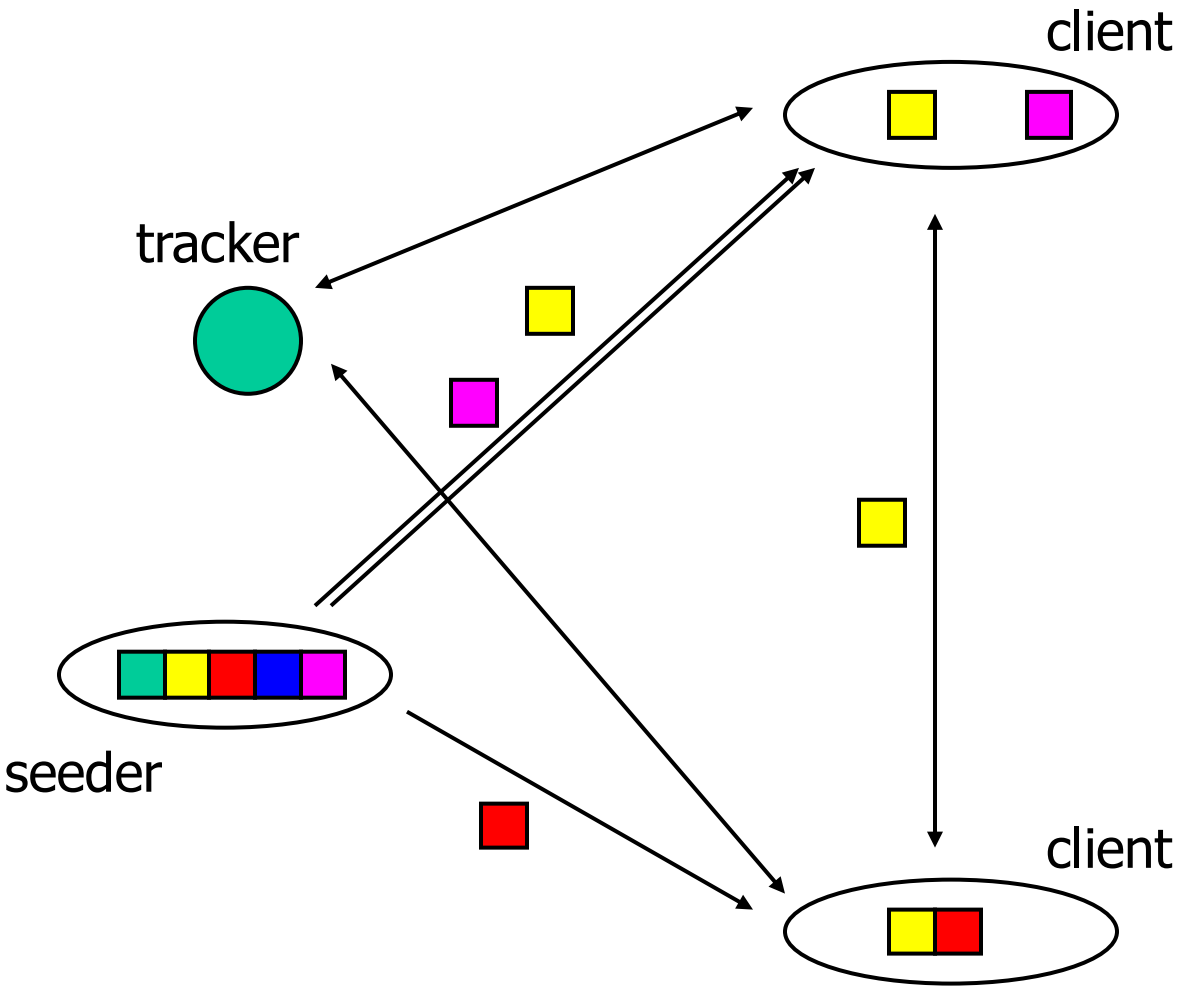
BitTorrent-versão original (simplificado)

