



Sistemas Distribuídos

Modelos e arquiteturas



Modelos fundamentais

Modelos fundamentais

- Explicitam quais são as entidades e características essenciais de um sistema
- Permitem-nos:
 - Generalizar o que é possível e impossível resolver nesse modelo (através de provas matemáticas)
 - Desenhar soluções mais facilmente, pois não pensamos nos detalhes de hardware, etc.
 - Provar matematicamente propriedades das nossas soluções
 - Fiabilidade, desempenho, escalabilidade, segurança
 - Determinar facilmente se determinada solução funciona num sistema em particular
 - Basta verificar se os pressupostos do modelo usado para a solução se verificam no sistema em particular

Modelos fundamentais

- Logo, antes de desenhar qualquer solução, é muito boa prática definir os modelos fundamentais!
- Três modelos fundamentais:
 - Modelo de interacção
 - Modelo de faltas
 - Modelo de segurança

Modelo de Interacção

Mais à frente no semestre, analisaremos modelos de interacção em maior detalhe

- Pressupostos sobre o canal de comunicação?
 - Latência, que inclui:
 - Tempo de espera até ter acesso à rede +
 - Tempo de transmissão da mensagem pela rede +
 - Tempo de processamento gasto em processamento local para enviar e receber a mensagem
 - Largura de banda
 - Quantidade de informação que pode ser transmitida simultaneamente pela rede
 - *Jitter*
 - Que variação no tempo de entrega de uma mensagem é possível?
 - Canal assegura ordem de mensagens?
 - Mensagem pode chegar repetida?
- E sobre os relógios locais?
 - Taxa com que cada relógio local se desvia do tempo absoluto



Modelo de Falhas

- Que componentes podem falhar?
- De que forma podem falhar?
- Por enquanto, assumiremos modelo simples:
 - Processos podem falhar silenciosamente
 - Mensagens podem perder-se na rede



Mais à frente no semestre, analisaremos outros modelos de falhas em maior detalhe

Modelo de Segurança

- Que ameaças existem sobre o sistema?
- Que ataques são possíveis?
- Por enquanto, assumiremos que não existem quaisquer ameaças sobre o sistema

Mais à frente no semestre, analisaremos modelos de segurança mais realistas



Modelos arquiteturais

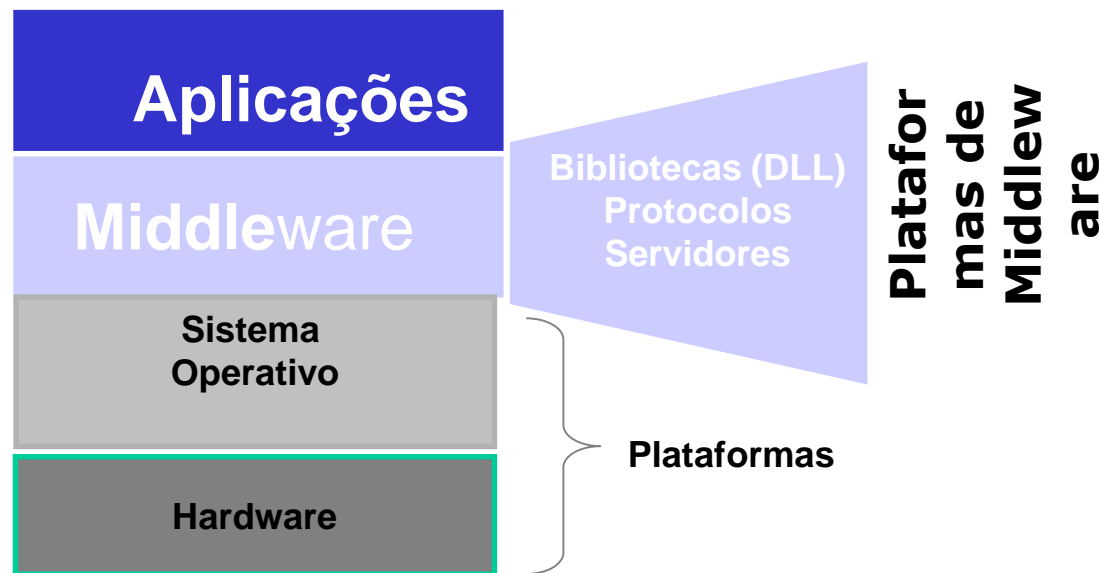
Quem são as entidades que comunicam através da rede num sistema distribuído?

