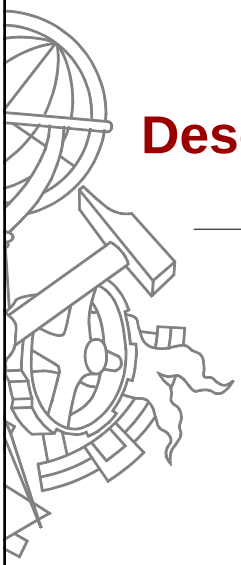


Ambientes de Desenvolvimento Avançados

http://www.dei.isep.ipp.pt/~jtavares/ADAV/ADAV.htm

Aula 3

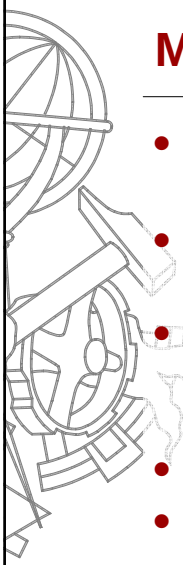
Engenharia Informática

2004/2005

José António Tavares
jrt@isep.ipp.pt

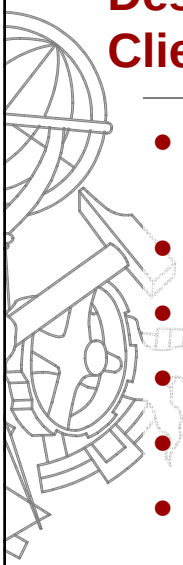
Introdução ao conceito Cliente-Servidor

Baseado num documento de *Alexandre Bragança - 1998/99*



Motivação para os componentes

- O desenvolvimento da WWW e *Internet*
  - Sistemas de serviços poucos coordenados
- Técnicas de projecto e linguagens Orientadas aos Objectos
- Movimento da computação baseada em *Mainframes* para computação do tipo Cliente-Servidor
- Ritmo rápido das mudanças tecnológicas
- Necessidades económicas de maximizar a reutilização



Desenvolvimento de sistemas Cliente-Servidor - Conteúdo

- Características de um sistema cliente-servidor
- Diferentes 'versões' de cliente-servidor
- Middleware
- Balanceamento cliente-servidor
- Cliente-servidor 2 camadas vs 3 camadas
- Componentes de uma arquitectura cliente-servidor

Características de um sistema Cliente-Servidor

- **Serviço**

  Cliente-servidor é uma relação entre processos que estão a correr em máquinas diferentes. O processo servidor é o fornecedor dos serviços. O cliente é o consumidor de serviços. Fundamentalmente uma arquitectura cliente-servidor implemente uma separação lógica de funções baseada no conceito de serviço.

- **Recursos partilhados**

  Um servidor pode servir vários clientes ao mesmo tempo e gerir os acessos a recursos partilhados.

- **Protocolos assimétricos**

  Existe uma relação de muitos-para-um entre clientes e servidor. Os clientes iniciam o diálogo através da requisição de um serviço. Os servidores esperam passivamente os pedidos dos clientes.

Características de um sistema Cliente-Servidor

- **Localização transparente**

  O servidor é um processo que pode residir na mesma máquina que o cliente ou numa máquina diferente que esteja ligada através de uma rede. Um programa pode ter o papel de cliente, servidor ou ambos.

- **Independência**

  O conceito inerente às arquitecturas cliente-servidor baseia-se em *software* que deve ser independente de *hardware* ou sistemas operativos.

- **Baseado na transmissão de mensagens**

  Clientes e servidores devem estar ligados de forma 'fraca', ou seja, não deve ser obrigatório que o servidor esteja a correr para que o cliente possa correr. Sistemas deste tipo são normalmente baseados em mensagens. A mensagem é o mecanismo de transporte para os pedidos e respostas dos serviços.

Características de um sistema Cliente-Servidor

- **Encapsulamento de serviços**

  Um servidor deve ser um programa 'especializado'. As mensagens transmitem o pedido de serviço ao servidor. O servidor é que deve ser responsável pela forma como implementa o serviço. A forma de implementar os serviços pode ser melhorada/alterada sem implicações ao nível dos clientes.

- **Escalabilidade**

  Os sistemas cliente-servidor podem evoluir facilmente quer por adição de novos clientes quer por evolução para novas máquinas servidoras mais potentes.

