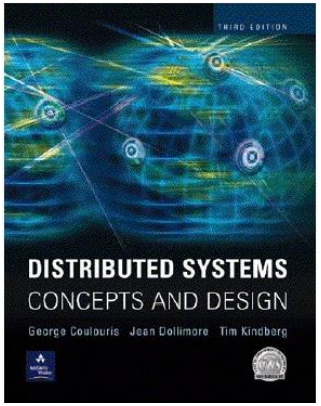
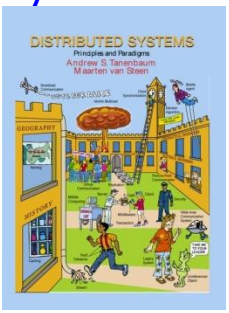
Modelos de Arquiteturas de Sistemas Distribuídos



Most concepts are drawn from Chapter 2

Tradução de material do Dr. Rajkumar Buyya por Rodrigo Malara

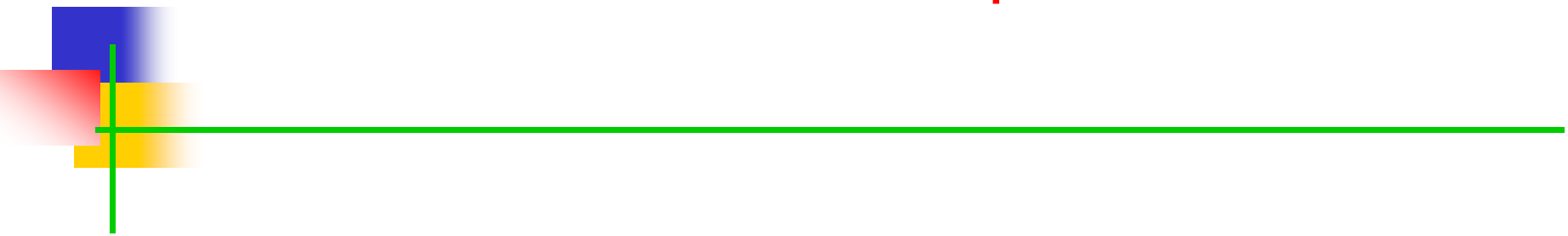
Senior Lecturer and Director of MEDC Course
Grid Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Dept. of Computer Science and Software Engineering
The University of Melbourne, Australia
http://www.cs.mu.oz.au/652



Some ideas from Chapter 1

Agenda

- Introdução
- Modelos de Arquiteturas
  - Camadas de Software
  - Arquiteturas de Sistemas
    - Cliente-Servidor
      - Clientes e um Único Servidor, Múltipols Servidores, Servidores Proxy + Cache, Peer-to-Peer.
    - Modelos Cliente-Servidor alternativos focados em:
      - Código móvel, Agentes móveis, “Network Computers”, “Thin Clients”, dispositivos móveis e conexão espontânea.
    - Desafios de Projeto / Requisitos
- Modelos Fundamentais – descrição formal
  - Modelos de Interação, Falha, e de Segurança.
- Sumário



Modelos de Arquitetura

Camadas de Software

Arquiteturas dos SDs

Interfaces e Objetos

Requisitos de Projeto

Modelos de Arquitetura – Intro [1]

- A arquitetura do sistema documenta sua estrutura
  - Uso de componentes especificados separadamente.
  - Objetivo da arq: Atender requisitos atuais e futuros.
  - Tornar o SD confiável, gerenciável, adaptável e viável financeiramente.

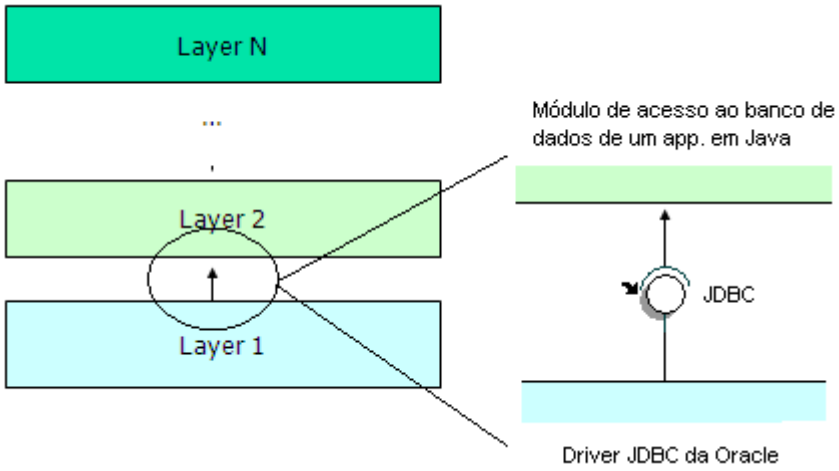
- Por quê definir um Modelo de Arquitetura ?
  - Simplifica e abstrai as funções de componentes individuais.
  - A disposição dos componentes conectados por uma rede de comunicação – definir padrões para a distribuição de dados e processos (workload).
  - Relacionamento entre componentes – ie. Funções e padrões de comunicação entre eles.

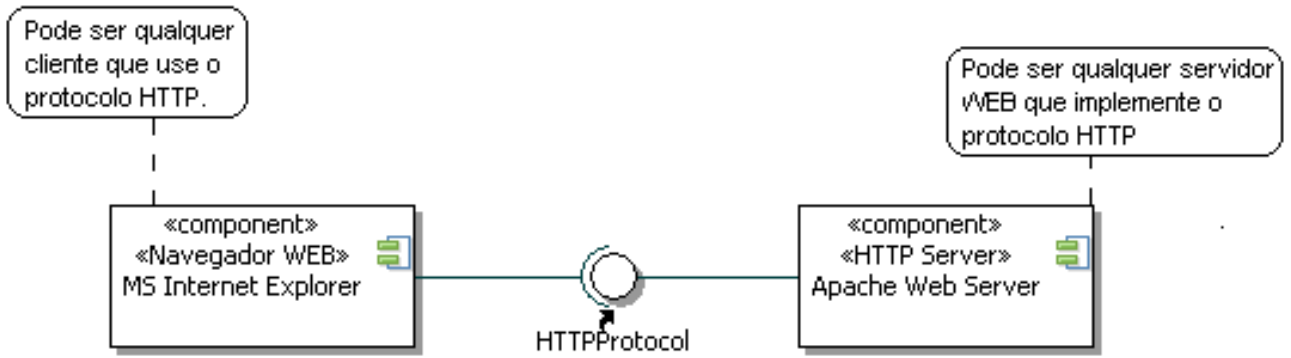
Modelos de Arquitetura – Simplificação Inicial

- Classificar os componentes como:
  - Processo Cliente
  - Processo Servidor
  - Processo Par (Peer)
    - Cooperam e se comunicam de maneira simétrica para realizar alguma tarefa.

