UNIVERSIDADE FEDERAL DO CEARÁ - Campus Quixadá
Cursos: SI, ES, RC, CC e EC
Código: QXD0043
Disciplina: Sistemas Distribuídos

Capítulo 2 – Modelos de Sistemas

Prof. Rafael Braga

Agenda

- Introdução
- Modelos físicos
- Modelos de arquitetura
- Modelos fundamentais
- Resumo

Motivação

- Sistemas utilizados em ambientes do mundo real devem ser projetados para operar corretamente nas mais variadas circunstâncias e diante de muitas possíveis dificuldades e ameaças
  - Variedades de modos de uso (demanda, requisitos de QoS)
  - Variedade de ambientes e plataformas (desempenho, escala)
  - Problemas internos e ameaças externas (falhas, segurança)
- Propriedades e questões de projeto comuns a diferentes tipos de sistemas distribuídos podem ser melhor estudadas e entendidas na forma de **modelos descritivos**
  - Cada modelo oferece uma descrição abstrata (ou seja, simples mas consistente com a realidade) de aspectos relevantes do projeto de um sistema distribuídos

Dificuldade e Desafios para Sistemas Distribuídos

- Modos de uso variando amplamente:
  - As partes componentes de sistemas estão sujeitas a variações em carga de trabalho (*workload*) – páginas web são acessadas muitas e muitas vezes por dia.
  - Algumas partes de um sistema podem ser desconectadas, ou fracamente conectadas por algum tempo
  - Algumas aplicações têm requisitos especiais como alta largura de banda latência, como aplicações multimídias

Dificuldade e Desafios para Sistemas Distribuídos

- Ampla gama de ambientes de sistemas:
  - Um SD deve acomodar HW heterogêneo, SOs e redes. As redes podem diferir amplamente em performance – redes sem fio operam em uma fração da velocidade de redes locais.
  - Sistemas diferindo amplamente em escalas – desde dezenas de computadores a milhões de computadores - devem ser suportados.

Dificuldade e Desafios para Sistemas Distribuídos

- Problemas Internos:
  - Clocks não sincronizados
  - Inconsistências em atualizações de dados
  - Muitos modos de HW e SW envolvendo os componentes individuais de um sistema.
- Ameaças Externas:
  - Ataques sobre a integridade e sigilo dos dados
  - Recusa de serviços (*denial of sevice - DoS*).

Para que Servem os Modelos?

- Capturam a “essência” de um sistema, permitindo que aspectos importantes do seu comportamento possam ser mais facilmente estudados e analisados pelos seus projetistas
  - Quais são os principais elementos do sistema?
  - Como eles se relacionam?
  - Que características afetam o seu comportamento (individual e colaborativo)?
- Ajudam a:
  - Tomar explícitas todas as pré-suposições sobre o comportamento esperado do sistema
  - Fazer generalizações sobre o que deve e não deve acontecer com o sistema, dadas essas pré-suposições

Tipos de modelos

- ***Modelos físicos*** são a maneira mais explicita de descrever um sistema.
  - Capturam a composição de hardware de um sistema
  - Capturam a composição das redes de interconexão
- ***Modelos arquitetônicos*** descrevem a estrutura organizacional dos componentes do sistema
  - Como interagem uns com os outros
  - Como são mapeados para a infraestrutura física (rede) subjacente
- ***Modelos fundamentais*** descrevem problemas e características chaves comuns ao projeto de todos os tipos de sistemas distribuídos
  - Mecanismos de interação e comunicação
  - Tratamento de falhas
  - Segurança

Agenda

- Introdução
- Modelos físicos
- Modelos de arquitetura
- Modelos fundamentais
- Resumo

Modelos Físicos

- Modelo físico básico
  - Definido no capítulo 1
- Gerações
  - Sistemas distribuídos primitivos
    - Anos 1980
    - Geralmente homogêneos
  - Sistemas distribuídos adaptados para a Internet
    - Anos 1990
    - Padrões abertos, middleware, CORBA e Web Services
  - Sistemas distribuídos contemporâneos
    - Computação móvel
    - Computação ubíqua
    - Computação em Nuvem

Agenda

- Introdução
- Modelos físicos
- Modelos de arquitetura
- Modelos fundamentais
- Resumo

Modelos de Arquitetura

- Elementos arquitetônicos
- Padrões arquitetônicos
- Plataformas de middleware disponíveis

Modelo de arquitetura

- Elementos arquitetônicos
  - Entidades em comunicação:
    - Processos e threads
    - Objetos
    - Componentes
    - Serviços Web
  - Paradigmas de comunicação
    - Comunicação entre processos (soquetes e multicast)
    - Invocação remota
    - Comunicação indireta

Modelo de arquitetura

- Comunicação com relação bilateral
  - Comunicação entre processos
  - Invocação remota
  - Protocolos requisição-resposta
  - Chamada de procedimento remoto
  - Invocação de método remoto
- Nesses casos os destinatários conhecem a identidade dos remetentes
- Na maioria dos casos as duas partes devem existir ao mesmo tempo.

Modelo de arquitetura

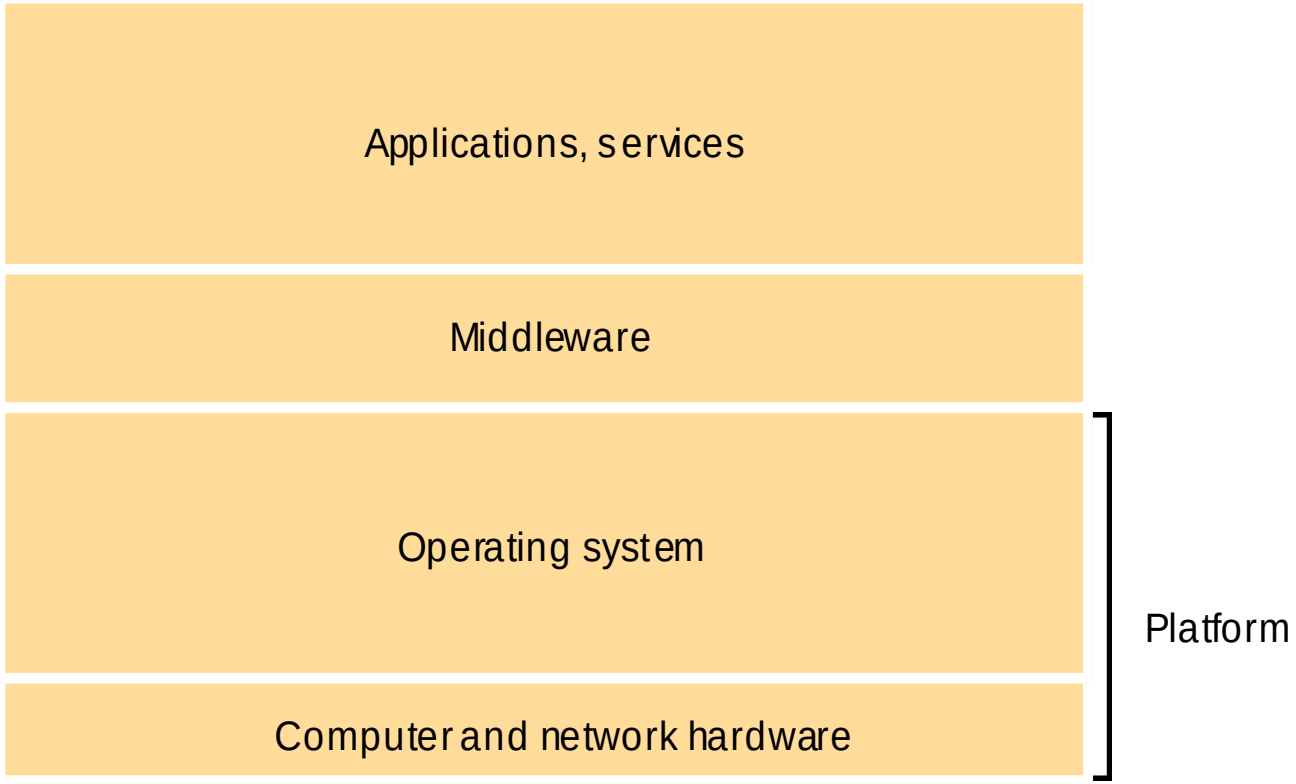
- A comunicação indireta é feita por intermédio de uma terceira entidade, possibilitando um alto grau de desacoplamento
  - Os remetentes não precisam saber para quem estão enviando (desacoplamento espacial)
  - Os remetentes e os destinatários não precisam existir ao mesmo tempo (desacoplamento temporal)

Modelo de arquitetura

- As principais técnicas de comunicação indireta incluem:
  - Comunicação em grupo
  - Sistemas publicar-assinar
  - Filas de mensagem
  - Espaços de tuplas
  - Memória compartilhada distribuída

Modelos arquitetônicos

- Modelos em camadas



Modelos

- Modelos em camadas
  - Plataforma:
    - Camadas de hardware e software
    - Fornecem serviços para as camadas que estão acima
  - Objetivo: levar a interface de programação do sistema a um nível que facilita a comunicação e a coordenação entre os processos
  - Exemplos:
    - Intel x86/Windows
    - Intel x86/Linux
    - PowerPC/MAC OS

Modelo de camadas

- Middleware
  - Camada de software que tem por objetivo mascarar o **heterogeneidade** e fornecer um modelo de programação conveniente para os programas dos aplicativos
  - Composto por um conjunto de processos ou objetos, em um grupo de computadores, que interagem entre si de forma a **implementar a comunicação** e oferecer suporte para **compartilhamento de recursos** a aplicativos distribuídos

Middleware

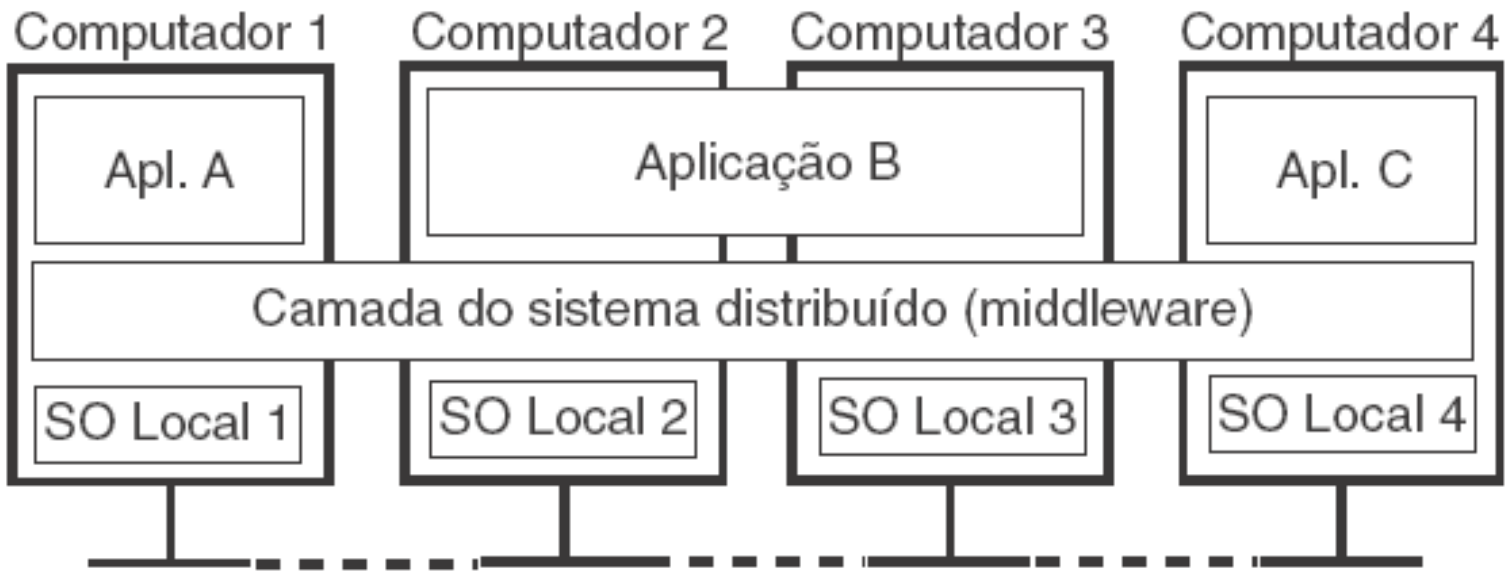


Figura 1.1 *Sistema distribuído organizado como middleware. A camada de middleware se estende por várias máquinas e oferece a mesma interface a cada aplicação.*