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

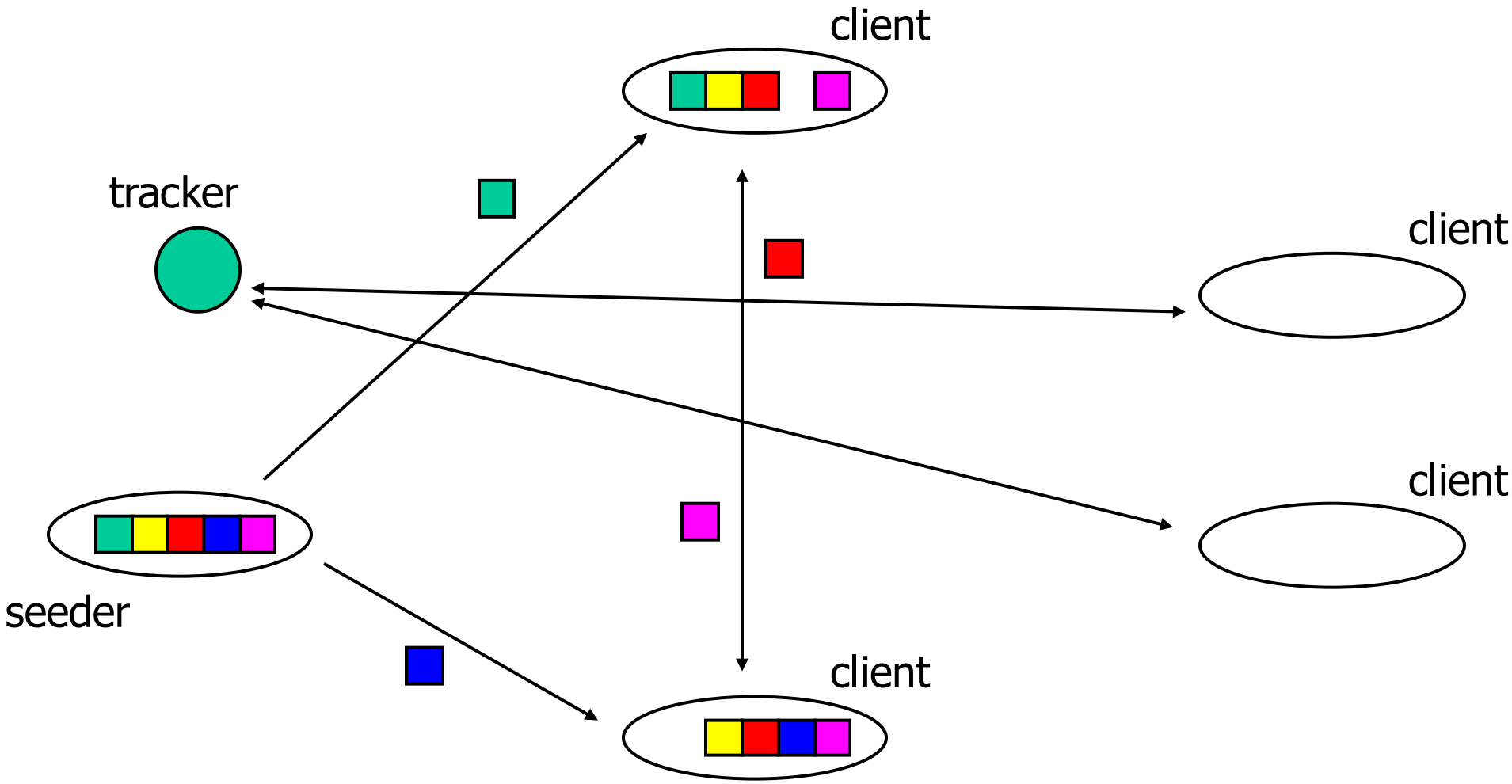
Arquitetura BitTorrent:

- 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?)