- **Integridade**

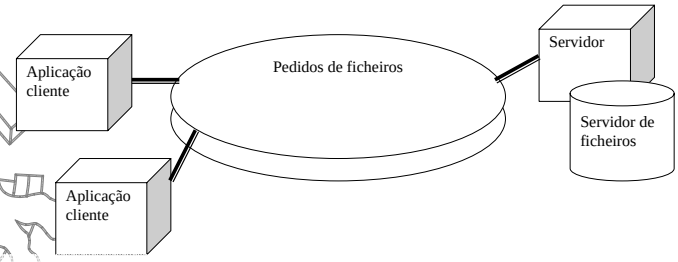
  O código e dados do servidor devem ser mantidos centralmente. Desta forma reduzem-se os custos de manutenção e aumenta-se a integridade dos dados.

Diferentes 'versões' de Cliente-Servidor

- Servidores de ficheiros
- Servidores de bases de dados
- Servidores de transacções
- Servidores de *groupware*
- Servidores de objectos
- Servidores de web

Diferentes 'versões' de Cliente-Servidor

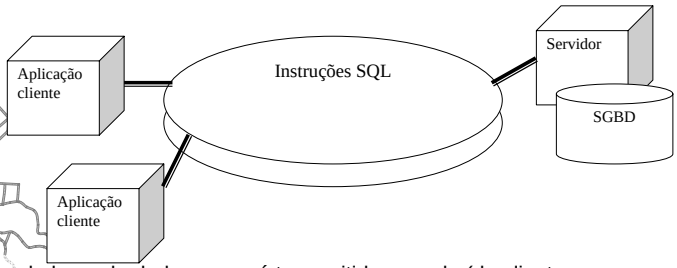
Servidores de ficheiros



Num sistema deste género o cliente executa pedidos de registos de ficheiros ao servidor de ficheiros através da rede.
É uma forma muito primitiva de serviço de dados e provoca uma troca muito elevada de mensagens pela rede.
São no entanto sistemas necessários para a partilha repositórios de ficheiros em rede (documentos, imagens, desenhos, etc.…).

Diferentes 'versões' de Cliente-Servidor

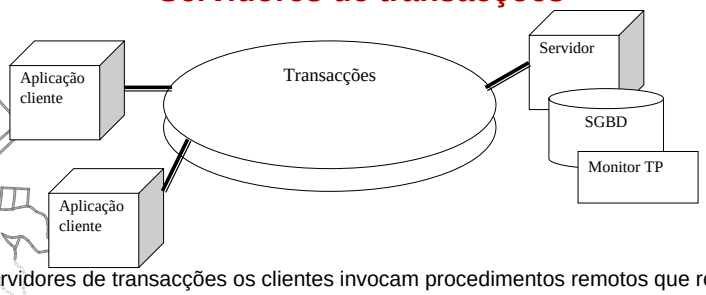
Servidores de bases de dados



Num servidor de base de dados o que é transmitido na rede (do cliente para o servidor) são instruções de SQL. O resultado da instruções de SQL são envidas para o cliente. O código que processa as instruções SQL e os dados residem na mesma máquina (o servidor). É o servidor que determina quais os registos resultantes da instrução e são apenas estes que são enviados pela rede.
Nos servidores de ficheiros todos os registos são enviados pela rede e é o cliente que determina aqueles que interessam.

Diferentes 'versões' de Cliente-Servidor

Servidores de transacções



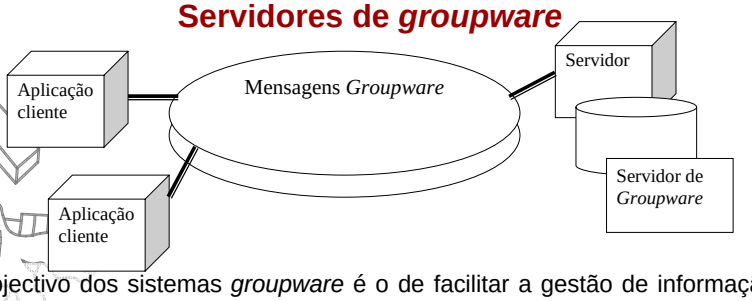
Com os servidores de transacções os clientes invocam procedimentos remotos que residem no servidor (com uma base de dados). Estes procedimentos remotos são constituídos por grupos de instruções SQL. As instruções do procedimento são executadas na totalidade ou então falha tudo.

Ao contrário do simples servidor de base de dados neste caso o programador tem que escrever código no cliente e no servidor.