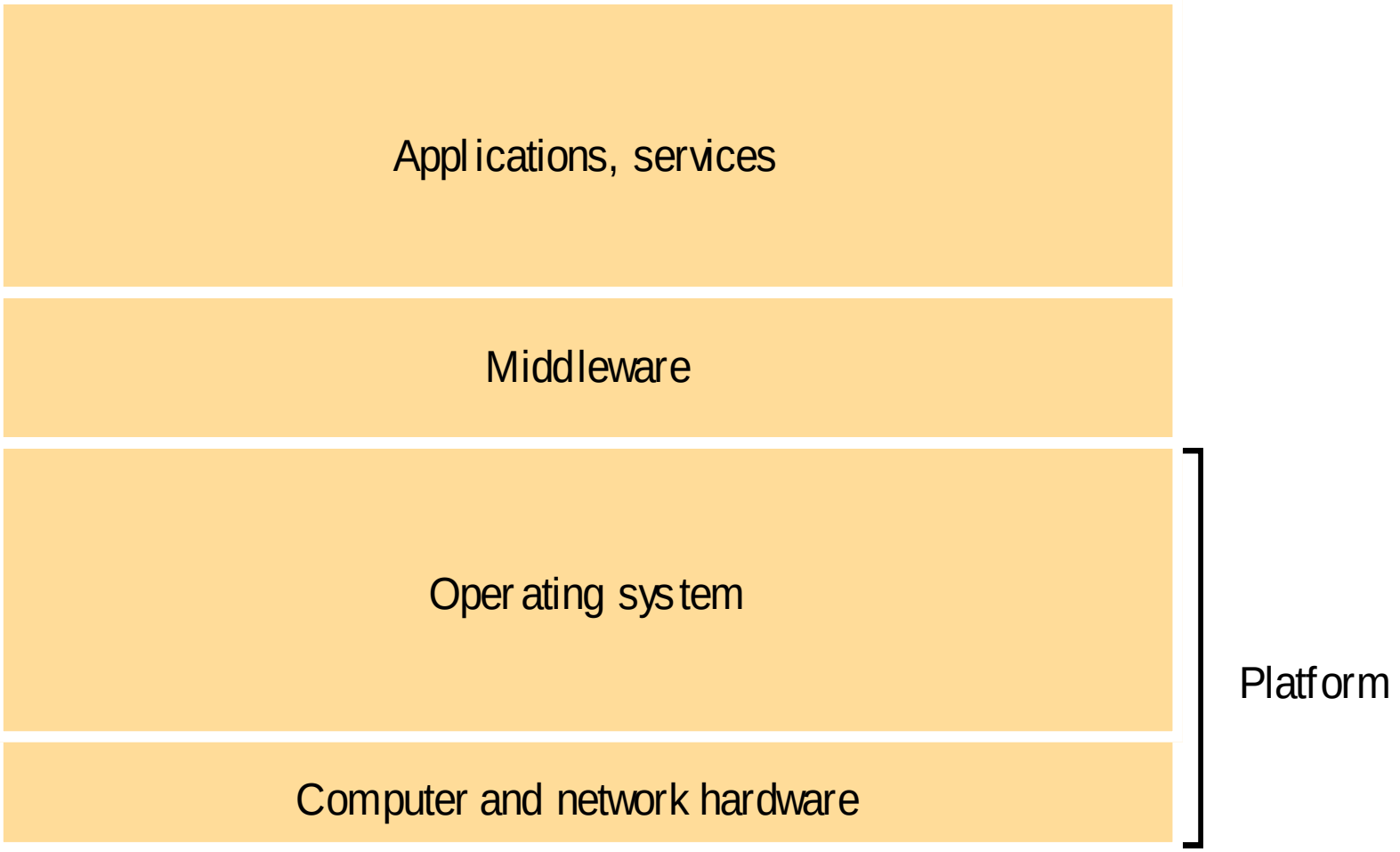
Arquitetura de Software e Camadas

- O termo arquitetura de software:
  - Originalmente se referia à estrutura de um software em *camadas* ou módulos em um único computador.
  - Mais recentemente em termos de *serviços* oferecidos e requeridos entre processos locais ou remotos.
- Quebrar a complexidade de sistemas, projetando-os usando camadas e serviços
  - Camada: um grupo de componentes que realizam uma certa função
  - Serviço: funcionalidade provida para a próxima camada.





Camadas de Serviços de Software e Hardware service layers em SDs



Plataforma

- Camadas mais baixas de hardware e software, geralmente são chamadas de plataforma para Sistemas Distribuídos e Aplicações.
- Estas camadas de baixo nível disponibilizam serviços para as camadas superiores.
- Exemplos
  - Intel x86 + Windows
  - Intel x64 + Linux
  - Intel x64 + Solaris
  - SPARC + SunOS
  - PowerPC + MacOS