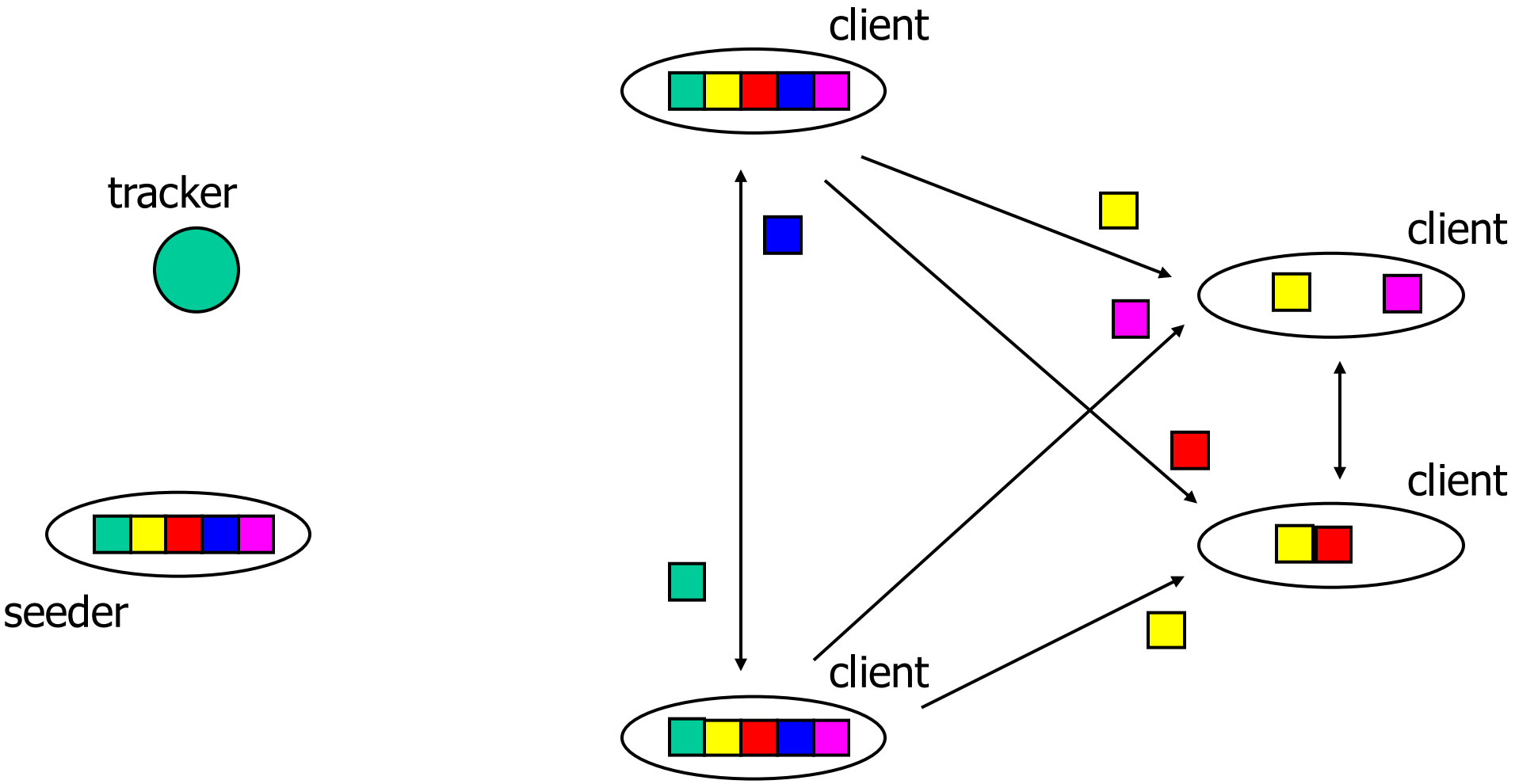
BITTORRENT



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 tipicamente no sistema Kademlia (circa 2002)

Petar Maymounkov and David Mazières. 2002. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS '01). Springer-Verlag, Berlin, Heidelberg, 53–65.

PARA SABER MAIS

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

Capítulo 2.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). "Chord: A scalable peer-to-peer lookup service for internet applications." ACM SIGCOMM computer communication review, 31(4), 149-160.

Petar Maymounkov and David Mazières. 2002. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric." In Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS '01). Springer-Verlag, Berlin, Heidelberg, 53–65.

João Leitão, José Pereira and Luís Rodrigues, "HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast," 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), Edinburgh, UK, 2007, pp. 419-429, doi: 10.1109/DSN.2007.56.

SISTEMAS DISTRIBUÍDOS

Aula 11: Capítulo 2

Arquiteturas e Modelos de Sistemas Distribuídos

NA ÚLTIMA AULA...

