

Sistemas Distribuídos

Rui Oliveira
Departamento de Informática
Universidade do Minho

Sistemas Distribuídos

(Apontamentos baseados no livro *Distributed Systems: Principles and Paradigms*, A. Tanenbaum e M. Van Steen)



Sistemas distribuídos

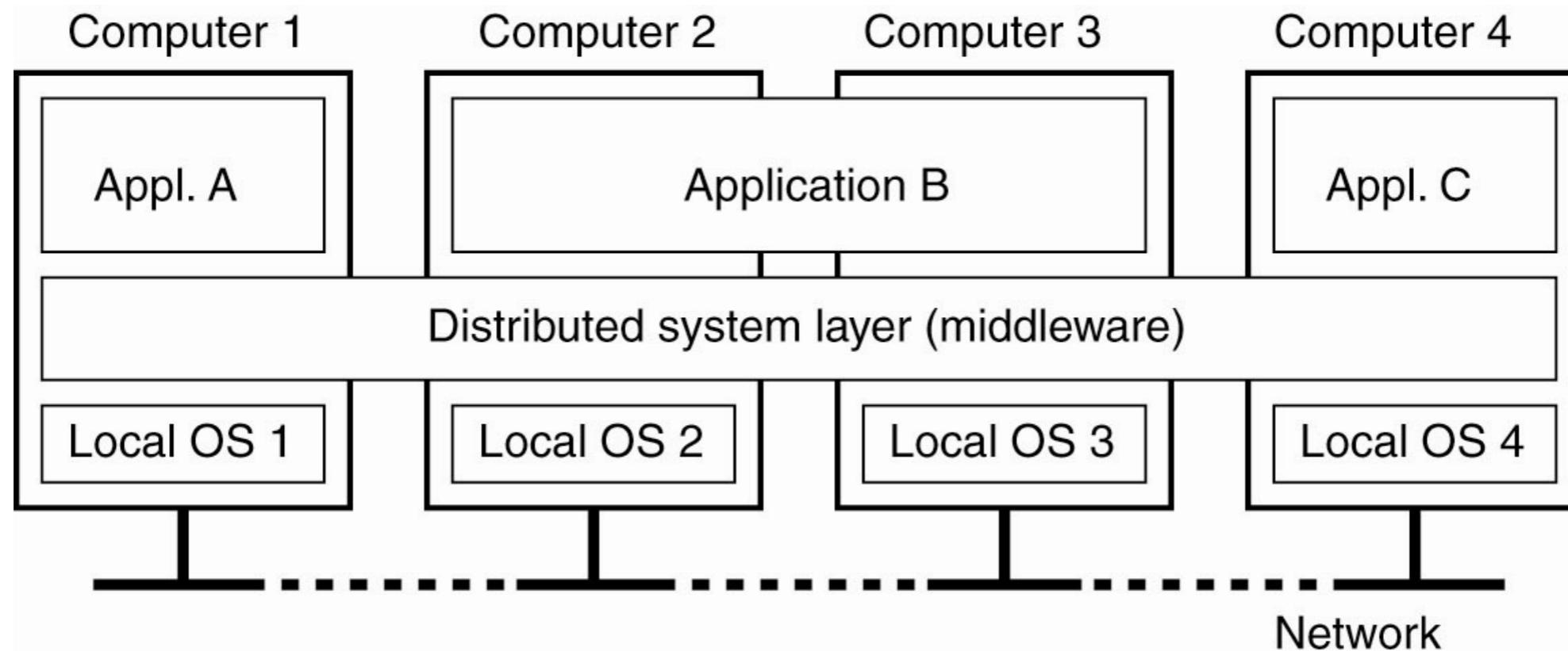
- Introdução
- Arquiteturas e Modelos
- Comunicação
- Sistemas de objetos distribuídos
- Sistemas de ficheiros distribuídos
- Sistemas Web

Sistemas distribuídos

- Introdução
- Arquiteturas e Modelos
- Comunicação
- Sistemas de objetos distribuídos
- Sistemas de ficheiros distribuídos
- Sistemas Web

- Um conjunto de computadores independentes que é apresentado aos utilizadores como um sistema único e coerente. [Tanenbaum & Van Steen]
- Um conjunto de computadores ligados em rede, com software que permite a partilha de recursos e a coordenação de actividades, oferecendo idealmente um ambiente integrado. [Coulouris et al.]
- É um sistema em que a falha de uma máquina da qual nunca se ouviu falar pode tornar a nossa própria máquina inútil. [Lamport]

Definição de Sistema Distribuído



Um sistema distribuído obtido por “middleware”.

A camada de middleware estende-se pelas várias máquinas e oferece a cada aplicação a mesma interface.

- Facilitar o acesso a recursos remotos/distribuídos
- Tornar a distribuição de recursos transparente
- Abertura e extensibilidade
- Escalabilidade

- Acesso: mascara diferenças na representação dos dados na forma como os recursos são acedidos.
- Localização: esconde a localização dos recursos
- Migração: esconde as mudanças de localização de recursos
- Recolocação: esconde as mudanças de localização de recursos durante a sua utilização
- Replicação: esconde a replicação de recursos
- Concorrência: esconde o acesso partilhado a recursos
- Falhas: mascara a falha e recuperação de recursos

- Um Sistema Distribuído diz-se aberto se oferece serviços de acordo com regras standard que descrevem a sua sintaxe e semântica
- Os serviços são normalmente descritos por linguagens de definição de interface – IDL – que capturam apenas a sintaxe
- A descrição dos serviços deve ser neutra relativamente a implementações promovendo a interoperabilidade e portabilidade dos sistemas
- Um sistema aberto é facilmente composto e estendido por componentes diferentes

• Vectores de escalabilidade

- Número de utilizadores e/ou processos (tamanho)
- Distância máxima entre nós (geográfica)
- Número de domínios administrativos (administração)

Técnicas para a escalabilidade

- Mascarar as latências de comunicação
 - Evitar esperar por respostas, comunicação assíncrona
- Distribuição
 - Dividir o esforço de computação
 - Colocar computação nos clientes
- Replicação e Caching
 - Disponibilizar cópias dos dados em vários nós

- Os canais de comunicação são fiáveis
- Os canais de comunicação são seguros
- A rede é homogénea
- A topologia da rede não se altera
- A latência da comunicação é nula
- A largura de banda é infinita
- O custo de comunicação é nulo
- Há apenas um domínio de administração

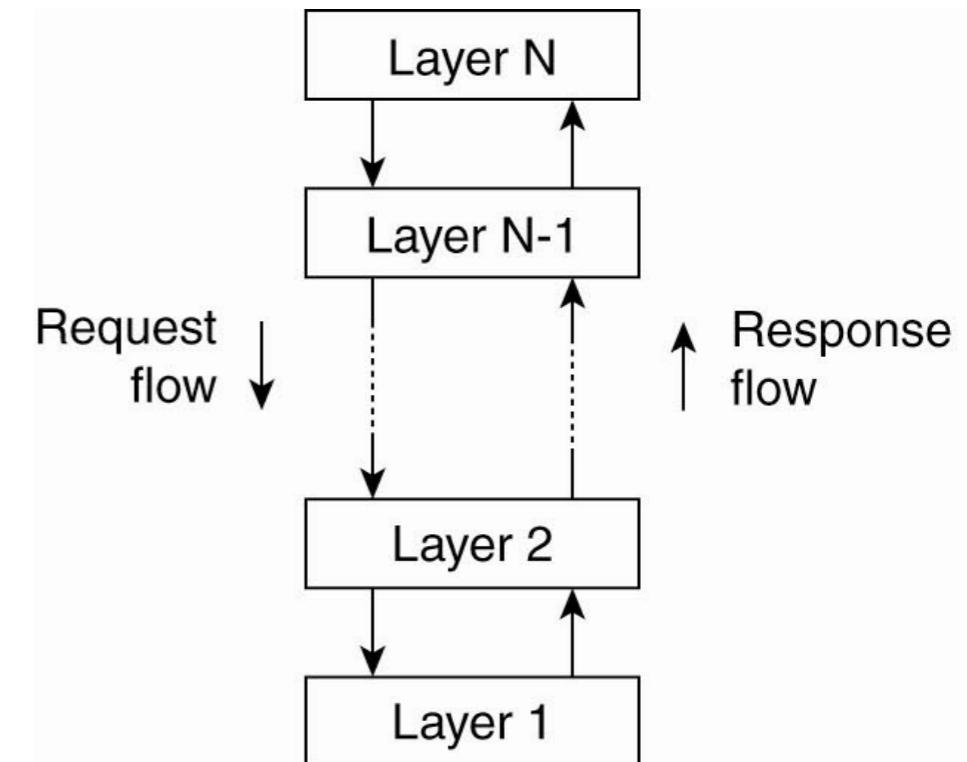
Sistemas distribuídos

-  Introdução
-  Arquiteturas e Modelos
-  Comunicação
-  Sistemas de objetos distribuídos
-  Sistemas de ficheiros distribuídos
-  Sistemas Web

- Organizar de forma lógica os diversos componentes do sistema e distribuí-los pelas diversas máquinas disponíveis.

- Tipos mais relevantes de arquiteturas:

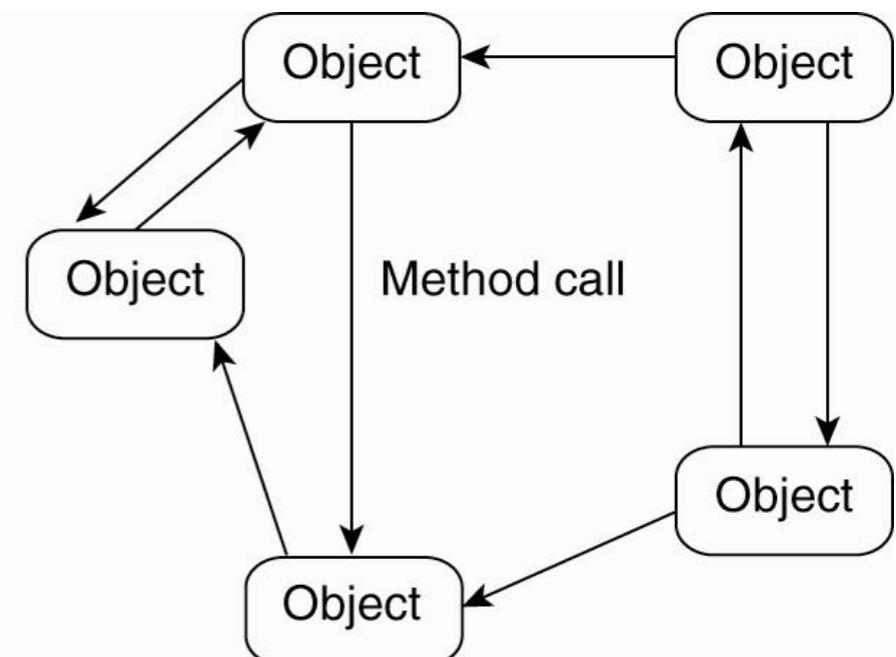
- Arquiteturas em camadas
- Arquiteturas baseadas em objetos
- Arquiteturas baseadas nos dados
- Arquiteturas baseadas em eventos



- Organizar de forma lógica os diversos componentes do sistema e distribuí-los pelas diversas máquinas disponíveis.

- Tipos mais relevantes de arquiteturas:

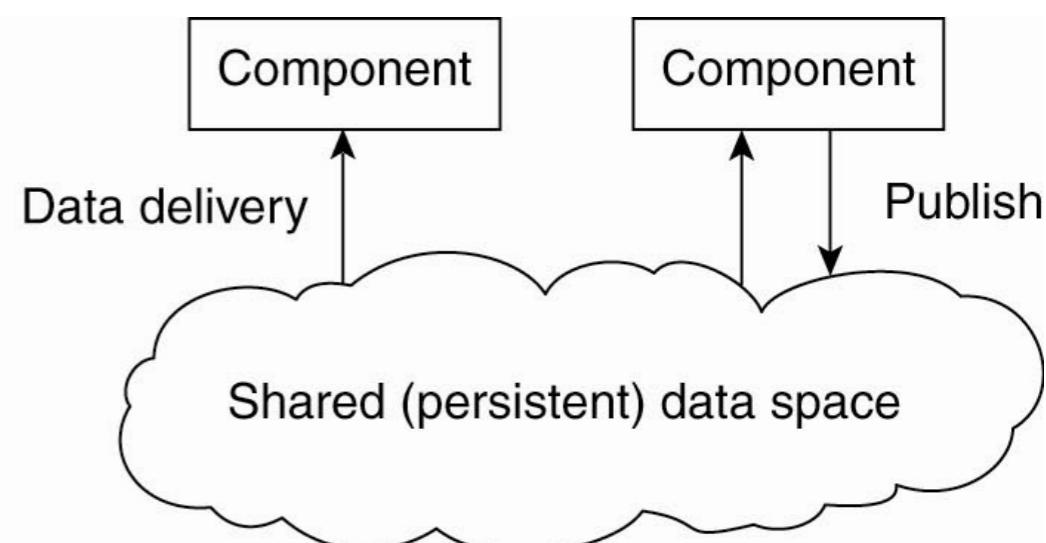
- Arquiteturas em camadas
- Arquiteturas baseadas em objetos
- Arquiteturas baseadas nos dados
- Arquiteturas baseadas em eventos



- Organizar de forma lógica os diversos componentes do sistema e distribuí-los pelas diversas máquinas disponíveis.

- Tipos mais relevantes de arquiteturas:

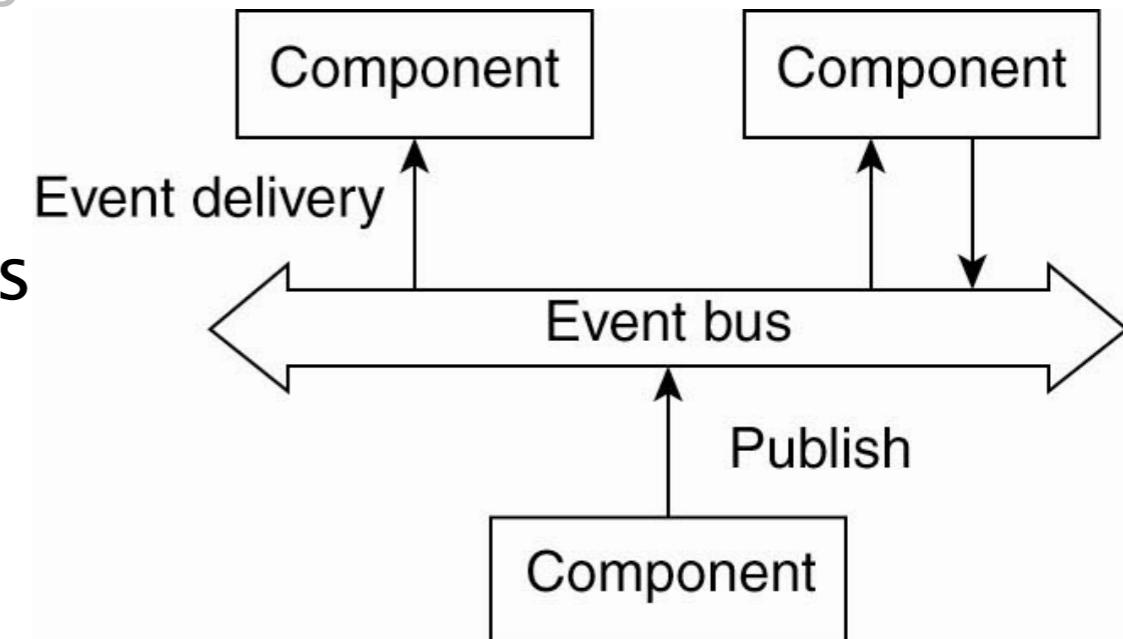
- Arquiteturas em camadas
- Arquiteturas baseadas em objetos
- Arquiteturas baseadas nos dados**
- Arquiteturas baseadas em eventos



- Organizar de forma lógica os diversos componentes do sistema e distribuí-los pelas diversas máquinas disponíveis.

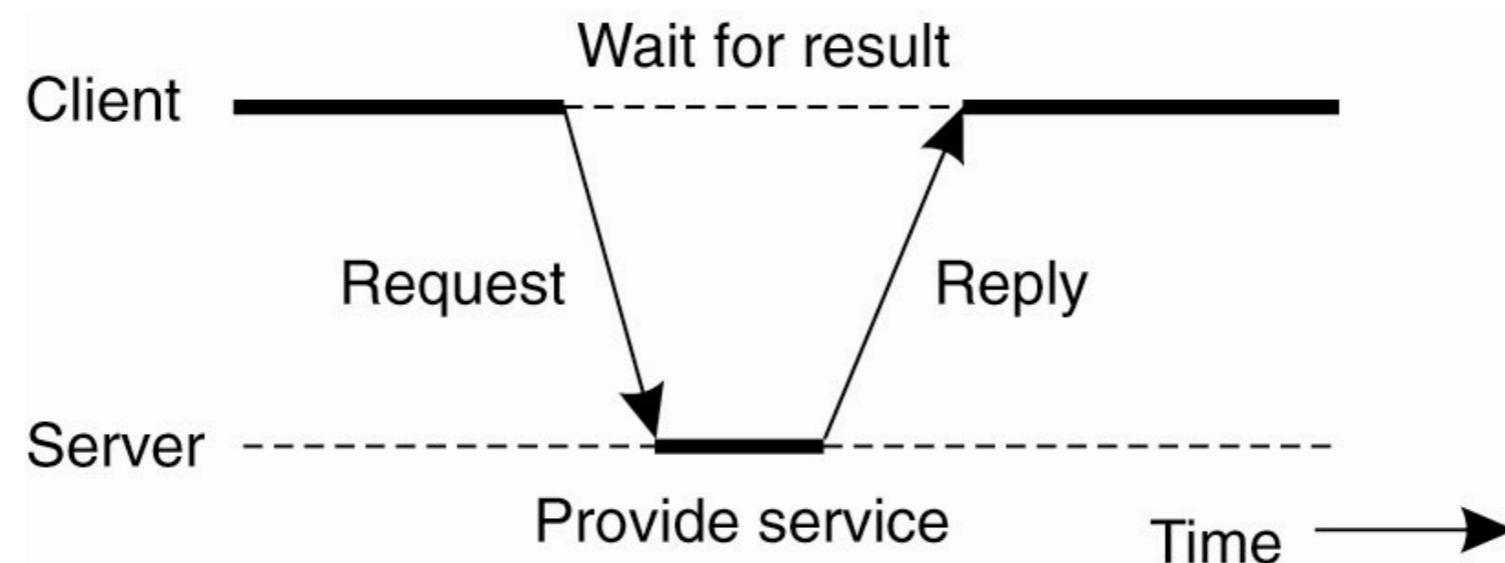
- Tipos mais relevantes de arquiteturas:

- Arquiteturas em camadas
- Arquiteturas baseadas em objetos
- Arquiteturas baseadas nos dados
- Arquiteturas baseadas em eventos