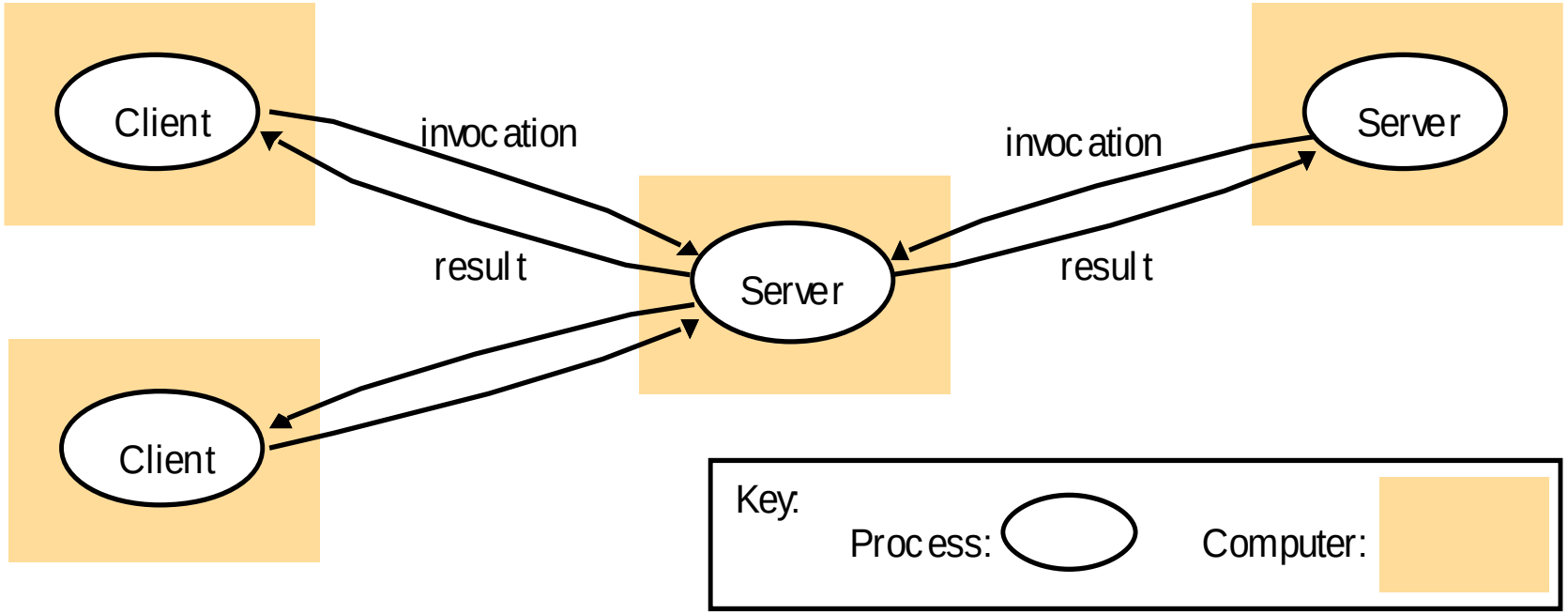
Middleware

- Camada de software cujo propósito é mascarar a heterogeinidade das plataformas e prover um ambiente de desenvolvimento adequado aos desenvolvedores de aplicações.
- RMI/RPC:
  - Sun RPC (Remote Procedure Calls)
  - OMG CORBA (Common Request Broker Architecture)
  - Microsoft D-COM (Distributed Components Object Model)
  - Sun Java RMI (Remote Method Invocation)
  - Webservices SOAP/XML e RESTful/JSON
- Outros Middlewares:
  - Oracle Fusion Middleware
  - JBoss Middleware
  - IBM WebSphere
  - Microsoft .NET
  - Sun JEE (Java Enterprise Edition) – Containers Web e de EJBs (Glassfish)

Arquitetura de SDs
Divisão de Responsabilidades

- O aspecto mais evidente do projeto de SDs é a divisão de responsabilidades entre componentes do sistema e a disposição dos componentes na rede.
- Aspectos mais importantes:
  - Performance
  - Confiabilidade
  - Segurança.

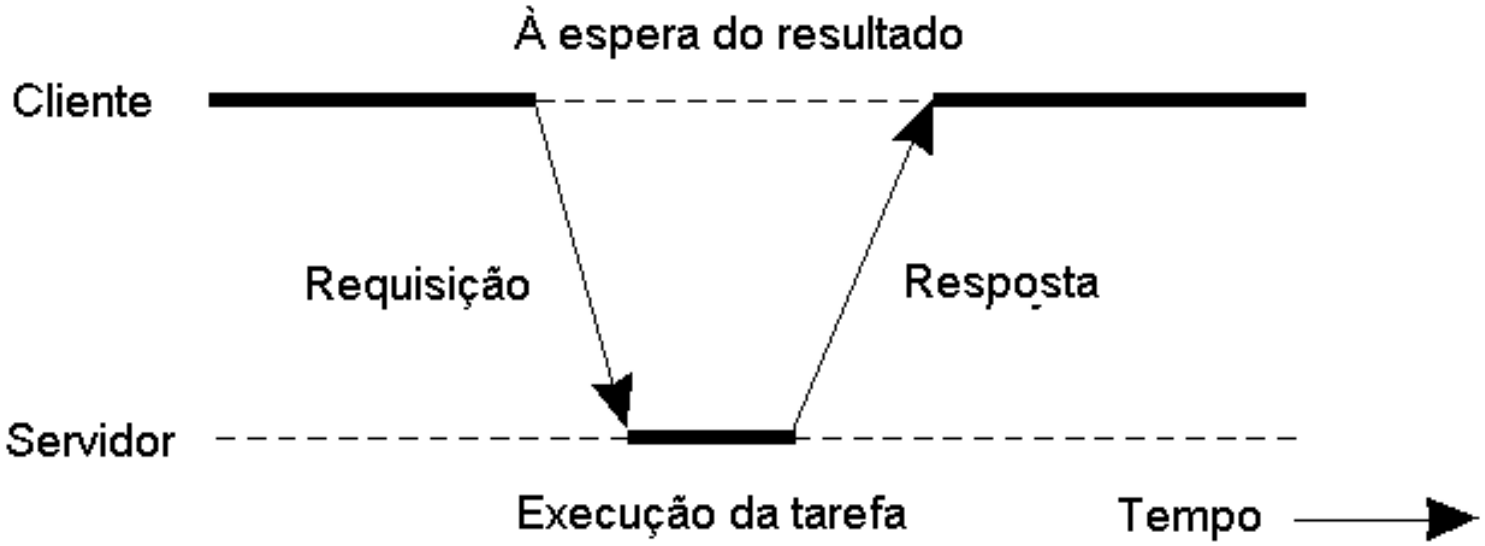
Modelo Básico Cliente Servidor: Clientes invocam servidores individuais