Modelo em camadas

- *Middleware*
  - Serviços
    - Invocação a métodos remotos
    - Comunicação entre um grupo de processos
    - Notificação de eventos
    - Replicação
    - Transações
  - Exemplos de *Middlewares* OO
    - CORBA
    - Java RMI
    - Web Services
    - DCOM (Disctributed Component Object Model) da Microsoft

Modelos Arquitetônicos

- Arquitetura de Sistemas
  - Foco na arquitetura – estrutura de alto nível do sistema, descrita em termos de componentes e seus relacionamentos
    - Base para garantir que a estrutura de sistema atenderá sua atual e provável futura demanda em termos de atributo de qualidade como confiabilidade, adaptabilidade, desempenho, gerência e etc.
  - Descrição simplificada e abstrata dos componentes de sistemas:
    - Funcionalidades (ou responsabilidades)
      - Ex: servidor, cliente, peer;
    - Distribuição física (recursos e carga de trabalho)
      - Ex: regras de particionamento e/ou replicação
    - Padrão de interação e comunicação
      - EX: cliente-servidor, ponto-a-ponto

Modelos Arquitetônicos

- Se preocupam com:
  - Confiança
  - Gerenciamento
  - Adaptabilidade
  - Custo
- Consideram:
  - Troca de componentes em uma rede de computadores
  - Inter-relação entre os componentes

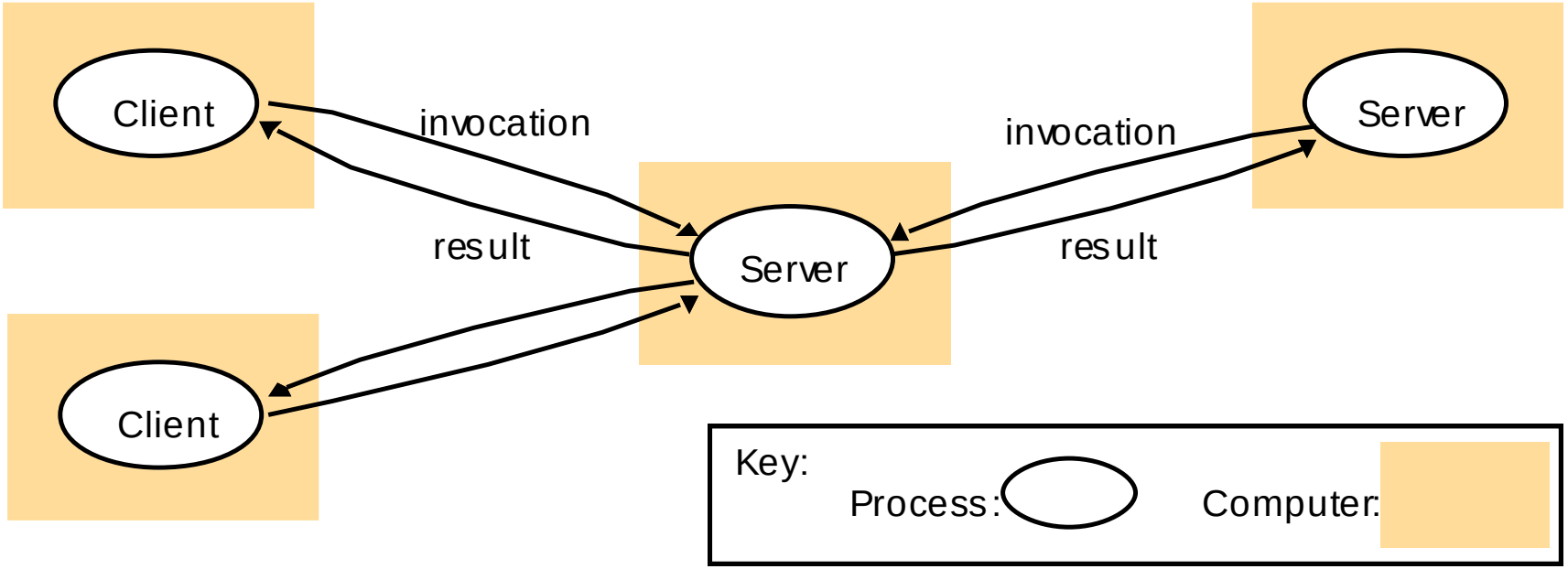
Modelos Arquitetônicos

- Classificados e estudados de acordo com as suas características comuns (“estilos arquitetônico“):
  - Cliente-servidor
  - Ponto-a-ponto
- Foco na divisão de responsabilidade entre os componentes do sistema e na alocação física desses componentes à infraestrutura de rede
  - Forte influência dos atributos de qualidade de sistema!
- Na prática, um mesmo sistema distribuído pode apresentar características de diferentes estilos (arquiteturas híbridas)

Estilo Cliente-Servidor

- Divisão das responsabilidade entre os componentes do sistema de acordo com dois papéis bem definidos:
  - Clientes
  - Servidores
- Servidores são responsáveis por gerenciar e controlar o acesso aos recursos mantidos no sistemas
- Clientes interagem com servidores de modo a terem acesso aos recursos que estes gerenciam
- Alguns servidores podem assumir o papel de clientes de outros servidores
  - Ex: Servidor web no papel de cliente de um servidor de nomes
- Continua sendo o modelo de sistema distribuído **mais estudado na prática**!

Clientes invocando servidores individuais



Camadas de Aplicação

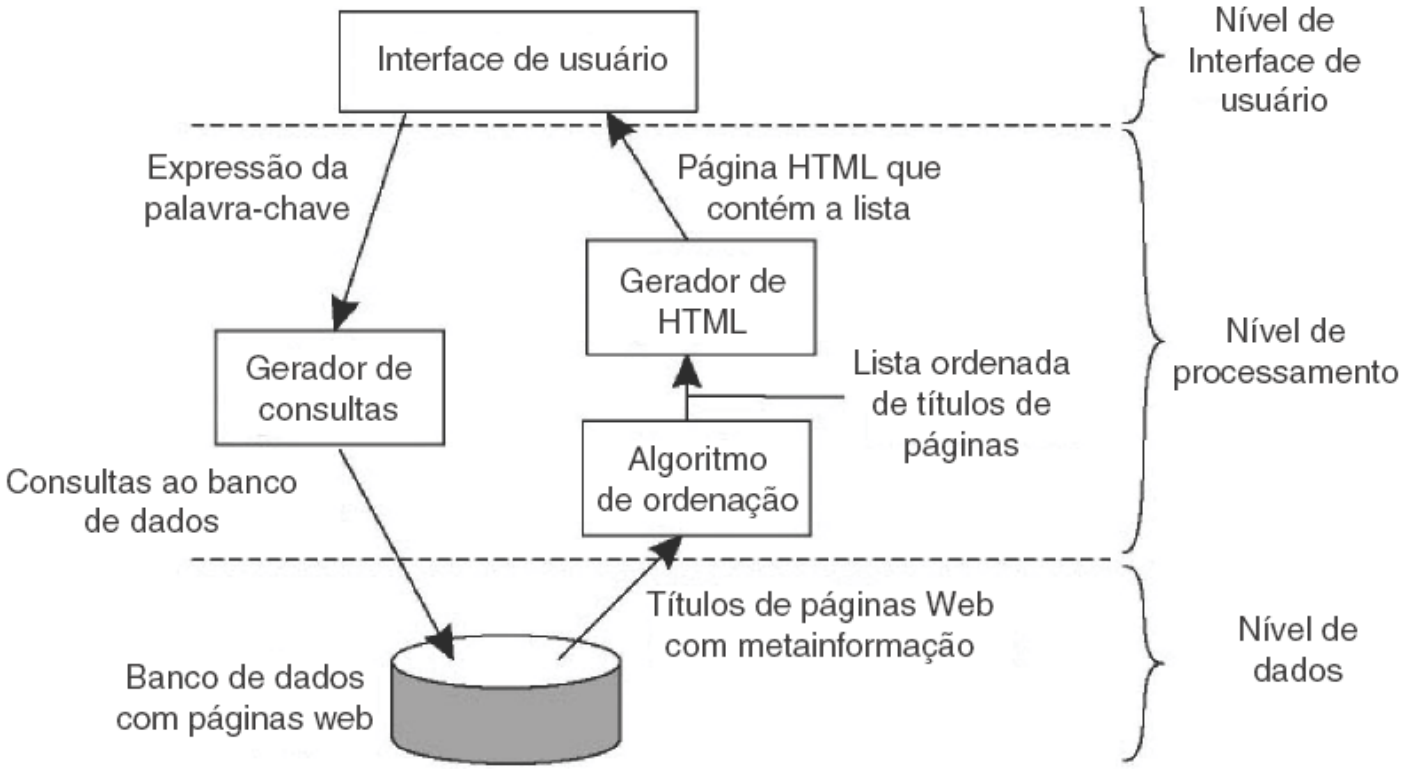
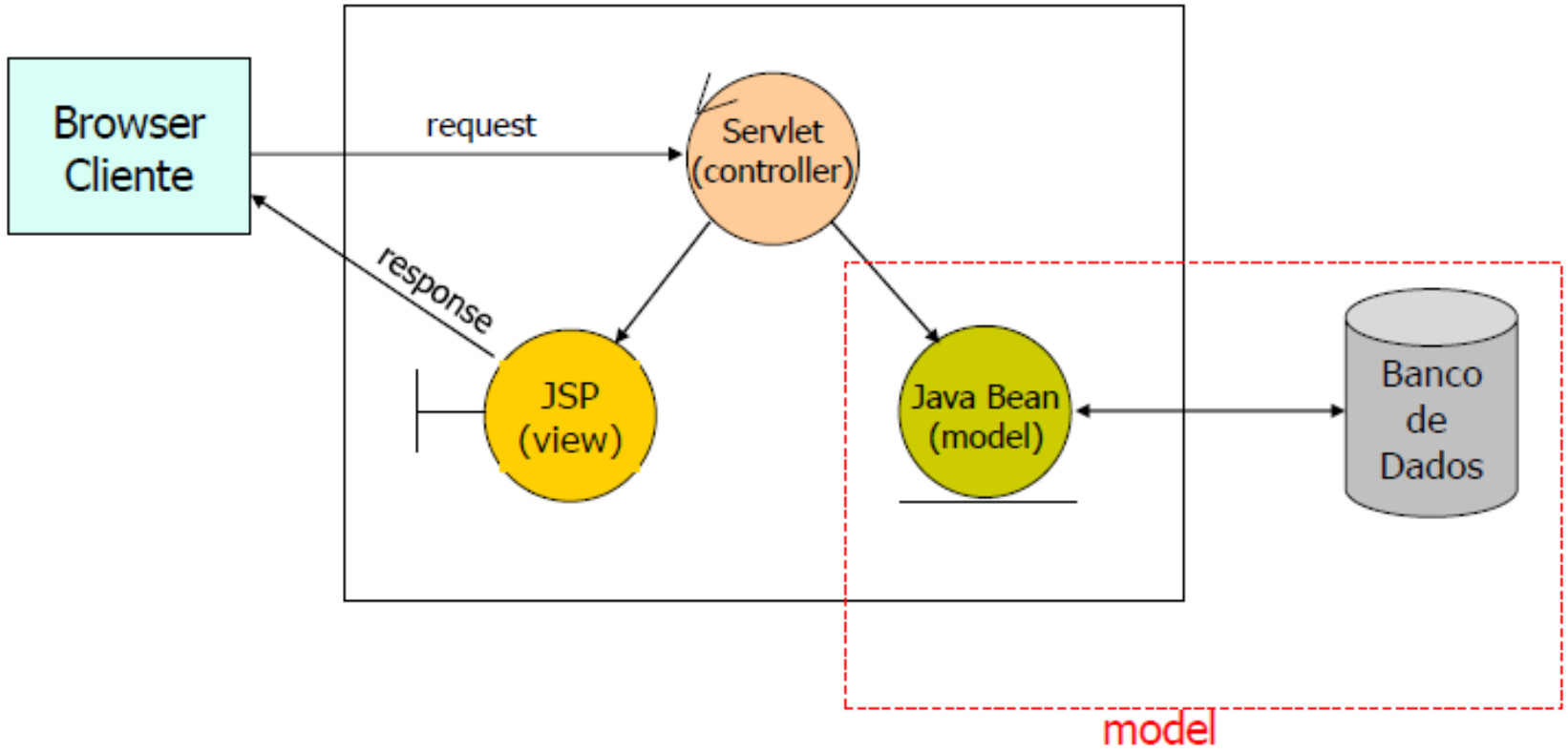


Figura 2.4 Organização simplificada de um mecanismo de busca da Internet em três camadas diferentes.