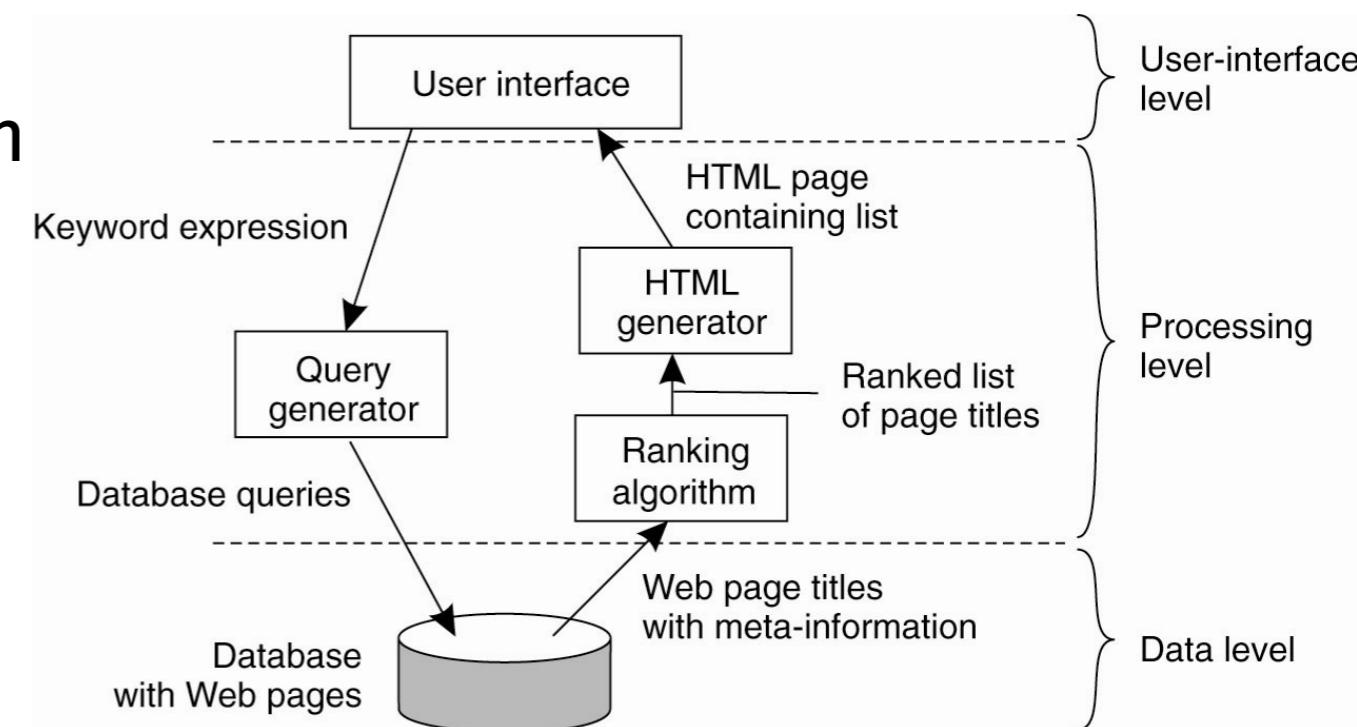


Modelo Cliente/Servidor

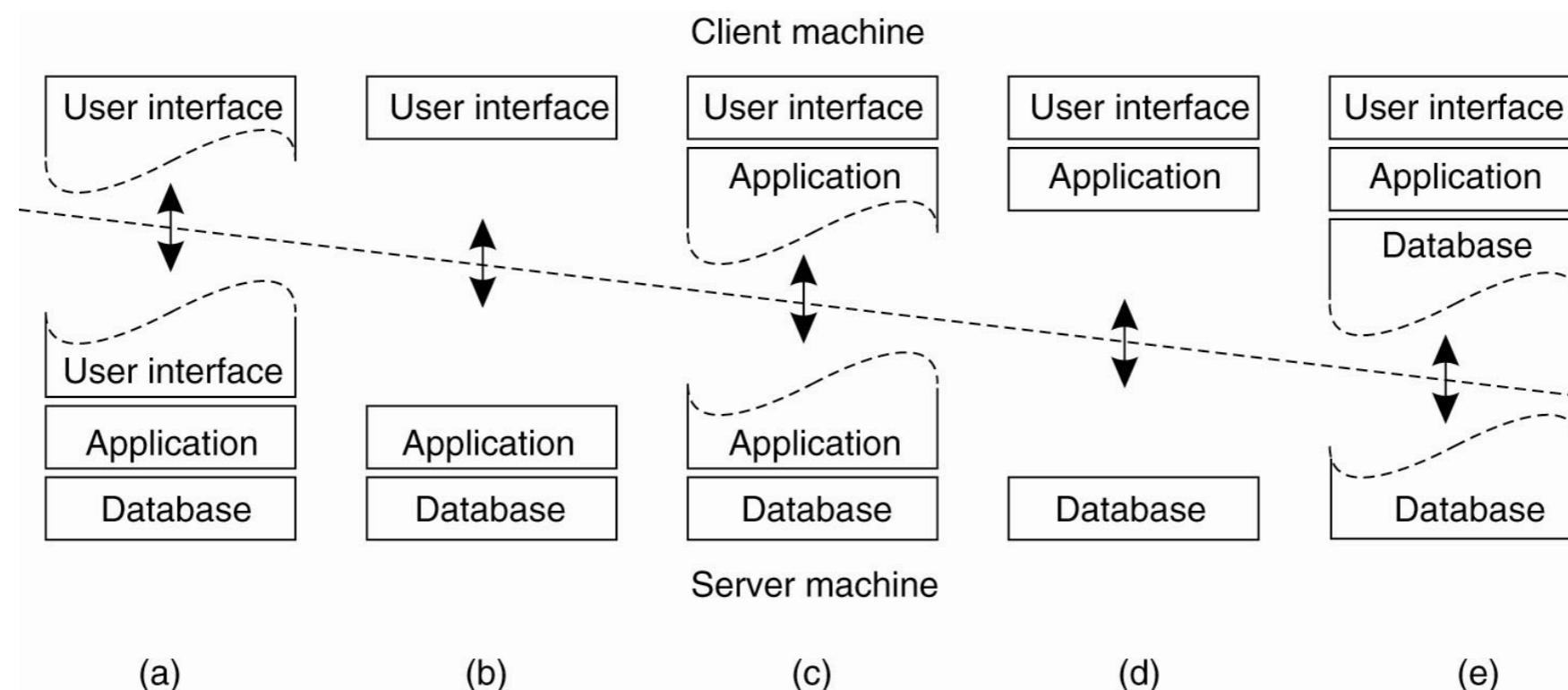
- Há processos que disponibilizam serviços: **servidores**
- Há processos que utilizam esses serviços: **clientes**
- Clientes e servidores podem estar em máquinas distintas
- A computação é uma sequencia de pares pedido/resposta



- A estruturação das aplicações é normalmente feita em **camadas**
- O exemplo típico é a estruturação em 3 camadas:
 - A camada responsável pela **interface com o utilizador**
 - A camada de **processamento** com a semântica da aplicação
 - A camada de **dados** que contém e gere todos os dados do utilizador a serem manipulados pela camada de processamento

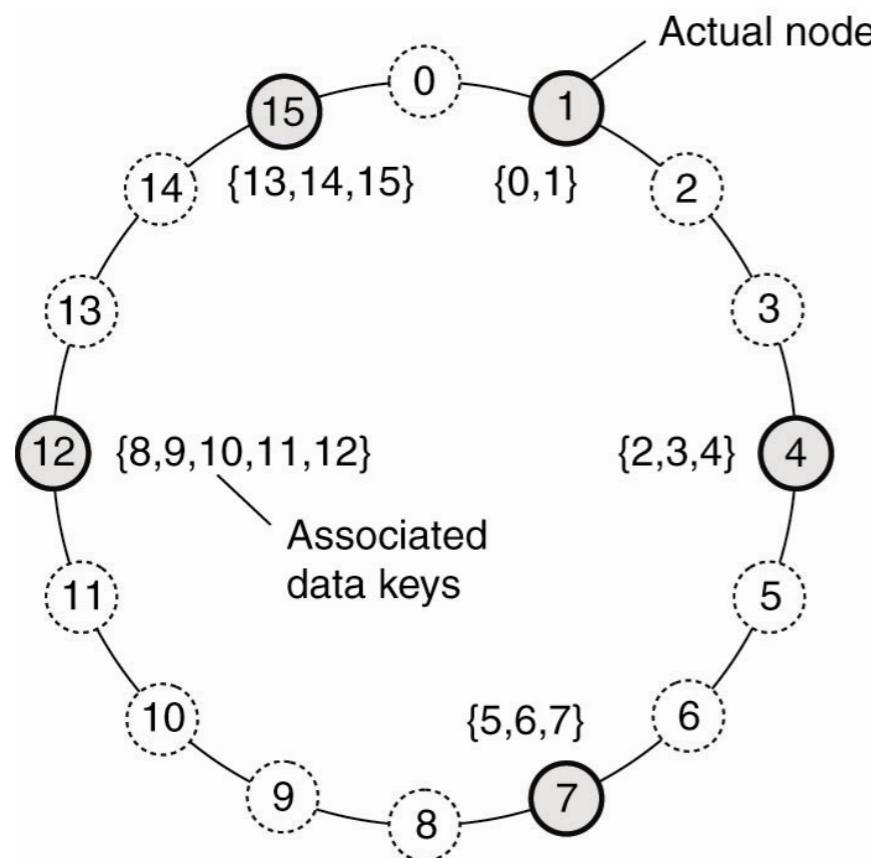


A divisão das diversas camadas por máquinas pode variar devido a fatores específicos da aplicação, das máquinas disponíveis, das condições dos canais de comunicação e ainda dos objectivos não funcionais pretendidos



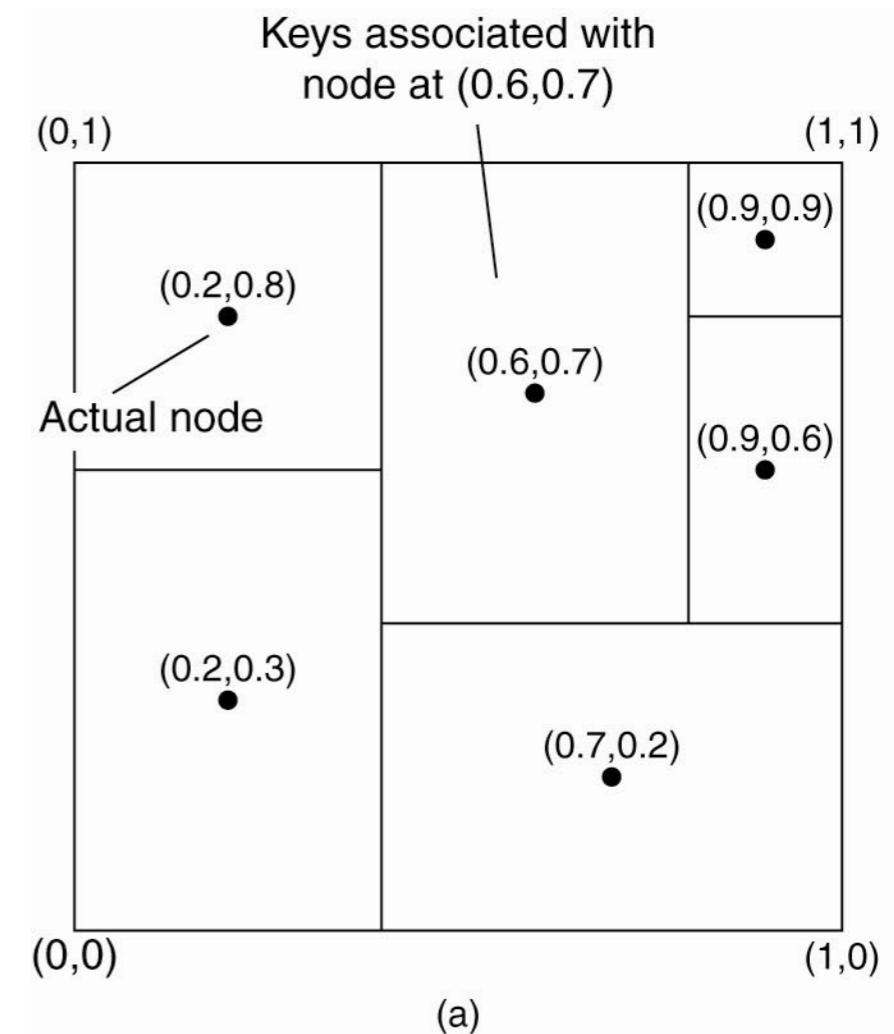
- Em alternativa às arquiteturas centralizadas, cliente/servidor abundam hoje sistemas organizados de forma totalmente descentralizada – sistemas par-a-par ou peer-to-peer (P2P)
- Estes sistemas podem ser estruturados, casos em que obedecem a uma estrutura específica para a distribuição dos **dados**, ou não-estruturados em que a interação entre pares é feita de forma aleatória.
- Em qualquer das formas a gestão e utilização do sistema é baseada numa rede lógica sobreposta à estrutura física do sistema – overlay network.

Exemplos de sistemas P2P estruturados, a estrutura é obtida através de uma DHT, tabela de Hash distribuída:



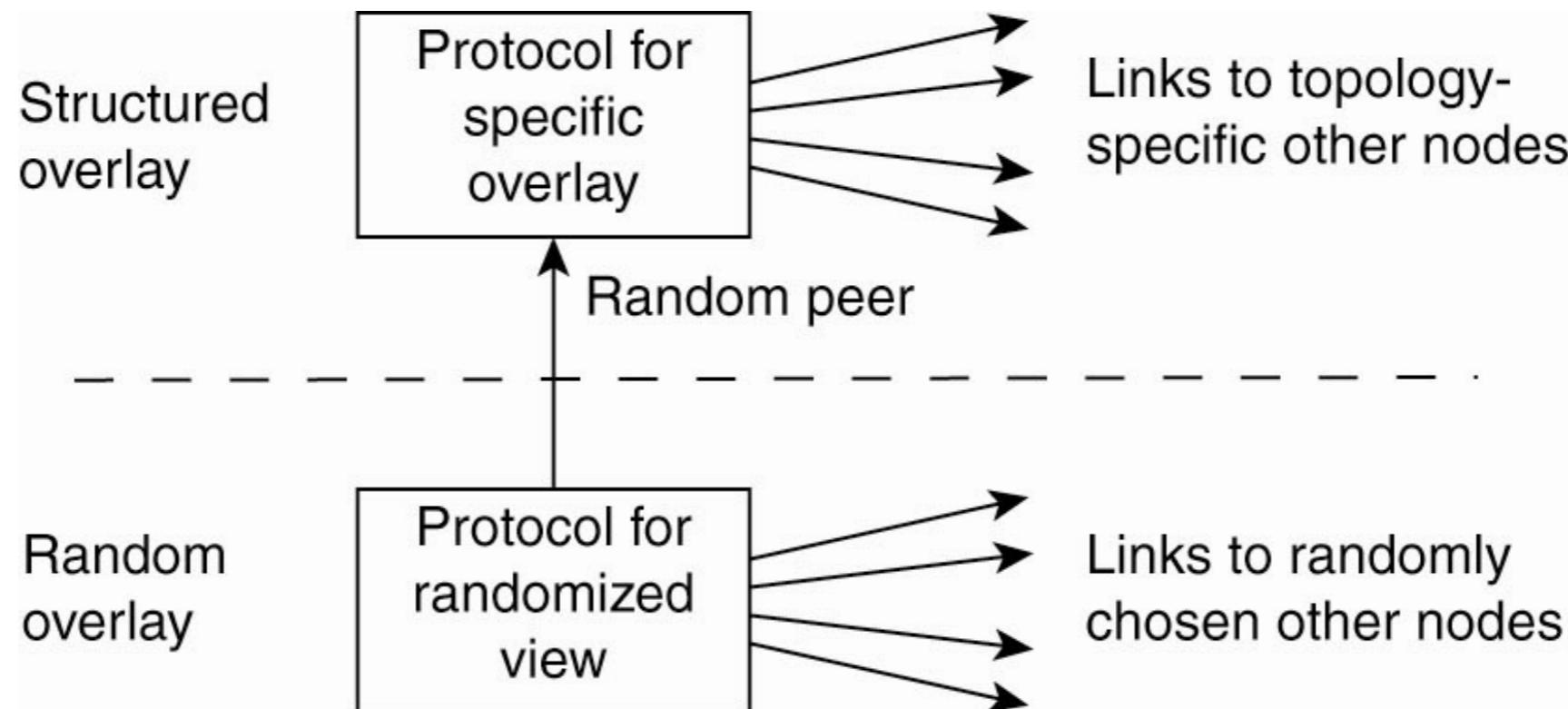
Chord

Can



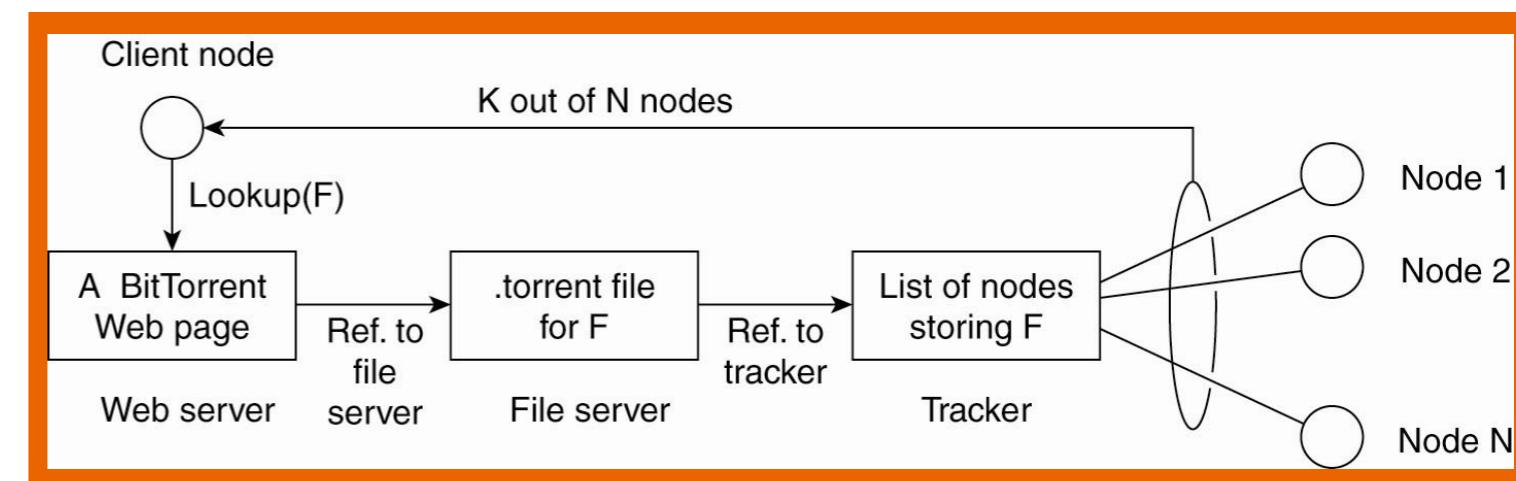
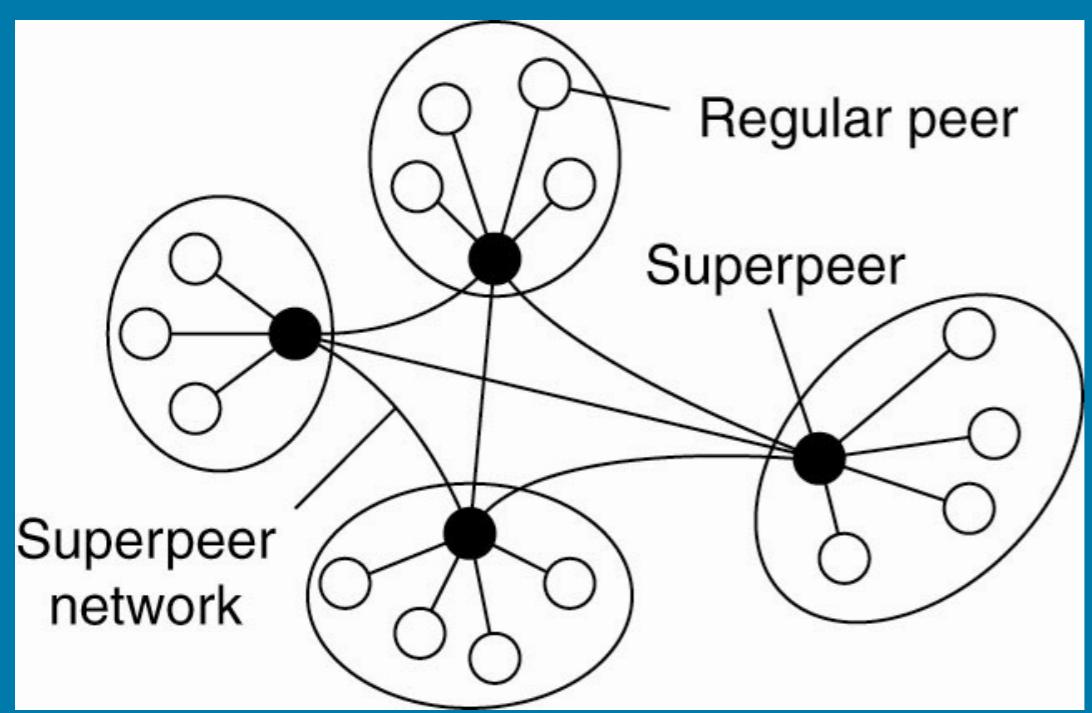
- Alternativamente, podemos ter sistemas P2P não-estruturados
- O princípio básico de funcionamento é o seguinte:
 - Cada nó mantém uma **vista parcial** da rede consistindo num conjunto razoavelmente pequeno de nós
 - Cada nó P periodicamente escolhe um nó Q da sua vista parcial
 - P e Q trocam informação e trocam nós das suas vistas parciais
 - A aleatoriedade com que as vistas parciais são mantidas é crucial para o funcionamento e robustez do sistema

