Computação Distribuída

Cap II – Arquitecturas de Sistemas Distribuídos

Licenciatura em Engenharia Informática

Universidade Lusófona

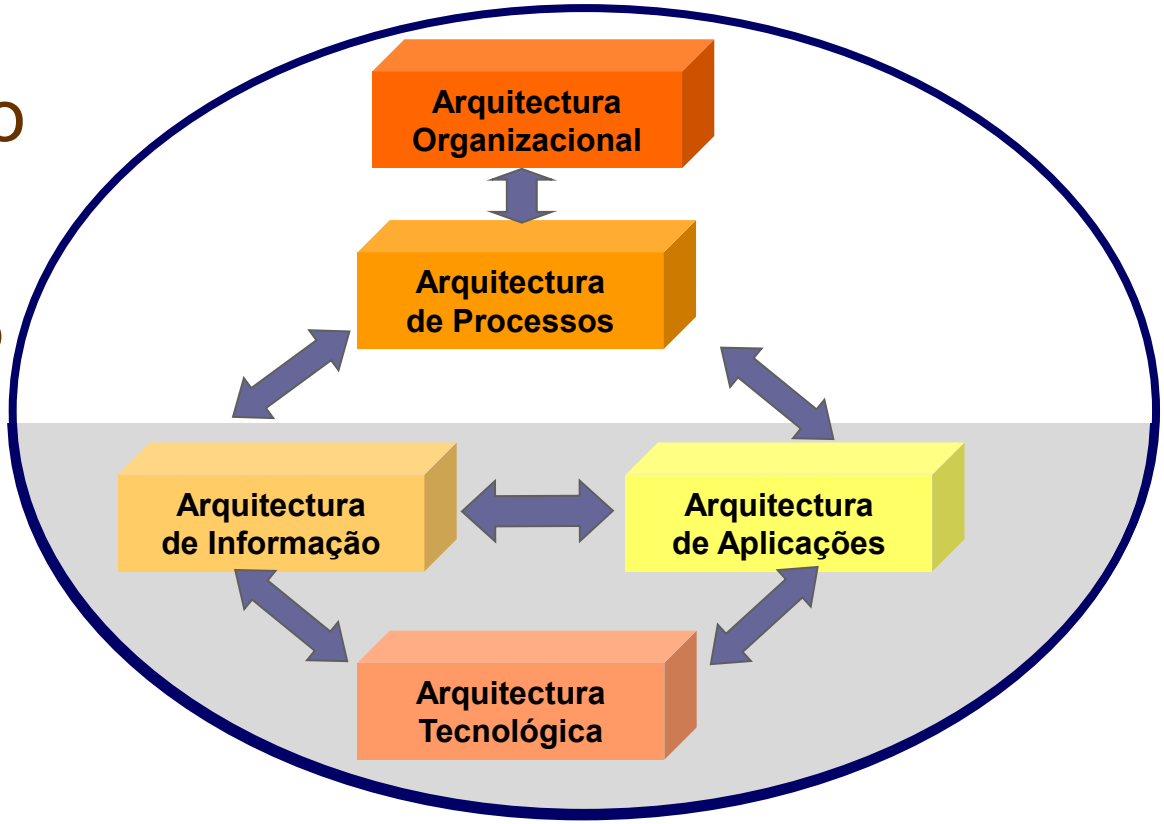
Prof. Paulo Guedes (paulo.guedes@ulusofona.pt)

Noção de Arquitectura

- A arquitectura define a estrutura de um sistema distribuído em termos de componentes especificáveis separadamente. Fornece um modelo para:
  - Especificação
  - Desenvolvimento
  - Funcionamento
  - Evolução
- O modelo de arquitectura
  - **Define a funcionalidade e localização dos componentes do sistema**
  - **Identifica as relações e os padrões de comunicação existentes entre eles**
- A correcta definição de uma arquitectura é importante para garantir:
  - Desempenho
  - Fiabilidade
  - Segurança
  - Modularidade
- Analogia: arquitectura de um edifício