Estes sistemas usualmente designam-se de OLTP (*Online Transaction Processing*)

Diferentes 'versões' de Cliente-Servidor

Servidores de *groupware*



O objectivo dos sistemas *groupware* é o de facilitar a gestão de informação semi-estruturada (ou não-estruturada) tal como texto, imagem, e-mail, etc.. Para além disso normalmente estes sistemas também implementam capacidades de automação de *workflow*.

Estes sistemas suportam-se sobre sistemas de transmissão de mensagens.

Existem diversos sistemas que embora se possam interligar são implementados de formas diferentes.

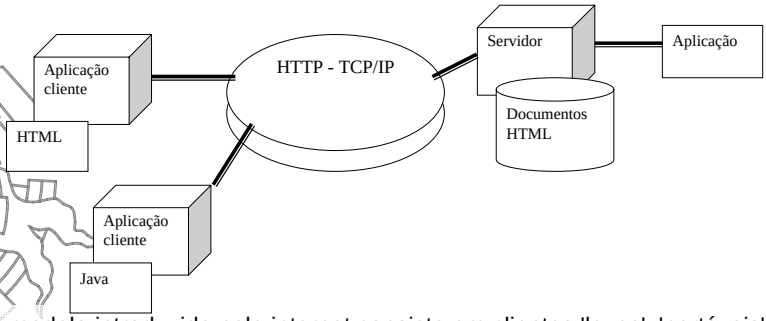
Diferentes 'versões' de Cliente-Servidor

Servidores de objectos

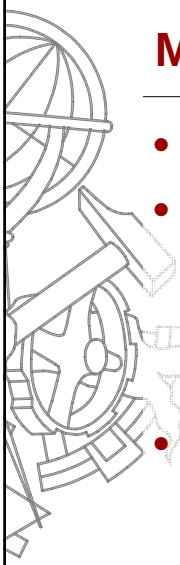
Um sistema deste tipo é implementado através de um conjunto de objectos que podem comunicar entre si. Objectos cliente comunicam com objectos servidores através do *Object Request Broker* (ORB).
Quando o cliente invoca um método num objecto remoto o ORB localiza a instância do objecto servidor, invoca o método e retorna o resultado ao objecto cliente.
As tecnologias concorrentes nesta área são o CORBA e o DCOM.

Diferentes 'versões' de Cliente-Servidor

Servidores de web

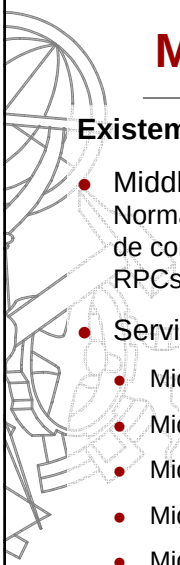


O novo modelo introduzido pela internet consiste em clientes 'leves', 'portáveis' e 'universais' que comunicam com servidores 'super pesados' (servem milhares ou milhões de clientes).
A comunicação é feita por um protocolo do tipo RPC designado por HTTP.



Middleware

- 'Aquilo' que se encontra entre o cliente e o servidor
- Começa com a API que o cliente utiliza para invocar um serviço, inclui a transmissão do pedido pela rede assim como da resposta. O Middleware não inclui o *software* que executa o serviço (esta é a função do servidor)
- O Middleware não inclui o interface (esta é uma função do cliente)



Middleware

Existem 2 categorias de middleware

- Middleware genérico
  Normalmente inclui tudo o que tem que ver com transporte (*stacks* de comunicação, serviços de directório, serviços de autenticação, RPCs, etc.).
- Serviços de Middleware
  - Middleware especifico de base de dados (ODBC…)
  - Middleware especifico de groupware (MAPI, VIM…)
  - Middleware especifico de serviços de objectos (CORBA, DCOM…)
  - Middleware especifico de internet (HTTP, SSL…)
  - Middleware de gestão especifico (ORB…)