- Na prática é comum as overlay networks serem criadas e mantidas em duas camadas:



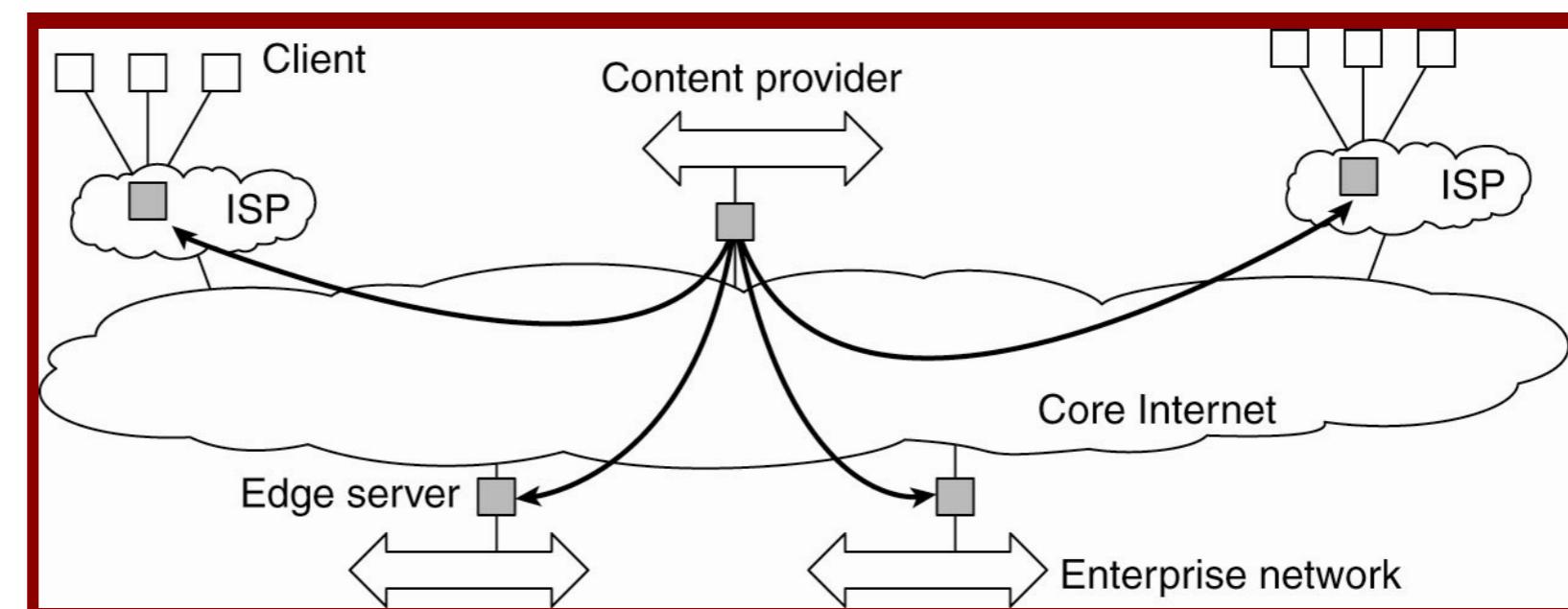
- A camada de cima seleciona os nós fornecidos pela de baixo conforme a estrutura pretendida

• Vários sistemas distribuídos combinam arquiteturas centralizadas e descentralizadas



BitTorrent

Superpeers

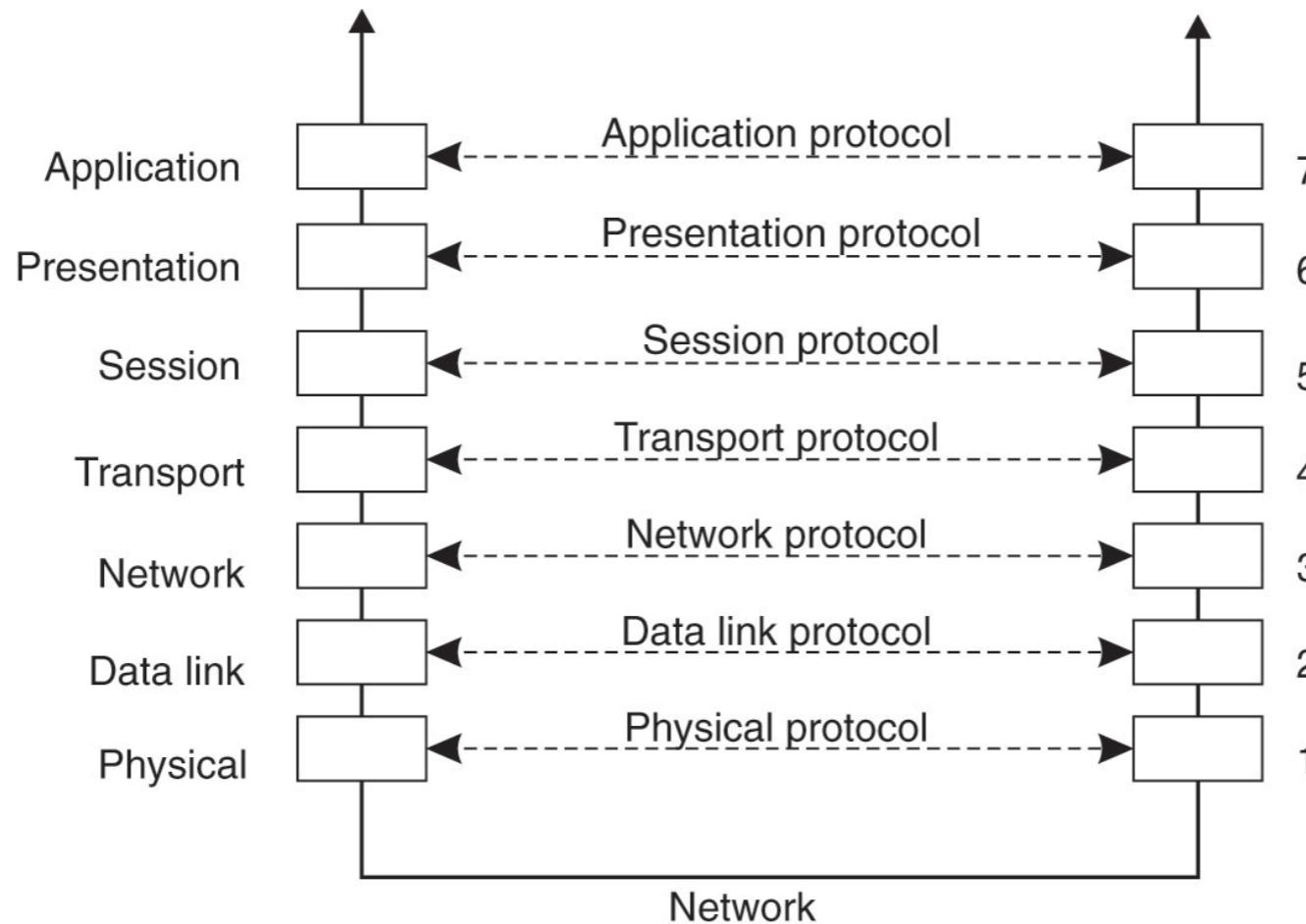


Edge-Servers

Sistemas distribuídos

-  Introdução
-  Arquiteturas e Modelos
-  Comunicação
-  Sistemas de objetos distribuídos
-  Sistemas de ficheiros distribuídos
-  Sistemas Web

O modelo básico de rede - pilha OSI



Tem algumas desvantagens:

- É orientado à passagem de mensagens apenas
- Viola a transparência de acesso

Níveis de baixo nível da pilha OSI

- Nível físico: contém a especificação e implementação dos bits transmitidos entre emissor e receptor
- Nível do canal de dados: define a transmissão de bits em “frames” que permitem a detecção e correção de erros e o controlo do fluxo
- Nível da rede: descreve o roteamento de pacotes numa rede de computadores
- Em sistemas distribuídos o nível de rede é normalmente o mais baixo que consideramos

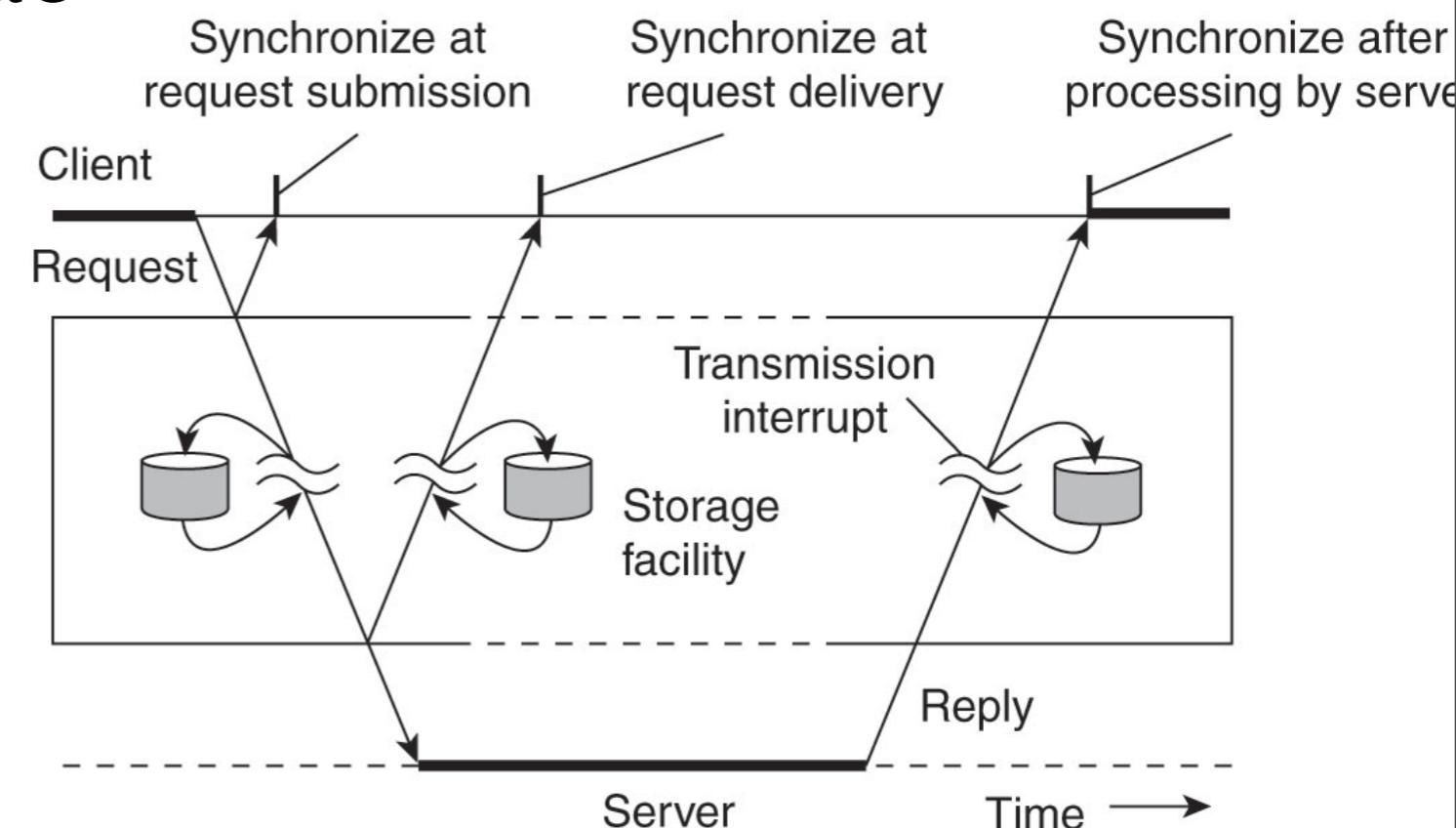
Nível de transporte:

- É o nível que de facto disponibiliza as primitivas de comunicação à generalidade dos sistemas distribuídos
- UDP: User Datagram Protocol – comunicação por datagramas sem garantias de entrega (não fiável)
- TCP: Transmission Control Protocol – comunicação no contexto de uma sessão e com garantias de entrega (melhor esforço)

Nível de Middleware

- Pretende oferecer um conjunto comum de serviços e protocolos para uso de muitas aplicações diferentes
- Protocolos de comunicação com várias características e várias garantias de entrega no que respeita à fiabilidade e à ordem
- “Marshaling/Unmarshaling” de dados
- Resolução de nomes (fundamental para a transparência de acesso)
- Mecanismos de segurança
- Caching e replicação (importantes para a escalabilidade)

Tipos de comunicação



Transiente vs. Persistente

- As mensagens são descartadas se não puderem ser entregues ou guardadas até ser possível a entrega

Síncrona vs. Assíncrona

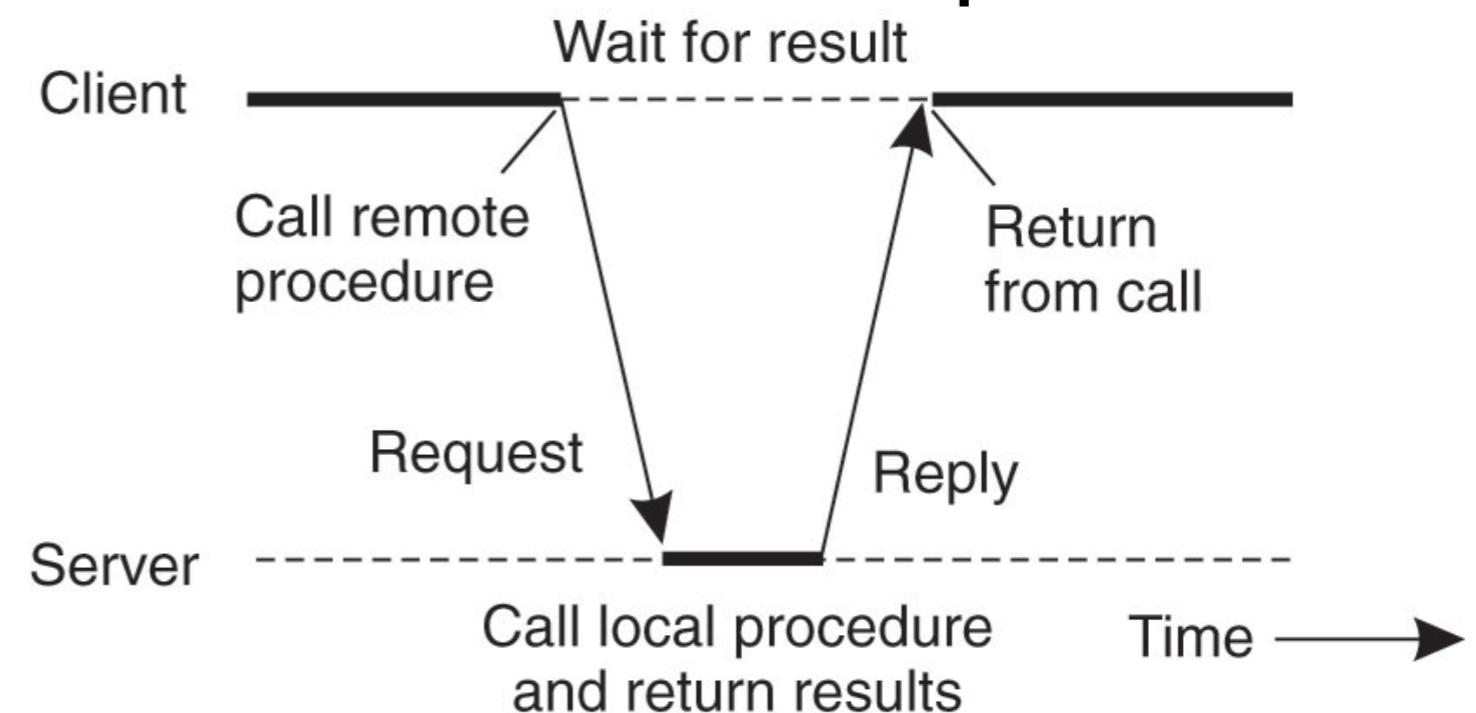
- O emissor poderá sincronizar ou não com o receptor

Tipos de comunicação

Exemplos:

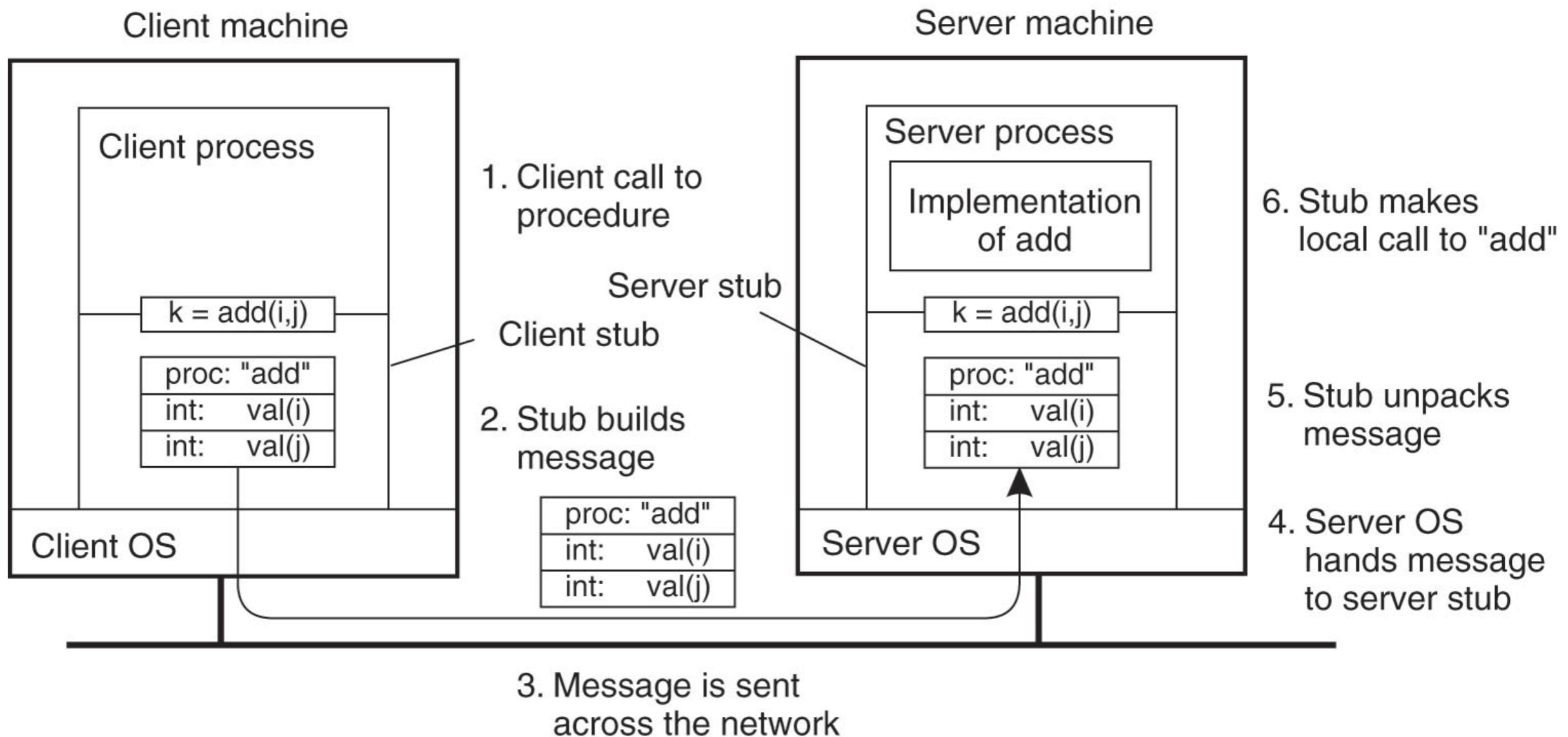
- Cliente/servidor: baseado normalmente num modelo de comunicação transiente e síncrono
 - O cliente e o servidor têm que estar simultaneamente activos
 - O cliente envia o pedido e aguarda/bloqueia até receber a resposta
 - O servidor aguarda por pedidos e processa-os
- “Messaging”: modelo normalmente associado a uma comunicação persistente e assíncrona
 - As mensagens são armazenadas numa fila
 - O emissor não aguarda pela resposta, mas prossegue processamento

- O modelo natural de programação utiliza a invocação de procedimentos (ou funções)
- Os procedimentos podem ser concebidos de forma a executarem isolados, sem (demasiada) dependência de outros procedimentos.
- A invocação de procedimentos remotos possibilita a execução de procedimentos em computadores distintos.