As vistas fundamentais da arquitectura

- Organização
- Processos
- **Informação**
- **Aplicações**
- **Tecnologia**



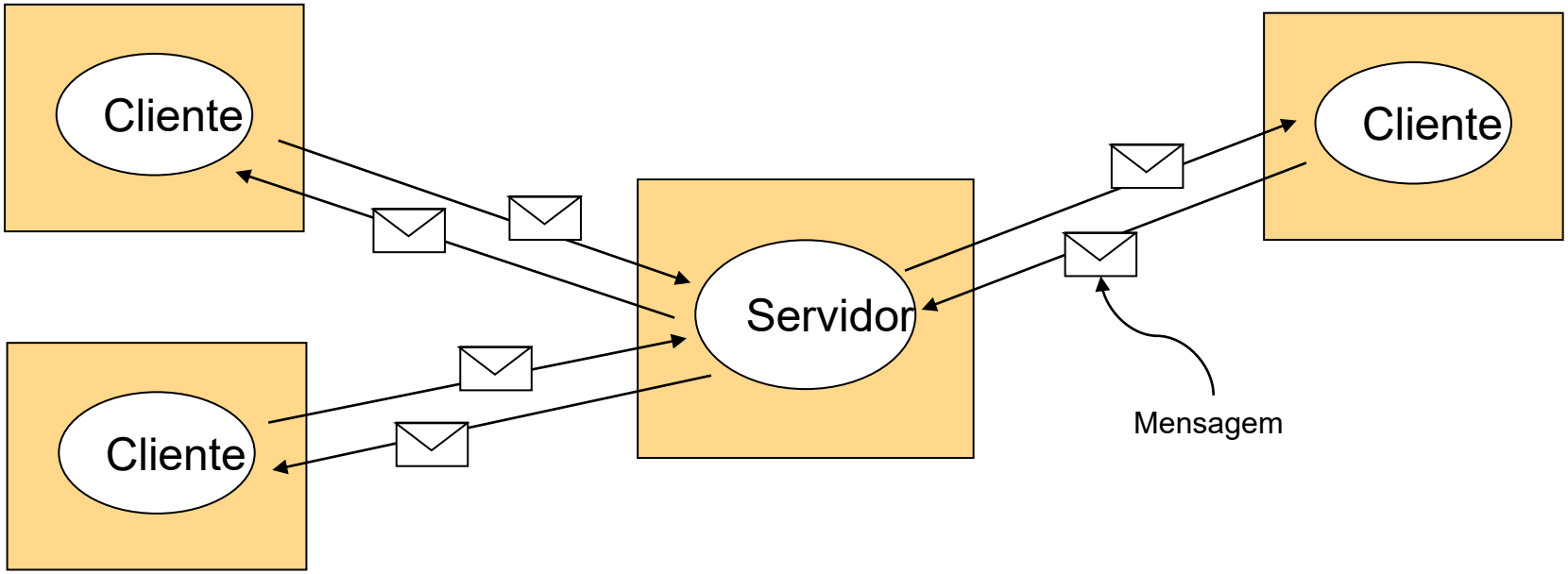
Componentes da Arquitectura

- Processo
  - Unidade de execução de programas num sistema
- Objecto
  - Unidade de encapsulamento de código e/ou dados
  - Os objectos são instanciados e executados em processos
- Serviço
  - Unidade funcional de distribuição, que gere recursos e fornece funcionalidades a aplicações e utilizadores
  - É implementado por objectos executados num ou vários processos
  - É definido por uma ***interface*** e um protocolo
  - Deve poder ser identificado e localizado
- Servidor
  - Componente que fornece um ou mais serviços, podendo abranger um ou mais componentes hardware
- Cliente
  - Entidade que invoca os serviços num servidor

Tipos de Arquitectura

- Cliente – Servidor
- Servidor com cache
- Peer to peer
- Código móvel
- Arquitecturas multi-nível
  - 2 níveis
  - 3 níveis
- Cloud computing

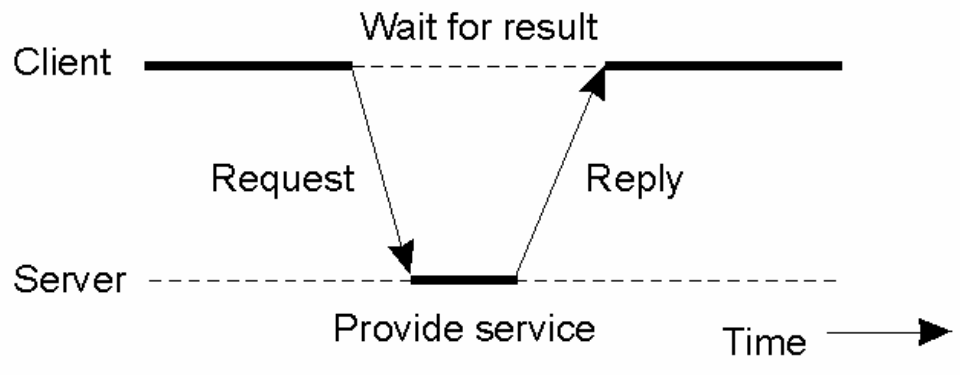
Arquitectura Cliente - Servidor



- Arquitectura tradicional dos Sistemas Distribuídos
  - Caracteriza um sistema em que só há dois tipos de componentes
    - Arquitectura simples e fácil de implementar
  - Servidor: executa operações invocadas pelos clientes retornando resultados
  - Cliente: invoca operações em servidores
  - Mensagem: dados trocados entre cliente e servidor obedecendo a um formato especifico do serviço

Características da arquitectura Cliente - Servidor

- Modelo de invocação de *request/reply* com envio e retorno de mensagens
- Modelo de interacção é geralmente síncrono: o cliente espera pelo retorno da invocação