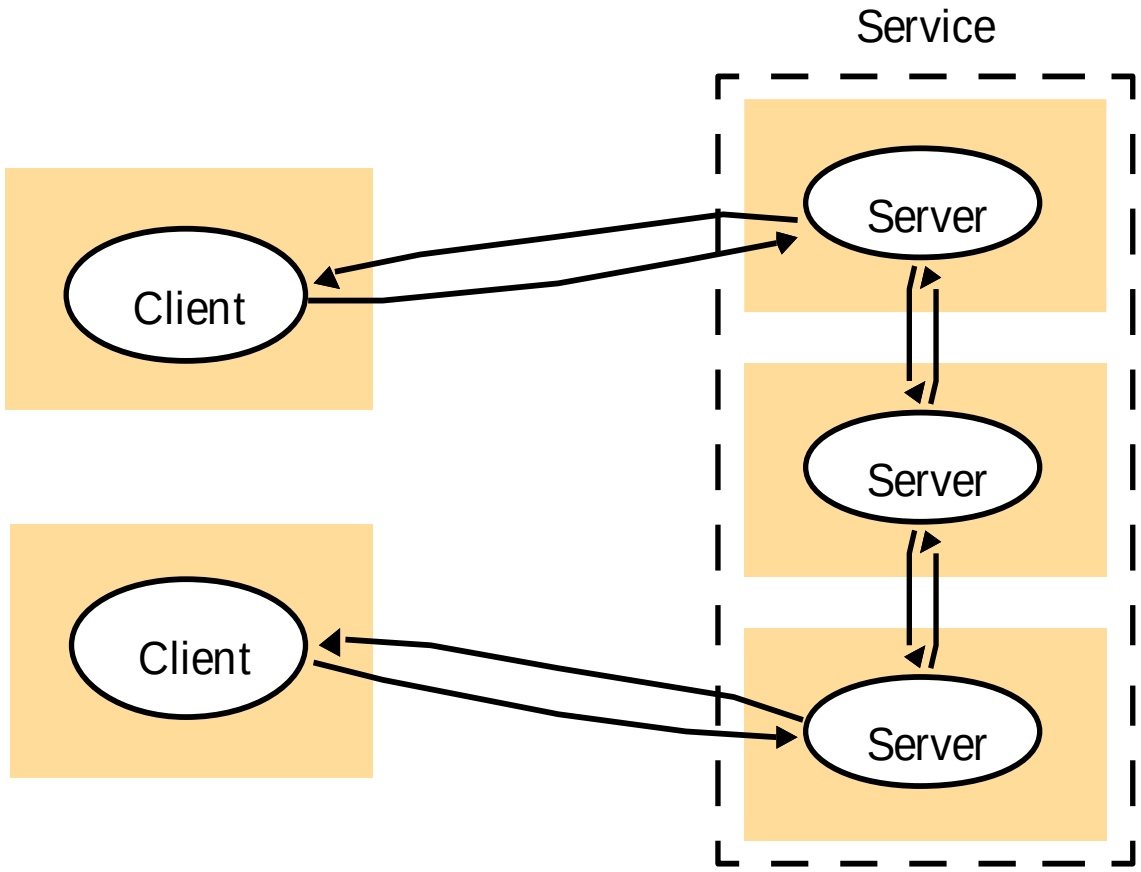
- Um processo cliente interage com um processo servidor individual (local ou remoto) para acessar informações ou recursos. O servidor pode usar serviços de outros servidores .
- Exemplo:
  - Um Servidor Web Server geralmente é cliente de um servidor de Bancos de dados.
  - Navegador → Mecanismo de busca -> Escavadores → Outros servidores Web.

Clientes e Servidores

- Interação padrão entre um cliente e um servidor.

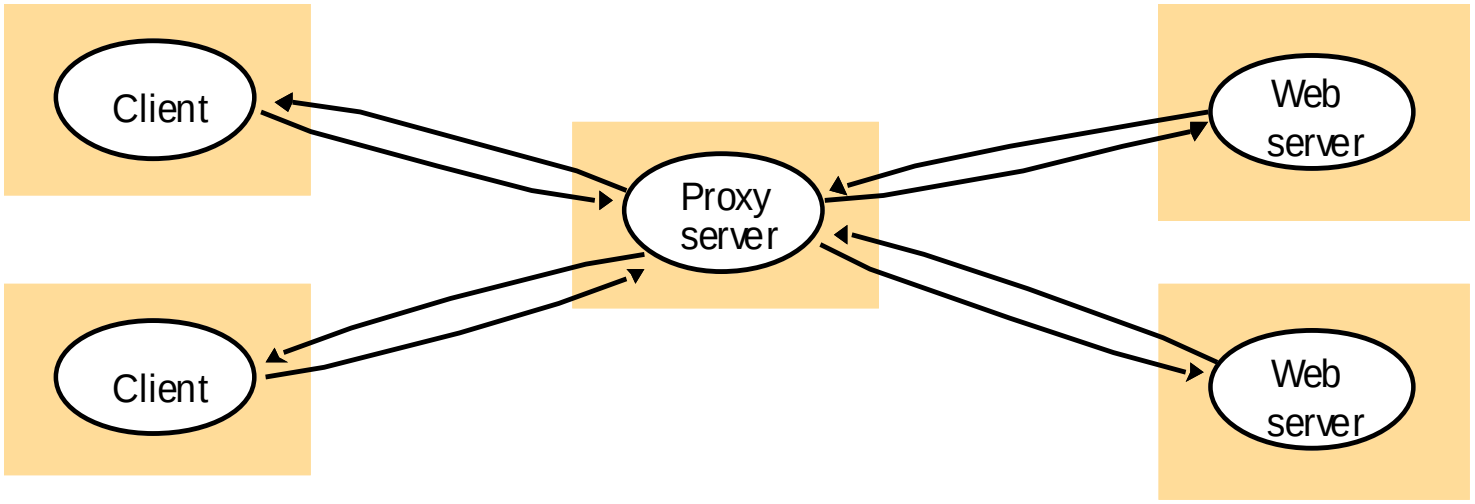


Um Serviço Prestado por Vários Servidores

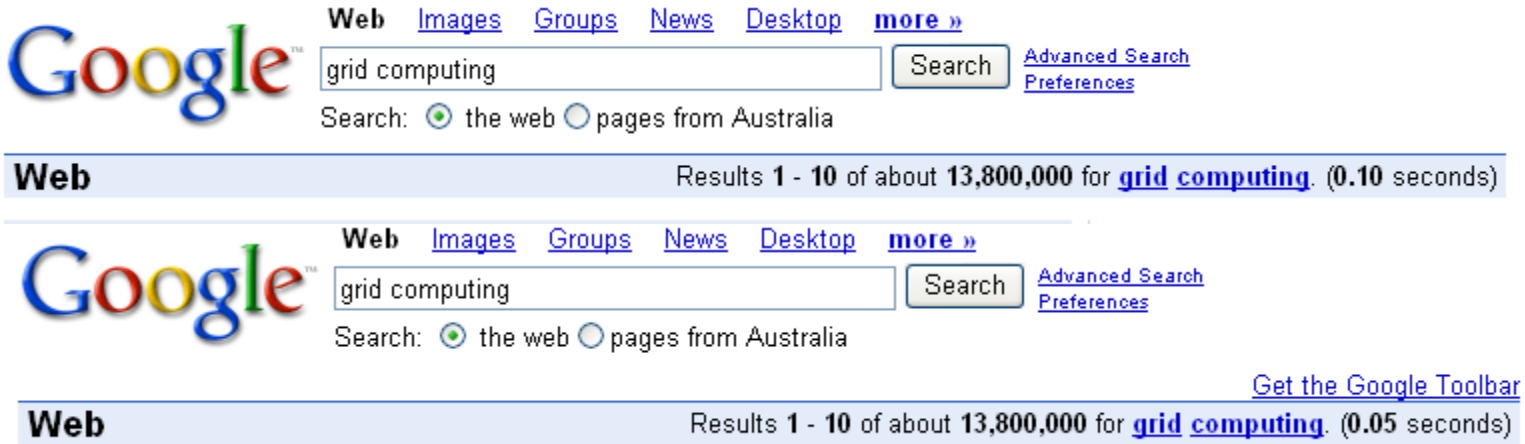


- Serviços podem ser providos por vários processos servidores computadores separados.
- Exemplo: Servidores WEB em Cluster e Aplicativos como o Google, Bancos de Dados Oracle paralelos.