- Processos ou tarefas
 - Nós
 - Objectos
 - Web Services
 - Componentes
- Por omissão, assumiremos sistema distribuído de **processos**
- Em alguns sistemas primitivos não existe a abstracção de processo ou tarefa
 - Exemplo: redes de sensores
 - Exemplo: objecto Java invoca método de outro objecto remoto
 - Veremos mais adiante na cadeira
 - (Fora do âmbito da cadeira)



Como comunicam estas entidades?

Nova camadas de software: o Middleware



Os Sistemas Distribuídos são suportados por diversas componentes frequentemente designadas por plataformas de Middleware

Comunicação directa

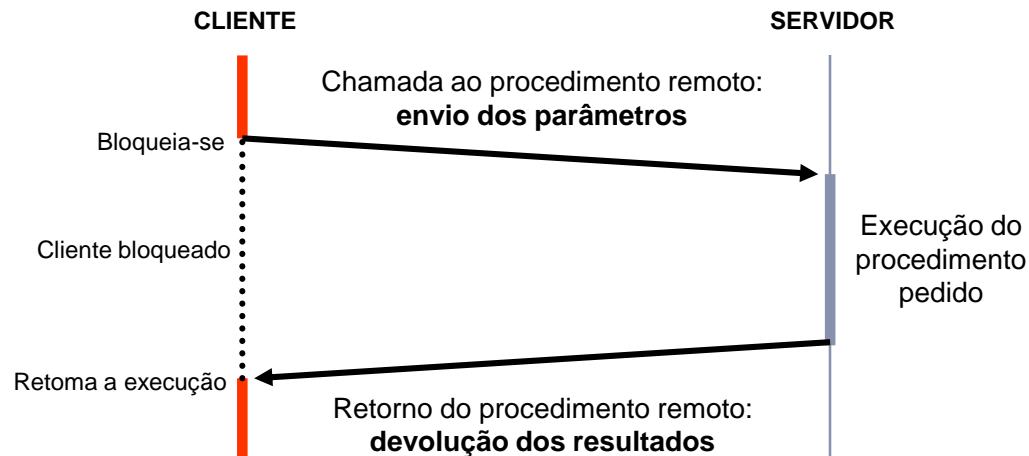
- Interface de comunicação entre-processos
- Invocação remota
 - Protocolos de pedido-resposta
 - Exemplo: HTTP

Estudaremos
ambos em breve

- Chamada remota de procedimentos
 - Programador define conjunto de procedimentos que servidor oferece
 - Cliente pode invocar esses procedimentos como se tratassem de chamadas locais
- Invocação remota de métodos
 - Semelhante a chamada remota de procedimentos, mas no mundo OO