Arquitetura MVC



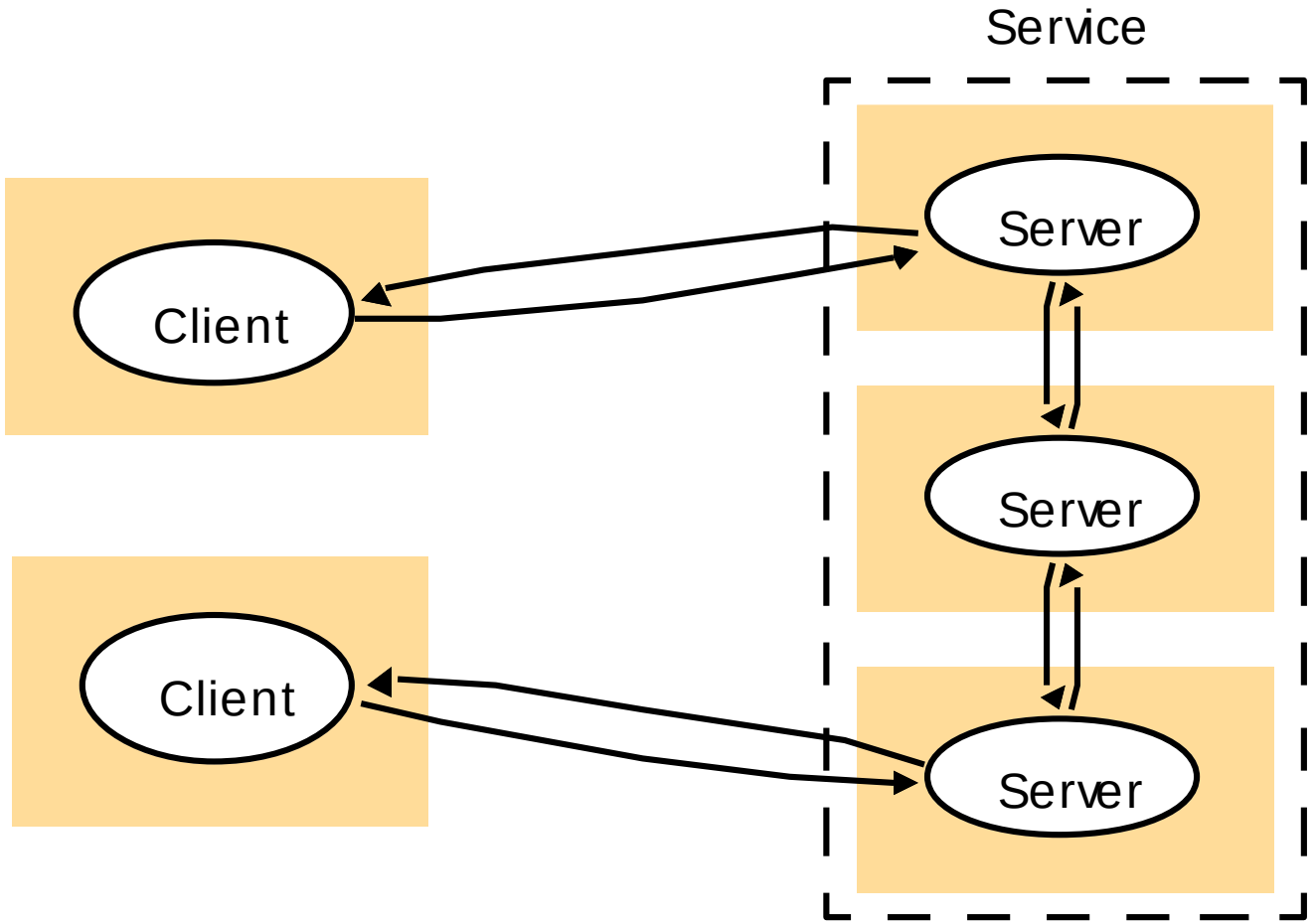
Estilo Cliente-Servidor

- Sete variações do estilo Cliente-Servidor mais estudadas:
  - Múltiplos servidores por serviço
  - Cache e servidores proxy
  - Clientes magros
  - Código móvel
  - Agentes móveis
  - Objetos distribuídos
  - Dispositivos móveis

Cliente Servidor com Múltiplos Servidores

- Cada serviço é implementado por um conjunto de servidores, possivelmente localizados em diferentes pontos da rede
- Servidores podem interagir entre si para oferecer uma visão global consistente do serviço para os clientes
- Técnicas mais utilizadas:
  - Particionamente – distribuição física dos recursos entre os vários servidores
    - Maior facilidade de gerência e maior escalabilidade
    - Ex: Clusters de servidores do portal UOL
  - Replicação – manutenção de cópias do mesmo recurso lógico em dois ou mais servidores
    - Maior desempenho e disponibilidade
    - Ex: Base de dados do Google

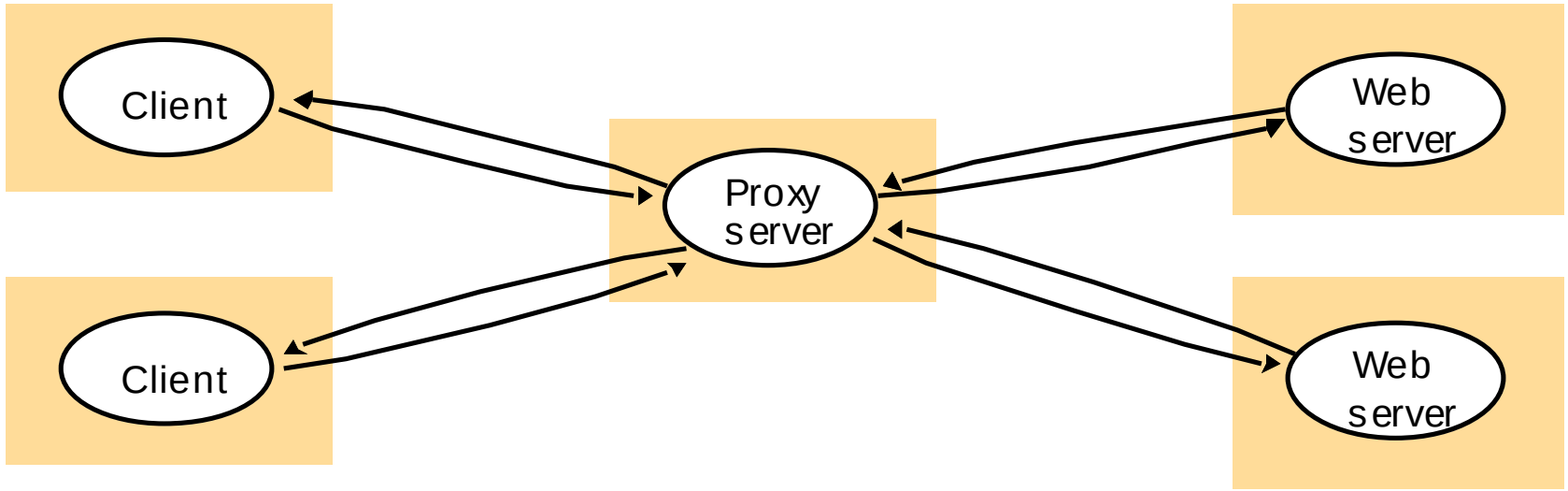
Um serviço provido por múltiplos servidores



Cliente servidor com Cache e Servidor de Proxy

- Cache
  - Repositório de cópias de objetos recentemente utilizados que está fisicamente mais próximo do que os objetos originais
  - Principais desafios:
    - Política de atualização (controla a entrada e saída de objetos no cache)
  - Localização física (nos clientes ou em um ou mais servidores proxy)
- Servidor proxy
  - Processo compartilhado por vários clientes que serve como cache para os recursos disponibilizados por outros servidores remotos
  - Principais funções
    - Reduzir o tempo de acesso
    - Aumentar a disponibilidade
    - Também utilizado para proteção, filtragem, adaptação, etc.

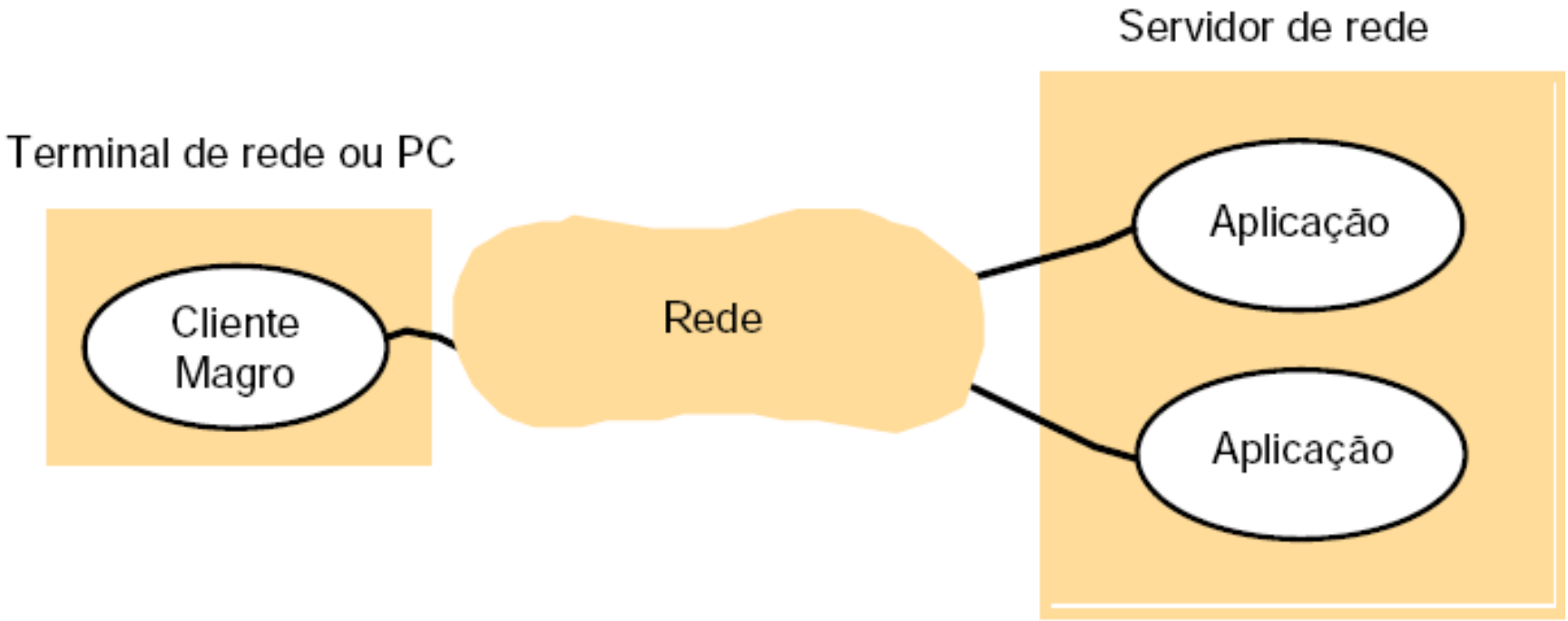
Servidor de Proxy Web



Cliente Servidor com Cliente Magro

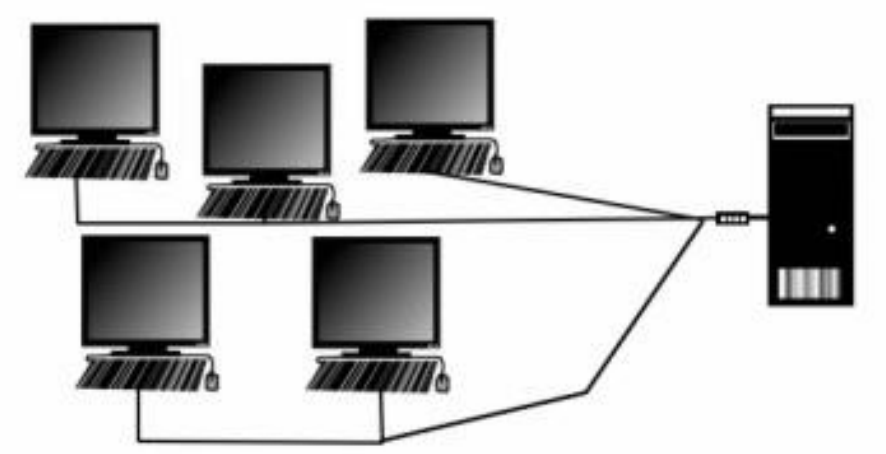
- Cliente Magro
  - Camada de software com suporte para interação local com o usuário, e que executa aplicações e solicita serviços exclusivamente a partir de servidores remotos
    - VNC (*Virtual Network Computing*)
    - LTSP (*Linux Terminal Server Project*)
  - A favor:
    - Baixo custo de *hardware* e *software* para os clientes
    - Maior facilidade de gerência e manutenção das aplicações
  - Contra:
    - Alto custo de hardware e software para os servidores
    - Centralização da carga de trabalho e do tráfego de mensagem
    - Risco de sobrecarga dos servidores e/ou da rede
    - Baixo desempenho para aplicações altamente interativas