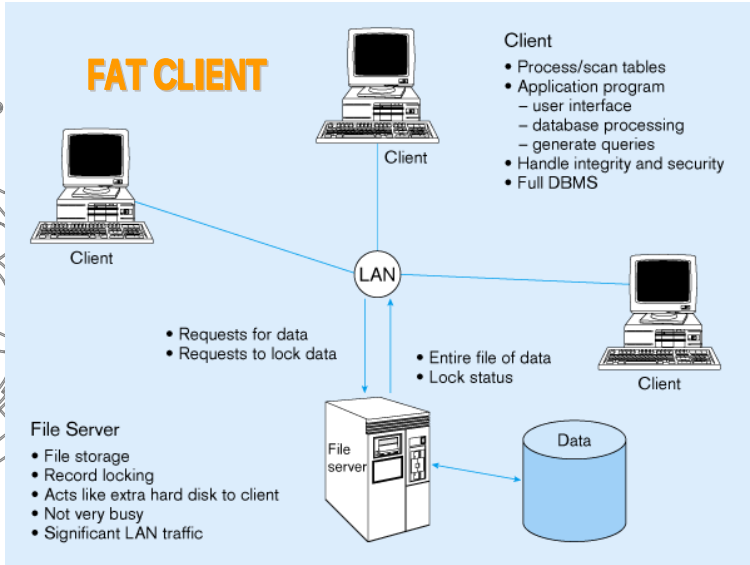
Componentes de uma arquitectura Cliente-Servidor

- **Cliente**
  Baseia-se fundamentalmente no interface gráfico da aplicação. Cada vez mais este é um Object Oriented User Interface (OOUI). Acede a serviços através do middleware
- **Servidor**
  Executa a parte de serviços da aplicação. Utiliza normalmente servidores específicos de software (SGBDs (relacionais ou OO), Monitores TP, servidores de groupware, servidores de objectos e servidores de Web)
- **Middleware**
  Executa no cliente e no servidor. Normalmente está dividido em stacks de transporte (TCP/IP, NetBios, IPX/SPX, SNA, …), Sistemas Operativos de Rede ou serviços de rede (NOS = serviços de directório, segurança, RPC, mensagens, etc.) e serviços específicos de middleware (ODBC, Mail, ORB, HTTP, …)

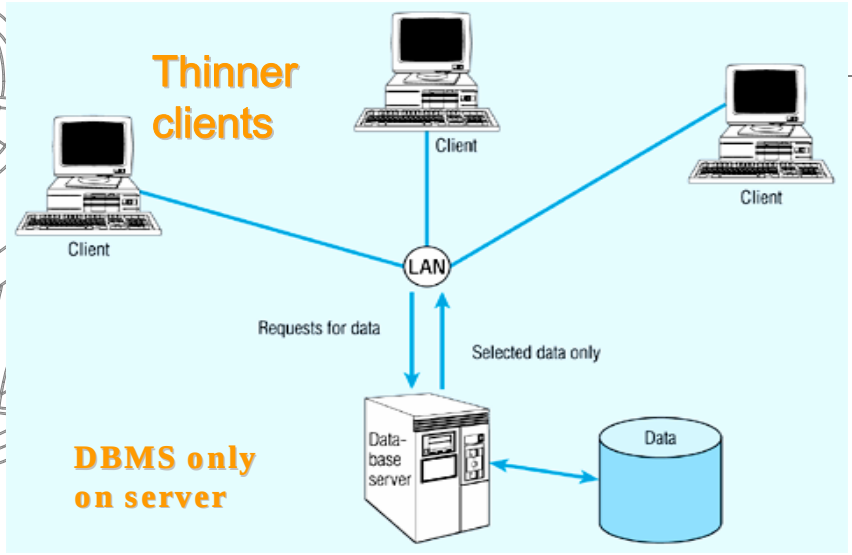
Balanceamento Cliente-Servidor

- Como distribuir uma aplicação entre cliente e servidor?
- Groupware e servidores de Web são servidores 'pesados'
- Servidores de ficheiros e de base de dados são exemplos de clientes 'pesados'
- Objectos distribuídos estão no meio (normalmente…)