Comunicação directa

Exemplo: chamada remota de procedimentos



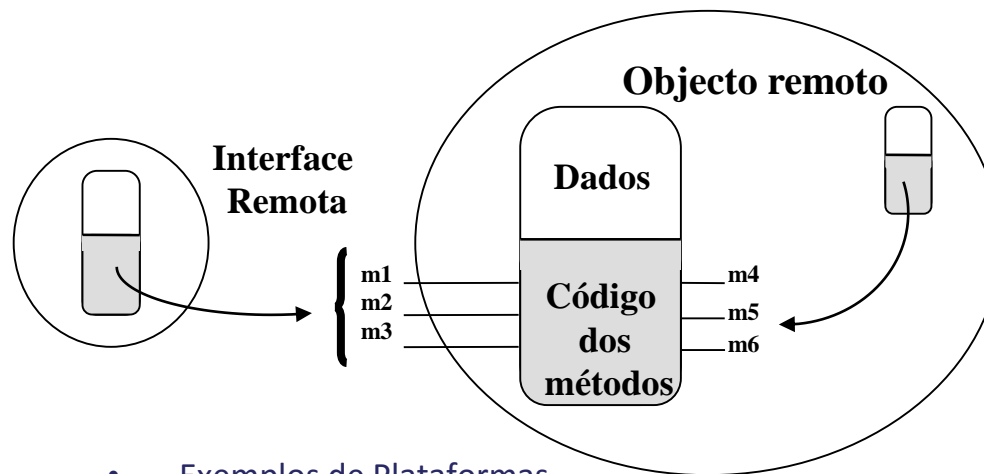
Comunicação directa

Exemplo: invocação remota de métodos

- As potencialidades da noção de objecto tornaram-na atractiva para descrever diversos conceitos em Eng. Informática
 - dando origem a uma tendência de evolução que se designa por OO de Object Oriented
- Diferenças entre a aproximação baseada em objectos e uma arquitectura cliente-servidor:
 - No RPC invocam-se funções, os dados são entidades separadas
 - Num sistema de objectos invoca-se uma função num determinado objecto que, como contém o seu próprio estado, torna indissociável a invocação da operação dos dados a que se aplica

Comunicação directa

Exemplo: invocação remota de métodos



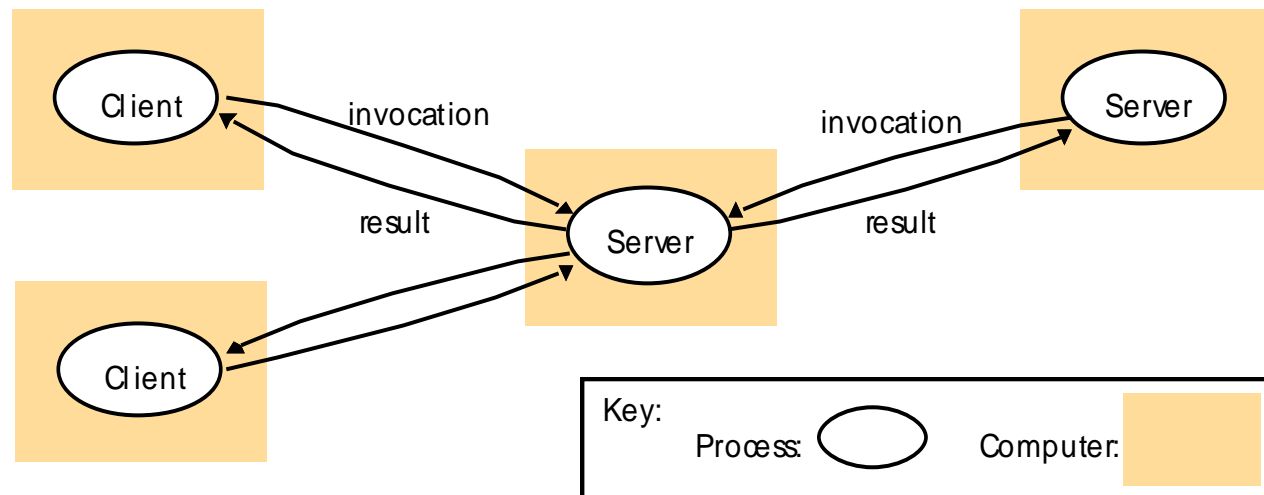
- Exemplos de Plataformas
 - RMI do Java
 - DCOM – Distributed Component Object Model - Microsoft
 - Common Object Request Broker Architecture (CORBA) - Object Management Group (OMG)