Cliente Servidor com Clientes Magros

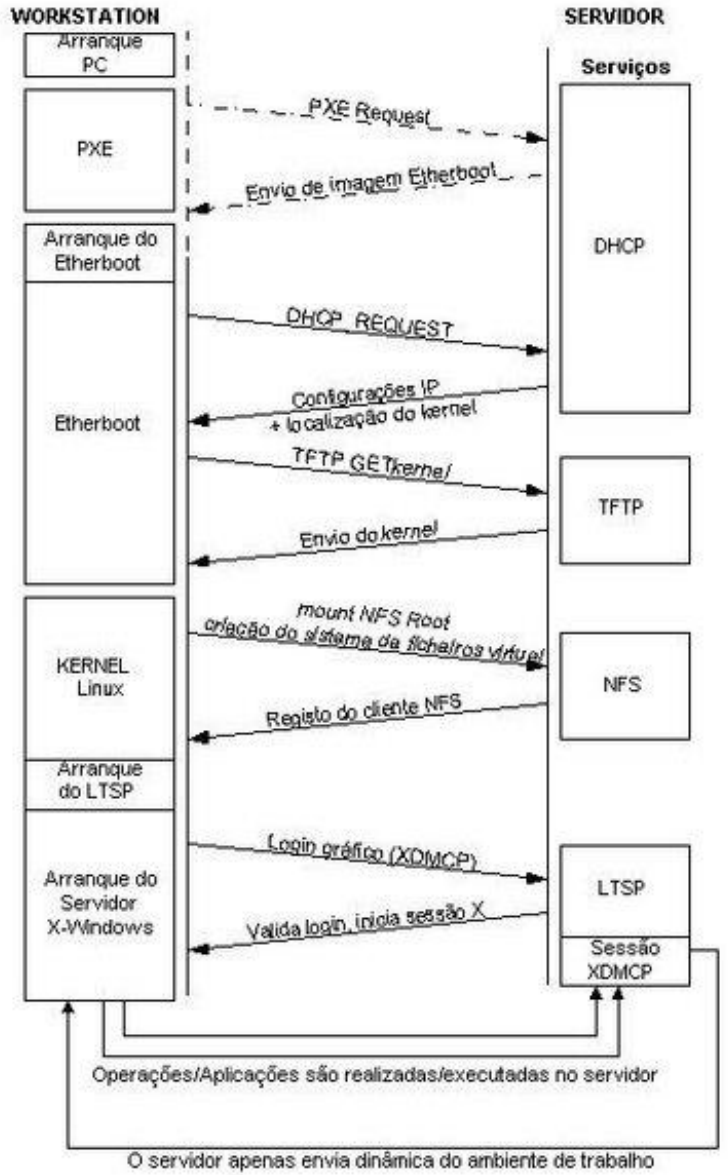


LTSP (Linux Terminal Server Project)

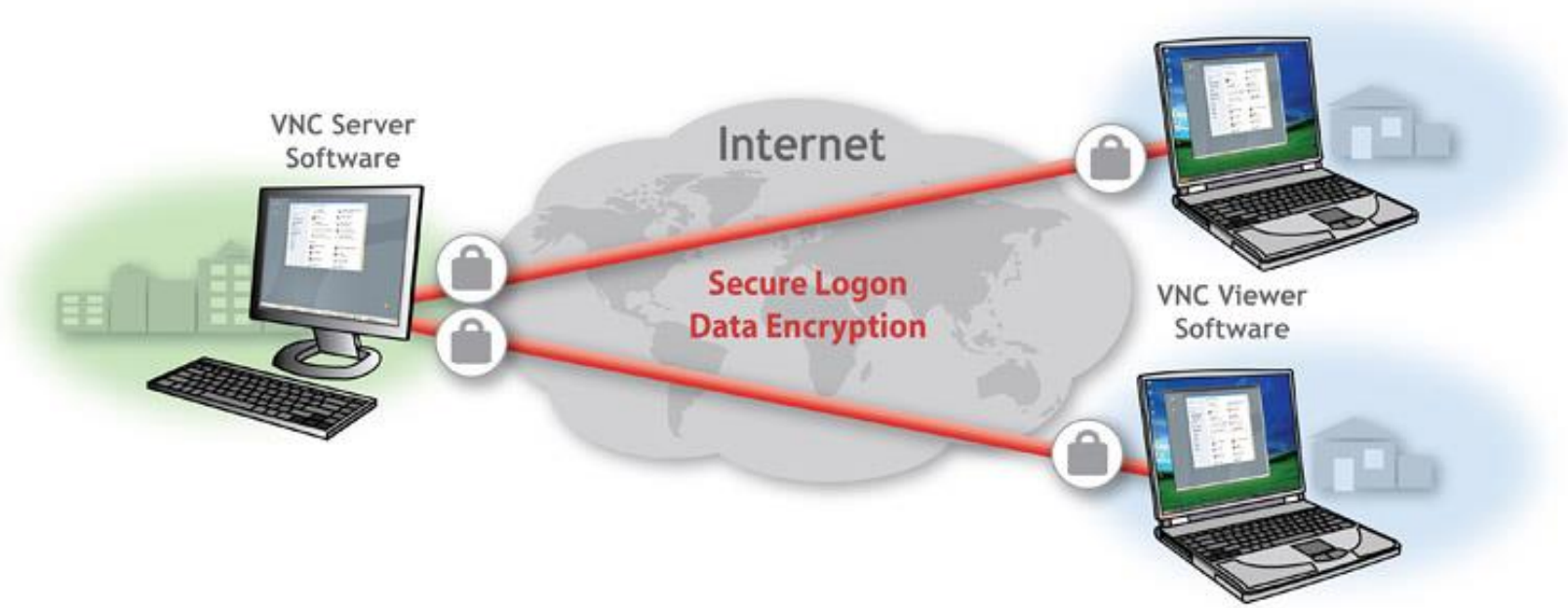
- Utiliza uma combinação de DHCP, TFTp e NFS para permitir que as estações rodem aplicativos instalados no servidor, mas realmente dêem boot via rede, baixando todos os softwares de que precisam diretamente do servidor.



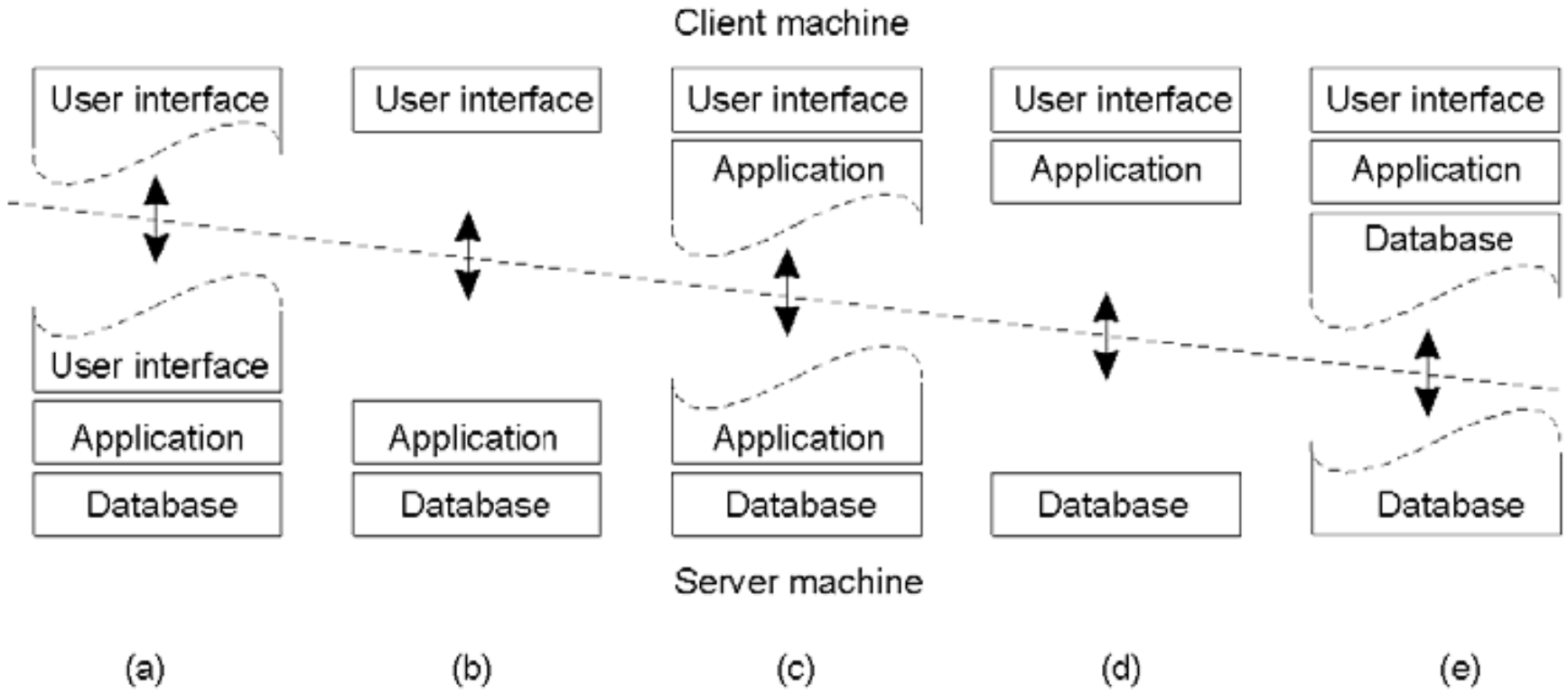
LTSP-BOOT


WORKSTATION
SERVIDOR
Arranque PC
Serviços
PXE
PXE Request
Envio de imagem Etherboot
DHCP
Arranque do Etherboot
DHCP REQUEST
Configurações IP + localização do kernel
Etherboot
TFTP GET kernel
TFTP
Envio do kernel
mount NFS Root
criação do sistema de ficheiros virtual
KERNEL Linux
NFS
Registo do cliente NFS
Arranque do LTSP
Login gráfico (XDMCP)
LTSP
Arranque do Servidor X-Windows
Valida login, inicia sessão X
Sessão XDMCP
Operações/Aplicações são realizadas/executadas no servidor
O servidor apenas envia dinâmica do ambiente de trabalho

VNC (*Virtual Network Computing*)



Clientes Magros e Seus Níveis



Cliente Servidor com Código Móvel

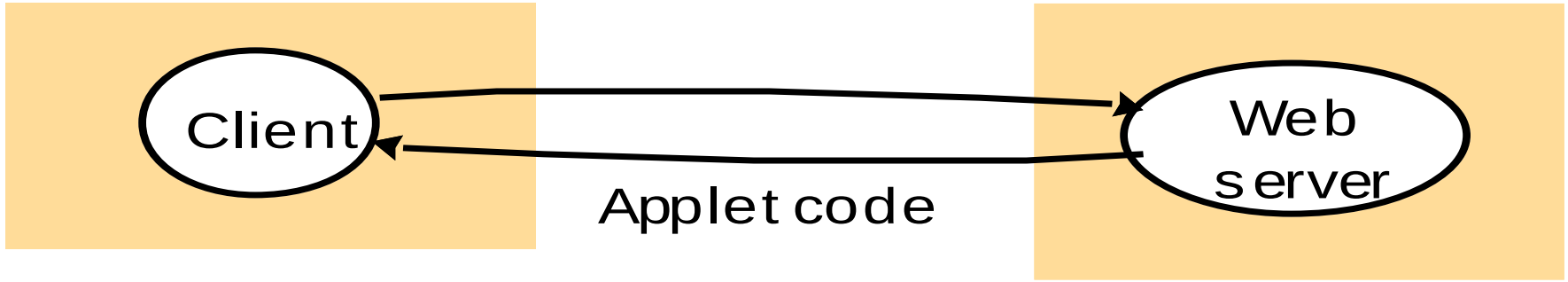
- Serviços oferecidos na forma de um código (programa) específico que deve ser descarregado do servidor
  - Aplicações clientes executam e interagem localmente com o código móvel recebido
  - Dependendo do serviço, código móvel pode interagir com um ou mais servidores em nome da aplicação cliente
  - Ex: Java Applets
- Principais benefícios
  - Redução do tempo de resposta para aplicações interativas
  - Maior facilidade de customização e atualização da interface de acesso ao serviço
  - Possibilidade de estender dinamicamente as funcionalidades das aplicações clientes

Cliente Servidor com Código Móvel

- Navegadores impedem que o Código Móvel:
  - Acesse arquivos locais
  - Impressoras
  - Sockets de Rede
- JVM
  - Classes carregadas por download são armazenadas separadamente das classes locais, impedindo sobrescrita por classes maliciosas
  - É verificada a validade dos *bytecodes.*
    - Composto de instruções de um conjunto específico?
    - Acessa endereços de memória inválidos?