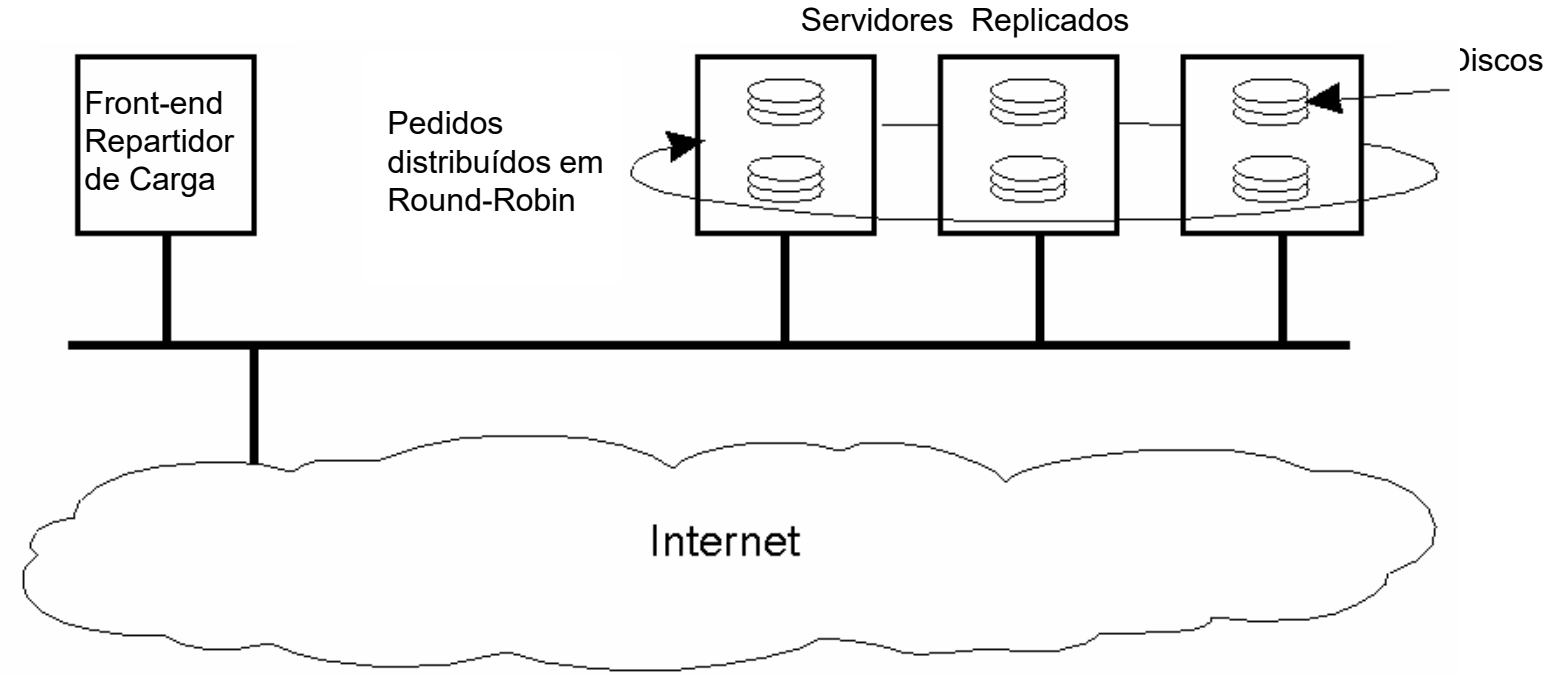


- Relação N para 1 pode trazer problemas de desempenho e fiabilidade
  - Toda a carga é concentrada no servidor: *bottleneck*
  - A escalabilidade é problemática
  - Ponto de falha único: servidor
- Na realidade, a arquitecturas utilizadas são geralmente variantes do C/S
  - Cliente/Servidor-Múltiplo
- Exemplos
  - Browser/Servidor Web
  - Etc…

Perguntas a que devo ser capaz de responder

- O que define a arquitetura de uma aplicação distribuída ?
- Quais são as componentes da arquitetura de uma aplicação distribuída ?
- Quais são as características da arquitetura cliente – servidor ?
- Porque é que esta arquitetura é tão utilizada em sistemas empresariais ?

Variantes Cliente - Servidor (i)