Papéis e responsabilidades

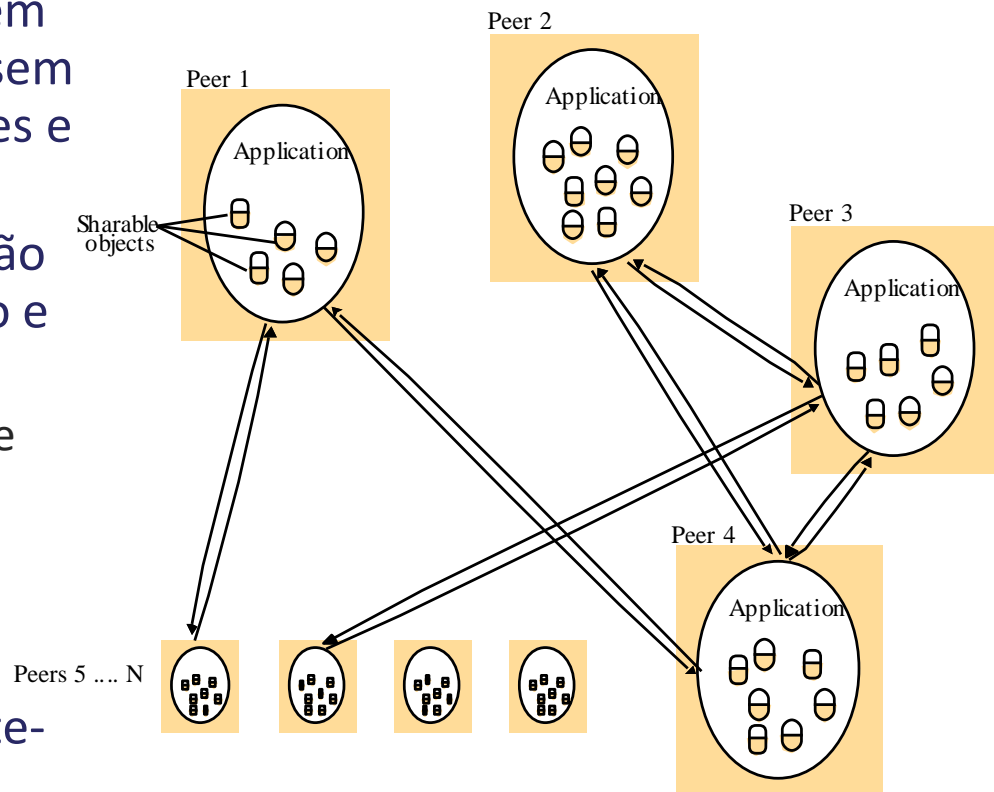
Modelo Cliente-Servidor

- Servidores mantêm recursos e servem pedidos de operações sobre esses recursos
- Servidores podem ser clientes de outros servidores
- Simples e permite distribuir sistemas centralizados muito directamente
- Mas pouco escalável: limitado pela capacidade do servidor e pela rede que o liga aos clientes