Variantes do Modelo Peer-to-Peer

- Não Estruturado (Aleatório)
- Estruturado & Distributed Hash Tables

Combinações do modelo Client-Servidor com Peer-to-Peer

- Servidor + P2P: Caso de Estudo - BitTorrent
- P2P Hierárquico: Caso de Estudo – Sistemas de download com pesquisa descentralizada

Na prática, tendem a existir mais componentes num sistema distribuído

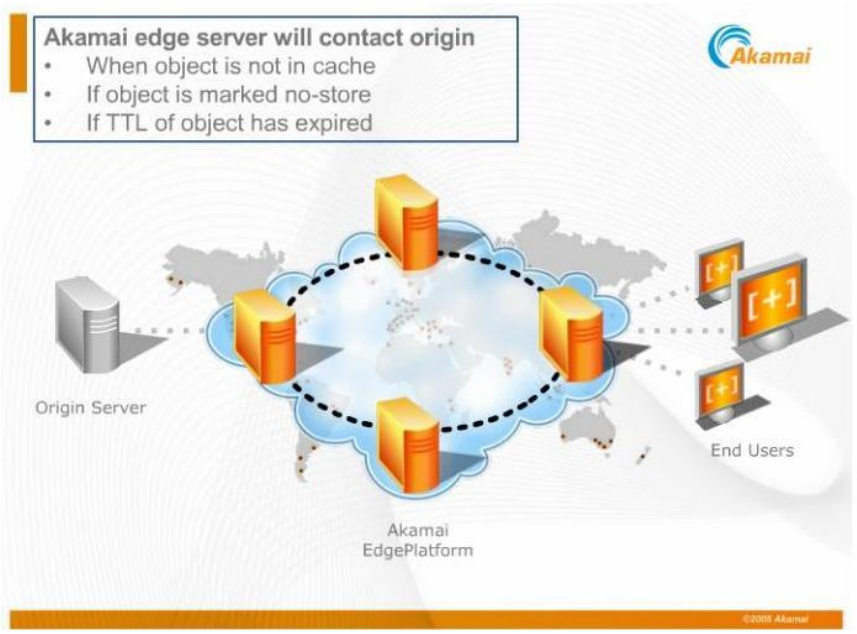
EDGE-SERVER

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.

Suportam **distribuição de conteúdos estáticos** de grande volume e/ou popularidade:

eg., streaming: Youtube, Netflix



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

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 interação, 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

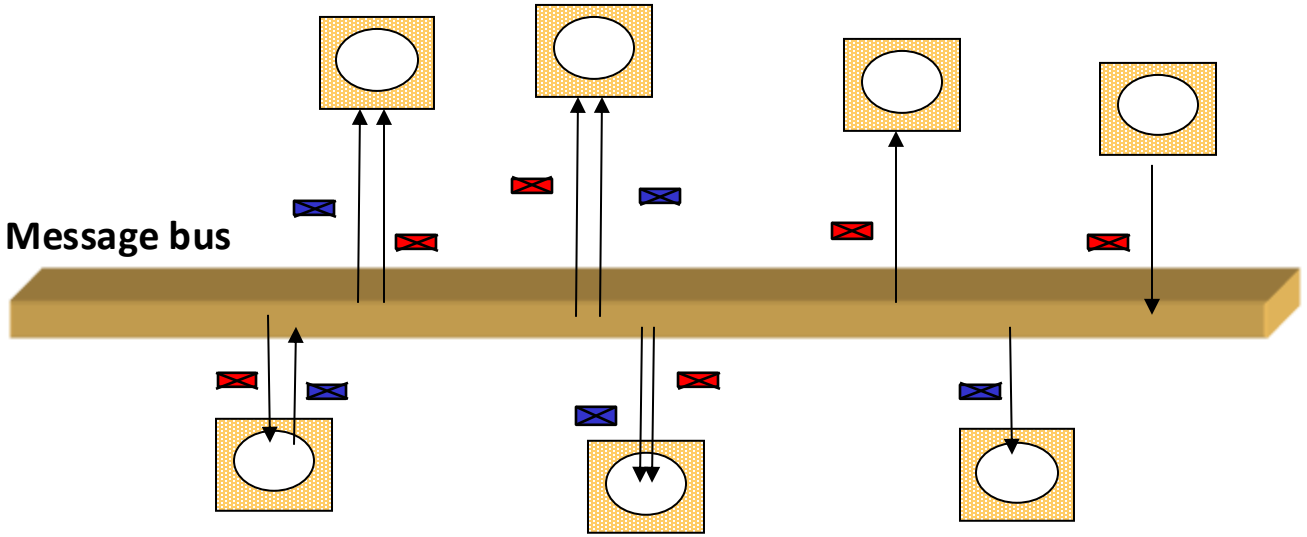
Como difundir informação sobre:

- Eventos dum jogo de futebol
- Cotações da bolsa

Propriedades

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

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

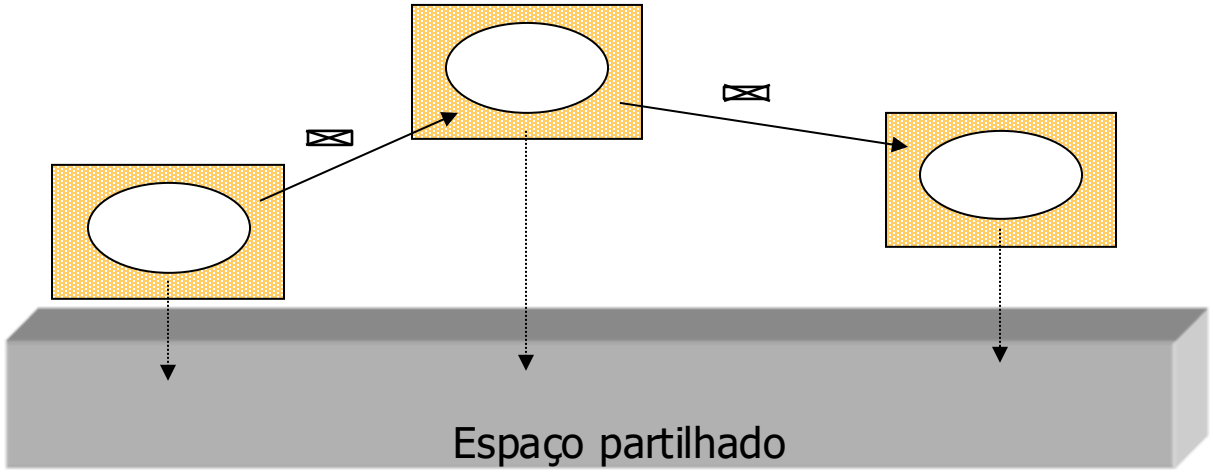


Modelo editor/assinante / "publish/subscribe" / pub/sub

- 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



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, DHT, 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

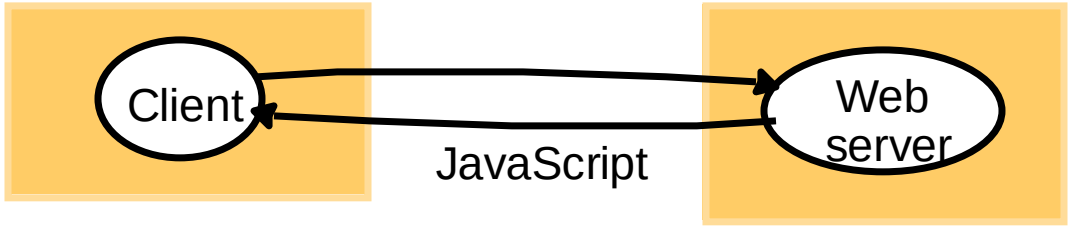
- Pedido (operação a invocar + parâmetros + inf. utilizador + ...)
- Resposta (resultado de operação + ...)
- Evento (num sistema de eventos)