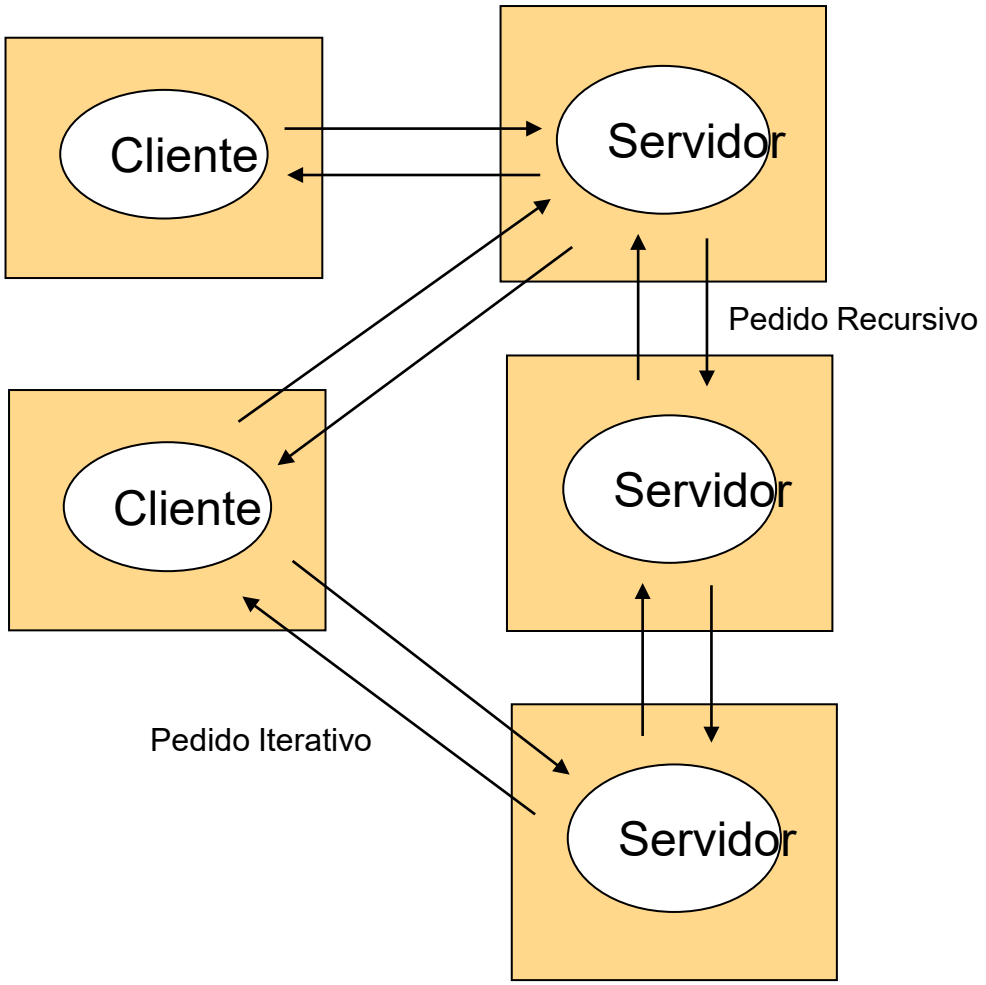
Servidores Redundantes

- O mesmo serviço é fornecido por vários servidores
  - Distribuição da carga, evitando ponto único de falha
  - Utilização de Repartidores de Carga (*Load-Balancers*)
  - Solução utilizada nos grandes portais
- Dificuldade em gerir a coerência de estado em cada réplica
  - Problema das Sessões

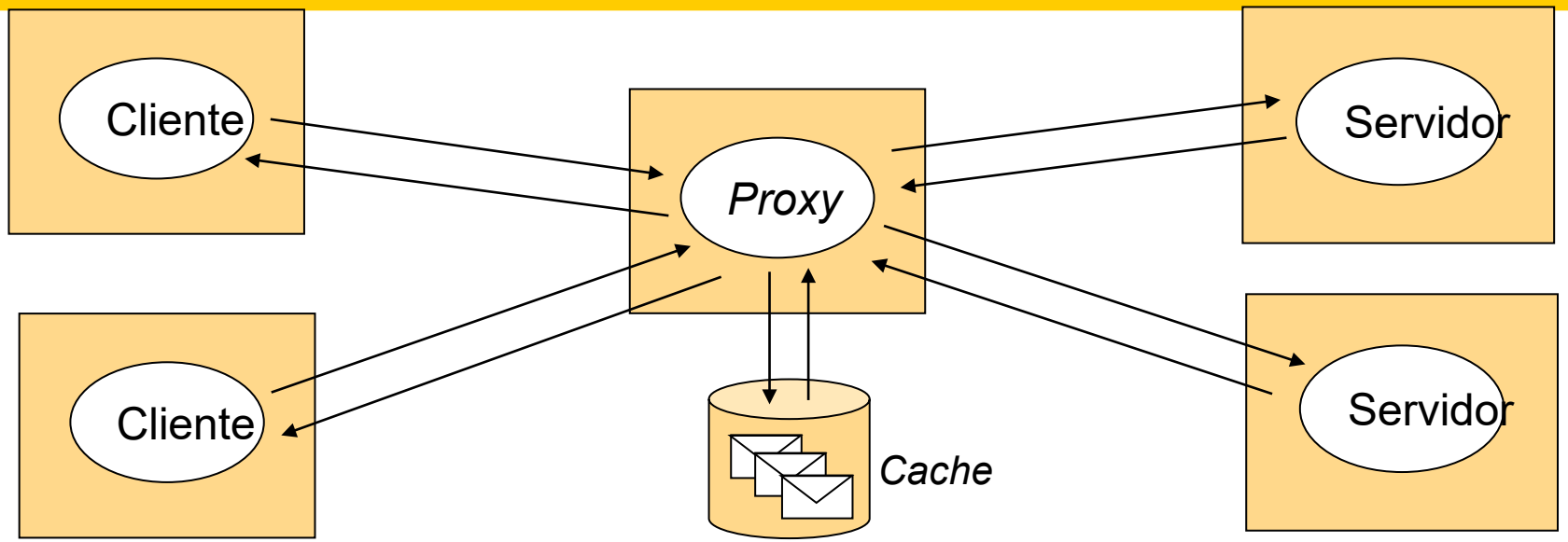
Variantes Cliente - Servidor (ii)