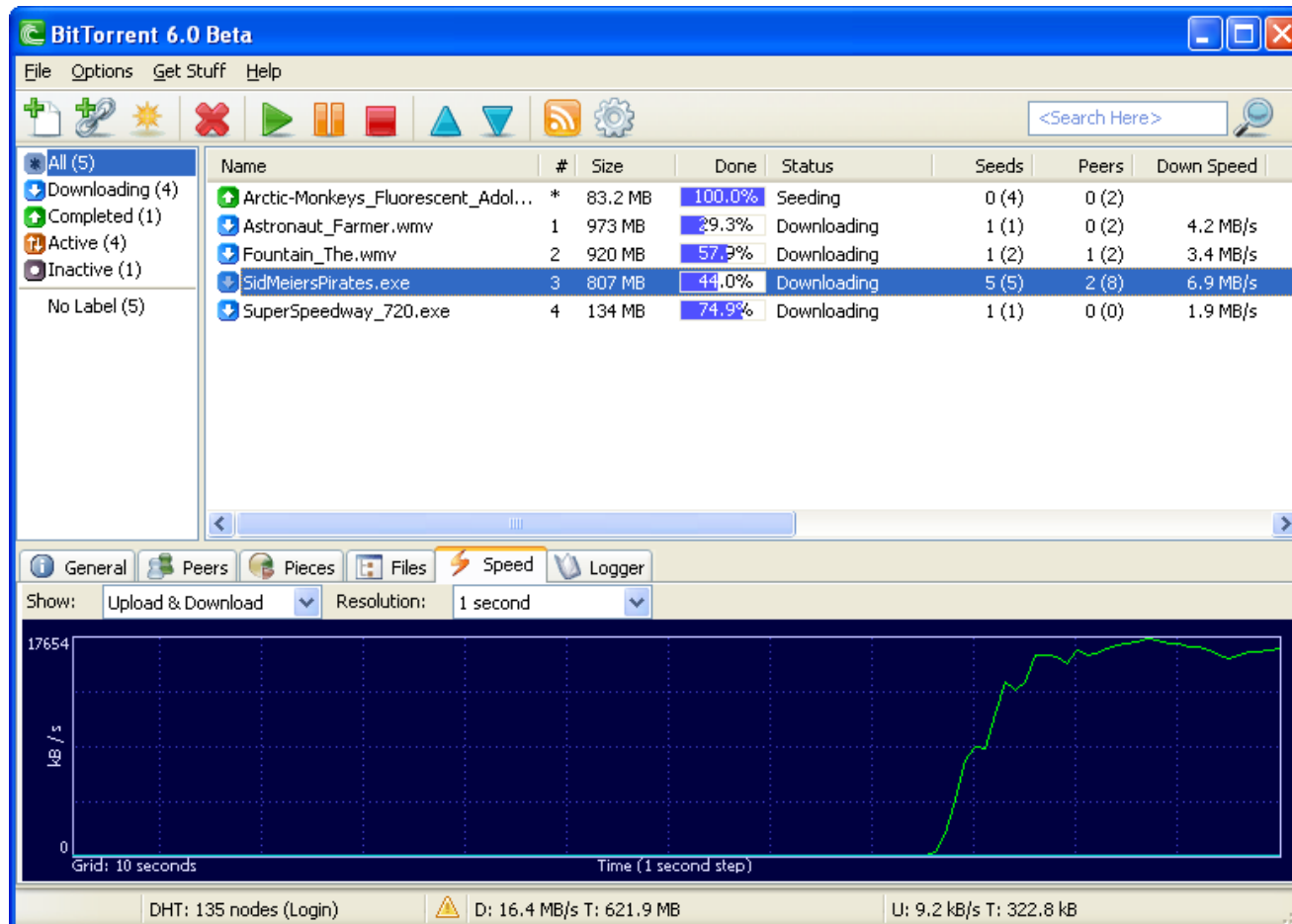


Modelo Entre-Pares (Peer-to-Peer)

- Todos os processos têm papéis semelhantes, sem distinção entre clientes e servidores
- Mais ampla distribuição de carga (computação e rede)
 - Maior escalabilidade
 - Sistema expande-se acrescentando mais pares
- Coordenação mais complicada que cliente-servidor



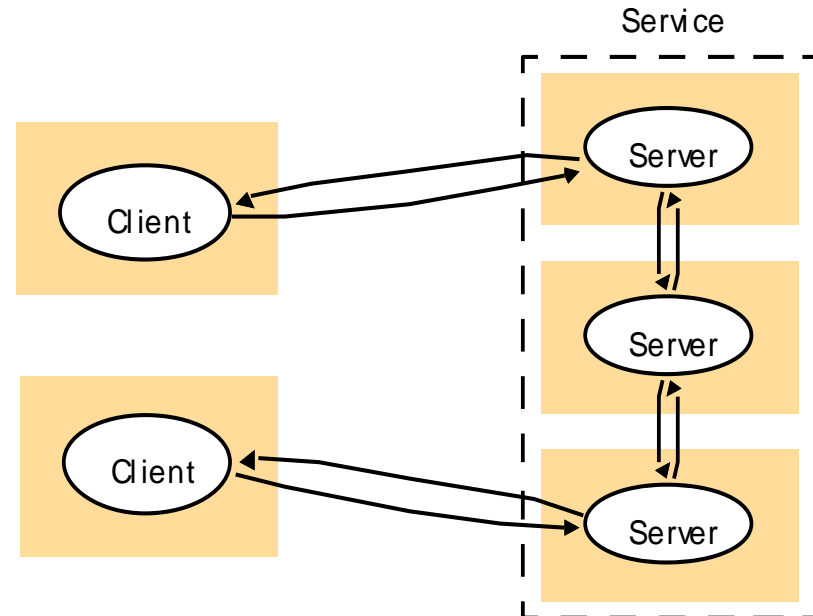
Entre-Pares (Peer-to-Peer)