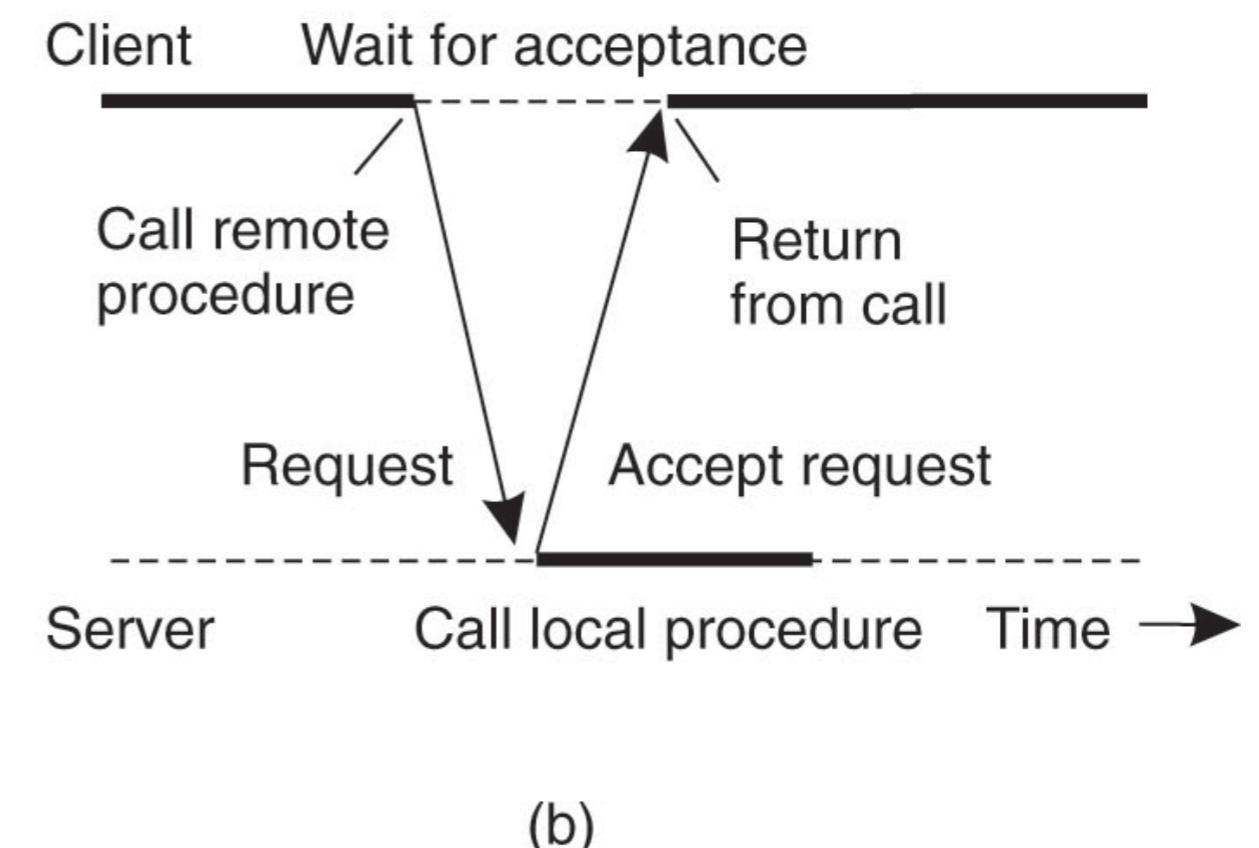
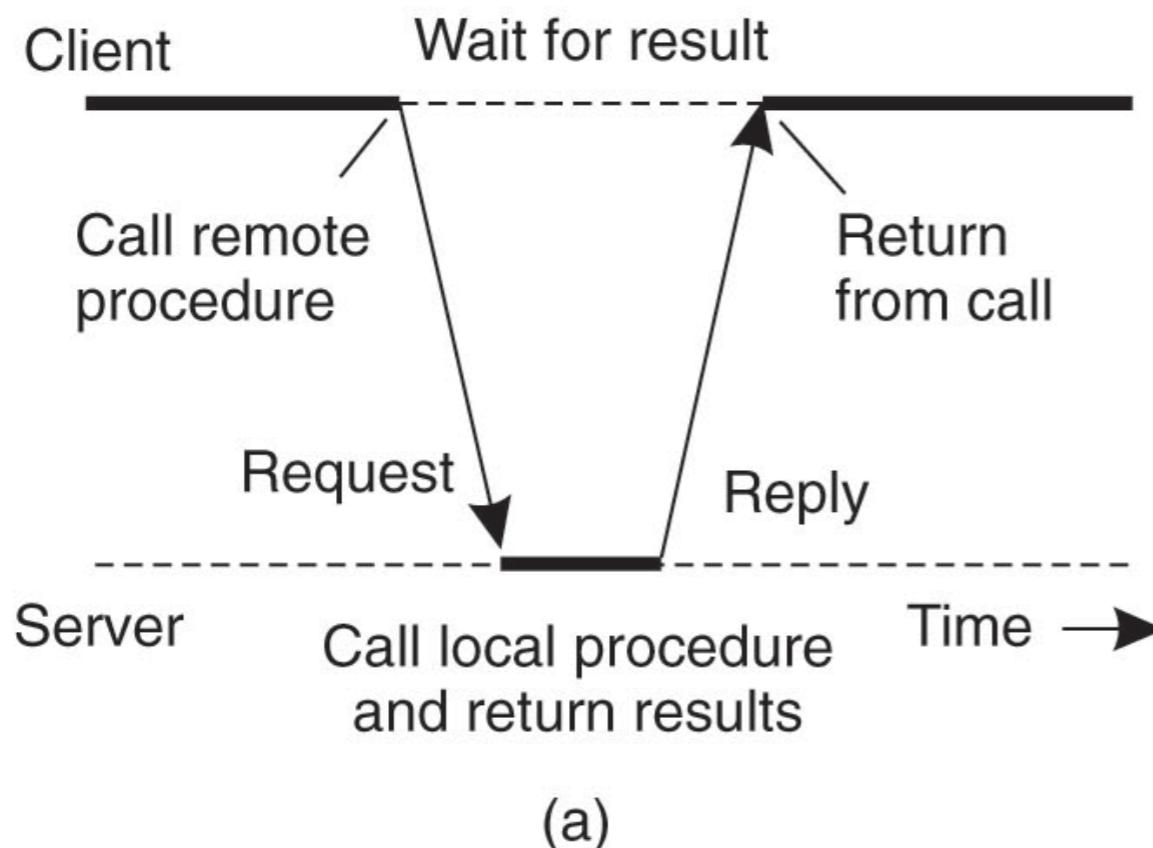




Funcionamento de um Remote Procedure Call

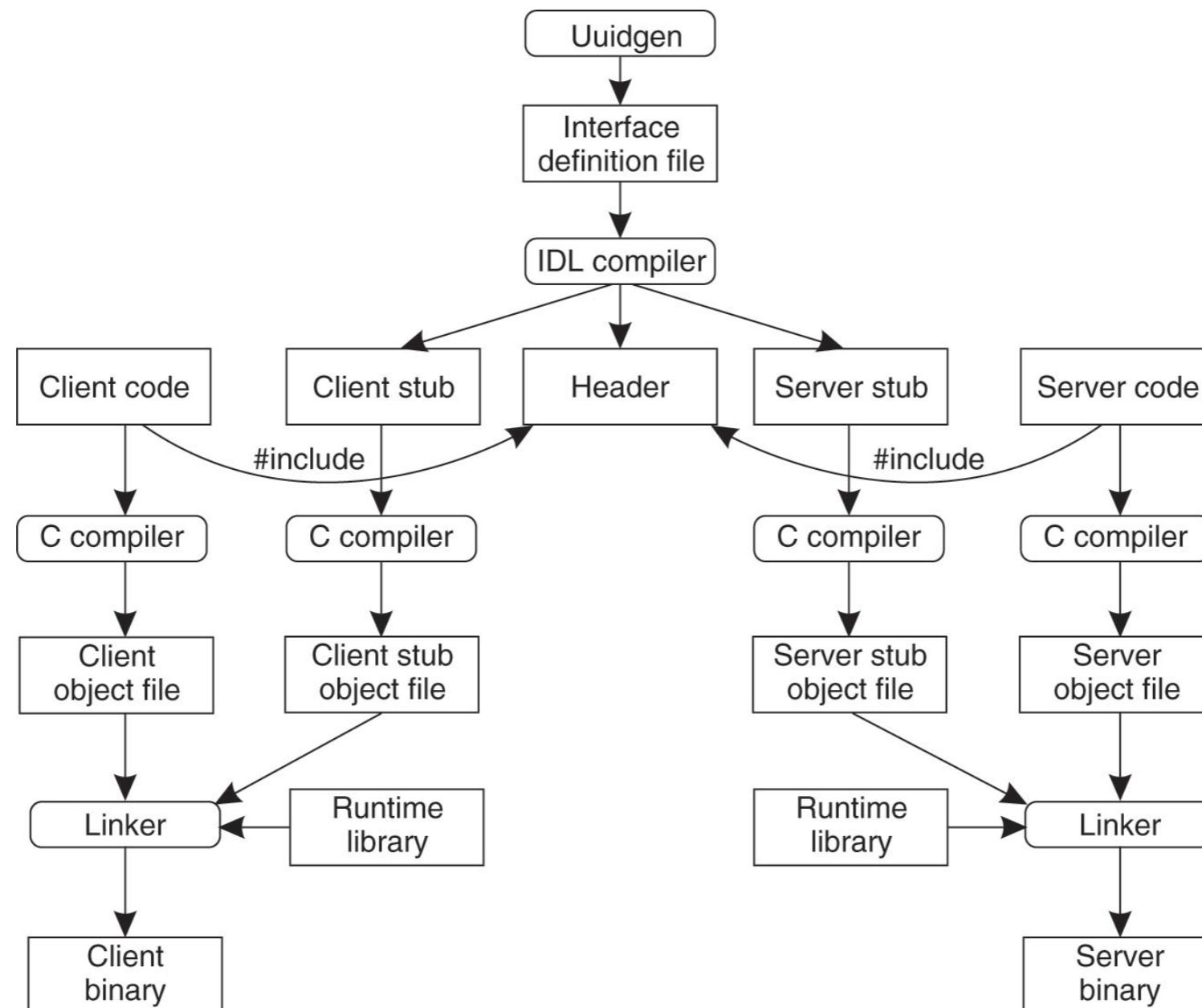


RPC Assíncrono



Alternativamente pode ser usado RPC síncrono deferido

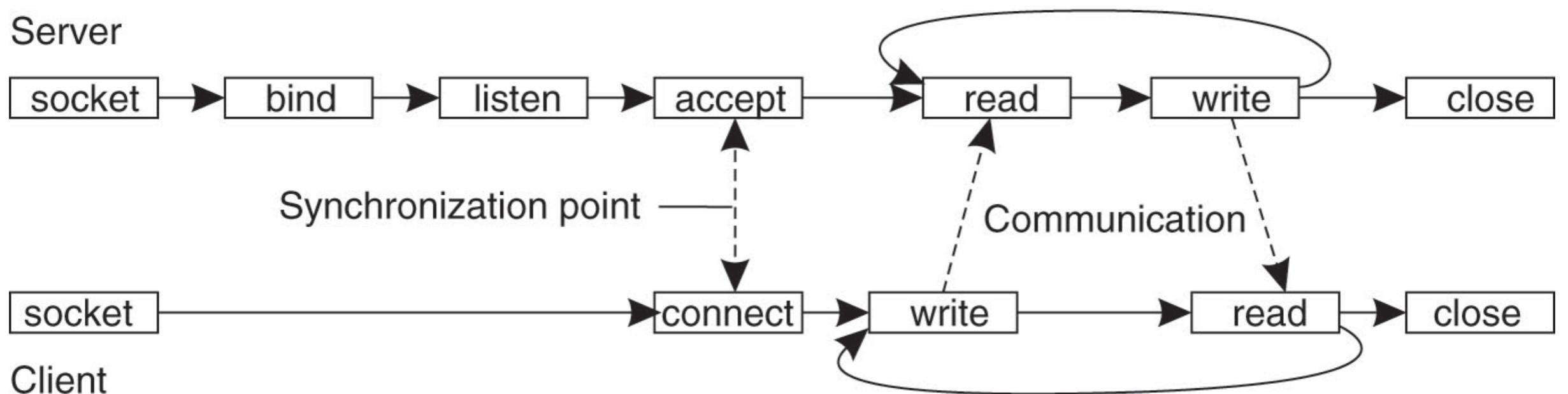
Criação de um RPC



Mensagens transientes: sockets

Primitive	Meaning
Socket	Create a new communication end point
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Mensagens transientes: sockets

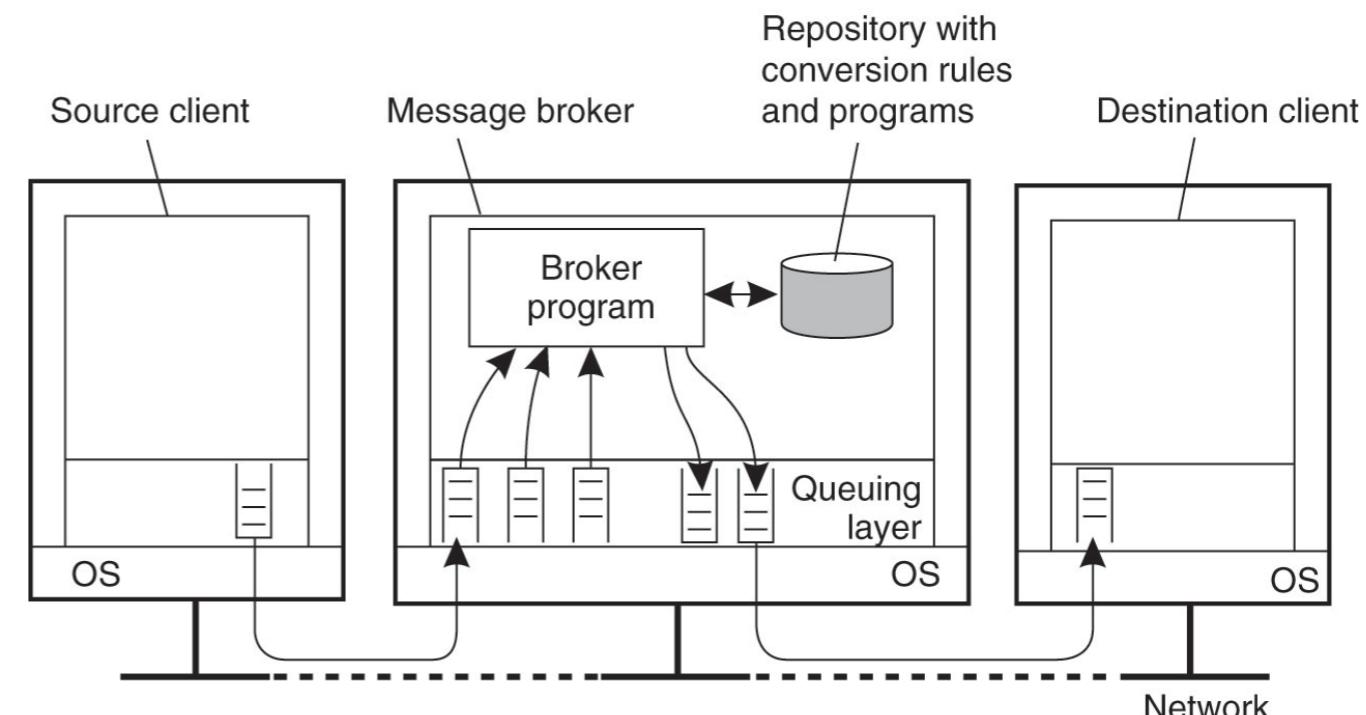


- Middleware orientado a mensagens
- Comunicação assíncrona e persistente através de filas de mensagens
- Interface básico:

Primitive	Meaning
Put	Append a message to a specified queue
Get	Block until the specified queue is nonempty, and remove the first message
Poll	Check a specified queue for messages, and remove the first. Never block
Notify	Install a handler to be called when a message is put into the specified queue

Middleware orientado a mensagens

- Os sistemas de filas de mensagens assumem um protocolo comum usado por todas as aplicações – i.e. a representação dos dados e da sua estrutura
- No entanto, uma das principais aplicações dos sistemas de filas de mensagens é integrar diferentes aplicações, desenvolvidas independentemente, num sistema de informação distribuído coerente. Para o efeito usa-se um Message Broker.