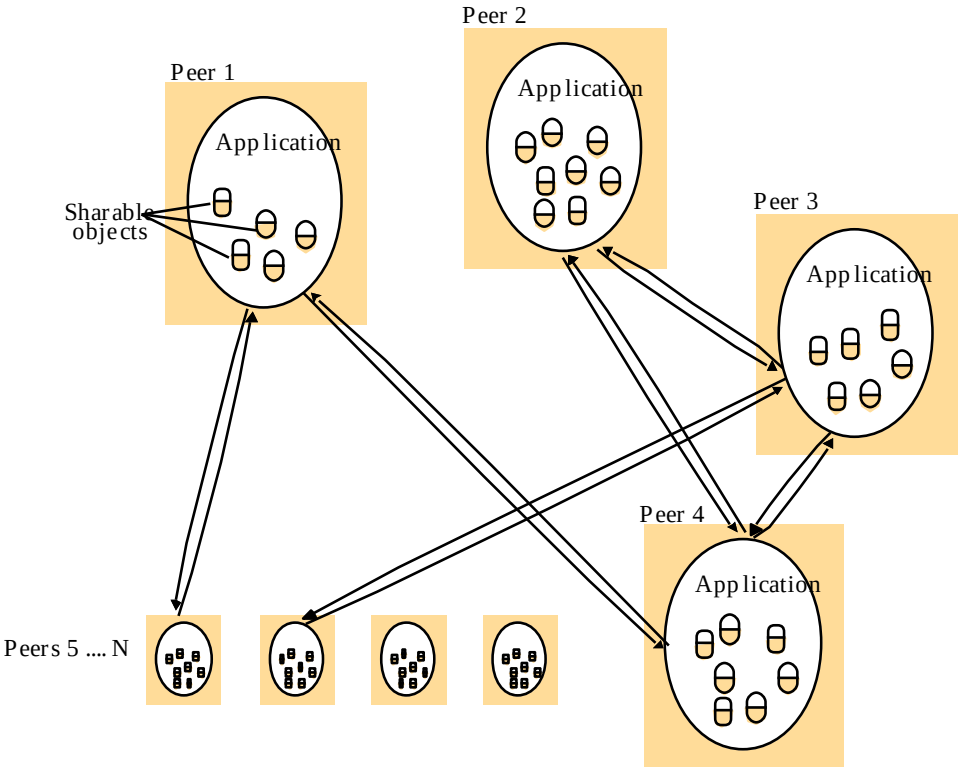
Servidores Proxy (transparência de replicação) e caches: Proxy para Servidor Web



- Cache: contém dados usados recentemente.



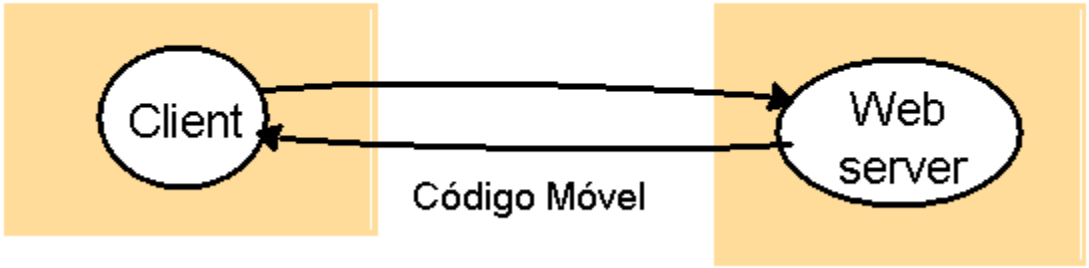
Processos Pares (Peer): Uma aplicação distribuída baseada em processos pares



- Todos os processos executam funções similares, cooperando como pares para realizar atividades distribuídas sem distinção entre clientes e servidores. Ex:
  - Sistemas de compartilhamento de arquivos como BitTorrent, etc.
  - “Quadro negro” distribuído – usuários em diversos computadores vêem e modificam uma mesma imagem interativamente.

Variantes do modelo Cliente Sevidor Código Móvel

a) Cliente solicita download de um Java Applet ou aplicativo Flash



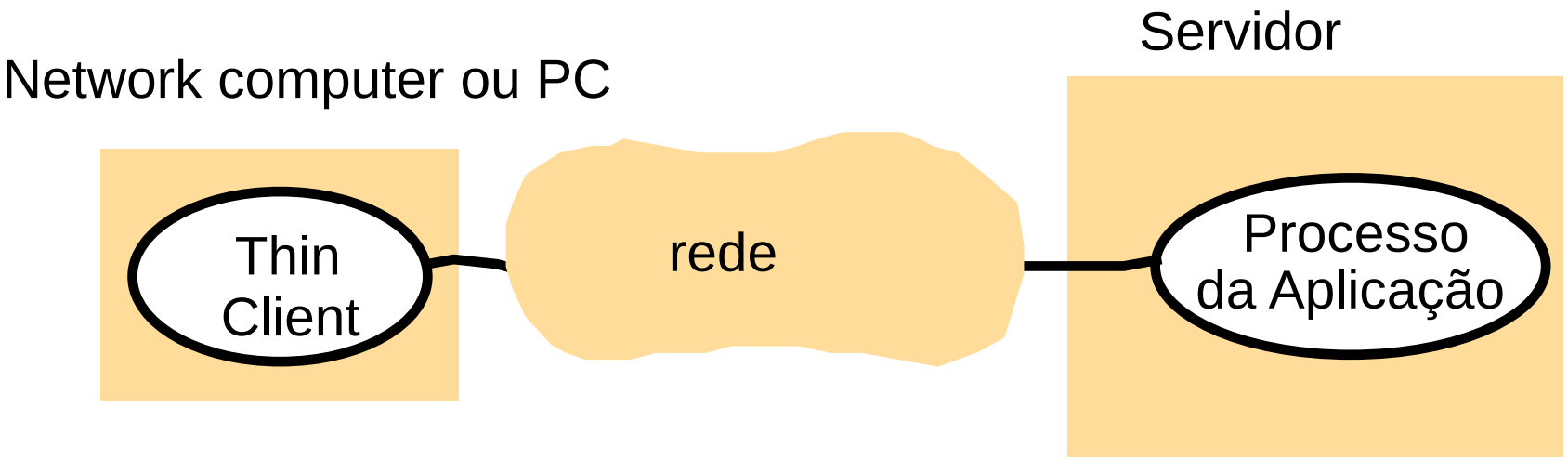
b) Cliente interage com o aplicativo

- Códigos móveis como Applets ou aplicativos em Flash são ameaças de segurança portanto os navegadores dão poucos direitos aos mesmos (ex: sem acesso aos arquivos locais).