Servidores Repartidos

- O serviço é repartido por vários servidores
  - Cada servidor implementa uma parte do serviço
  - Distribuição da carga, evitando ponto único de falha
  - Solução utilizada no serviço de nomes (DNS)
- A invocação dos servidores pode ser feita
  - directamente pelos clientes por redirecção
    - Serviço iterativo
  - Pelo servidor inicial
    - Serviço recursivo



Utilização de *Proxy* e *Cache*

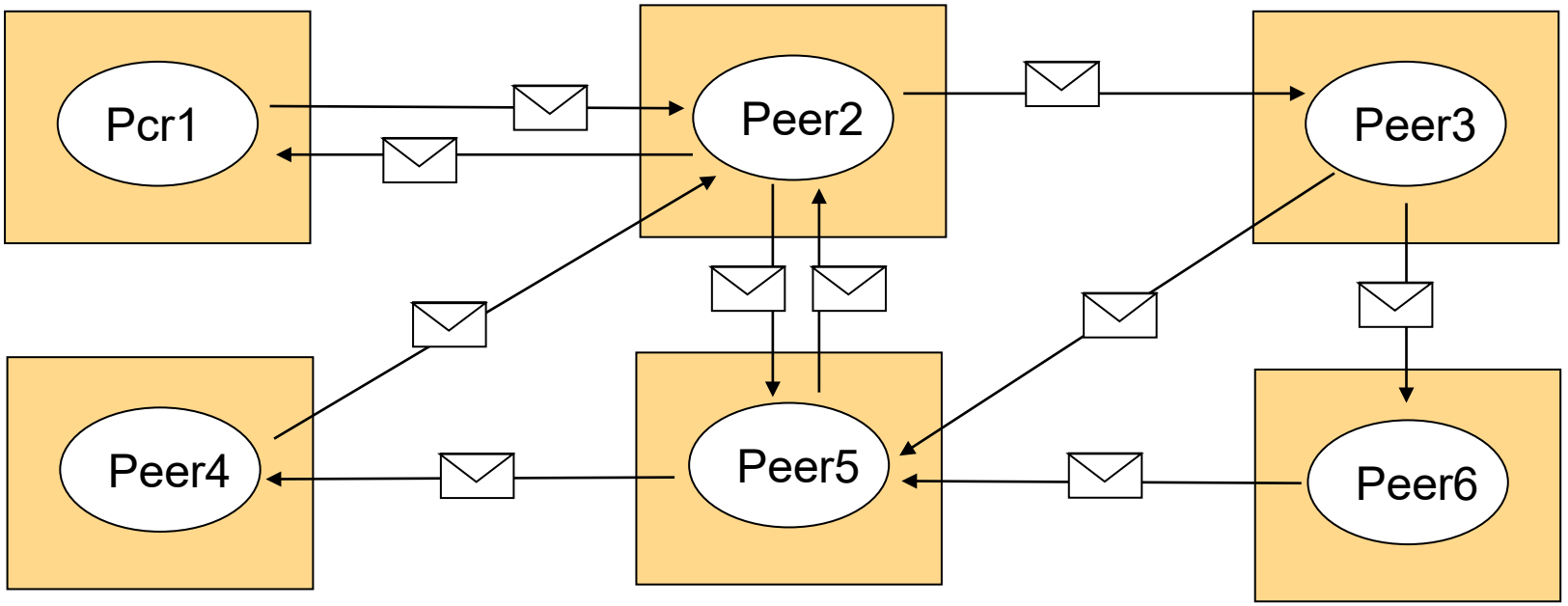


- O *proxy é um* representante de um serviço situado na proximidade imediata dos clientes
  - Simultaneamente cliente e servidor
- O *proxy* gere um *cache* que é um repositório de objectos mais frequentemente utilizados
  - Permite diminuir os tempos de acessos aos objectos e a carga dos servidores, introduzindo um certo grau de tolerância a falhas
  - A gestão do *cache* e a sua dimensão são essenciais para garantir a sua eficácia, devendo existir mecanismos para assegurar a sua coerência
- Ex: Proxy Web Server, *cache* NFS

Perguntas a que devo ser capaz de responder

- Quais são as motivações para ter servidores replicados ?
- Porque é que a programação de servidores replicados pode ser mais complexa ?
- Com servidores repartidos, que é uma resposta iterativa e uma resposta recursiva ?
- Quais são as vantagens da utilização de um servidor proxy com cache ?