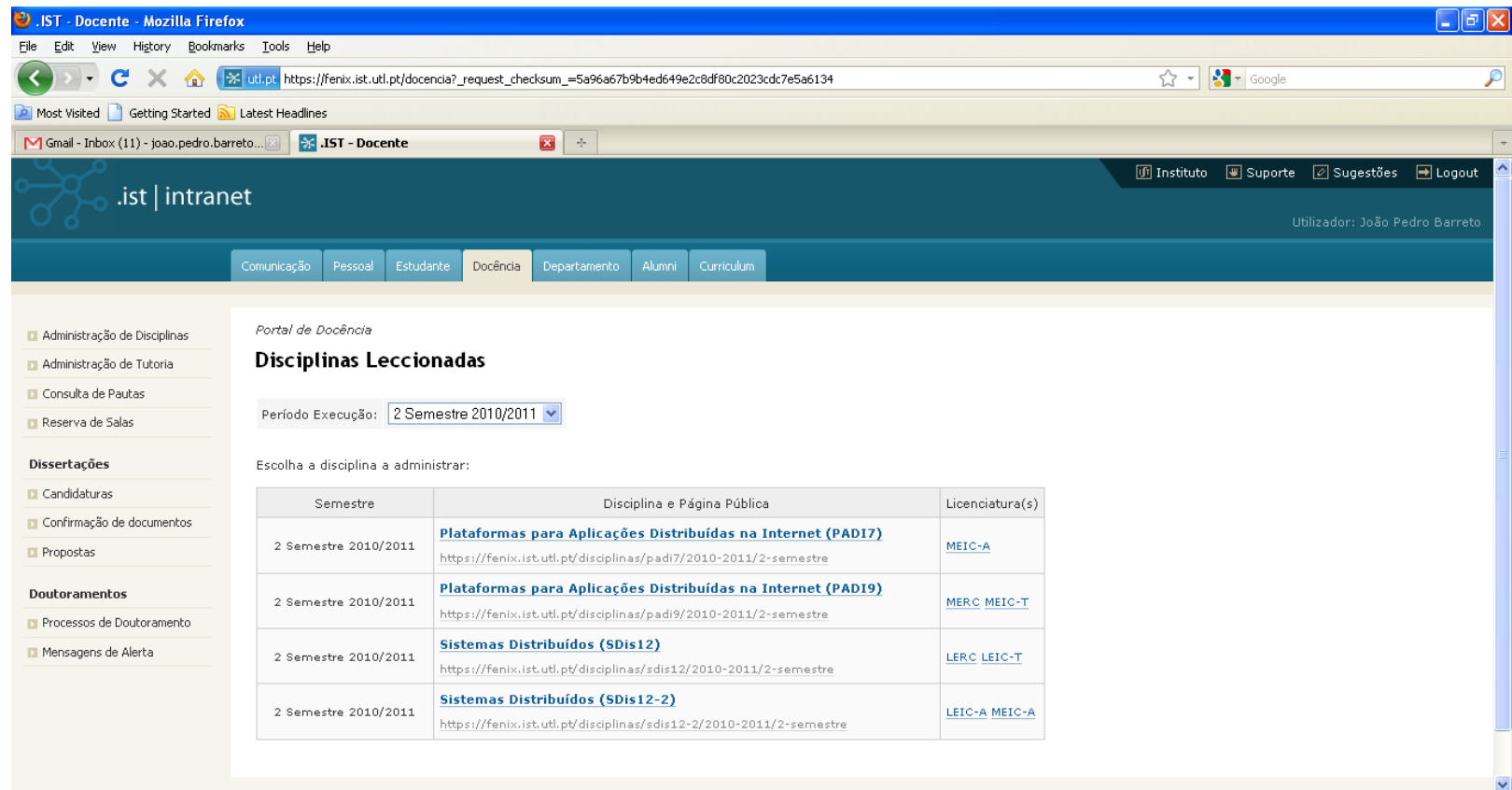
Como mapear objectos e serviços no modelo físico?

Serviço Oferecido por Múltiplos Servidores

- Distribui carga do servidor por múltiplos servidores
- Duas opções:
 - Particionamento: cada servidor mantém uma partição do conjunto de objectos
 - Replicação: todos os servidores mantêm réplicas do mesmo conjunto de objectos