Variantes do modelo Cliente Servidor: Agentes Móveis

- Programa em execução (código e dados) são transferidos de um computador para outro pela rede realizando algum processamento solicitado por um processo cliente
- – vantagens: flexibilidade
  - Processamento focado em comunicação: processo pode migrar para próximo do cliente
  - Processamento focado em CPU: processo pode migrar para computador com CPU livre.
  - Processamento focado em funcionalidade: processo percorre computadores que provêm determinada funcionalidade
    - Ex: Instalar atualizações de software
- Ameaça de segurança potencial aos recursos dos computadores visitados.
- EX: DCOS (dcos.io) – Plataforma de containerização Single Image.
  - Cluster se comporta como computador único para containers.

Thin clients and compute servers



- Network computer: faz download do SO e dos aplicativos usando a rede e roda localmente (evita degradação do servidor).
  - Mapeamento de disco a partir da rede. Soluções CITRIX
- Thin clients: IGU baseada em janelas exibidas na máquina do usuário e executa a aplicação em computador remoto.
  - Padrão de janelas Unix: X-11
  - Muito flexível.

Resumindo…

- O uso de CS impacta a arquitetura de SDs:
  - Distribuição de responsabilidades
  - Mecanismos de sincronização entre cliente e servidor
  - Estabelecimento de serviços
    - Tipos pré-determinados de requisições e respostas
- Modelo Básico CS: a responsibilidade é alocada estáticamente.
  - Sistemas de Bancos de Dados são responsáveis por dados e não por páginas web.
- Processos pares: a responsabilidade é alocada dinâmicamente:
  - Compartilhamento de arquivos (mp3). Busca, envio, recepção, etc..

Requisitos de Projeto
Desafios em Sistemas Distribuídos

- Questões de Performance
  - Responsividade
    - Suporte a clientes interativos
      - Usar caching e replicação
  - Rendimento
  - Balanceamento de carga e exatidão
- Qualidade de Serviço (Quality of Service or QoS):
  - Confiabilidade
  - Segurança
  - Performance adaptativa.
- Questões de Dependência:
  - Corretude, segurança, tolerância a falhas
  - Continuidade do funcionamento na presença de falhas em hardware, software, e redes.

Visão Geral da Aula (II)

- Modelos Fundamentais: descrição formal das propriedades comuns em todos os modelos arquiteturais.
- Endereçar: sincronização (temporização), atraso na comunicação, falhas, aspectos de segurança são endereçados por:
  - Modelo de Interação – lida com performance e a dificuldade de se estipular limites de tempo em um SD.
  - Modelo de Falha – especificação dos tipos de falha que podem ocorrer nos processos CS.
  - Modelo de Segurança – discute possíveis ameaças aos processos e aos canais de comunicação.

Modelo de Interação

- Computação ocorre dentro de Processos;
- Processos interagem através da troca de mensagens, resultando em:
  - Comunicação (fluxo de informações)
  - Coordenação (sincronizatção e ordenação de atividades) entre processos.
- Dois fatores significantes afetam a interação de processos em SDs:
  - Performance da comunicação geralmente é uma característica limitante.
  - Não é possível manter uma noção global de tempo.

Modelo de Interação:
Performance do Canal de Comunicação

- Tipos de canais de comunicação. Ex:
  - Streams ou canais de fluxos de dados: Sockets
  - Passagem de mensagens simples em uma rede: HTTP, MPI.
- Performance na Comunicação em Redes de Computadores :
  - Latência:
    - Atraso entre o início da transmissão da mensagem por um processo e o início da recepção da mensagem pelo outro processo.
  - Largura de banda:
    - Quantidade de informação que pode ser transmitida em um determinado período de tempo.
    - Canais de communicação que usem a mesma rede devem compartilhar a largura de banda disponível.
  - Jitter
    - Variação no tempo que se leva para entregar uma série de mensagens. É relevante para sistemas multimídia.

Modelo de Interação:
Relógios de computadores e eventos temporizados

- Cada computador em um SD tem seu relógio interno, que pode ser usado por processos locais para obter a data e hora atual.
- Mas, dois processos rodando em computadores diferentes podem associar diferentes momentos aos seus eventos.
- Se dois processos pegarem o momento ao mesmo tempo, os relógios deles podem retornar momentos diferentes.
- Mesmo que os relógios forem inicializados com a mesma data e hora inicialmente, os relógios internos vão variar e precisarão de correções

Modelo de Interação: Duas variantes

- Em um SD é difícil estabelecer limites de tempo
- SD Síncrono – difícil de se atingir:
  - O tempo que se leva para executar uma instrução precisaria ter limites inferiores e superiores de tempo.
  - Cada mensagem transmitida pela rede deveria ser recebida dentro de um limite máximo de tempo.
  - Cada processo teria um relógio local cujo deslize da data e hora reais seriam conhecidos.
- SD Assincrono: Sem limites quanto à:
  - Velocidade de execução dos processadores
  - Atrasos da transmissão de mensagens
  - Deslizes dos relógios em relação à hora real.

Modelo de Interação: Ordenação de Eventos

- Em vários SDs, é necessário saber quando um evento ocorreu antes, depois ou ao mesmo tempo que outro.
  - Mesmo que não haja relógios sincronizados, a execução de um sistema pode ser descrita em termos de eventos e sua ordenação.
- Ex: Sistema de Compra e Venda de Ações na Bovespa
  - Cada ordem (compra ou venda) recebe um número seqüencial
  - Ele determina qual é a próxima ordem a ser cumprida.