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 javascript code



b) client interacts with the web application

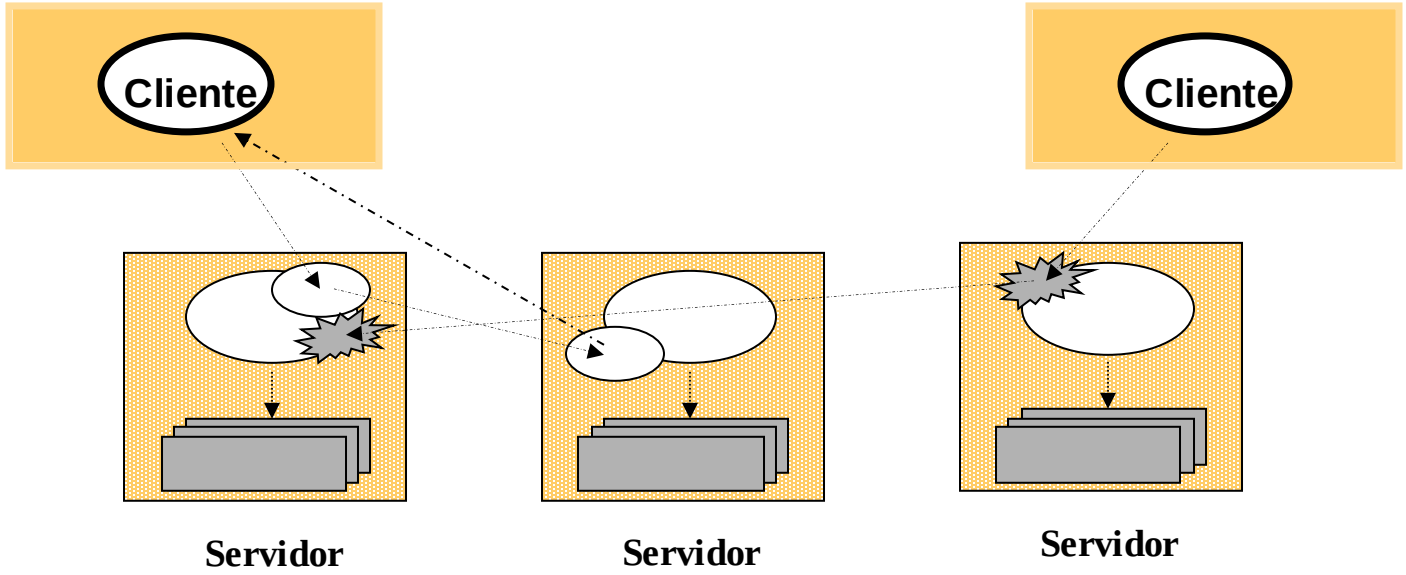
A execução do código no cliente pode:

- 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 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

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

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

EXEMPLO: TRABALHO PRÁTICO

Qual o modelo de falhas.

Falhas dos nós?

Não, no primeiro trabalho assume-se que os nós não falham.

Falhas de comunicação?

Sim, falhas por **omissão, temporárias**, i.e., as mensagens podem-se perder durante um período de tempo limitado.

SEGURANÇA NUM SISTEMA DISTRIBUÍDO (MODELO DE SEGURANÇA)

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

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.

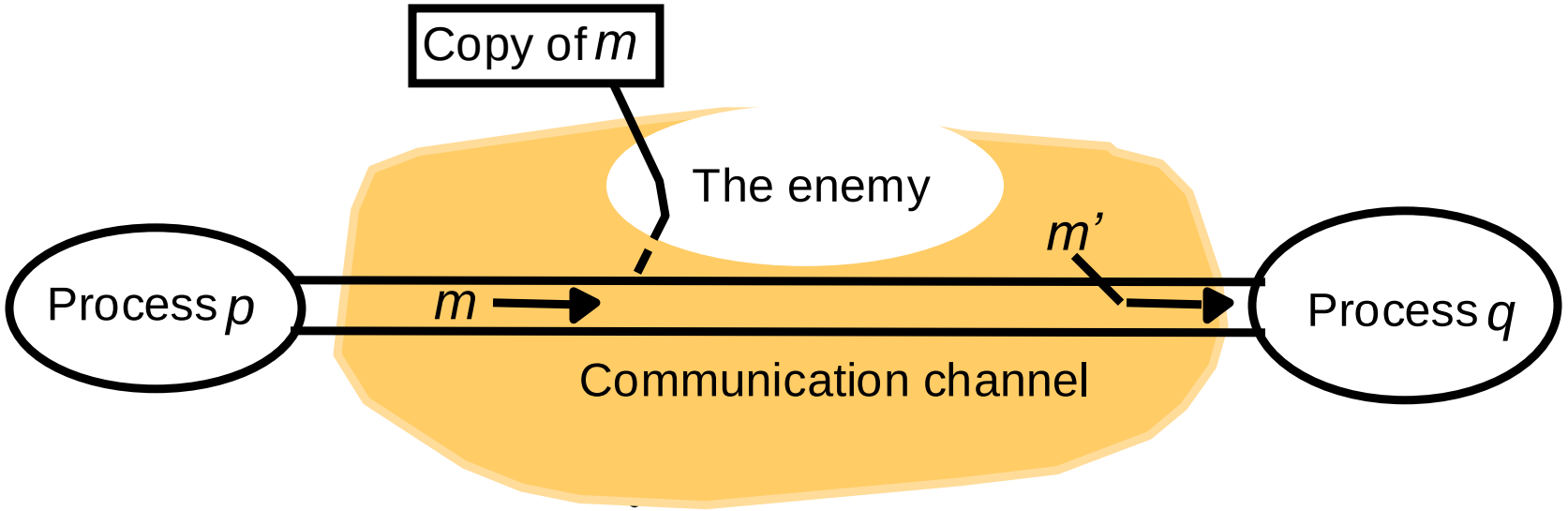
SEGURANÇA NUM SISTEMA DISTRIBUÍDO (MODELO DE SEGURANÇA)