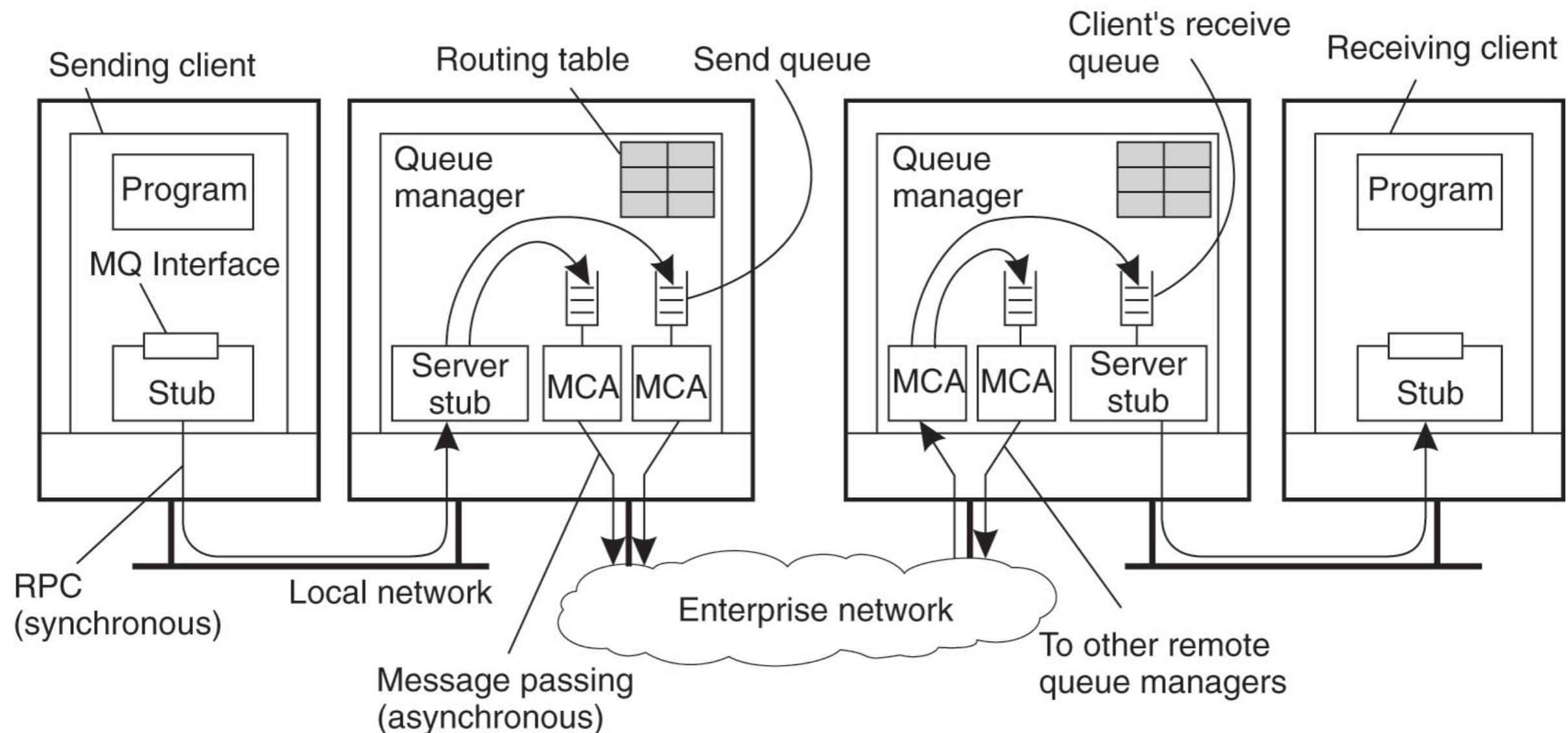


Middleware orientado a mensagens

- Exemplo: IBM WebSphere MQ
- As mensagens são colocadas e retiradas de filas de mensagens
- A filas de mensagens são controladas por um gestor de filas
- As mensagens pode ser colocadas em filas locais ou no gestor através de RPC
- As mensagens são transferidas entre filas através de canais
- Os canais são estabelecidos por dois Message Channel Agents
- Um MCA trata de estabelecer o canal de rede (eg. TCP), empacotar/desempacotar as mensagens e enviar/receber através do canal de rede

Middleware orientado a mensagens

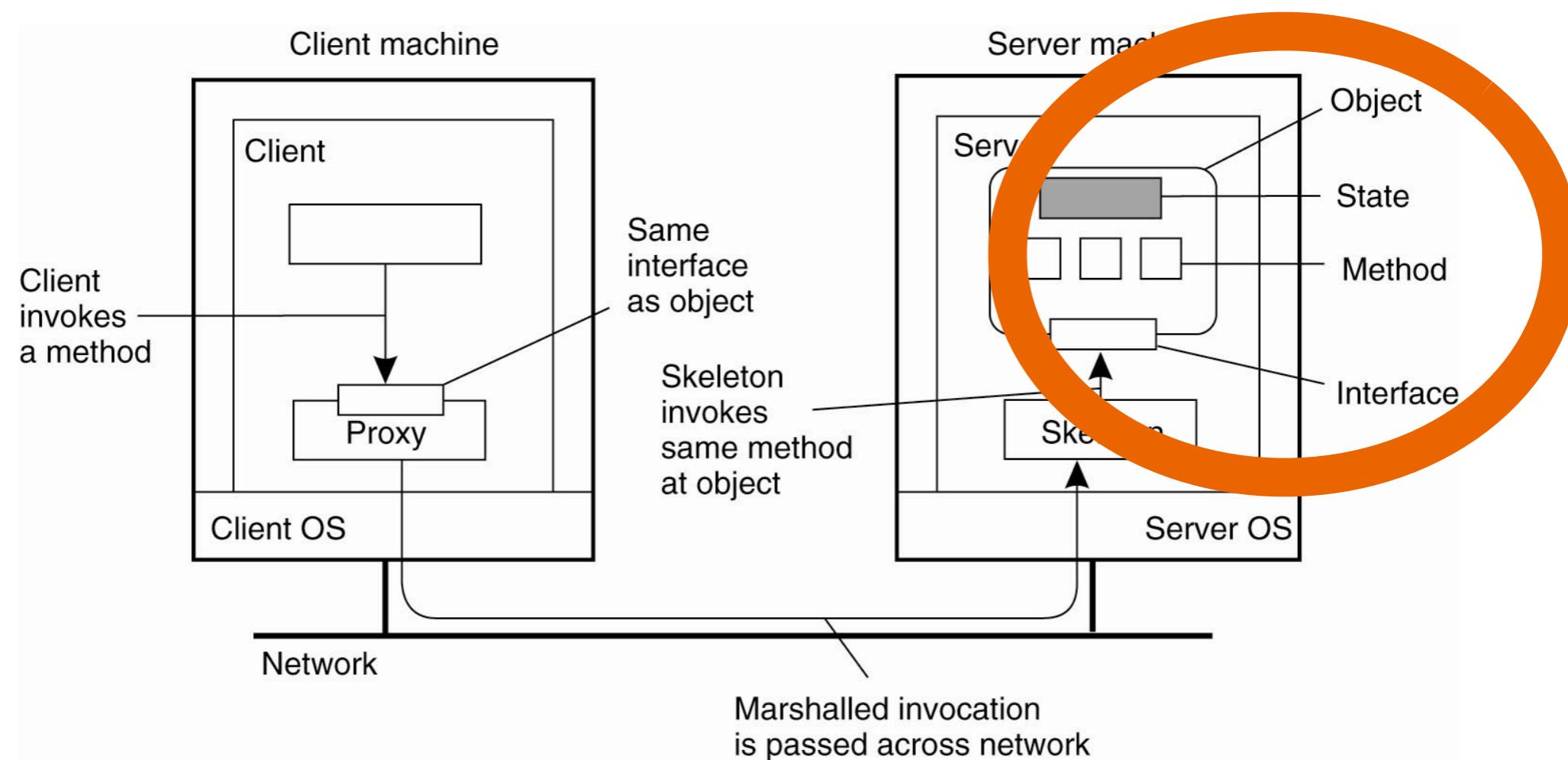
Exemplo: IBM WebSphere MQ



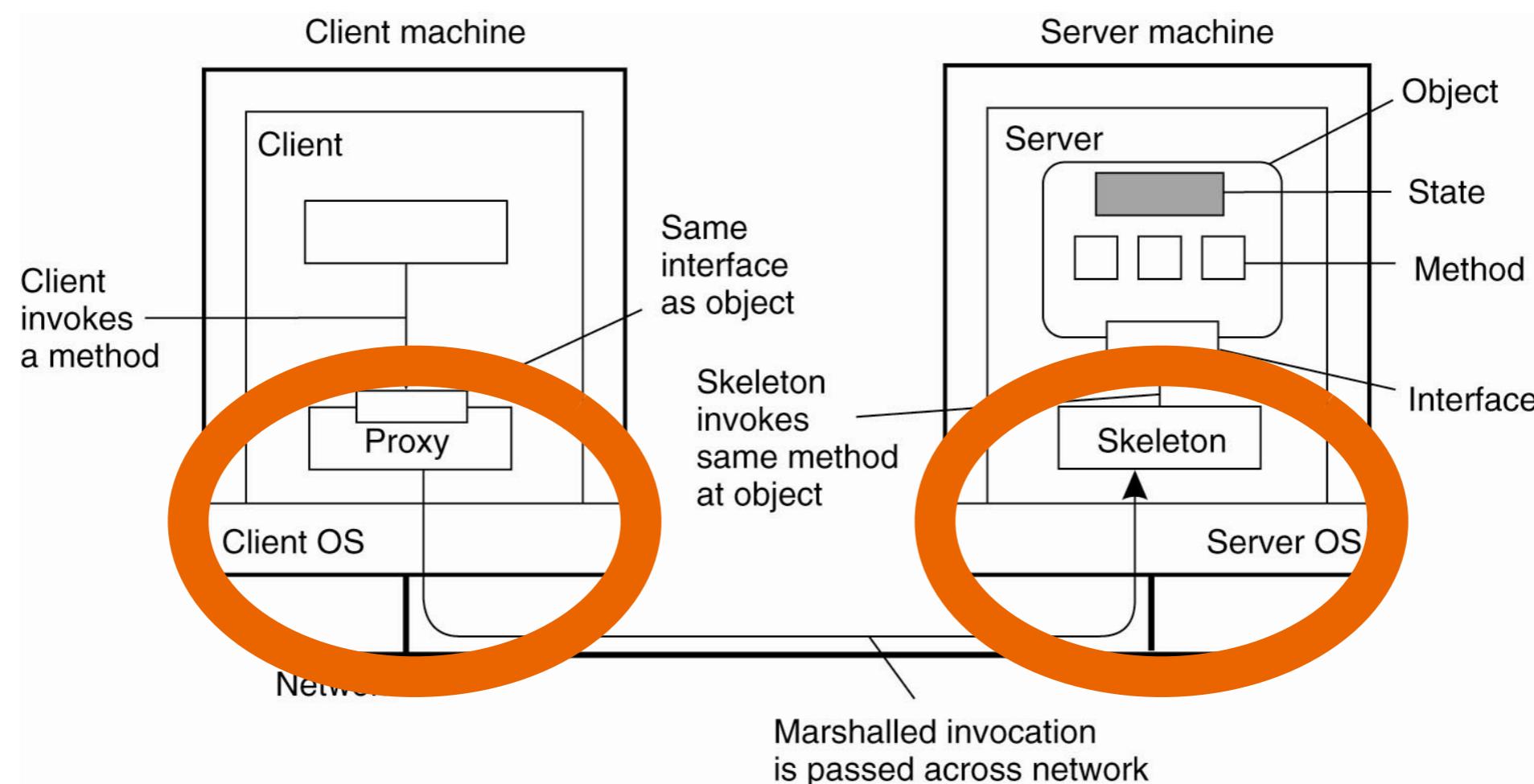
Sistemas distribuídos

-  Introdução
-  Arquiteturas e Modelos
-  Comunicação
-  **Sistemas de objetos distribuídos**
-  Sistemas de ficheiros distribuídos
-  Sistemas Web

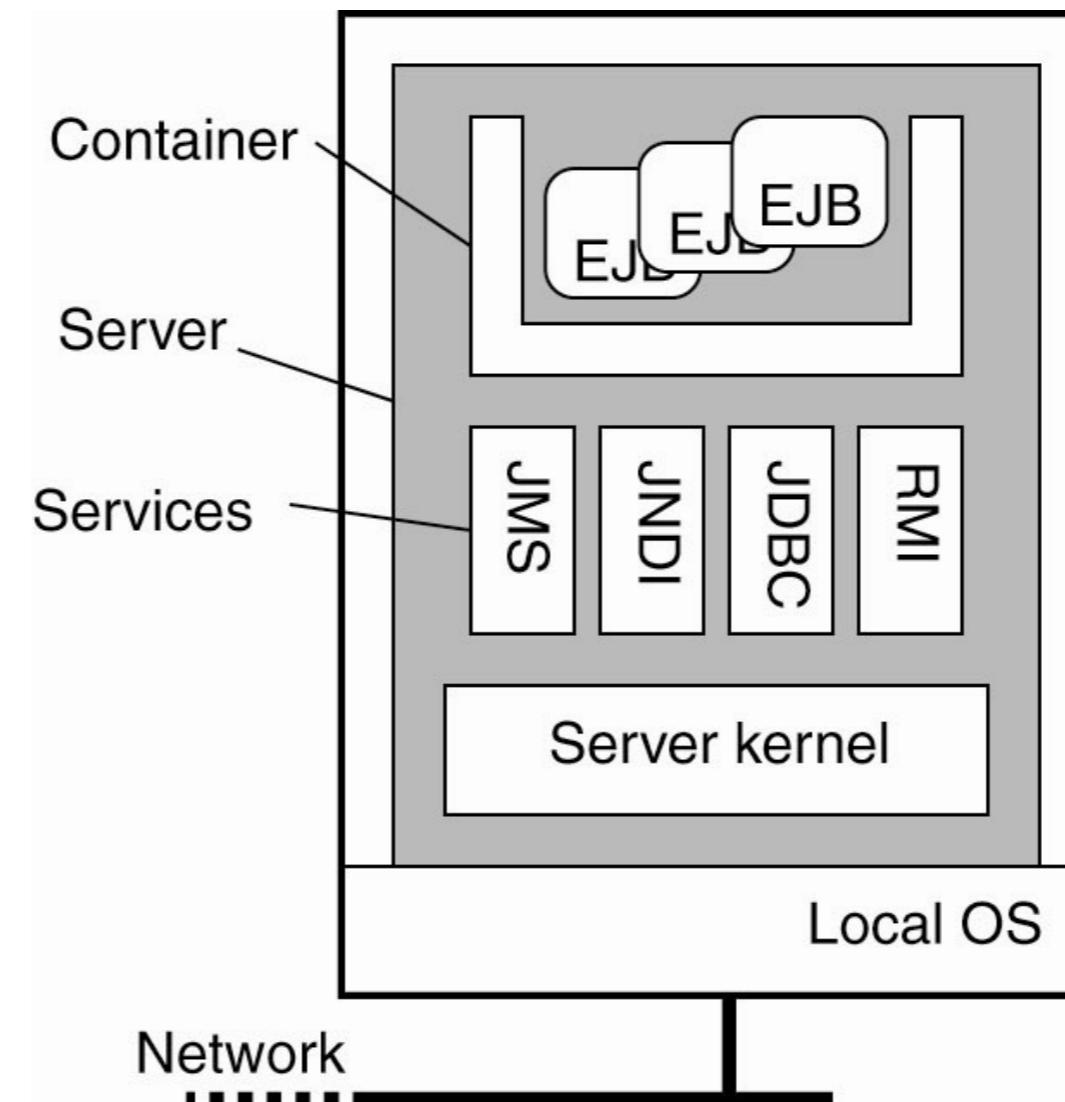
- Os objectos encapsulam **estado** e **métodos**
- Os métodos são agrupados em **interfaces**
- As interfaces são a única parte visível aos clientes



- O **stub** ou **proxy** é uma implementação da interface que representa localmente o objecto remoto
- O **skeleton** atende todas as invocações remotas e procede às respectivas invocações locais



- EJBs são objetos Java oferecidos por um servidor especial que permite diferentes formas de invocação por parte dos objetos clientes remotos



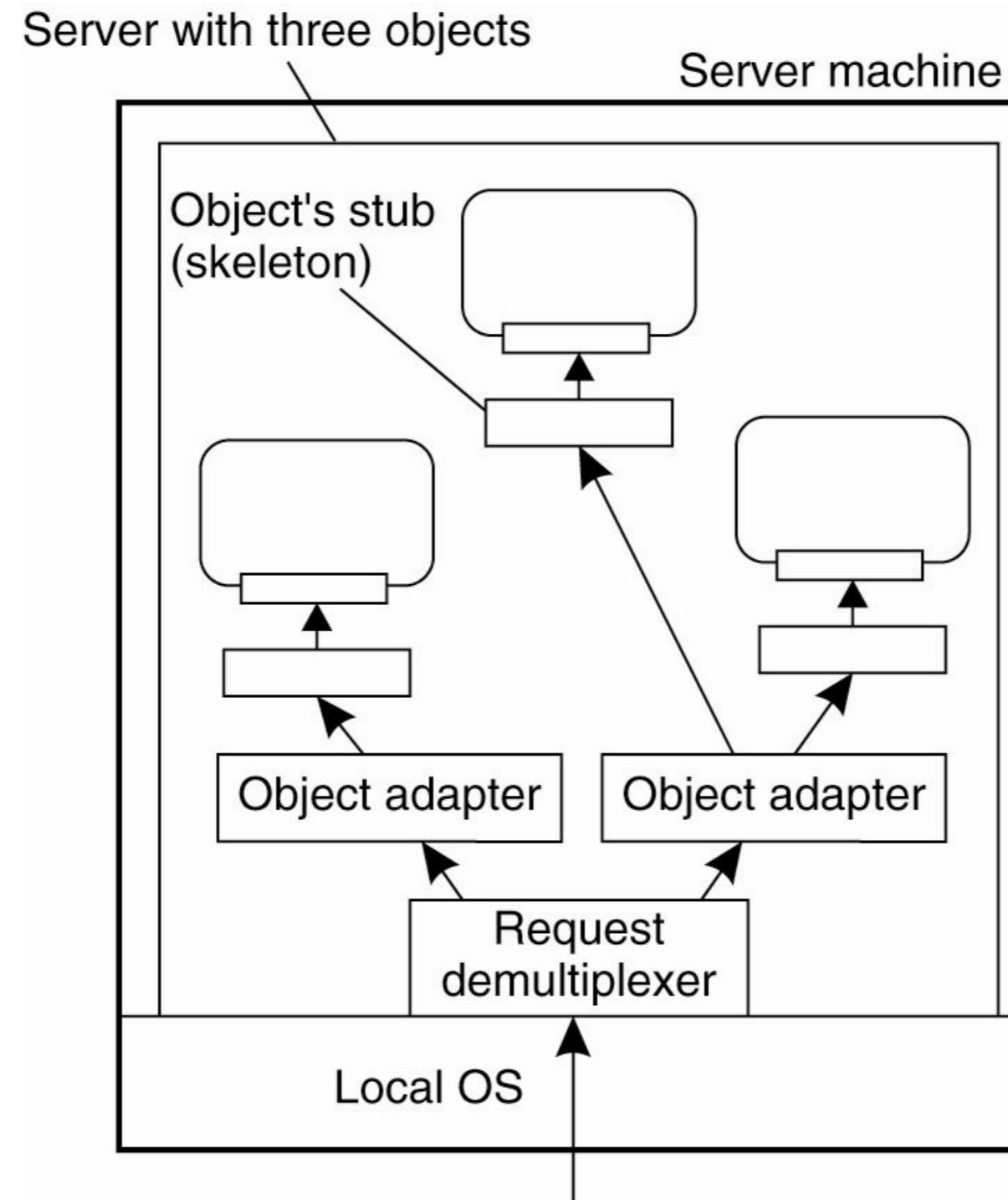
4 tipos de EJBs:

- **Stateless session bean:** objecto de existência transiente, são criados, executam a sua função, retornam o resultado e desaparecem. Duram apenas durante a invocação e não têm estado persistente. Exemplo: executar uma interrogação SQL e retornar o resultado ao cliente.
- **Stateful session bean:** objecto transiente mas que mantem estado sobre o cliente durante a sessão. Exemplo: carrinho de compras numa aplicação de vendas online.
- **Entity bean:** objeto que modela entidades da aplicação, com estado persistente e disponível durante quaisquer sessões. Exemplo: objeto com o perfil do cliente.
- **Message-driven bean:** objetos que reajem a certos tipos de mensagens. Utilizados na implementação de sistemas editor/subscritor.

- Um servidor de objectos, ao contrário de um servidor tradicional, não oferece um serviço específico mas mantém um conjunto de objetos que, esses sim, disponibilizam variados serviços.
- Um servidor de objetos tem como papel principal num sistema de objectos distribuídos, gerir um conjunto de objetos e intermediar os pedidos que são realizados aos objetos.
- Um servidor de objetos é multi-threaded podendo atribuir uma thread a cada objeto ou uma thread a cada invocação.