Balanceamento cliente-servidor



Balanceamento cliente-servidor



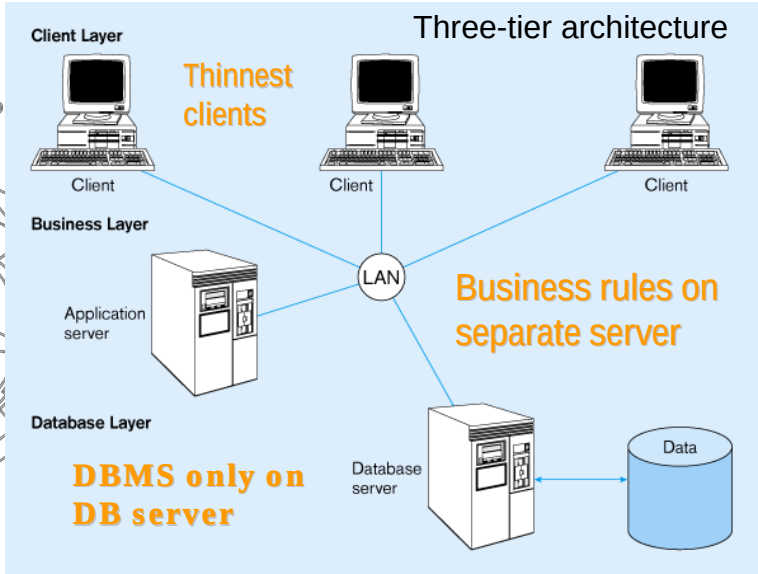
Cliente-servidor 2 camadas vs 3 camadas (2-*tier* vs 3-*tier*)

- As aplicações podem normalmente ser divididas logicamente em interface, lógica de negócio e base de dados…
- No chamado cliente-servidor a 2 níveis a lógica de negócio está dividida entre cliente e servidor
- No cliente-servidor a 3 níveis (ou **n** níveis) existe um nível independente só para a lógica de negócio. Esta solução permite um desempenho e uma evolução superior.
- O exemplo por excelência desta arquitectura são as soluções de objectos distribuídos. As soluções Web também se enquadram normalmente desta categoria.

Cliente-servidor 3 camadas

- Em geral, estes incluem outra camada do tipo servidor entre o cliente e o usual servidor;
- Este servidor adicional pode ser usado para diferentes propósitos;
- Frequentemente os programas de aplicação residem no servidor adicional (*Application Server*)
- Na maior parte dos casos os cliente apenas contêm a Interface com o Utilizador e fazem um processamento mais leve. O armazenamento de dados é limitado ou não existe.

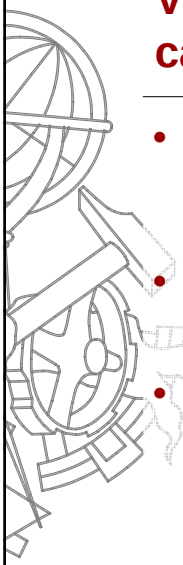
Cliente-servidor 3 camadas



Vantagens das arquitecturas de 3 camadas

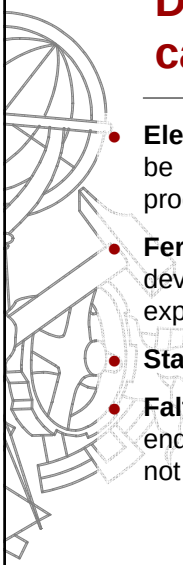
- **Escalabilidade** – middle tier can be used to reduce the load on a database server by using a transaction processing (TP) monitor to reduce the number of connections to a server, and additional application servers can be added to distribute application processing
- **Flexibilidade tecnológica** – easier to change DBMS engines – middle tier can be moved to a different platform. Simplified presentation interfaces make it easier to implement new interfaces
- **Redução de custo a longo prazo** – use of off-the-shelf components or services in the middle tier can reduce costs, as can substitution of modules within an application rather than a whole application

Vantagens das arquitecturas de 3 camadas

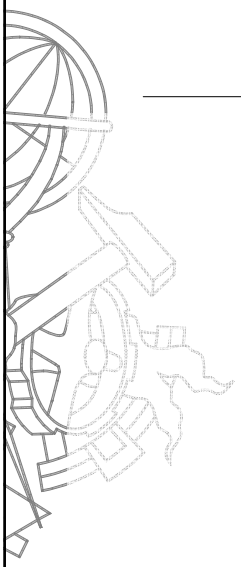


- **Melhor adaptação dos sistemas às nessidades de negócio** – new modules can be built to support specific business needs rather than building more general, complete applications
- **Serviço de apoio aos clientes melhorado** – multiple interfaces on different clients can access the same business process
- **Vantagem competitiva** – ability to react to business changes quickly by changing small modules of code rather than entire applications

Desafios para as arquitecturas de 3 camadas



- **Elevados custo de curto prazo** – presentation component must be split from process component – this requires more programming
- **Ferramentas, treino e experiência** – currently lack of development tools and training programmes, and people experienced in the technology
- **Standards incompatíveis** – few standards yet proposed
- **Falta de ferramentas compatíveis do utilizador final** – many end-user tools such as spreadsheets and report generators do not yet work through middle-tier services



Questões

?