Caixa de Entrada do usuário A

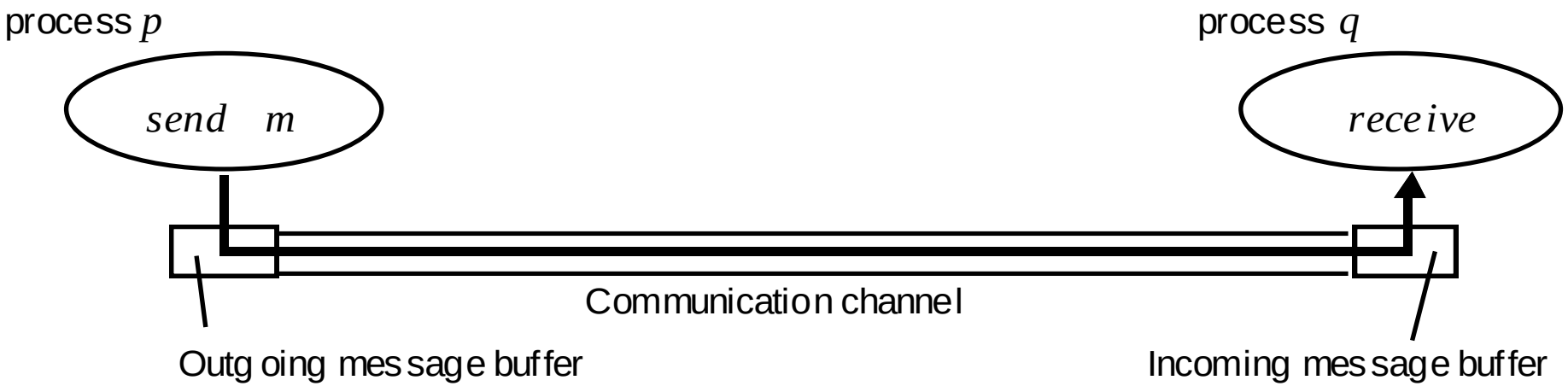
Item	*De*	*Assunto*
23	Z	Re: Reunião
24	X	Reunião
26	Y	Re: Reunião

- Devido ao mecanismo independente de entrega de e-mails, mensagens podem ser entregues fora de ordem.
- Se as mensagens carregarem as horas de envio, é possível ordenar a caixa de entrada do usuário corretamente.

Modelo de Falha

- Em um SD, ambos: processos e canais de comunicação podem falhar.
- Tipos de falhas:
  - Falhas de Omissão
  - Falhas Arbitrárias
  - Falhas de Temporização

Processos e Canais



- Falha de Omissão: um canal de comunicação não transportou a mensagem do processo P para o processo Q.

Falhas de Omissão e Arbitrárias

Classe da falha	*Afeta*	*Descrição*
Parada	Processo	Processo para e permanece parado. Outros processos podem detectar esse estado.
"Morte"	Processo	Processo trava e termina sua execução. Outros processos podem não ter condições de detectar essa falha.
Omissão	Canal	Uma mensagem enviada não é entregue para o receptor
Arbitrário	Processo ou canal	Não há um padrão na falha. Processo ou canal Se comportam de maneira inesperada arbitráriamente, sem uma causa ou periodicidade definida.

Falhas de temporização

Válidas apenas para Sistemas Distribuídos Síncronos

Classe de Falha	*Afeta*	*Descrição*
Relógio	Processo	Relógio do processo excede taxa limite superior de deslizamento do tempo real
Performance	Processo	Processo excede o limite de tempo de execução entre 2 dois passos.
Performance	Canal	A transmissão de uma mensagem demora mais que o limite superior de tempo destinado para isso

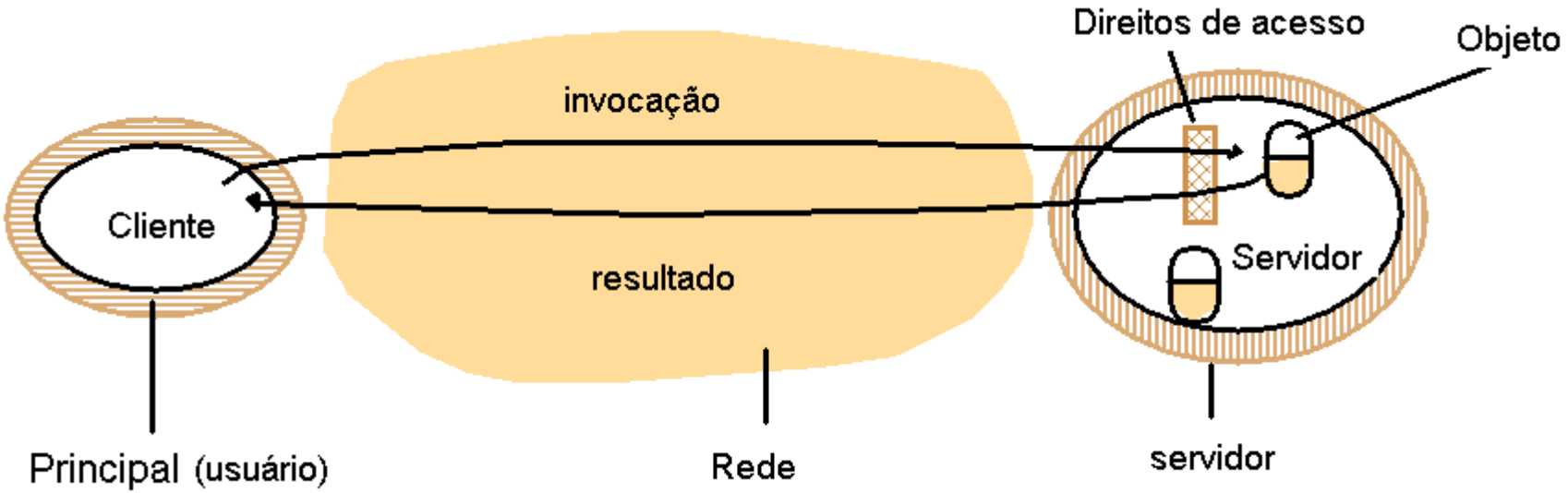
Mascarando Falhas

- É possível construir serviços confiáveis usando componentes que exibem falhas.
  - Ex: múltiplos servidores que armazenam réplicas dos dados permitindo que o serviço continue a ser prestado mesmo que um deles trave.
- O conhecimento dos tipos de falha de um componente pode permitir que as falhas deste componente sejam mascaradas:
  - Checksums são usados para detectar mensagens corrompidas e solicitar o envio das mesmas novamente.