Arquitectura P2P (*Peer-to-Peer*)



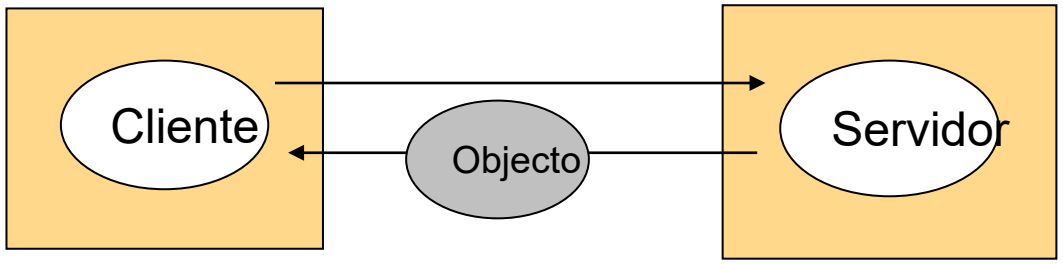
- Arquitectura baseada em componentes que desempenham papéis idênticos (*pares*)
  - Interagem cooperativamente assumindo o papel de cliente e/ou servidor simultaneamente
  - Permite explorar a capacidade de processamento e armazenamento de múltiplos computadores numa rede

Características P2P

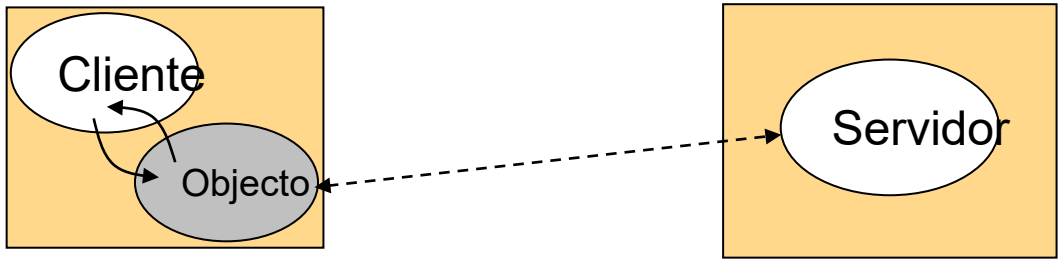
- Modelo de interacção complexo
  - Dificuldade de implementação
  - Administração problemática
- Dispersão e número elevado de componentes
  - Problemas de Segurança
- Vantagens
  - Escalabilidade facilitada
  - Maior fiabilidade do serviço como um todo
    - Não existe ponto único de falha
- Os problemas de registo e pesquisa de serviço necessitam de um serviço centralizado, ou a utilização de algoritmos distribuídos complexos
  - Tema de investigação actual
  - Arquitecturas Híbridas com um ponto único de registo
  - *Overlay* de Roteamento Distribuído
- Muitas aplicações P2P Web utiliza(ra)m este último modelo
  - O BitTorrent é um dos exemplos mais conhecidos