Serviço Oferecido por Múltiplos Servidores



Portal de Docência

Disciplinas Leccionadas

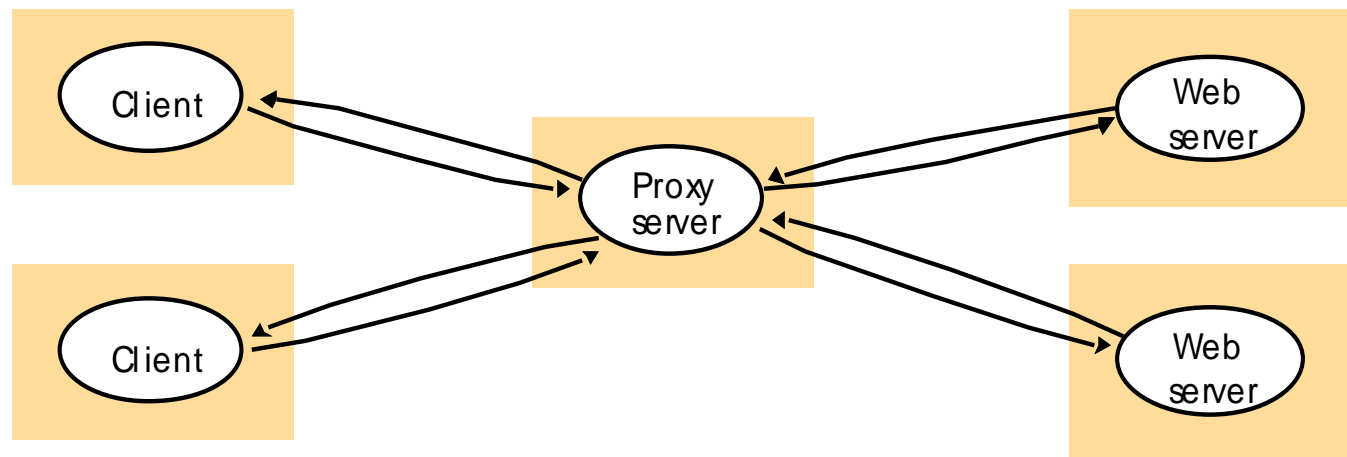
Período Execução: 2 Semestre 2010/2011

Escolha a disciplina a administrar:

Semestre	Disciplina e Página Pública	Licenciatura(s)
2 Semestre 2010/2011	Plataformas para Aplicações Distribuídas na Internet (PAD17) https://fenix.ist.utl.pt/disciplinas/padi7/2010-2011/2-semester	MEIC-A
2 Semestre 2010/2011	Plataformas para Aplicações Distribuídas na Internet (PAD19) https://fenix.ist.utl.pt/disciplinas/padi9/2010-2011/2-semester	MERC MEIC-T
2 Semestre 2010/2011	Sistemas Distribuídos (SDis12) https://fenix.ist.utl.pt/disciplinas/sdis12/2010-2011/2-semester	LERC LEIC-T
2 Semestre 2010/2011	Sistemas Distribuídos (SDis12-2) https://fenix.ist.utl.pt/disciplinas/sdis12-2/2010-2011/2-semester	LEIC-A MEIC-A

Servidores Proxy e Caches

- Mantêm cópias de sub-conjunto dos objectos num computador mais próximo dos clientes
- Melhor desempenho e disponibilidade
- Outros objectivos: por exemplo, acesso ao exterior através de firewall



Servidores Proxy e Caches

Index of active cache

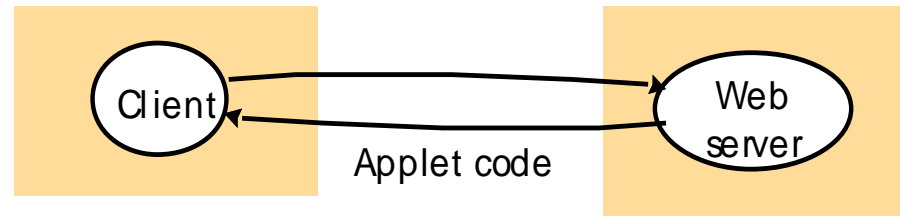
Open URL Find Print Close

Number of cache entries: 355

URL ▲	Date last accessed ▲
http://ad-adex3.flycast.com/server/_ad/WebstatCom/Web	06/21/2000 00:36:31
http://www.altavista.com/css/av.css	05/30/2000 09:09:01
http://cs2000.compuserve.de/	06/06/2000 12:29:43
http://webstat.com/img/t_rside.gif	06/30/2000 11:29:56
http://www.qvc.com/images/b_Jewelry.gif	04/30/2000 01:20:01
http://www.qvc.com/qvc/gif/check.gif	04/30/2000 01:20:00
http://www.kast.cz/kast1.gif	05/30/2000 09:11:18
http://www.webstat.com/img/nada.gif	02/21/2000 21:47:30
http://www.webstat.com/img/box_top.gif	02/13/2000 12:32:17
http://www.webstat.com/images/br.gif	06/30/2000 11:59:43
http://newapps.internet.com/ads/2000/02/zero_468x60.gif	02/09/2000 11:46:38
http://us.yimg.com/images/new2.gif	04/30/2000 00:41:54