Cliente Servidor com Código Móvel

a) client request results in the downloading of applet code



b) client interacts with the applet

Cliente Servidor com Código Móvel

- Desafios de projeto:
  - Heterogeneidade do código móvel e da arquitetura de execução das aplicações clientes
    - Solução: máquinas virtuais padronizadas embutidas nas aplicações clientes
  - Riscos de segurança na execução do código móvel
    - Solução: limitar as ações do código móvel ou executá-lo em um ambiente isolado do restante da rede
  - Atrasos causados pelo tempo de transferência do código móvel e pelo tempo de inicialização do seu ambiente de execução
    - Solução: transferência do código em formato compactado; cache de códigos recentemente utilizados; pré-inicialização do ambiente de execução

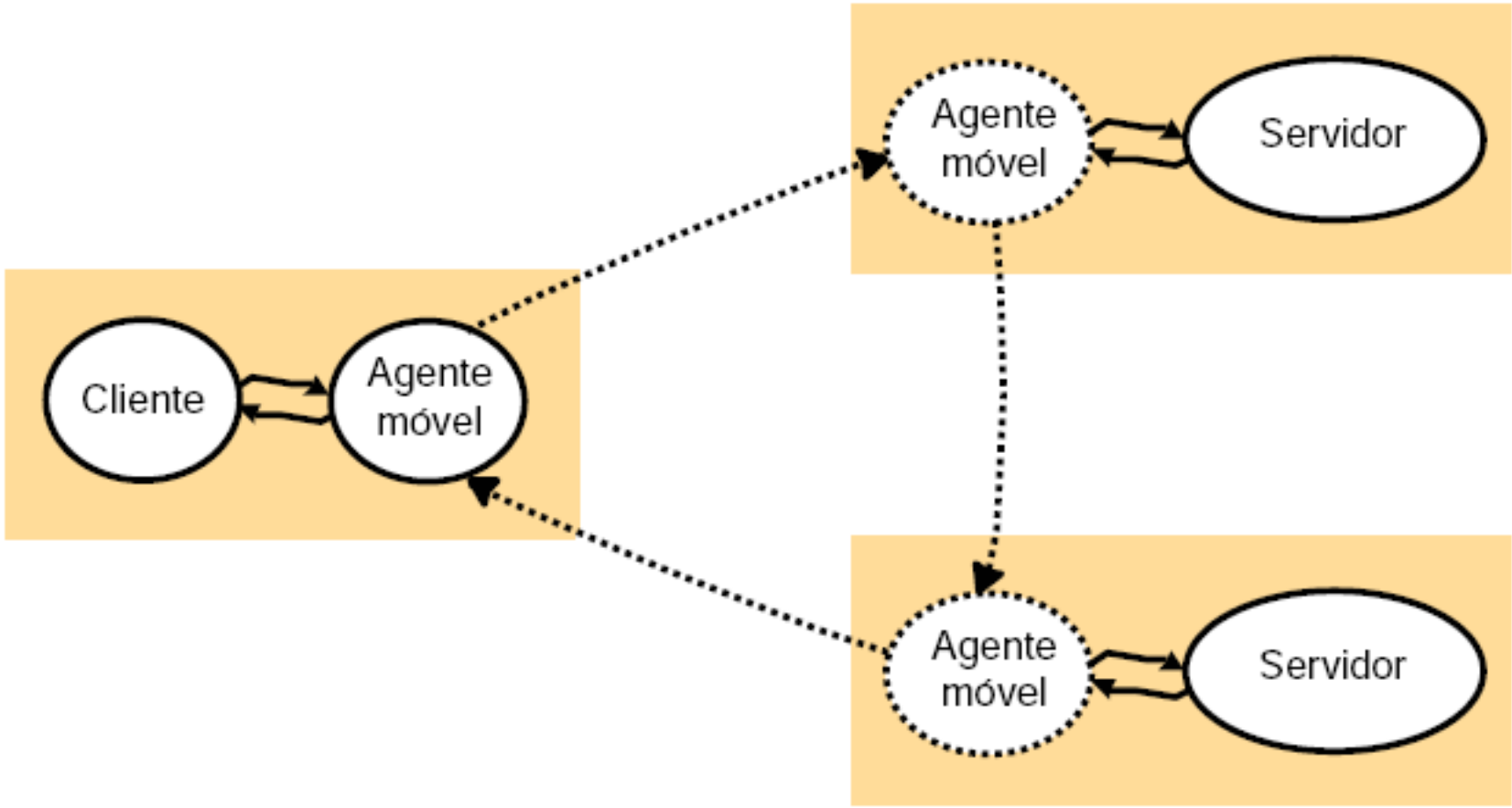
Cliente Servidor com Agente Móvel

- Agente móvel
  - Programas em execução (código + dados) que circula pela rede solicitando serviços em nome de um usuário ou de uma aplicação cliente
    - Ex.: agente para coleta de dados, busca e comparação de preços de produtos, instalação de software, etc
  - O acesso aos serviços é feito localmente pelo agente, ou de locais fisicamente próximos (da mesma rede local) aos servidores
- Benefícios:
  - Redução dos custos e do tempo de acesso
    - Acesso antes remoto agora passa a ser local
  - Maior tolerância a falhas de comunicação
    - Conexão necessária apenas durante a transferência do agente
  - Melhor distribuição do tráfego de mensagens na rede

Cliente Servidor com Agente Móvel

- Agente móvel
  - Programas em execução (código + dados) que circula pela rede solicitando serviços em nome de um usuário ou de uma aplicação cliente
    - Ex.: agente para coleta de dados, busca e comparação de preços de produtos, instalação de software, etc
  - O acesso aos serviços é feito localmente pelo agente, ou de locais fisicamente próximos (da mesma rede local) aos servidores
- Benefícios:
  - Redução dos custos e do tempo de acesso
    - Acesso antes remoto agora passa a ser local
  - Maior tolerância a falhas de comunicação
    - Conexão necessária apenas durante a transferência do agente
  - Melhor distribuição do tráfego de mensagens na rede

Cliente Servidor com Agente Móvel



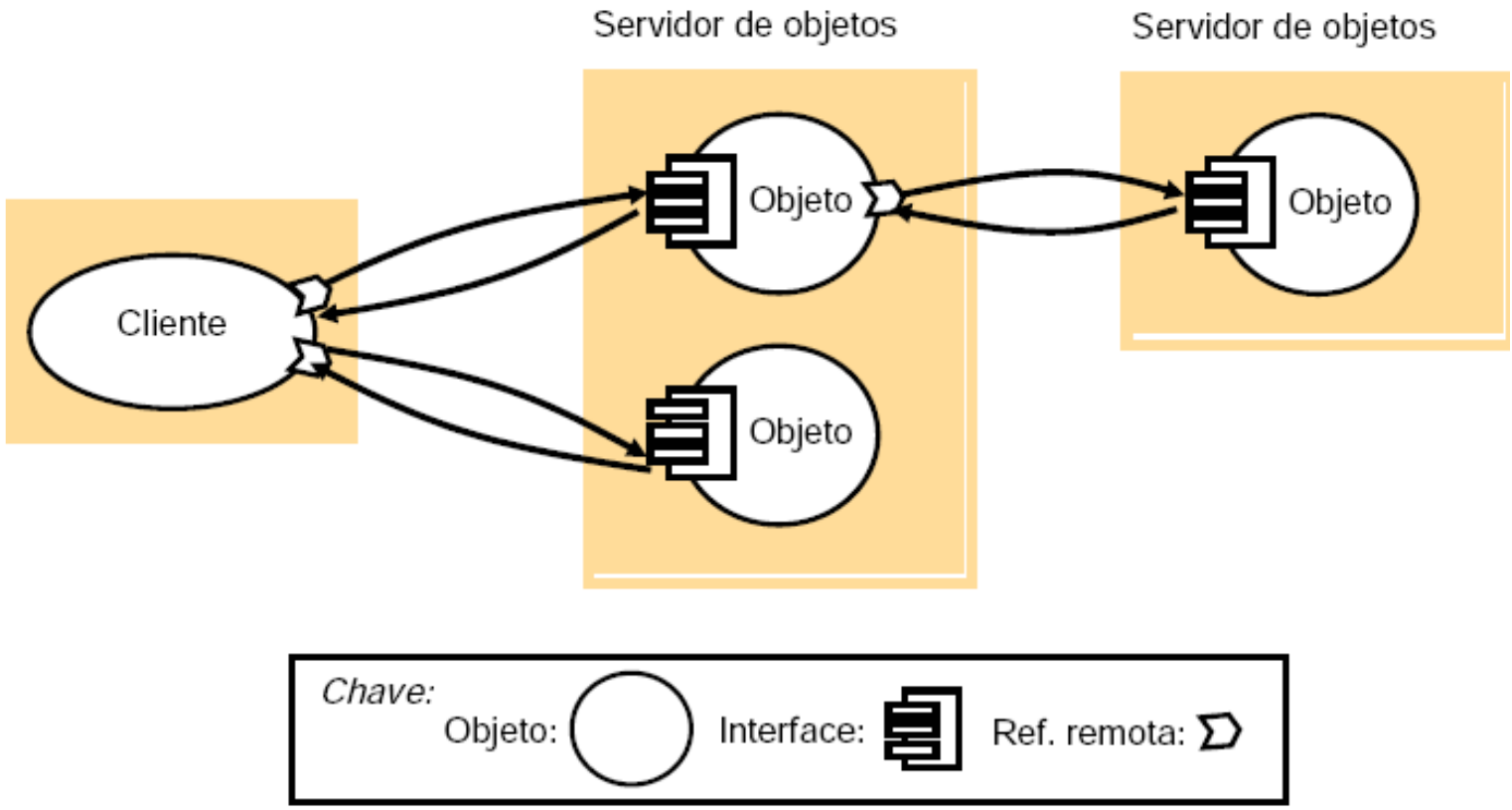
Cliente Servidor com Agente Móvel

- Desafios de projeto:
  - Heterogeneidade do código e da arquitetura de execução dos agentes
    - Solução: ambientes de execução padronizados em cada ponto do sistema
  - Riscos de segurança na execução dos agentes
    - Solução: restringir a entrada a agentes certificados e executá-los em um ambiente isolado ou com acesso aos recursos locais rigorosamente controlado
  - Riscos de interrupção dos agentes devido à negação de acesso por parte dos servidores
    - Solução: utilizar agentes certificados e com permissão de acesso aos recursos requisitados
  - Atrasos causados pela tempo de transferência dos agentes Solução: transferência dos agentes em formato compactado