Código Móvel

1. O pedido do cliente desencadeia o *download* de um objecto executável

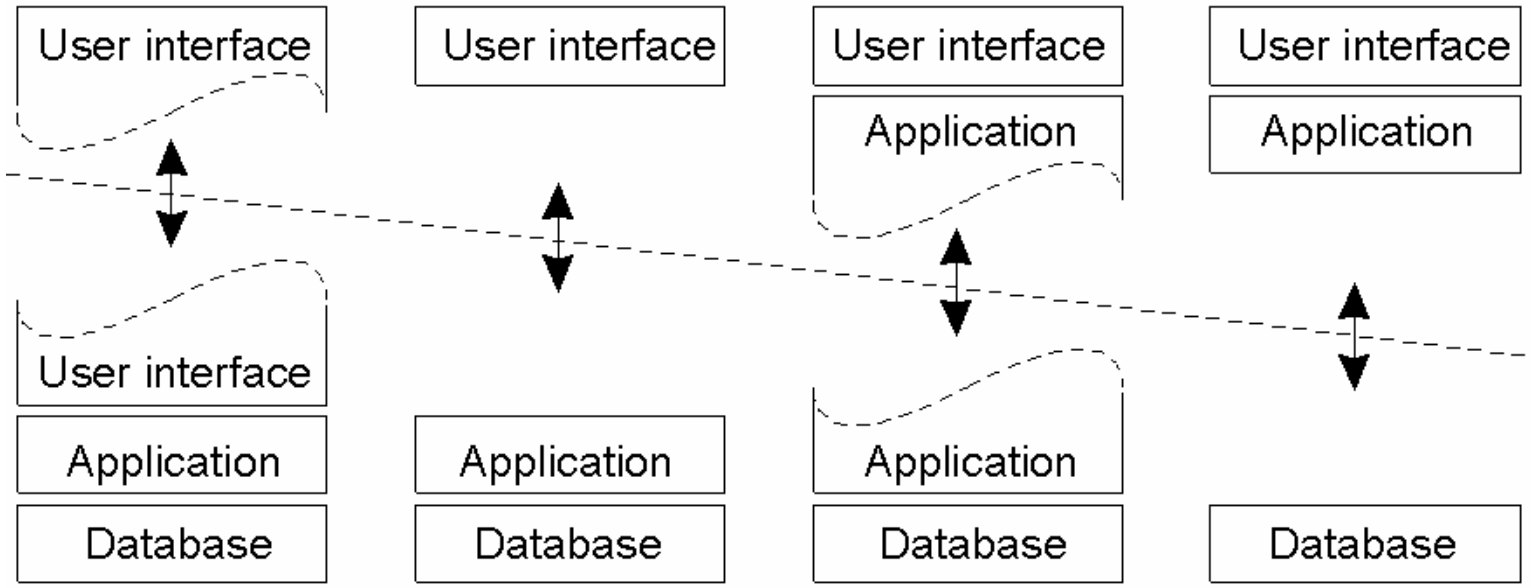


2. O cliente interage localmente com o objecto



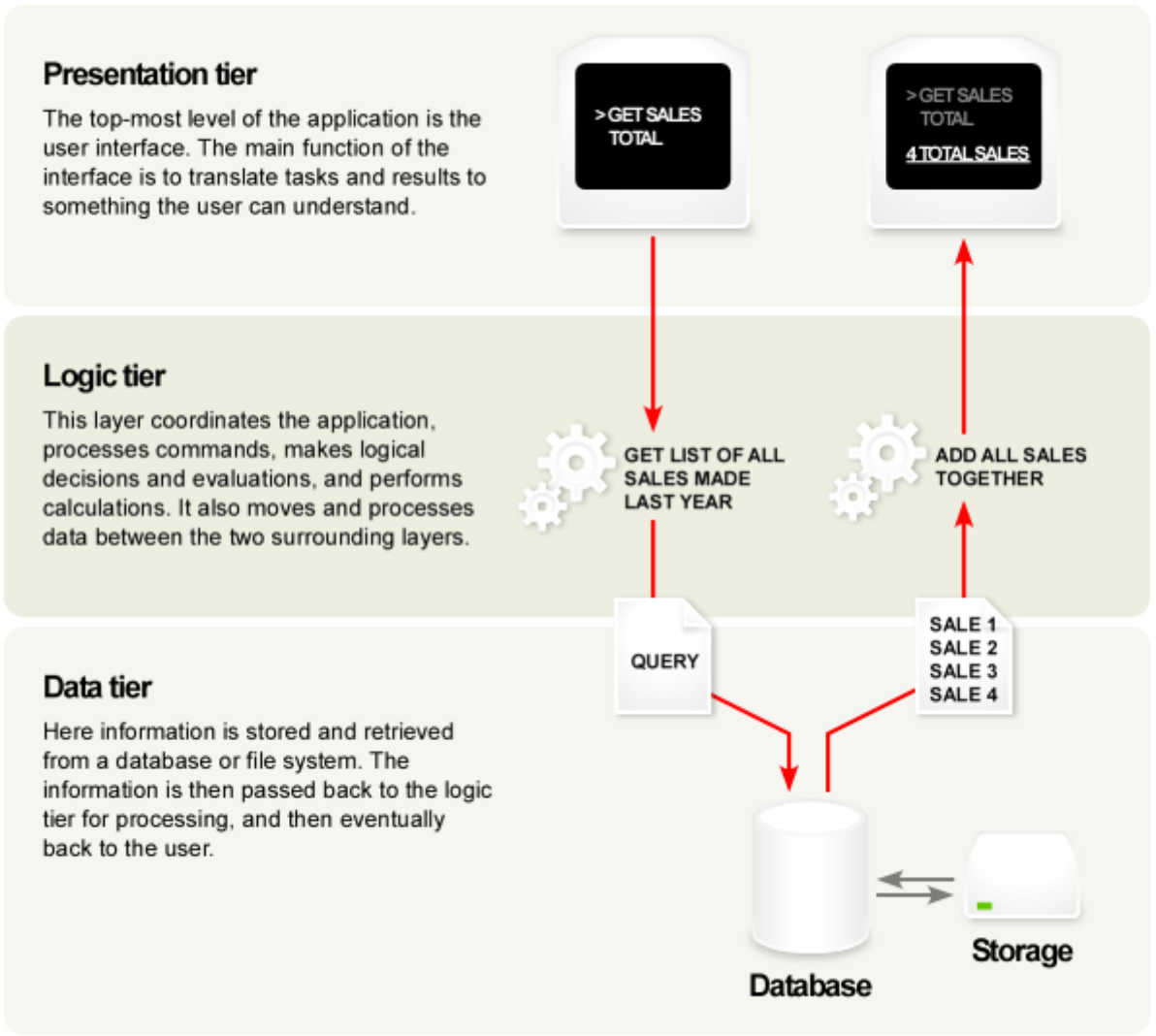
- O objecto é enviado para o cliente e executado localmente (Ex: Applet Java)
- A interacção com o cliente é feita localmente com o objecto
  - Melhora o desempenho e interactividade
  - Incrementa dinamicamente a funcionalidade do cliente
- O objecto pode interagir com o servidor para prestar serviços adicionais
- Segurança:
  - Necessidade de regras de carregamento e acesso do código móvel a recursos locais