- **Servants:** as implementações propriamente ditas dos objetos, a parte funcional.
- **Skeletons:** stub do lado servidor, responsável por reestruturar (unmarshall) as invocações e linearizar (marshall) e enviar os resultados.
- **Object Adapter:** gestor do objetos oferecidos pelo servidor de objetos
 - Inspecciona as invocações que chegam
 - Assegura que o objeto desejado se encontra ativado (servant)
 - Encaminha a invocação ao skeleton apropriado
 - É responsável por gerar as referências para os objetos que mantém

Servidores de Objetos

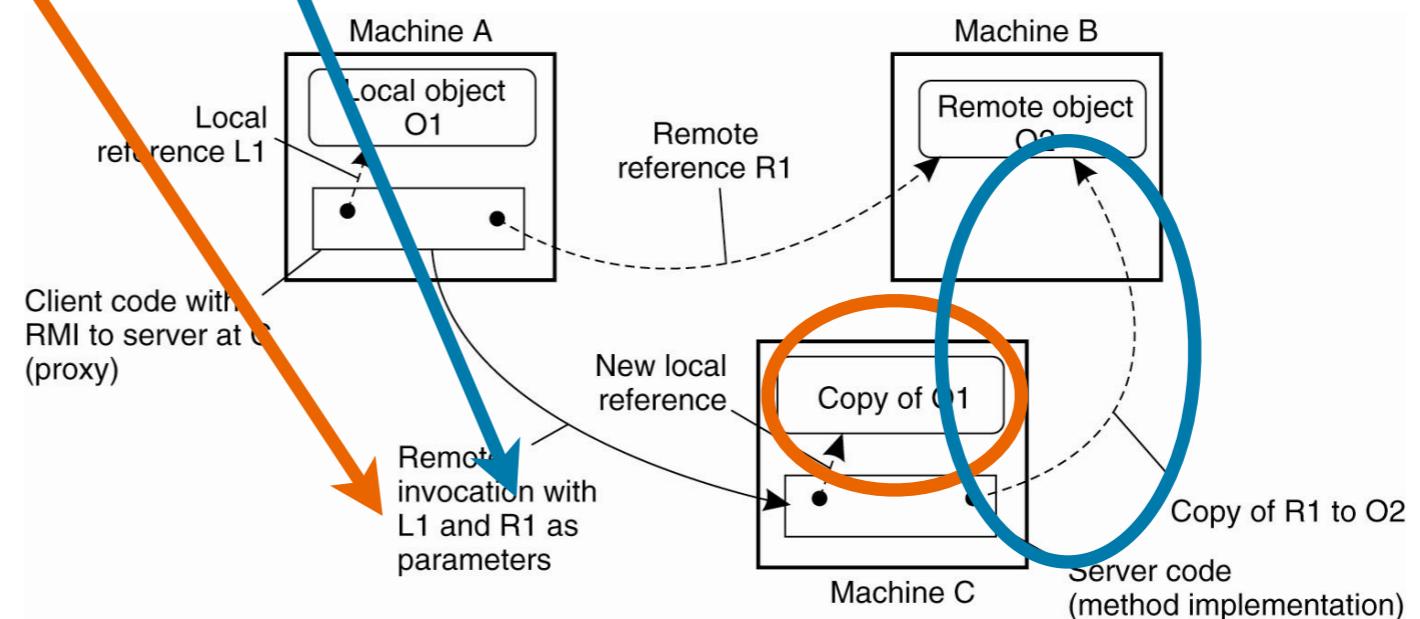


- Os sistemas de objetos distribuídos disponibilizam meios para que um cliente aceda e invoque métodos de um objeto remotamente.
- Os mecanismos baseiam-se essencialmente nos conceitos de RPC.
- Uma diferença importante porém é que em sistemas de objetos distribuídos, a referência a um objeto
 - é válida em todo o sistema,
 - é um tipo de dados de **primeira classe**, i.e., pode ser passada como argumento e resultado de invocações.
- Pelo facto da implementação de um objecto não ser exposta pela sua referência a transparência na distribuição é maior.

- Na posse de uma referência para um objeto o cliente tem que se associar – **bind** – ao objecto antes de poder proceder a qualquer invocação.
- Uma referência identifica o servidor, o objeto e o protocolo de comunicação
- O cliente carrega o código stub
- O stub é instanciado e inicializado para o objeto específico
- A associação pode ser feita
 - implicitamente: ocorre na primeira invocação do cliente sobre a referência do objeto
 - explicitamente: o cliente associa-se ao objeto antes de efetuar qualquer invocação

- Funcionamento de uma invocação remota de métodos (RMI)
 - Assumamos que tanto o stub como o skeleton estão carregados e inicializados
 - O cliente invoca o método no stub
 - O stub lineariza (marshall) o pedido e envia ao servidor
 - O servidor assegura-se que o objeto está ativo:
 - Cria se necessário um processo para correr o objeto
 - Carrega o objeto nesse processo
 - Invoca a inicialização do objeto, etc...
 - O pedido é reestruturado (unmarshall) pelo skeleton do objeto e o método pretendido é invocado no objeto
 - O resultado da invocação ao objeto é linearizado e enviado de novo ao cliente
 - O stub reestrutura o resultado e passa-o ao cliente.

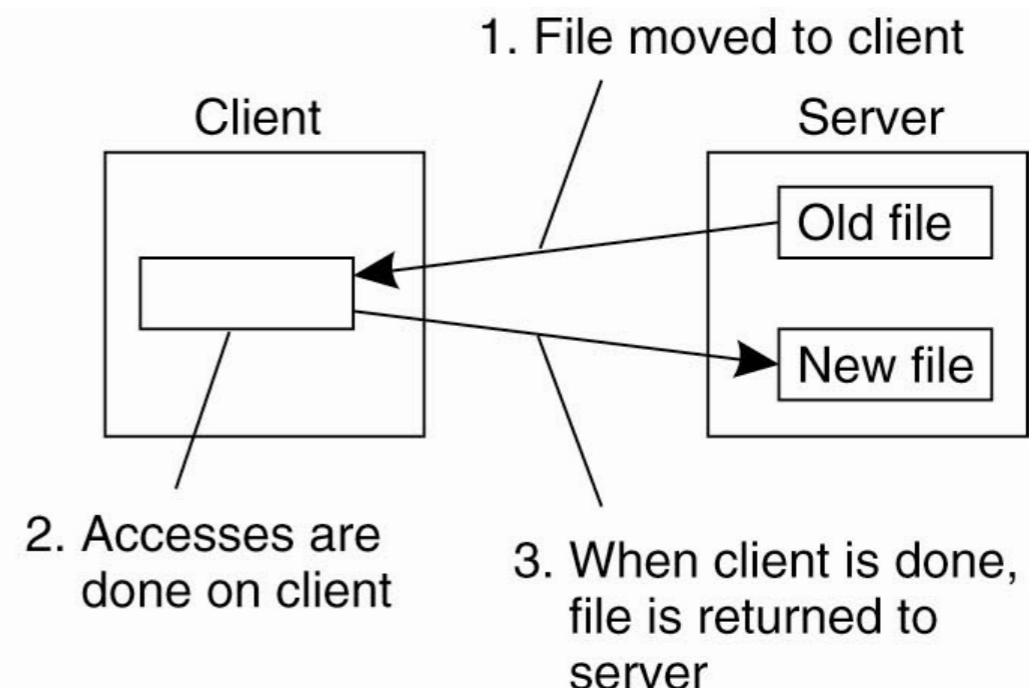
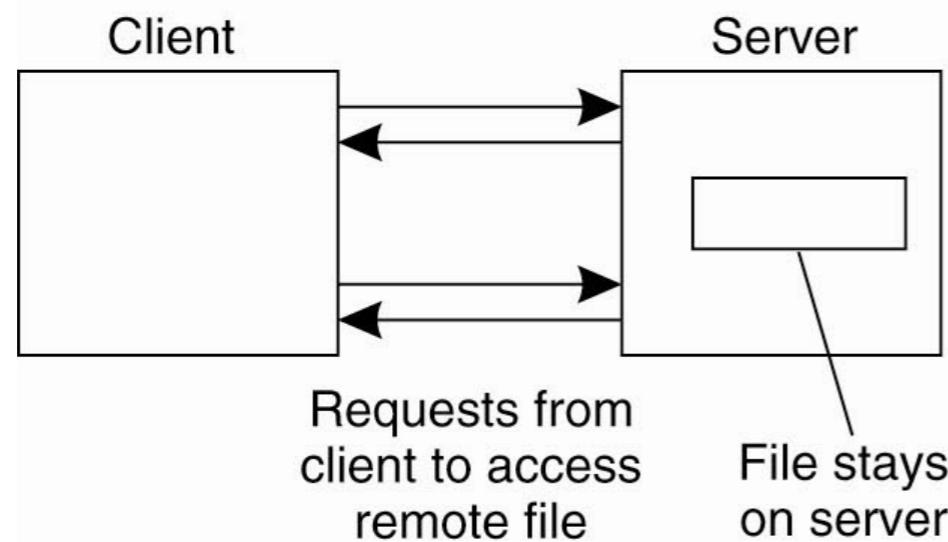
- A passagem de argumentos em RMI pode ser feita de duas formas:
 - Objetos por referência: o objeto invocado pode simplesmente usar a referência, associando-se ao respectivo objeto e invocando por sua vez métodos remotos. No fim, desassocia-se do objeto.
 - Objectos por valor: o cliente pode passar um objecto completo como parâmetro:
 - O objeto tem que ser linearizado (estado e métodos, ou apenas estado se o o servidor invocado conhecer a implementação do objeto).
 - O servidor reestrutura o objeto passando a ter uma cópia do objeto original.



Sistemas distribuídos

-  Introdução
-  Arquiteturas e Modelos
-  Comunicação
-  Sistemas de objetos distribuídos
-  **Sistemas de ficheiros distribuídos**
-  Sistemas Web

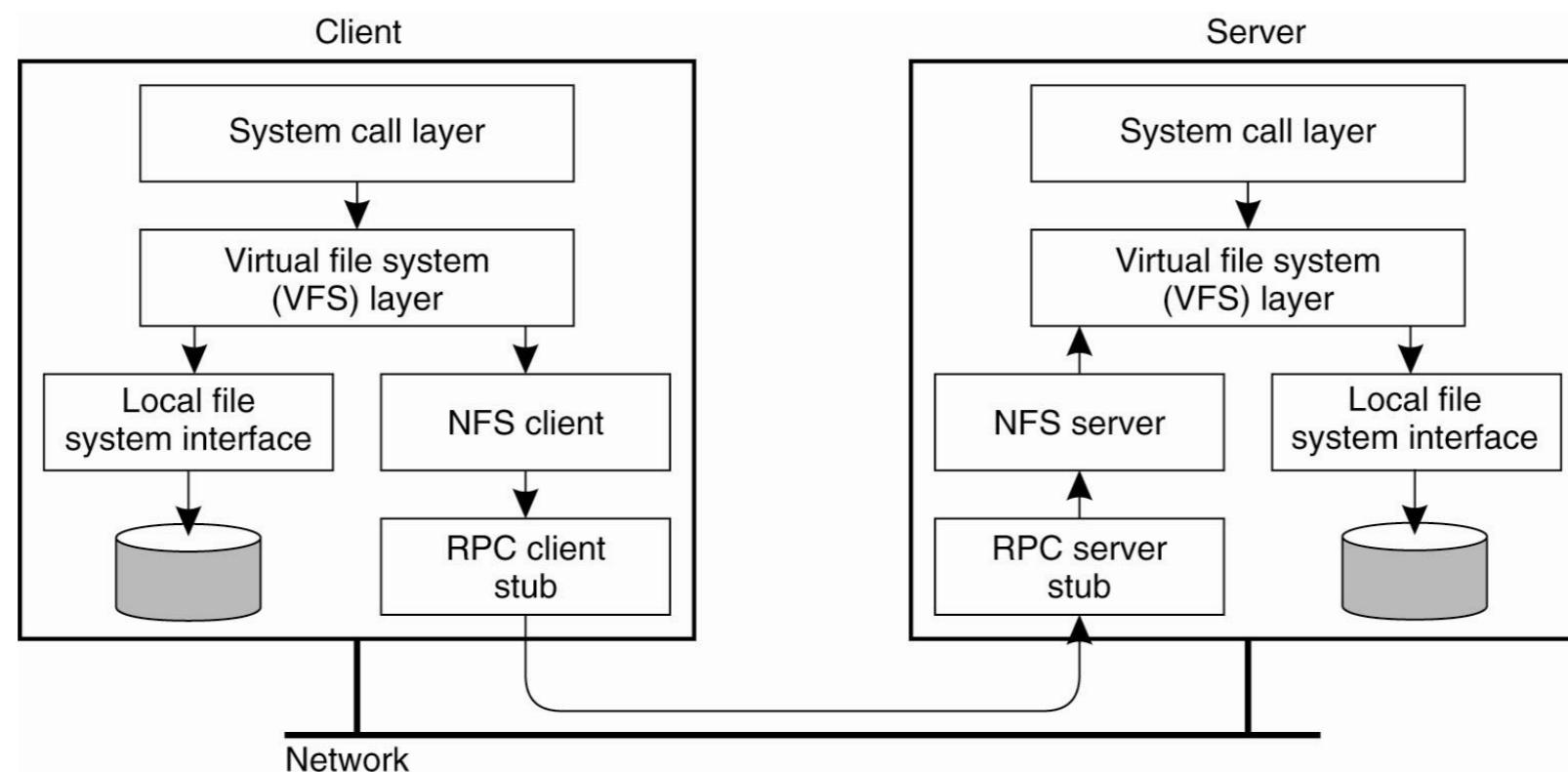
Têm como objetivo principal tornar transparente o acesso a sistemas de ficheiros remotos



Modelo de acesso remoto

Modelo download/upload

O NFS é implementado através da abstração de um Virtual File System, atualmente utilizado em praticamente todos os sistemas operativos de utilização geral

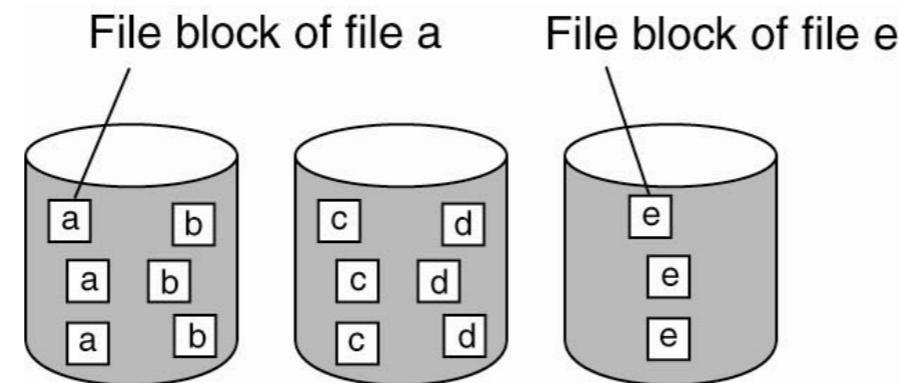


Será o NFS realmente um sistema de ficheiros?

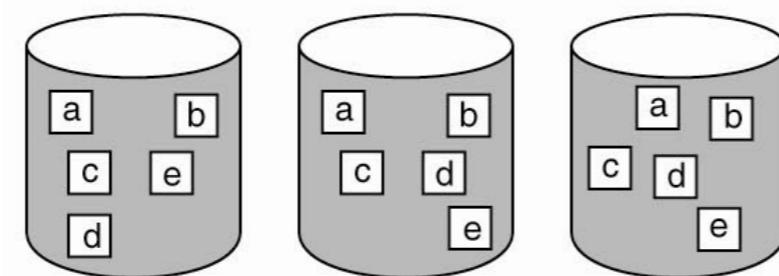
Exemplo: Operações do Network File System

Oper.	v3	v4	Description
Create	Yes	No	Create a regular file
Create	No	Yes	Create a nonregular file
Link	Yes	Yes	Create a hard link to a file
Symlink	Yes	No	Create a symbolic link to a file
Mkdir	Yes	No	Create a subdirectory
Mknod	Yes	No	Create a special file
Rename	Yes	Yes	Change the name of a file
Remove	Yes	Yes	Remove a file from a file system
Rmdir	Yes	No	Remove an empty subdirectory
Open	No	Yes	Open a file
Close	No	Yes	Close a file
Lookup	Yes	Yes	Look up a file by means of a name
Readdir	Yes	Yes	Read the entries in a directory
Readlink	Yes	Yes	Read the path name in a symbolic link
Getattr	Yes	Yes	Get the attribute values for a file
Setattr	Yes	Yes	Set one or more file-attribute values
Read	Yes	Yes	Read the data contained in a file
Write	Yes	Yes	Write data to a file

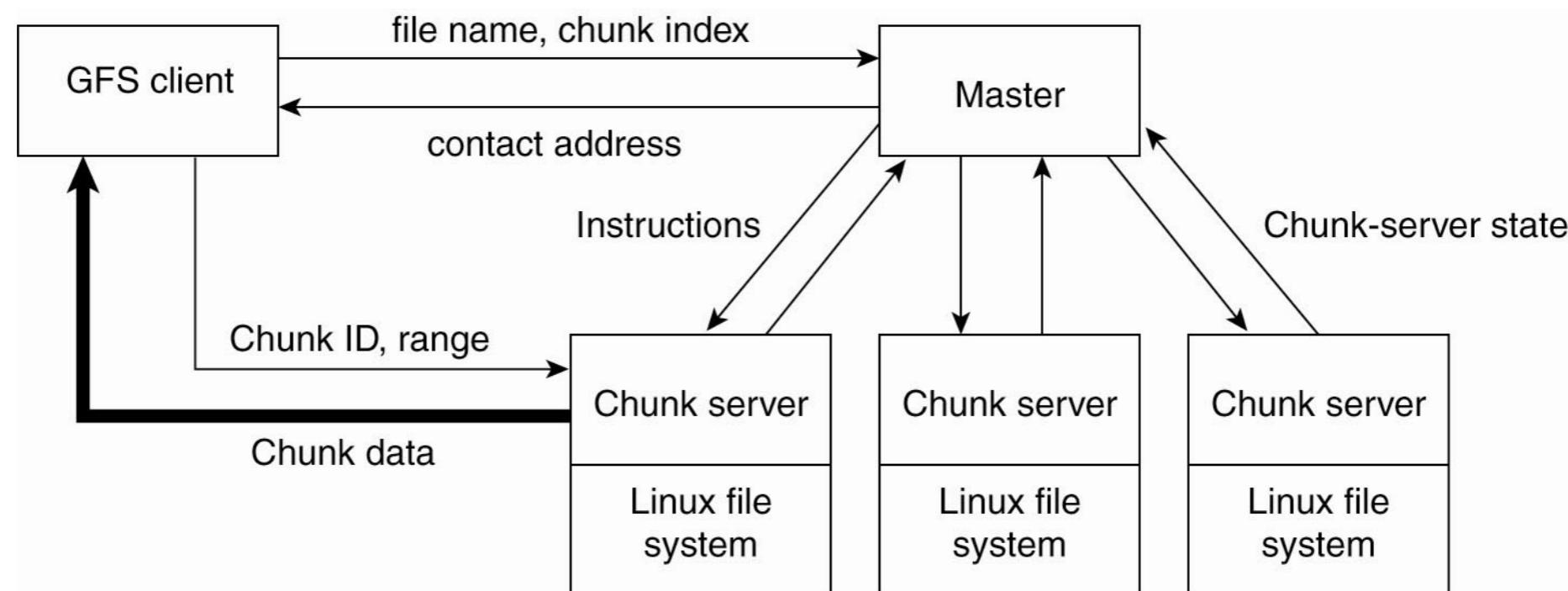
- Com grandes volumes de dados, o modelo centralizado de cliente/servidor é limitado.
- Para aumentar a velocidade de acesso paralelizam-se os sistemas de ficheiros