Cliente Servidor com Objetos Distribuídos

- Objetos encapsulados em processos servidores
  - Objetos acessados por outros processos (clientes) através de referências remotas para uma ou mais de suas interfaces
  - Referência remota permite invocar remotamente os métodos disponíveis na interface do objeto referenciado
- Implementação na forma de middleware orientada a objetos (ex.: CORBA, Java-RMI, EJB, .NET)
  - Diferentes mecanismos para **criar**, **executar**, **publicar**, **localizar**, e **invocar** objetos remotos
  - Diferentes serviços de suporte
    - Transação, persistência, replicação, segurança, etc

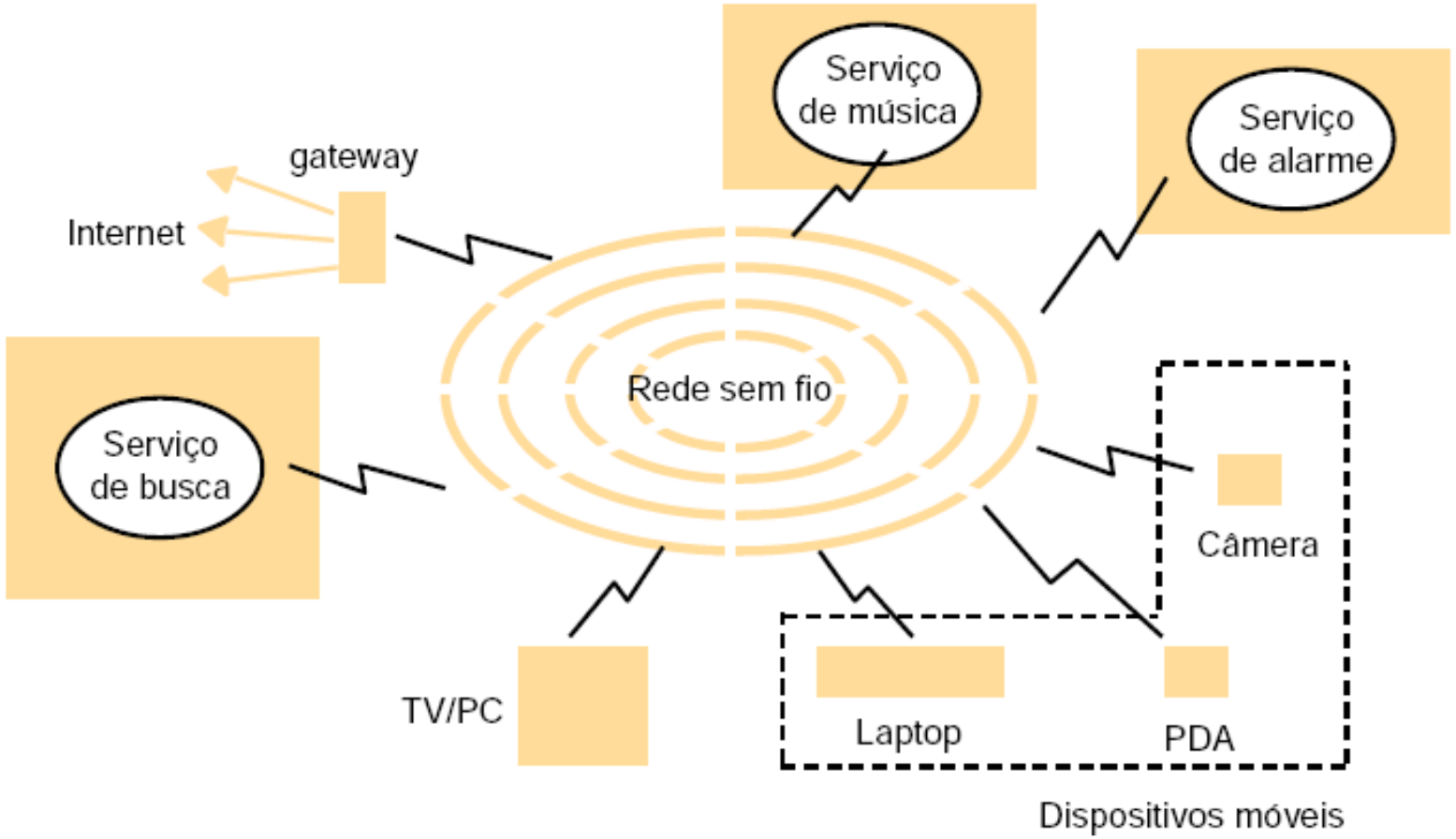
Cliente Servidor com Objetos Distribuídos



Cliente Servidor com Dispositivo Móvel

- Formado por aplicações clientes que executam em dispositivos móveis (PDAs, laptops, celulares, etc) e acessam servidores da rede fixa através de uma infraestrutura de comunicação sem fio
  - Diferença para as variações com código móvel e agentes móveis?
- Principais benefícios:
  - Fácil conexão dos dispositivos a uma nova rede local
    - Inclusão de novos clientes sem a necessidade de configuração explícita
  - Fácil integração dos clientes aos serviços locais
    - Descoberta automática de novos serviços (sem intervenção do usuário)
- Desafios de projeto:
  - Identificação de recursos independente de sua localização física
  - Limitações de processamento, tempo de conexão e throughput
  - Privacidade e segurança

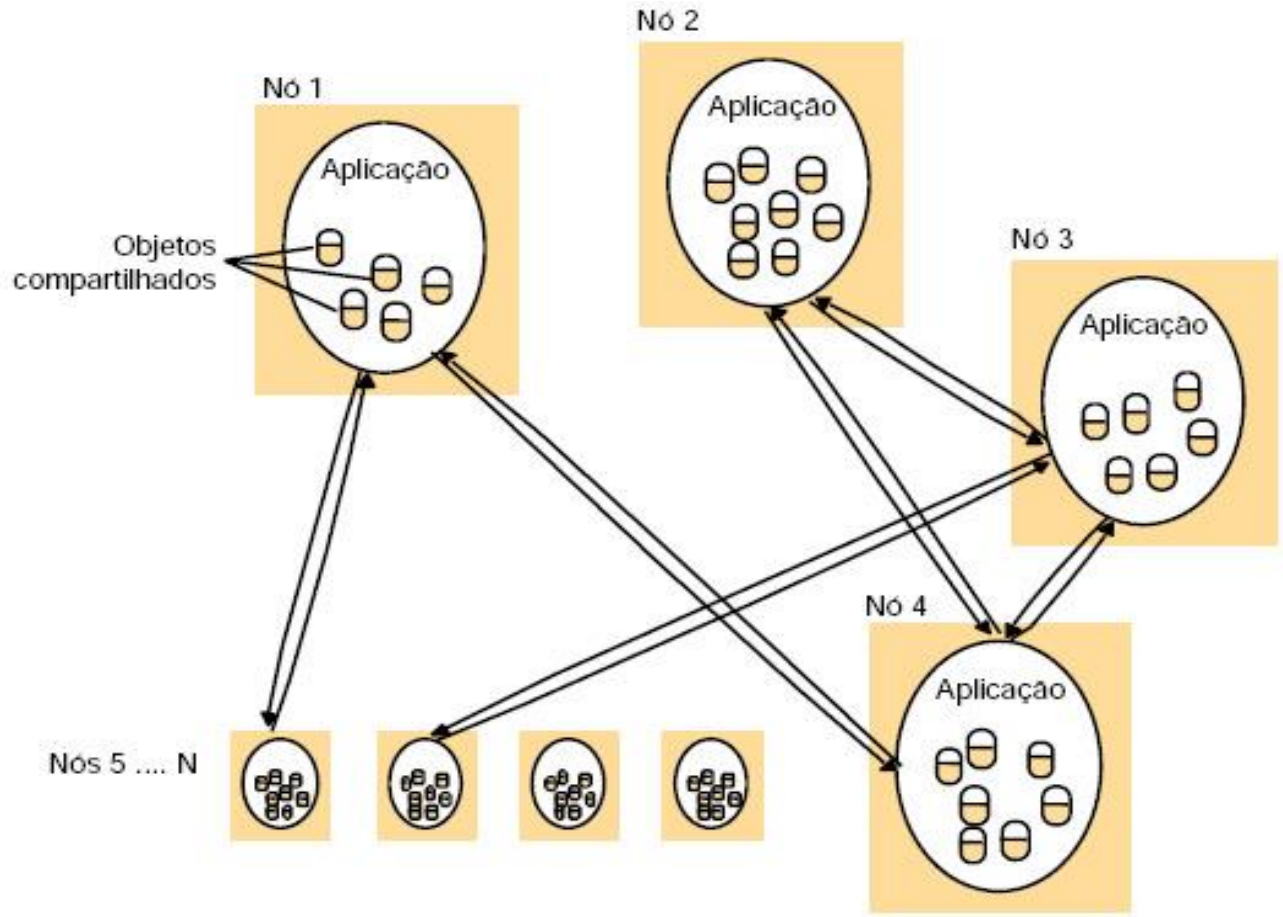
Cliente Servidor com Dispositivos Móvel



Estilo Ponto-a-Ponto

- Todos os processos (**nós**) envolvidos em uma mesma tarefa ou atividade exercem papéis similares, **interagindo cooperativamente** como “parceiros” (**peers**) uns dos outros
  - Criado para suprir as conhecidas deficiências de escalabilidade do modelo CS tradicional
- O objetivo principal é explorar os recursos (hardware & software) de um grande número de máquinas/usuários interessados em realizar uma determinada tarefa ou atividade
  - Uso pioneiro no compartilhamento de arquivos de áudio (Napster)
  - Sucesso do Napster abriu caminho para vários outros sistemas e middleware P2P de propósito geral (KaZaA, Gnutella, Emule, JXTA, etc)

Estilo Ponto-a-Ponto



Estilo Ponto-a-Ponto

- Características dos sistemas P2P:
  - Arquitetura totalmente **distribuída** (sem qualquer controle centralizado)
  - **Sem distinção** entre clientes e servidores (cada nó é cliente e servidor ao mesmo tempo)
  - Nós podem **trocar recursos** diretamente entre si
  - Nós são **autônomos** para se juntarem ao sistema ou deixá-lo quando quiserem
  - Interação entre nós pode ser feita utilizando mecanismos apropriados para **comunicação em grupo** (difusão seletiva, notificação de eventos)

Estilo Ponto-a-Ponto

- Características dos sistemas P2P:
  - Seu projeto garante que cada usuário contribua com recursos para o sistema
  - Seu correto funcionamento não depende da existência de quaisquer sistemas administrados de forma centralizada
  - Podem oferecer um grau limitado de anonimato para provedores e usuários de recursos
  - Um problema importante dos sistemas p2p é a distribuição de objetos de dados em muitos hosts e subsequente acesso a eles de uma maneira que equilibre a carga de trabalho e garanta a disponibilidade sem adicionar cargas indevidas.