Arquitecturas Multi-Nível (*Multitier)*



- A partição das funcionalidades entre cliente e servidor pode ser feita a vários níveis:
  - Terminais alfanuméricos clássicos
  - Terminais X – cliente corre OS e um servidor X
  - *Network Computers* – cliente corre OS mas não tem disco
  - Browser (*Thin Client)* – cliente só executa interface utilizador
  - Aplicações empresariais interactivas têm geralmente 3 camadas

Three-Tier Architecture

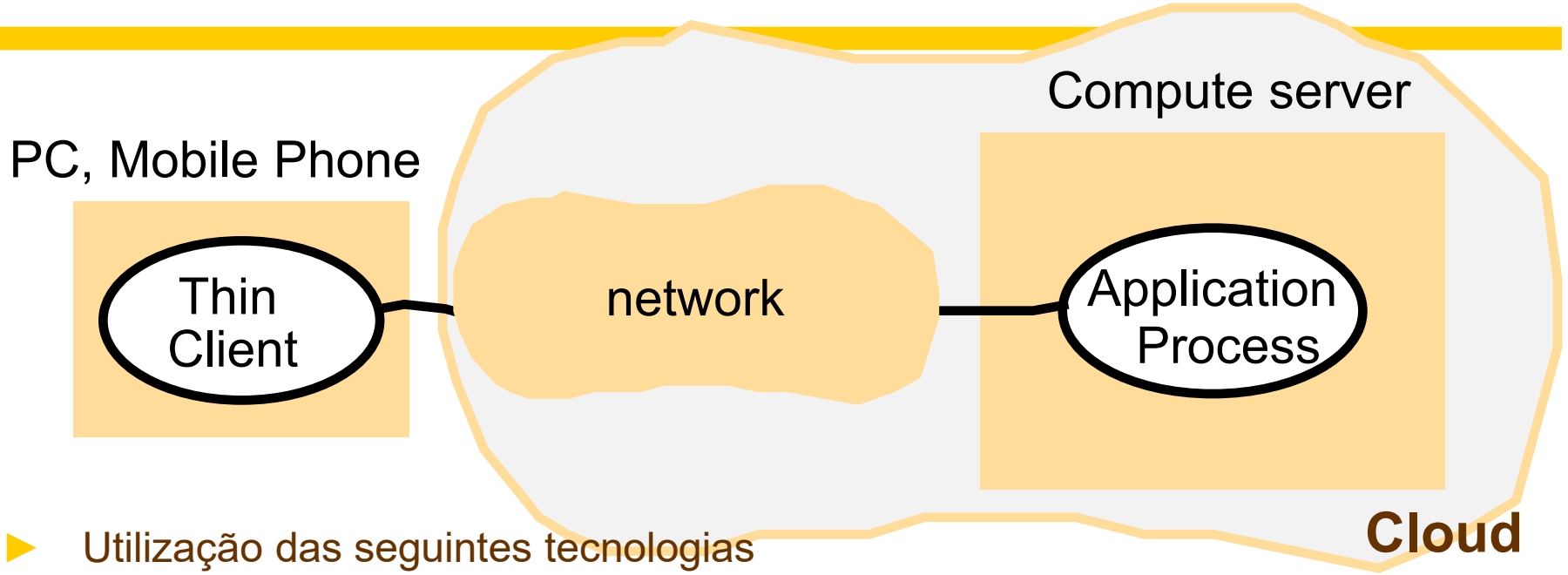


Apresentação

Lógica Aplicacional

Repositório de Dados

Cloud Computing



- Utilização das seguintes tecnologias
  - Computador standalone com sistema mínimo e browser
  - Virtualização de Sistemas (*hypervisor*)
  - Virtualização do Desktop
  - Gestão de Identidade Distribuída
- Modelos de utilização
  - IaaS (Infrastructure as a Service)
  - PaaS (Platform as a Service)
  - SaaS (Software as a Service)

Perguntas a que devo ser capaz de responder

- Em que situações faz sentido utilizar uma arquitetura peer-to-peer ? E em que situações não faz sentido utilizá-la ?
- Que exemplos práticos de código móvel conhece ?
- Quais são os componentes de uma arquitetura de 3 camadas ?
- O que muda nestas arquiteturas com a computação na Cloud ?