Modelo de Segurança

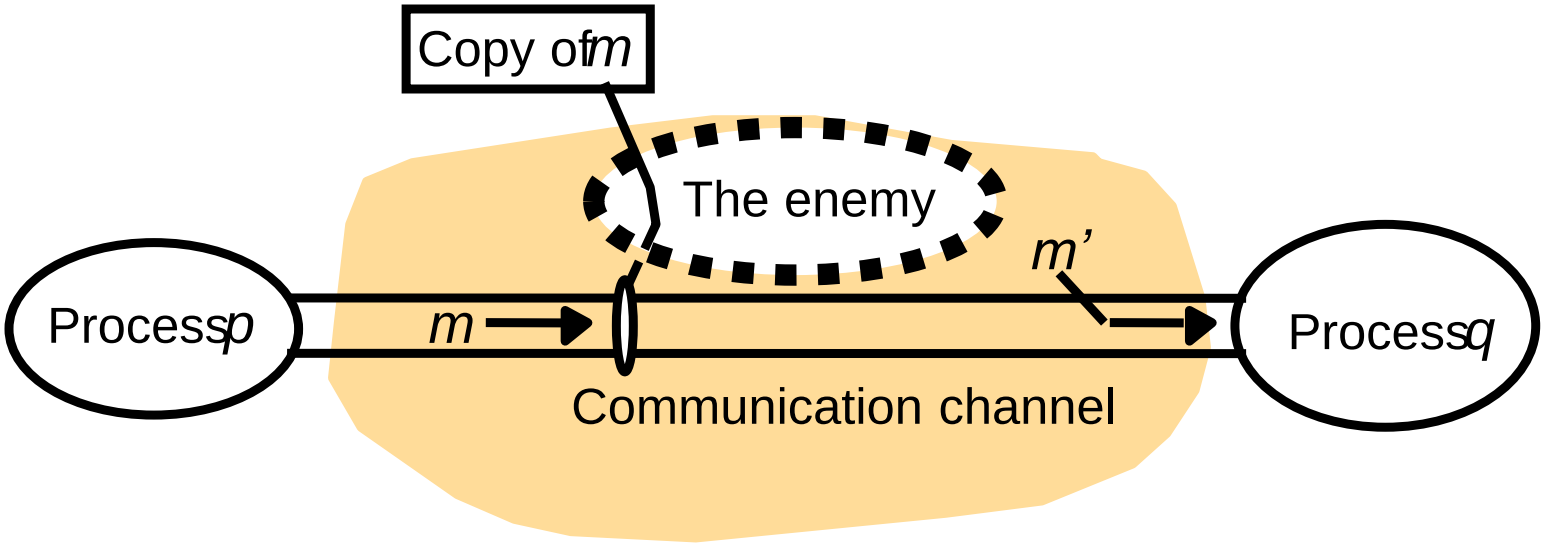
- A segurança de um SD pode ser atingida através da proteção dos processos, dos canais de comunicação e dos recursos ou objetos encapsulados pelos serviços, evitando acesso não-autorizado.

Protegendo objetos: Objectos e “Principals”



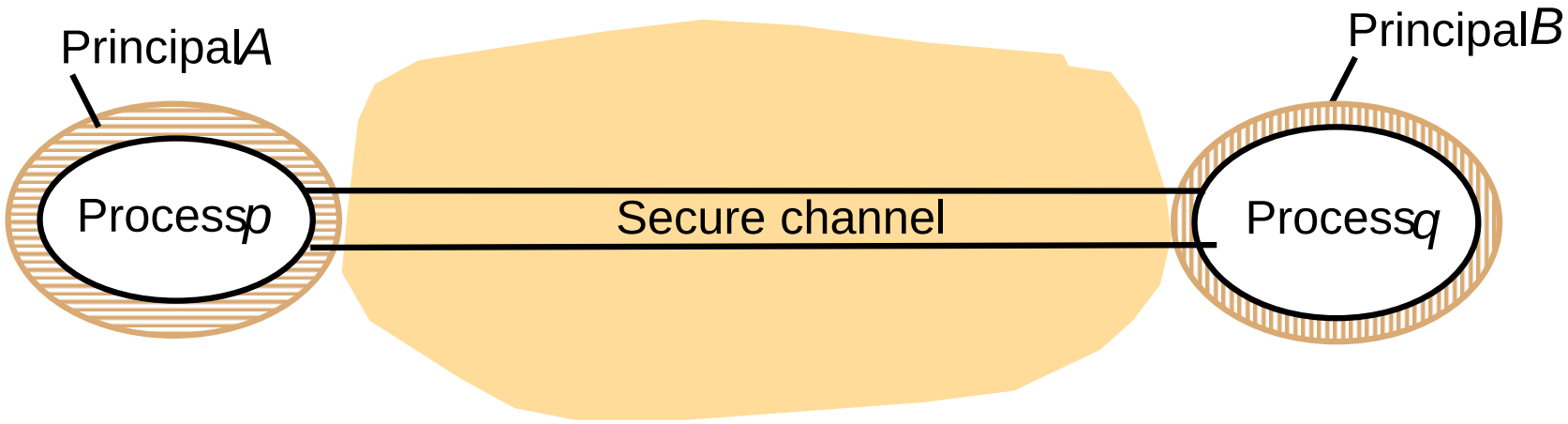
- Usar direitos de acesso que definem quem tem permissão para utilizar um determinado objeto.
- O servidor deve:
  - Autenticar: Verificar a identidade do “principal” (usuário) que solicita cada operação.
  - Autorizar: Verificar se o “principal” tem direitos de realizar a operação solicitada e rejeitar os que não têm permissão.

O Inimigo



- Para modelar ameaças de segurança, definimos que um inimigo é capaz de enviar uma mensagem a qualquer processo ou ler/copiar mensagens trafegando entre processos.
- Ameaças de um potencial inimigo: ameaças a processos, aos canais de comunicações ou negação de serviço (Denial of Service).

Eliminando ameaças de segurança



- Criptografia e autenticação são usadas para proteger os canais de comunicação.
- Cada processo sabe a identidade do “principal” para o qual o processo do outro lado está executando e pode autorizar ou não antes de realizar alguma operação.
- Utilizar Certificados Digitais para detectar se os parceiros são eles mesmos.

Sumário

- A maioria dos SDs são arranjados de acordo com um dos modelos arquiteturais:
  - Cliente-Servidor
    - Clientes e um Único Servidor, Múltipols Servidores, Servidores Proxy + Cache, Peer-to-Peer.
  - Modelos Cliente-Servidor alternativos focados em:
    - Código móvel, Agentes móveis, “Network Computers”, “Thin Clients”, dispositivos móveis e conexão espontânea.
- Modelos Fundamentais – descrição formal
  - Modelos de Interação, Falha, e de Segurança.
- Os conceitos discutidos nesse módulo devem ser estudados mais a fundo e considerados ao se arquitetar sistemas distribuídos