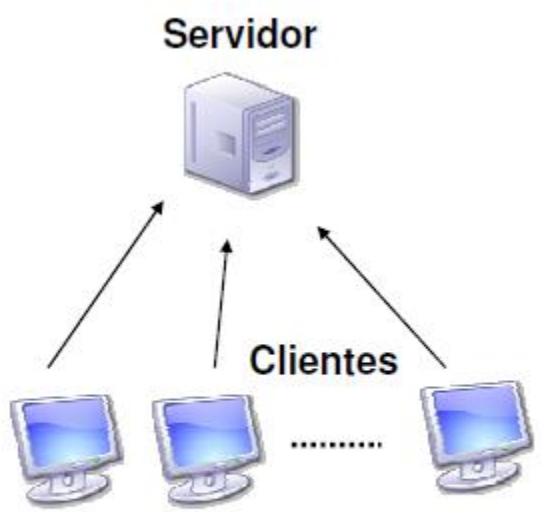
Estilo Ponto-a-Ponto

- Exemplos
  - Primeira Geração
    - Napster
  - Segunda Geração
    - Freenet
    - Gnutella
    - Kazaa
    - Bit-Torrent
  - Terceira Geração – *Middleware*
    - Pastry
    - CAN
    - Chord
    - Tapestry

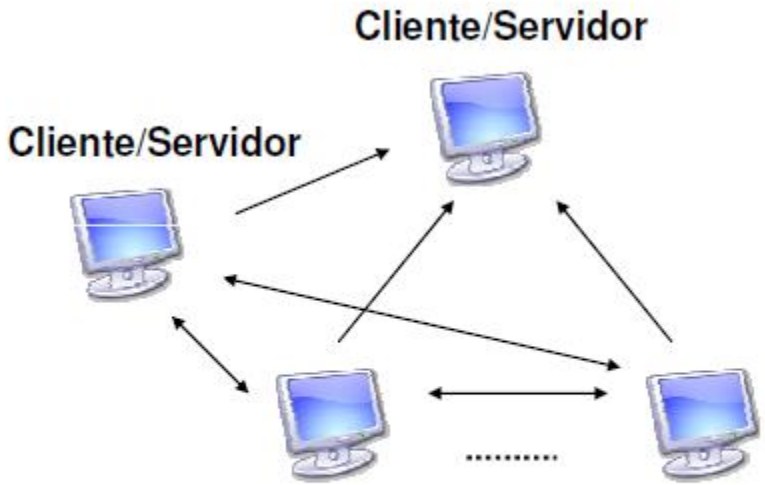
Classificação de aplicações p2p

- Compartilhamento de arquivos
  - Napster, Gnutella, Freenet, Oceanstore, PAST, Freehaven, KaZaA
- Computação distribuída
  - Seti@home, Entropia, Parabon, Popular Power, …
- Trabalho colaborativo
  - Mensagens instantâneas (Jabber, MSN, AIM, YahooMessager!, …)
  - Groupware(Groove, Netmeetingetc)

Estilo Ponto-a-Ponto vs Cliente-Servidor



(a) Cliente/Servidor Típico



(b) Ponto-a-Ponto Típico

Agenda

- Introdução
- Modelos físicos
- Modelos de arquitetura
- Modelos fundamentais
- Resumo

Modelos Fundamentais

- Independente de ser cliente/servidor ou *p2p, todos os modelos possuem características* comuns:
  - São constituídos de processos
  - Esses processos se comunicam através do envio de mensagens em uma rede de comunicação
  - Requisitos de projeto semelhantes
    - Desempenho e confiabilidade das redes e processos
    - Segurança dos recursos compartilhados

Modelos Fundamentais

- Foco em três importantes aspectos de projeto:
  - Mecanismo de interação
  - Tratamento de falhas
  - Segurança
- Utilizados para ajudar a planejar, entender e analisar o comportamento esperado do sistema
- Principais benefícios:
  - **Correção** antecipada de erros
  - Investigação, avaliação e reuso de diferentes **alternativas de projeto**
  - **Menor custo** de desenvolvimento, manutenção e evolução

Modelos Fundamentais

- Modelos de Interação:
  - Como os processos se comunicam?
  - Deve refletir o fato de que a comunicação ocorre com atrasos
- Modelos de Falha:
  - Define e classifica as falhas
- Modelos de Segurança:
  - Define e classifica as formas que ataques de agentes externos ou internos podem assumir

Modelo de Interação

- Descreve as formas de **interação e coordenação** entre os componentes do sistema
  - Programa Simples vs Sistema Distribuído
- Influenciado pela:
  - Capacidade de **comunicação da rede**
    - **Latência (transmissão, acesso à rede, S.O.)**
    - Largura de banda
    - **Instabilidade (*jitter)***
  - Ausência de estado global
    - Impossibilidade de acordo sobre a mesma **noção de tempo**
    - Dificuldade para **sincronizar relógios através da rede**
- Duas variantes em relação ao tempo:
  - Modelo **síncrono**
  - Modelo **assíncrono**

Modelo de Interação

- Descreve as formas de **interação e coordenação** entre os componentes do sistema
  - Processos interagem entre si através da **troca de mensagens** para coordenar suas atividades
  - Um algoritmo **distribuído** define os passos necessários para essa coordenação
  - Cada processo possui seu próprio **estado**
    - Ausência de estado global
  - O desempenho do canal de comunicação subjacente não é previsível (atraso, banda, *jitter)*
  - Não é possível manter uma noção global de tempo única/precisa

Modelo de Interação

- Desempenho do canal de comunicação
  - Latência: Atraso entre o início da transmissão da mensagem para um processo e o início da recepção dessa mensagem pelo outro processo
    - soma de várias componentes:
      - transmissão + acesso à rede + propagação
  - Largura de banda
    - Quantidade total de informação que pode ser transmitida em uma unidade de tempo
  - *Jitter*
    - Variação do atraso
    - Muito importante para dados de tempo real

Relógios e temporização de eventos

- Cada computador possui seu próprio relógio
  - Pode ser usado para marcar o tempo de eventos locais
- Mas sem significância global
  - Cada relógio marca um tempo diferente
  - Defasagem entre os relógios
- Técnicas para sincronização de relógios podem ser aplicadas
  - GPS, algoritmos de sincronização

Modelo Síncrono

- Características:
  - O tempo para **executar cada passo de um processo tem limites inferior e superior conhecidos**
  - Cada **mensagem transmitida** por um canal de comunicação é recebida dentro de um **limite conhecido** de tempo
  - Cada processo tem um relógio local cuja **taxa de desvio** do tempo real tem um **limite conhecido**
- Vantagens:
  - **Mais fácil** para **programar e analisar** o comportamento dos processos
    - ex.: *timeouts* para detectar falhas
- Desvantagens:
  - Dificuldade de definir e garantir **valores realistas para os** limites de tempo (***timeout)***
  - Resultados de análise e simulação podem não ser confiáveis

Modelo Assíncrono

- Características:
  - **Sem limites** conhecidos para
    - **Velocidade** de execução dos processo
    - **Atraso na transmissão** das mensagens
    - Taxa de **desvio dos relógios**
- Vantagens:
  - Mais realista
  - Soluções assíncronas também são válidas para o modelo síncrono
- Desvantagens:
  - Mais difícil de implementar e analisar
- Exemplos?

Ordenação de Eventos

- Em muitos casos, a execução de um sistema distribuído pode ser descrita em **termos dos eventos** ocorridos no sistema e da ordem em que eles ocorreram
  - Problema:
    - como saber se um determinado evento (ex.: envio ou recebimento de uma mensagem) de um determinado processo ocorreu **antes, após ou concorrentemente** a um outro evento de um outro processo (possivelmente remoto)?
- Como exemplo, considere o seguinte cenário de troca de mensagens entre um grupo de quatro usuários (X, Y, Z e A) de correio eletrônico
  1. O usuário X envia uma mensagem *m1 com o assunto* "Encontro";
  2. Os usuários Y e Z respondem enviando as mensagens *m2* e *m3, respectivamente, com o assunto "Re: Encontro".*

Ordenação de Eventos

- (Continuação do exemplo)
  - Em tempo “real”, X envia *m1 primeiro; em seguida, Y recebe* e lê *m1, e então responde enviando m2; por fim, Z recebe e* lê tanto *m1 quanto m2, e responde enviando m3*
  - Porém, devido a atrasos na rede, as três mensagens podem ser entregues a alguns usuários na ordem errada. Por exemplo, o usuário A poderia receber as mensagens na seguinte ordem:

Caixa de Entrada de A	
De	**Assunto**
Z	Re:Encontro
X	Encontro
Y	Re:Encontro

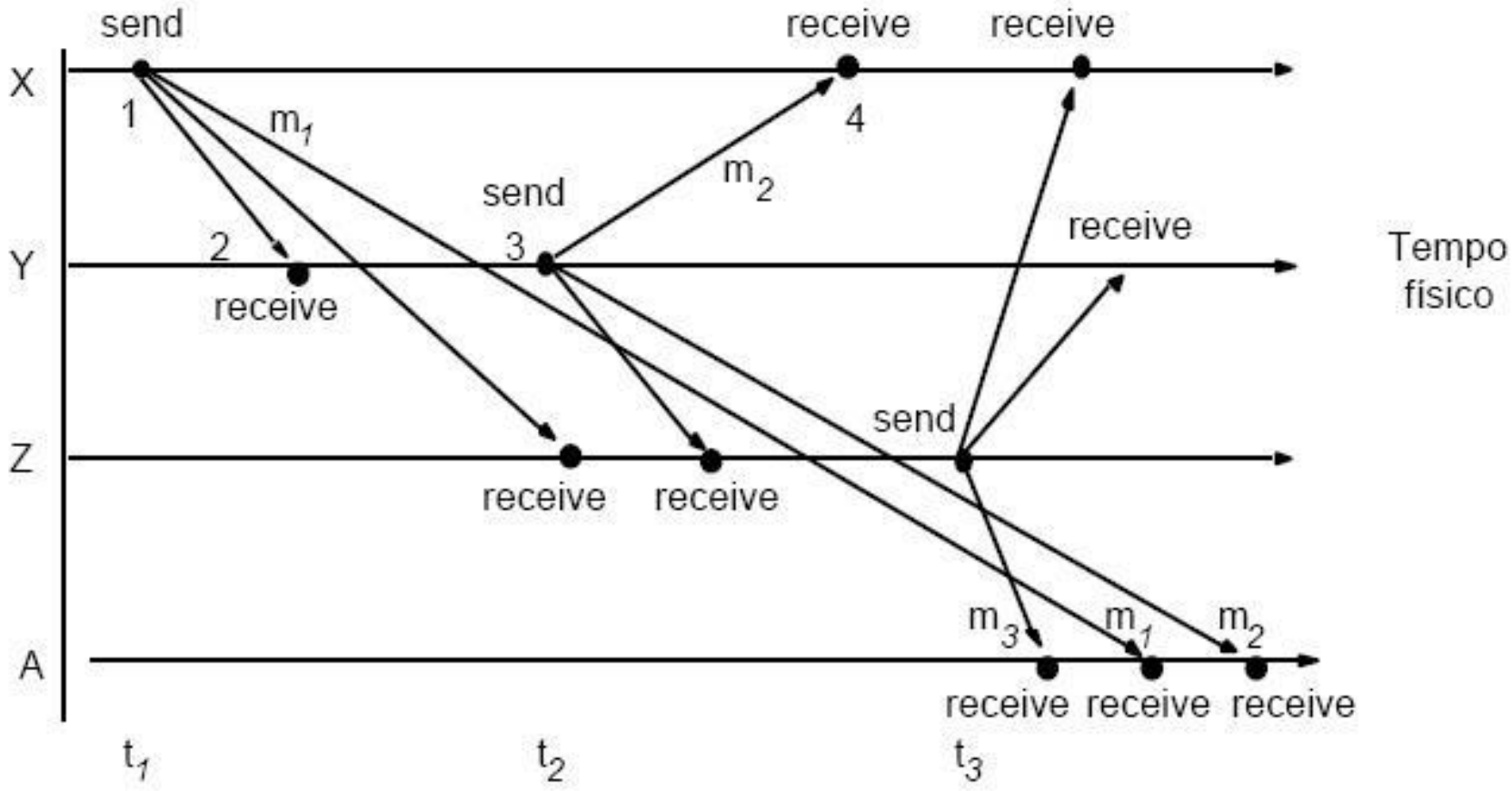
Ordenação de Eventos

- **Tentativa de solução:**
  - **Embutir a hora local do** processo remetente em cada mensagem enviada. Assim, as mensagens recebidas poderiam ser ordenadas pelo processo de destino, com **base no valor da hora embutido** em cada uma
  - **Problemas?**

Ordenação de Eventos

- Como é impossível sincronizar relógios perfeitamente em um sistema distribuído
  - *Lamport* propôs a noção de **tempo lógico** como mecanismo de ordenação de eventos em sistemas distribuídos
  - O conceito de tempo lógico permite estabelecer uma ordem parcial entre eventos ocorridos em processos executando em diferentes computadores, **sem a necessidade de recorrer a relógios "reais"**
  - Com base na relação de **causa-e-efeito** entre eventos

Ordenação de Eventos



Modelo de Falha

- Descreve as maneiras através das quais podem ocorrer **falhas nos processos e canais de comunicação,** de modo a facilitar o entendimento dos seus efeitos no sistema
- ***Falhas por Omissão***
  - Processo: processo termina inesperadamente
    - ***Fail-stop:*** *outros processos podem detectar a falha com certeza (através de timeouts): requer sistema síncrono*
  - Canal: mensagens enviadas não são recebidas

Modelo de Falha

- ***Falhas Arbitrárias***
  - *Um processo ou canal arbitrariamente omite* passos de processamento que deveriam ser executados, ou executa passos não intencionados (mais grave tipo de falha, também conhecido como **falha bizantina)**
  - *Qualquer tipo de erro pode acontecer:*
    - *Canal: mensagens podem ser omitidas, alteradas, duplicadas, etc*
    - *Processo: passos em um algoritmo podem ser omitidos...*
      - ***Não é possível detectar a falha***
- ***Falhas de Temporização*** *(apenas para o modelo síncrono)*
  - *Um* processo ou canal não atende os limites de tempo que lhes são estabelecidos (geralmente causada pela sobrecarga dos processos ou da rede)
  - Violação das suposições sobre a temporização de eventos em sistemas síncronos

Falhas por Omissão e Arbitrárias

Classe	*Afeta*	*Descrição*
Falha-e-pára	Processo	Processo interrompe sua execução em definitivo. Outros processos podem vir a detectar este estado.
Pane	Processo	Processo interrompe sua execução em definitivo. Outros processos podem não ser capazes de detectar este estado.
Omissão	Canal	Uma mensagem inserida no *buffer* de saída do remetente nunca chega ao *buffer* de entrada do destinatário.
Omissão-envio	Processo	Um processo completa um *envio*, mas a mensagem não é inserida no seu *buffer* de saída.
Omissão-receb.	Processo	Uma mensagem é inserida no *buffer* de entrada de um processo, mas o processo não a recebe.
Arbitrária (Bizantina)	Processo ou canal	Processo/canal exibe um comportamento arbitrário: envio ou recebimento de mensagens arbitrárias em momentos arbitrários; omissões; interrupção ou ação incorreta de um processo.