Código Móvel (Applets)

a) client request results in the downloading of applet code

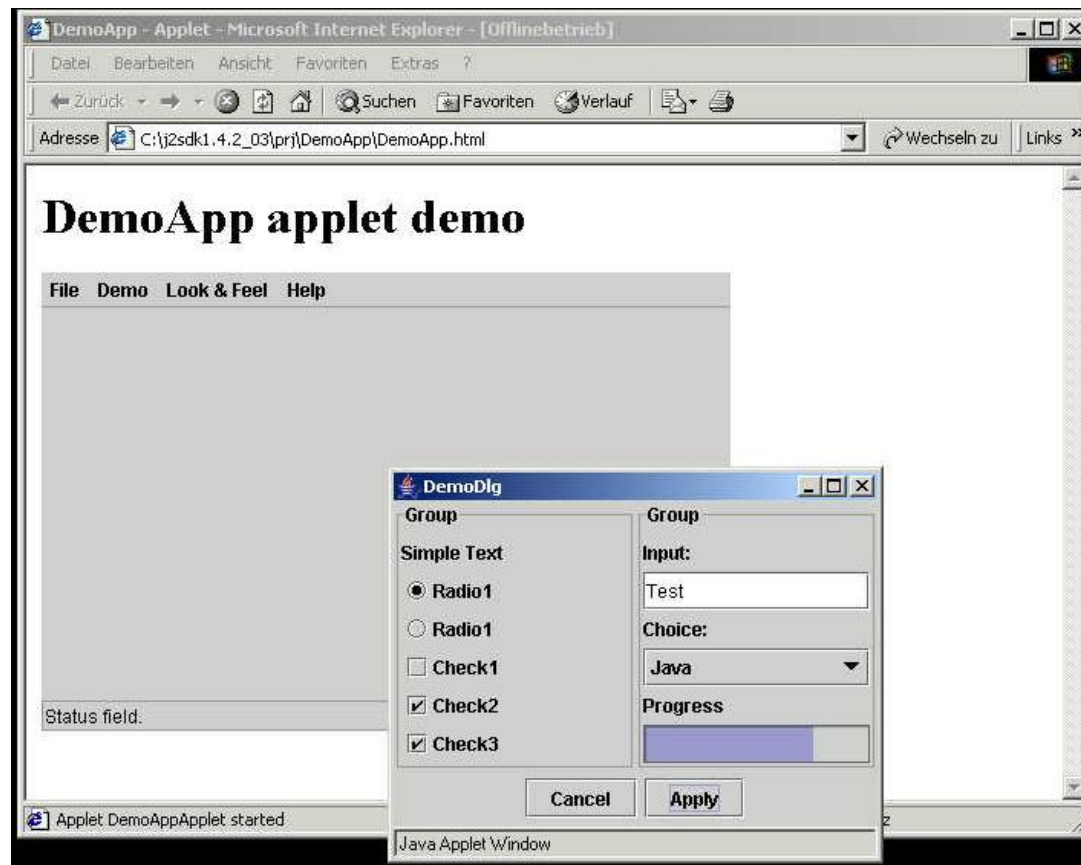


b) client interacts with the applet



- Parte do código do servidor é transferido para o cliente e executado localmente
- Execução não sofre com atrasos de rede e variações de largura de banda
- Bom desempenho de aplicações interactivas

Código Móvel (Applets)



Agentes móveis

- Programa em execução (código+dados) que viaja de um computador para outro na rede
- Executa alguma tarefa em nome de alguém
- Em cada computador, invoca serviços locais (e.g. acesso a BD local para consultar informação local)
- Comparado com a solução de ter um cliente remoto a invocar os mesmos serviços remotamente:
 - Menor custo e tempo de comunicação