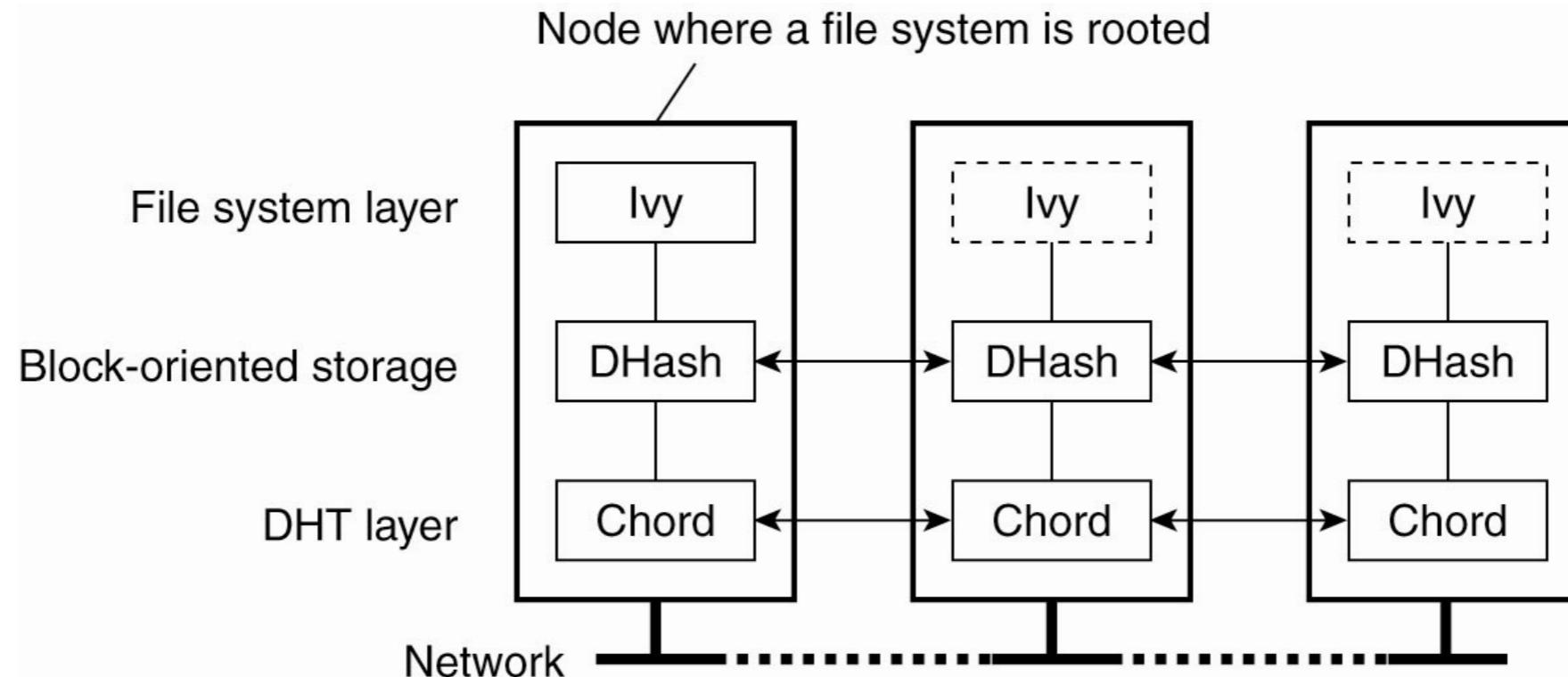
- Podendo-se além disso partir, “fatiar”, os ficheiros



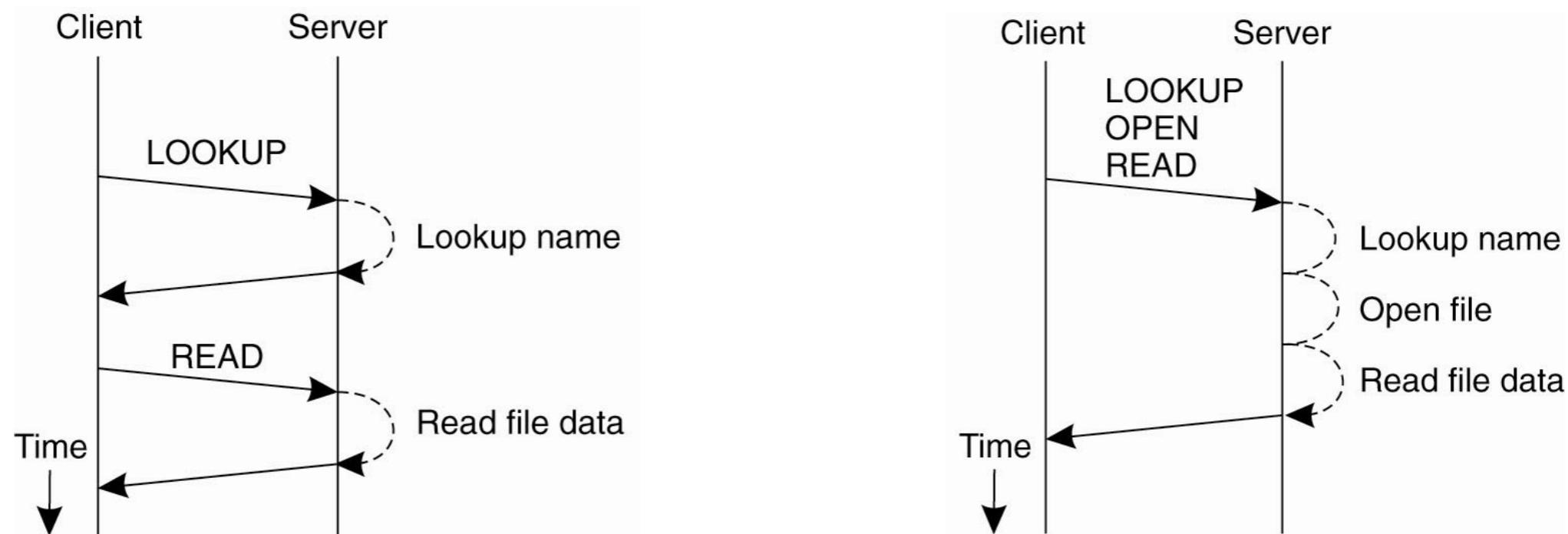
- O GFS divide os ficheiros em partes grandes de 64MB e distribui-as por vários servidores
- O servidor master mantém apenas uma tabela em memória com meta-informação (ficheiro+parte, servidor da parte)



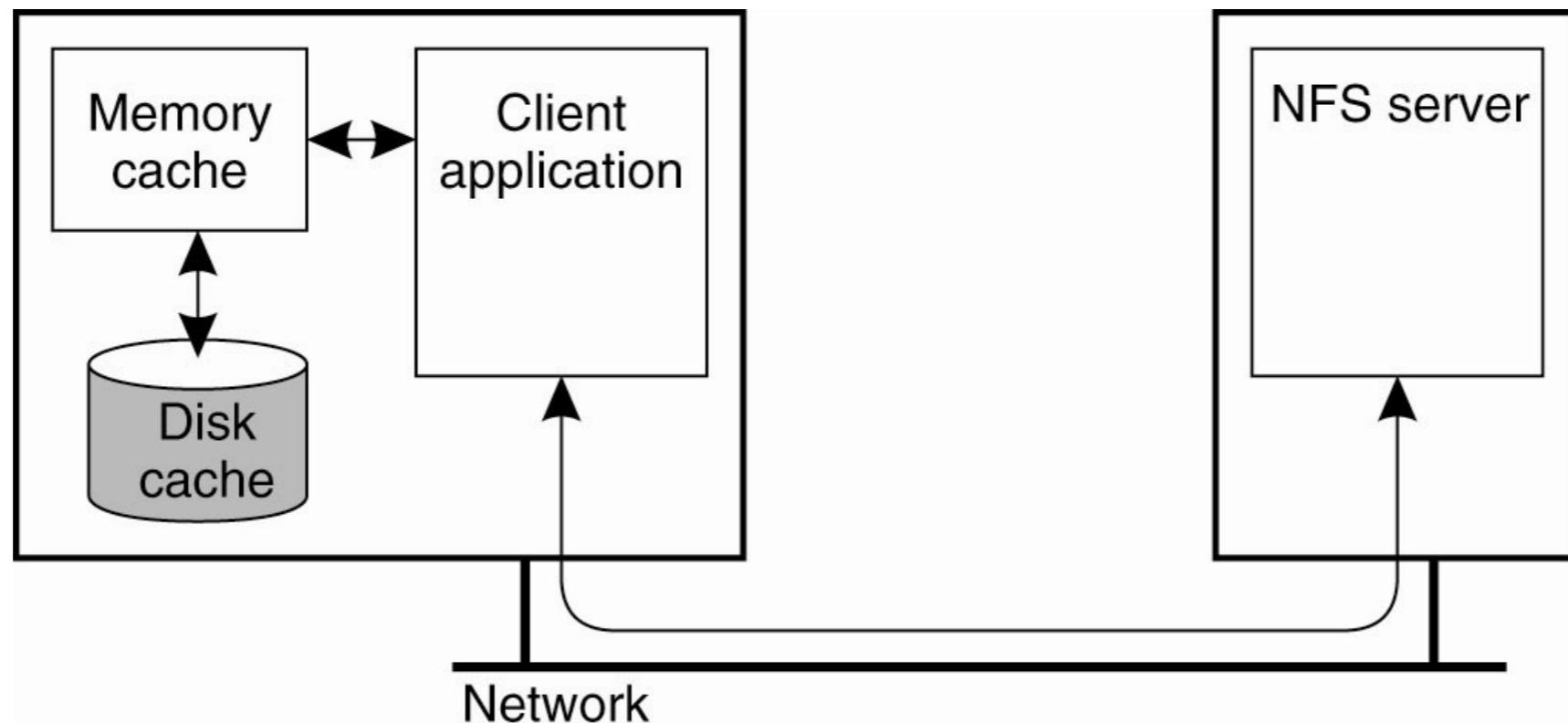
- A ideia é distribuir partes/blocos de ficheiros pelos vários nós de um sistema P2P:
- Um block com conteúdo D é guardado pelo nó que corresponde ao valor de hash $h(D)$
- Um ficheiro corresponde a um diário de operações sobre blocos, uma sequência de pares (bloco, $h(D)$)



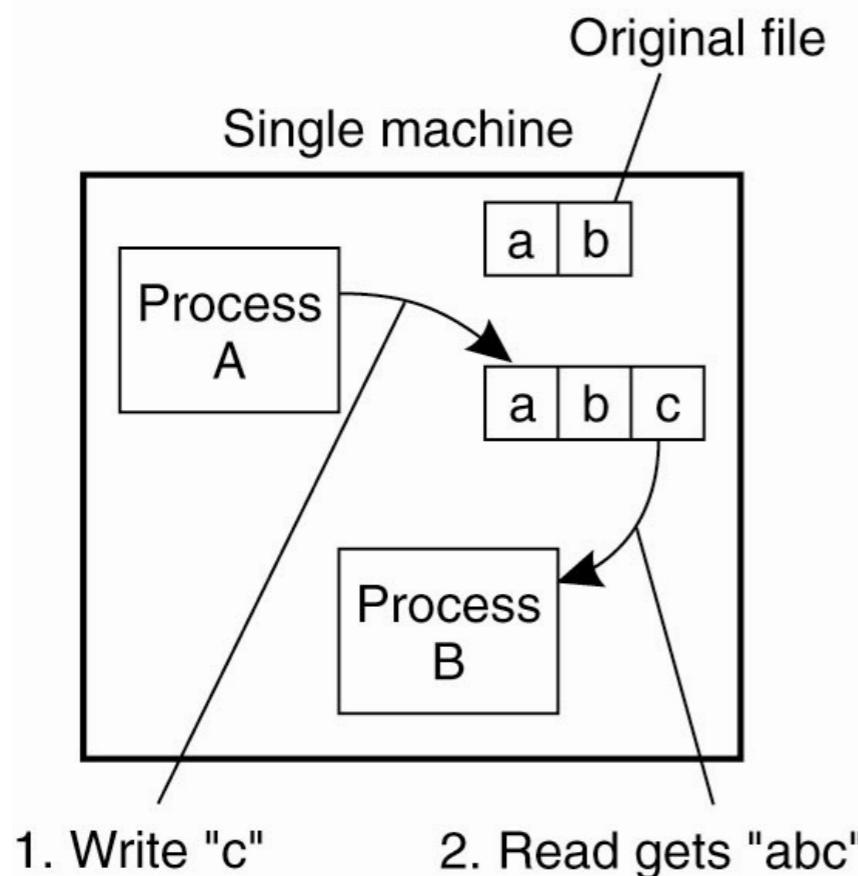
- Os sistemas de ficheiros distribuídos assentam tipicamente em RPCs. Em redes de larga escala (distribuição geográfica) têm que ser tidas em conta algumas optimizações:
- Uma é o agrupamento (batching) de comandos:



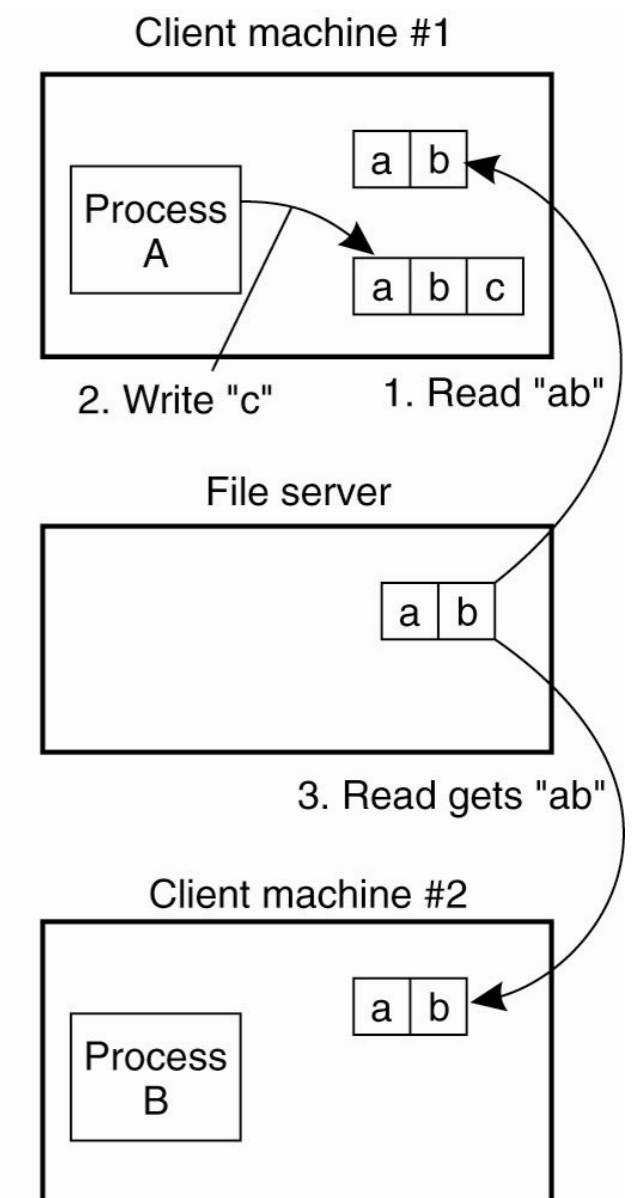
- A outra é caching de ficheiros ou blocos:



Num SFD é necessário ter em conta a ordem das operações concorrentes de escrita e leitura na coerência de dados que se



Sistema local



Sistema distribuído

● Semântica em UNIX

- uma leitura devolve o valor da última operação de escrita. Apenas se consegue obter em modelos de acesso remoto em que há apenas uma cópia dos ficheiros.

● Semântica Transacional

- o sistema oferece suporte para transações sobre ficheiros individuais. A implementação levanta grandes desafios para permitir o acesso a ficheiros fisicamente distribuídos.

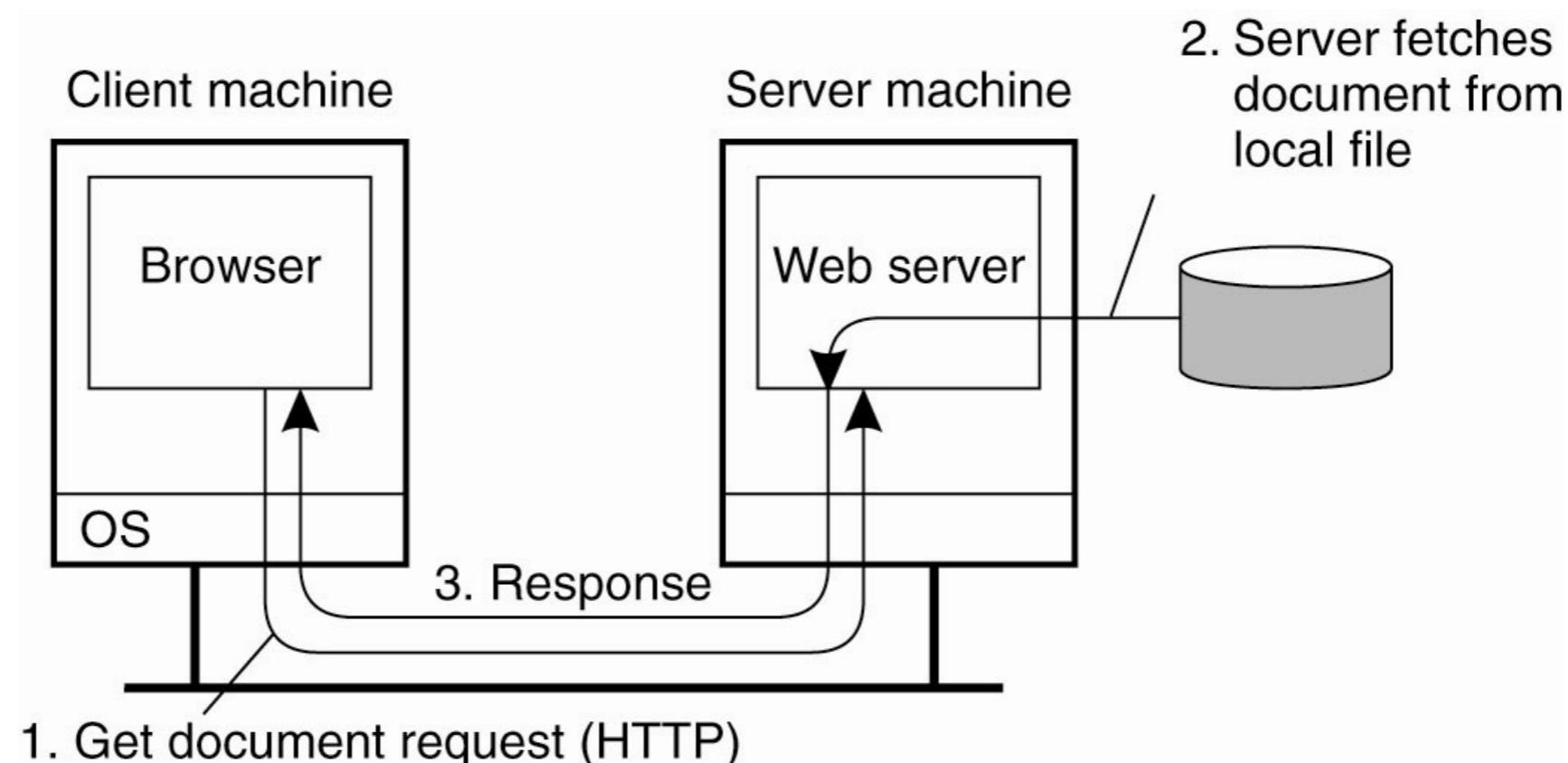
● Semântica de Sessão

- os efeitos de operações de escrita e leitura são visíveis pelo cliente que abriu uma sessão de acesso ao ficheiro. A implementação de controlo de sessões concorrentes poderá ser conservadora (locks) ou otimista (certificação no fecho das sessões).