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

Por outra palavras, consiste em **definir o modelo do atacante**.

Enumerando o que o atacante pode ou não fazer, sem que o sistema fique comprometido.

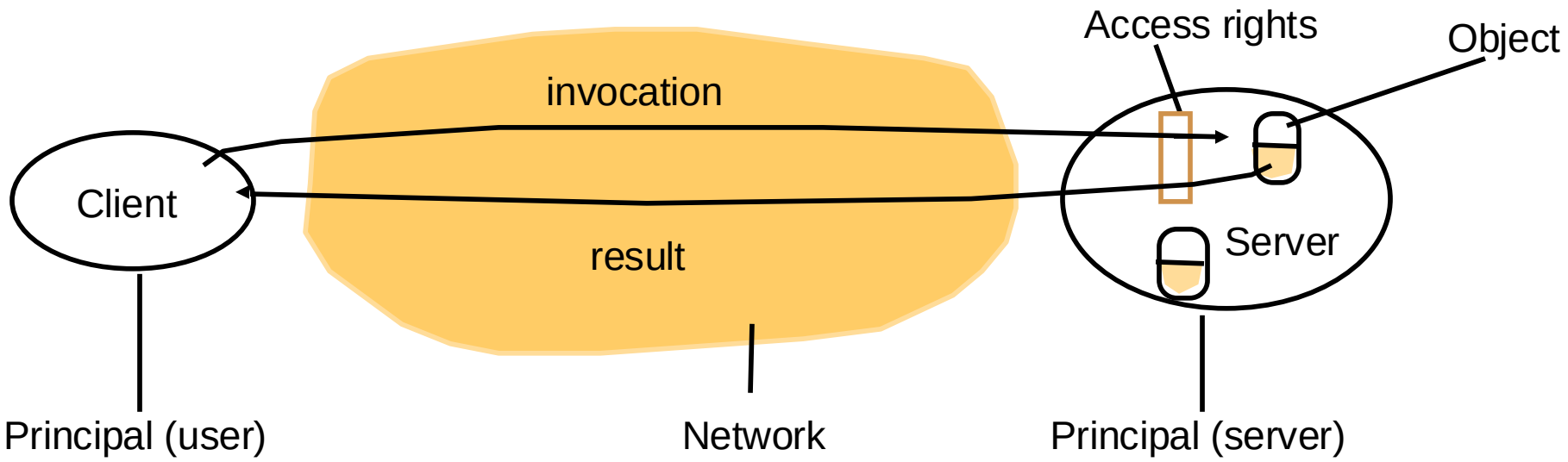
AMEAÇAS BÁSICAS AOS CANAIS DE COMUNICAÇÃO (E CONSEQUENTEMENTE AOS PROCESSOS)



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.

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

OUTRAS AMEAÇAS

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

Distributed Denial of service: versão em o ataque de negação de serviço é desencadeado de forma distribuída (e tipicamente descentralizada) por vários agentes simultaneamente.

Muito difícil de prevenir e muito dispendioso

e.g., Akamai DDoS Protection

PARA SABER MAIS

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

Capítulo 2.