Falhas de Temporização

Classe	*Afeta*	*Descrição*
Relógio	Processo	Relógio local do processo excede os limites de sua taxa de desvio do tempo real.
Desempenho	Processo	Processo excede os limites do intervalo esperado entre dois passos.
Desempenho	Canal	A transmissão de uma mensagem atrasa além do limite estabelecido.

Lidando com falhas

- Mascaramento de falhas
  - Por exemplo, utilizando múltiplos servidores replicados para ocultar a falha de uma das réplicas
- Confiabilidade da comunicação
  - Detecção de mensagens com erro (CRCs)
  - Detecção de mensagens perdidas e retransmissão
  - Detecção de mensagens duplicadas

Modelo de Segurança

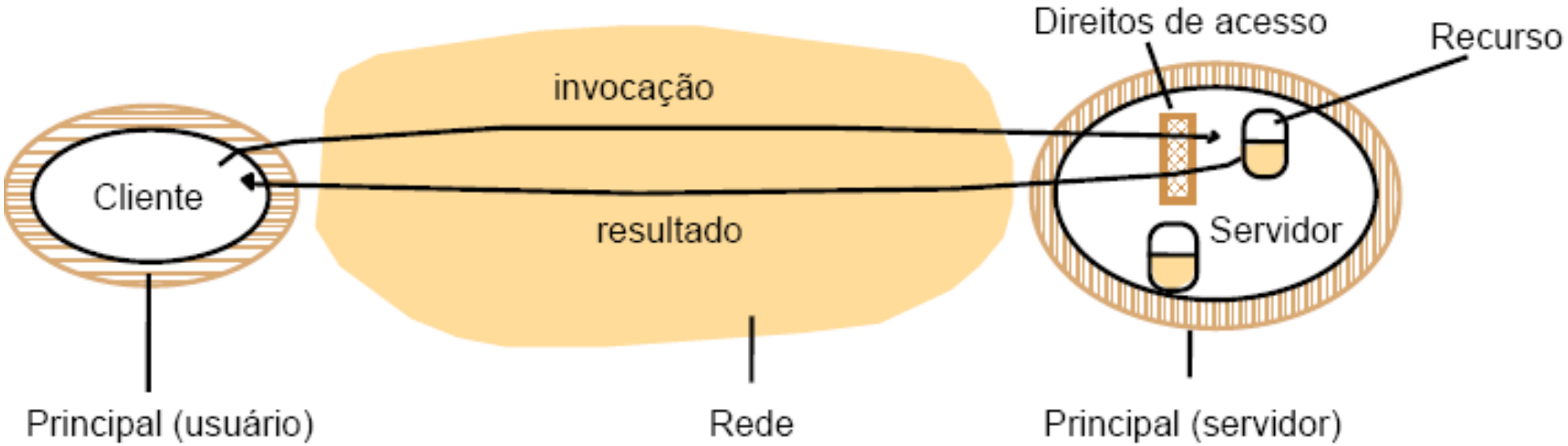
- Descreve os mecanismos utilizados para garantir a **segurança dos processos e de seus canais de comunicação, e para proteger os recursos que os** processos encapsulam contra acessos não autorizados
- Principais questões:
  - Proteção dos recursos
  - Segurança dos processos e de suas interações
- Desafios:
  - Uso de técnicas de segurança implica em **custos substanciais de processamento e de gerência**
- Necessidade de uma **análise** cuidadosa das possíveis fontes de ameaças, incluindo **ambientes externos** ao sistema (rede, físico, humano, etc.)

"Encryption software is expensive...so we just rearranged all the letters on your keyboard."

Proteção de Recursos

- Conceitos envolvidos:
  - ***Direitos de acesso –*** *especificação de quem pode* realizar as operações disponíveis para um determinado recurso
  - ***Principal –*** *entidade (usuário ou processo) autorizada* para solicitar uma operação, ou para enviar os resultados de uma operação para o principal solicitante
- Responsabilidade compartilhada entre clientes e servidores
  - Servidor **verifica a identidade** do principal por trás de cada invocação e **checa** se ele tem direitos de acesso suficientes para realizar a operação solicitada
  - Cliente **verifica a identidade** do principal por trás do servidor para **garantir** que os resultados vêm do servidor requisitado

Proteção de Recursos



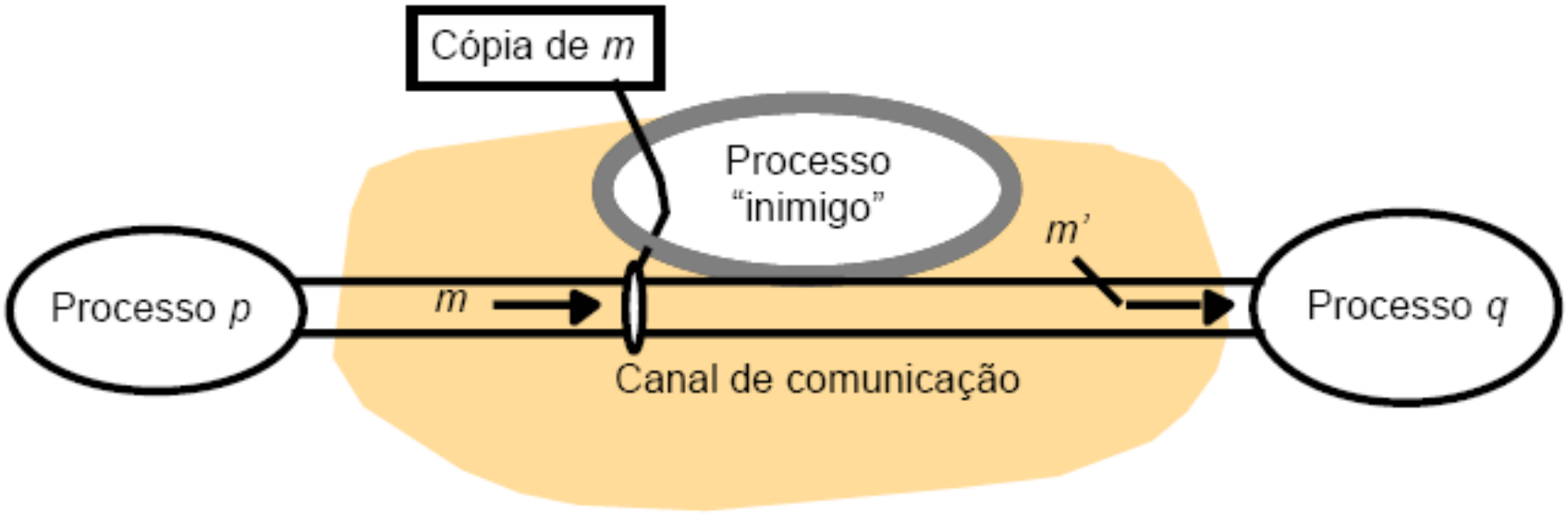
Modelo de Segurança

- Medidas de proteção aplicadas a Objetos (ou processos)
  - Direitos de acesso às interface dos objetos
  - Identidade dos objetos (*principals) e autenticação*
- Canais de comunicação
  - Encriptação dos dados transmitidos
  - Modelo de canal seguro
    - Autenticação dos objetos que se comunicam através do canal
    - Privacidade e integridade dos dados
    - Marcas de tempo nas mensagens para impedir *replay*

Segurança dos Processos e Interações

- **Natureza** aberta da rede e dos serviços de comunicação **expõe** os processos a ameaças e ataques "**inimigos"**
- Principais tipos de ameaça:
  - ***Aos processos***
    - *cliente e servidores podem não* ***ser*** capazes de determinar a identidade dos processos com os quais se comunicam
  - ***Aos canais de comunicação***
    - *mensagens podem ser* indevidamente copiadas, alteradas, forjadas ou removidas enquanto transitam pela rede
  - ***Negação de serviço***
    - *envios excessivos de mensagens ou* invocações de serviços através da rede, resultando na sobrecarga dos recursos físicos do sistema e prejudicando seus usuários
  - ***Mobilidade de código***
    - *ameaças disfarçadas na forma de* código móvel que deve ser executado localmente pelos clientes ("Cavalo de Tróia")

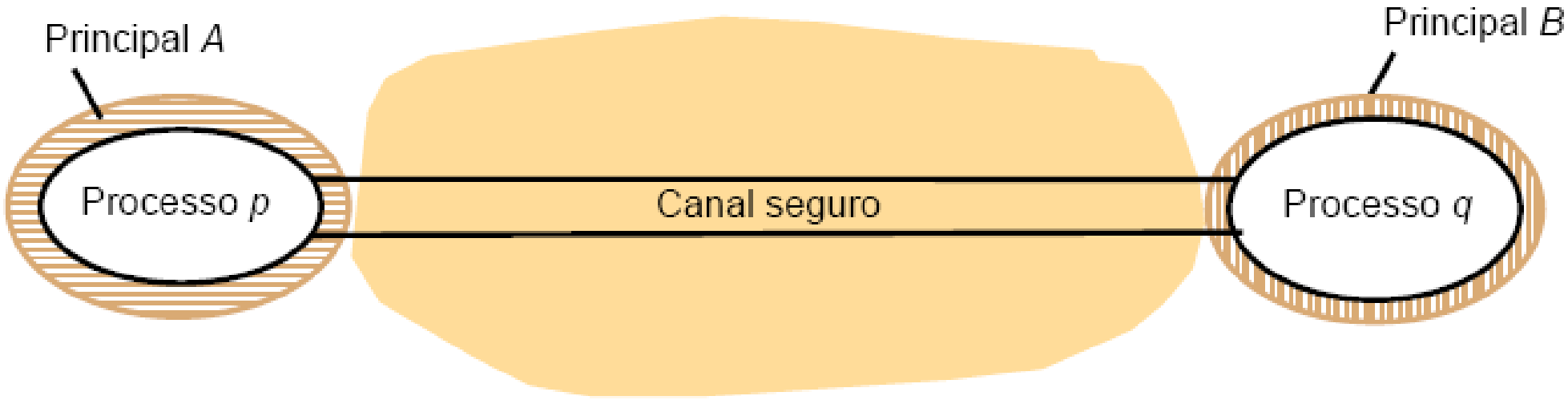
Segurança dos Processos e Interações



Comunicação Segura

- Principais mecanismos:
  - **Criptografia –** processo de **"embaralhamento"** de uma mensagem de modo a esconder seu conteúdo de usuários não autorizados
  - **Autenticação –** utiliza **chaves secretas e criptografia** para garantir a identidade dos processos clientes e servidores
  - **Canal seguro –** *canal de comunicação conectando um par* **de processos**, onde cada processo conhece e confia na identidade do principal em nome do qual o outro processo está executando
    - Geralmente implementado como uma **camada extra de serviço** sobre os serviços de comunicação existentes
    - Utiliza mecanismos de **autenticação e criptografia** para garantir a privacidade e a integridade das mensagens transmitidas através do canal
    - Também pode **garantir a entrega e ordem** de envio das mensagens
    - Ex.: VPN, SSL

Comunicação Segura



Exercícios

- Fazer todos os exercícios do capítulo 2 e entregar na próxima aula em modo manuscrito.