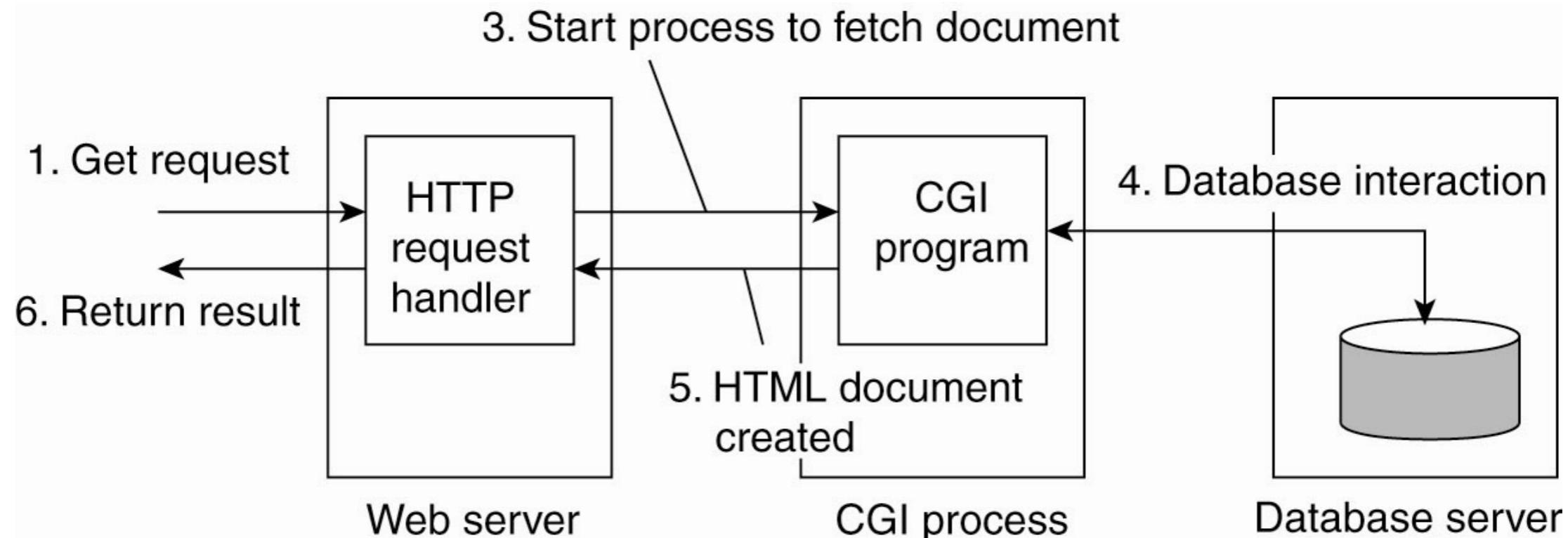
Sistemas distribuídos

-  Introdução
-  Arquiteturas e Modelos
-  Comunicação
-  Sistemas de objetos distribuídos
-  Sistemas de ficheiros distribuídos
-  Sistemas Web

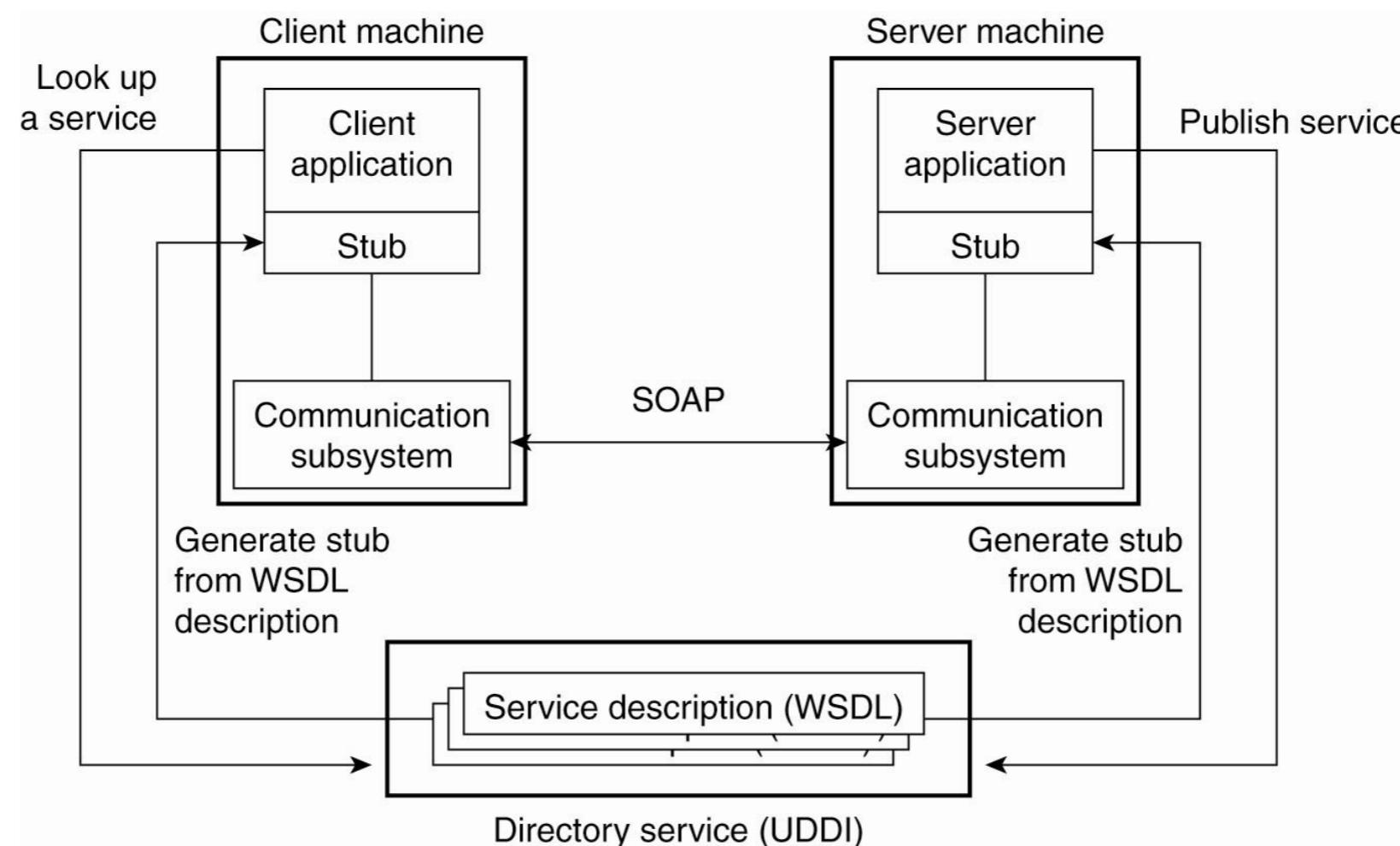
- A WWW é um sistema cliente/servidor enorme, com milhões de servidores, cada um disponibilizando milhares de “documentos”:
 - texto, HTML, XML, PDF, PS, imagens, áudio e vídeo, aplicações
 - programas/scripts que são executados do lado do cliente



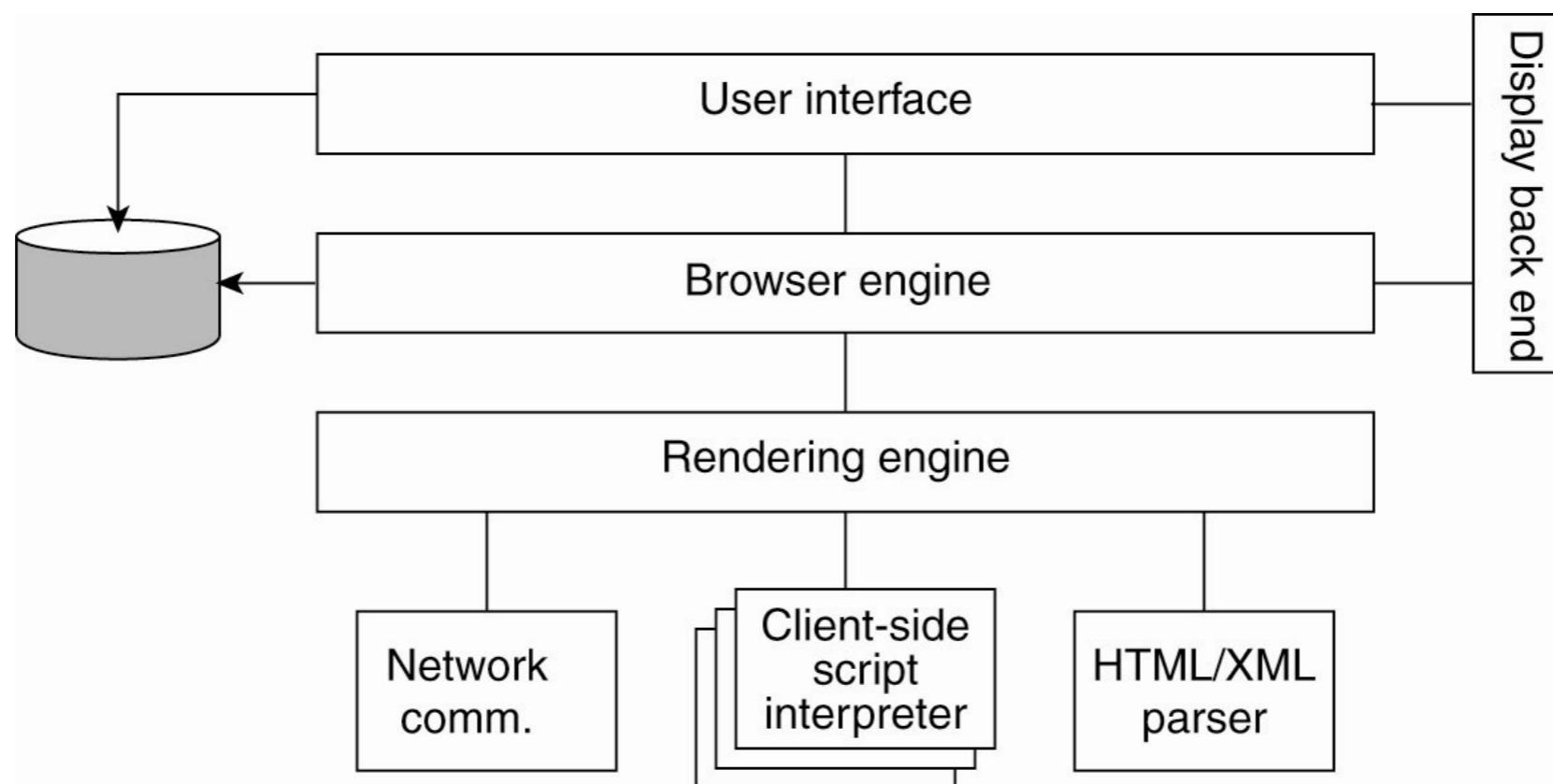
● Desde cedo que os sítios Web se organizaram em 3 camadas:



Para lá da utilização típica utilizador-sítio Web, os sítios Web começaram a oferecer serviços (funcionalidades/procedimentos com APIs bem definidas). Deu-se a standarização dos Web



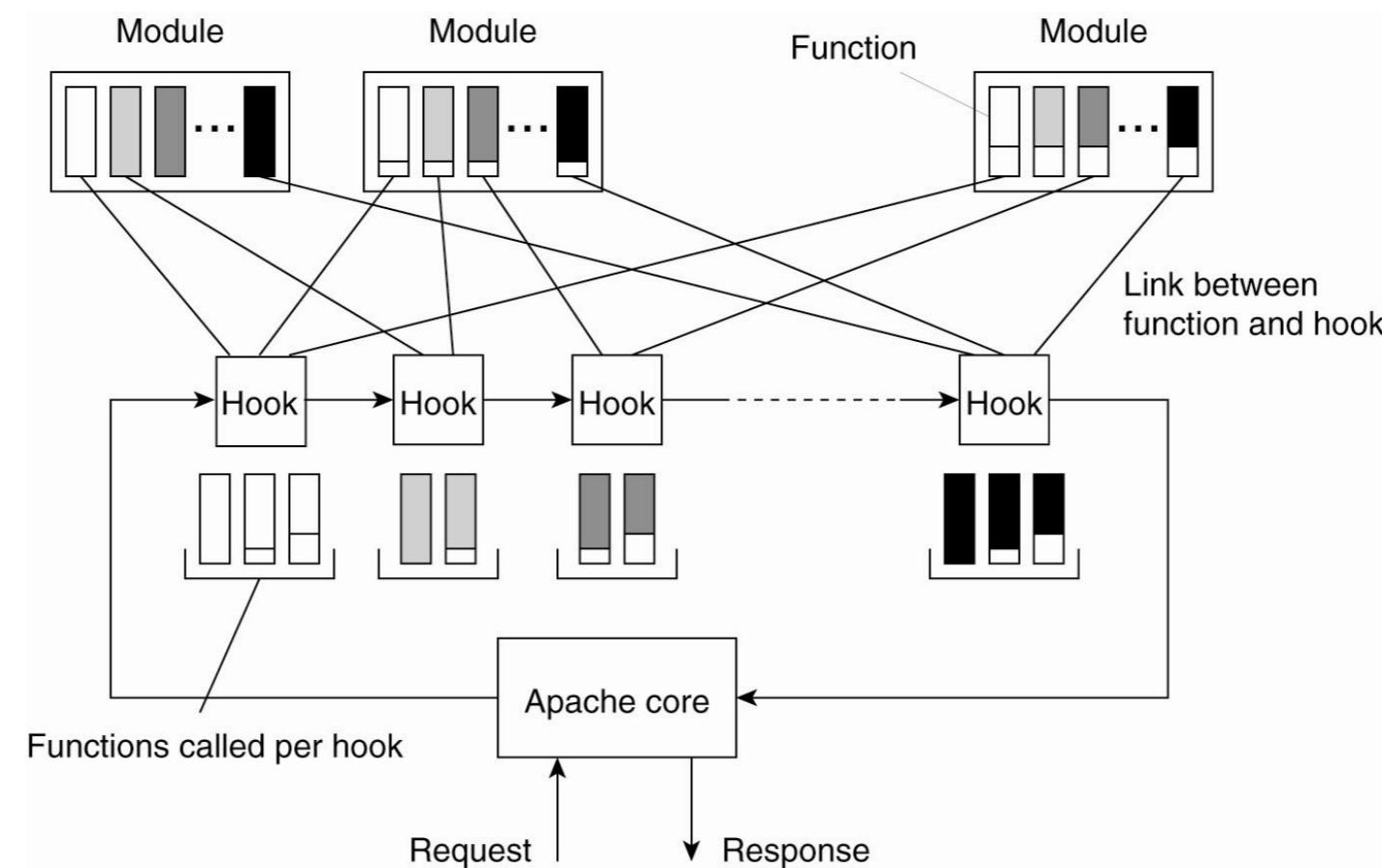
Os browsers são o software mais importante da Web do lado do cliente. Começaram por ser simples, limitando-se a mostrar HTML, que também começou por ser muito simples...



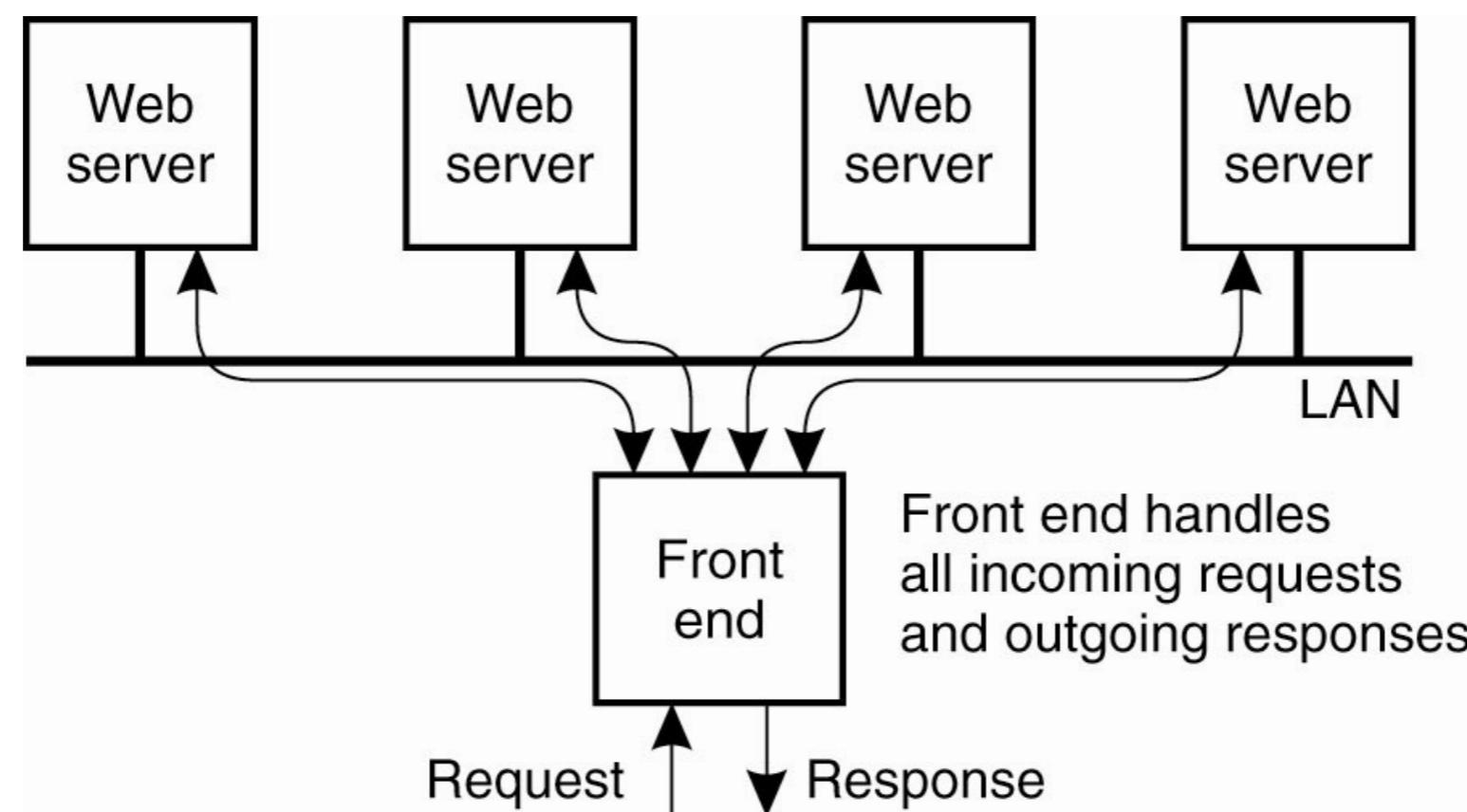
- Entre os clientes e os servidores, é típico existirem servidores intermediários, Web proxies, cuja função poderá ser de caching de documentos, pré-processamento de pedidos (ex.: Amazon Fire) ou adaptação de protocolos:



- Mais de metade dos sítios Web usa Apache server
- O servidor é organizado modularmente de acordo com os passos necessários ao processamento dos pedidos HTTP:



- De modo a aumentar o desempenho e a disponibilidade, os servidores WWW são normalmente conjuntos de servidores disponibilizando o serviço de forma transparente.



- A comunicação em sistemas Web é geralmente baseada em HTTP, um protocolo de transporte simples com as seguintes mensagens:

Operation	Description
Head	Request to return the header of a document
Get	Request to return a document to the client
Put	Request to store a document
Post	Provide data that are to be added to a document (collection)
Delete	Request to delete a document

Exemplos informação no cabeçalho de mensagens

Header	C/S	Contents
Accept	C	The type of documents the client can handle
Accept-Charset	C	The character sets are acceptable for the client
Accept-Encoding	C	The document encodings the client can handle
Accept-Language	C	The natural language the client can handle
Authorization	C	A list of the client's credentials
WWW-Authenticate	S	Security challenge the client should respond to
Date	C+S	Date and time the message was sent
ETag	S	The tags associated with the returned document
Expires	S	The time for how long the response remains valid
From	C	The client's e-mail address
Host	C	The TCP address of the document's server
If-Match	C	The tags the document should have
If-None-Match	C	The tags the document should not have
If-Modified-Since	C	Return a document only if it has been modified since the specified time
If-Unmodified-Since	C	Return a document only if it has not been modified since the specified time
Last-Modified	S	The time the returned document was last modified
Location	S	A document reference to which the client should redirect its request
Referer	C	Refers to client's most recently requested document
Upgrade	C+S	The application protocol sender wants to switch to
Warning	C+S	Information about status of the data in the message

- O protocolo de standard de facto para a comunicação de Web services é o SOAP – Simple Object Access Protocol
- É baseado em XML
- Depende do protocolo de transporte sobre o qual é usado (HTTP, SMTP, etc.)
- Pode ser usado quer em estilo de troca de mensagens: enviar documento e obter resposta, ou em estilo RPC servindo para invocar um Web service.

Exemplo de uma mensagem SOAP

```
env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
      <n:priority>1</n:priority>
      <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```