SSC150 – Sistemas Computacionais Distribuídos

Introdução aos Sistemas Distribuídos

1ª aula
04/03/10

Profa. Sarita Mazzini Bruschi
sarita@icmc.usp.br

Slides baseados no material de:
Prof. Rodrigo Mello (USP / ICMC)
Prof. Edmilson Marmo Moreira (UNIFEI / IESTI)

Motivação

- “A moda hoje entre os alunos de graduação em Ciência da Computação tem muito a ver com o aprendizado e exercício de aspectos fundamentais da Computação.
  É moda hoje em dia aprender Java, Corba e tudo sobre a Internet. Nossos alunos são inevitavelmente contaminados pela fase de realização tecnológica por que passa o mundo. Todos nós somos, de ma maneira ou de outra, afetados pelo que ocorre hoje em Computação e Tecnologia.
  Não há nada de errado com o aprendizado de tecnologias da moda. Porém, a mensagem desse curso é que fundamentos foram, são e serão importantes para o entendimento não somente da tecnologia da moda, mas também das que já foram moda e das que serão moda”

  Buzato, Escola Reginal de Informática, ICMC, 2001

  Curso de Algoritmos Distribuídos e Aplicações

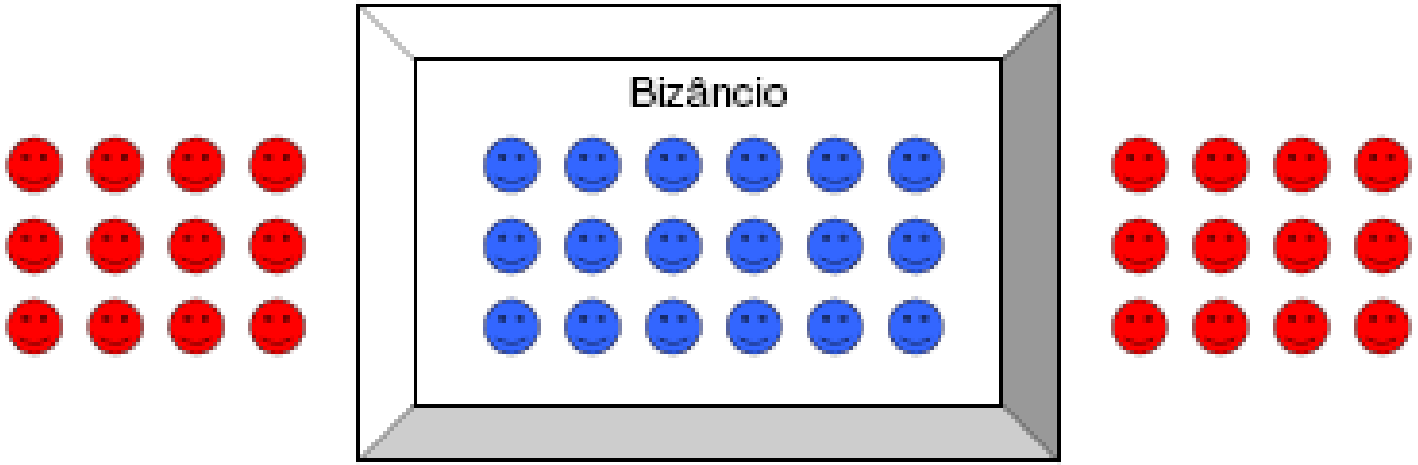
Motivação

- O Problema dos Generais Bizantinos
  - Artigo: “Bizantine General Problem”
    Lamport, Shostak, Pease
    ACM Transaction on Programming Language and Systems
    1982

O Problema dos Generais Bizantinos Problema com 2 Generais

- A origem desse problema remonta à época em que Bizâncio, hoje conhecida como Istambul, na Turquia, era uma das principais cidades na rota do comércio do Oriente. Naquela época, o domínio da cidade garantia o controle da rota de comércio terrestre entre o Oriente e o Ocidente. Portanto, Bizâncio era uma cidade bastante disputada.
- Historiadores comentam que em uma dessas disputas, Bizâncio, sob controle do Exército Azul, estava cercada por duas divisões do Exército vermelho. A batalha pelo domínio da cidade avançou para uma situação de impasse onde o Exército Vermelho venceria somente se suas divisões atacassem exatamente ao mesmo tempo.

O Problema dos Generais Bizantinos
Problema com 2 Generais



O Problema dos Generais Bizantinos
Problema com 2 Generais

- O problema do General Vermelho consistia em enviar mensageiros para o Capitão Vermelho, acampado do outro lado do campo de batalha, avisando o momento exato do ataque. O problema do General Azul era impedir que as duas divisões do Exército Vermelho combinassem o momento do ataque.
- Condições para o combate e vitória
  - A comunicação entre os militares dos exércitos ocorre somente via mensageiros.
  - O Exército Vermelho ganha se suas tropas atacarem exatamente ao mesmo tempo.
  - O Exército Azul ganha caso contrário.
- Mensageiros de uma divisão vermelha devem atravessar as cercanias de Bizâncio, que ainda são dominadas pelo Exército Azul, para chegarem à outra divisão e vice-versa.

O Problema dos Generais Bizantinos
Problema com 2 Generais

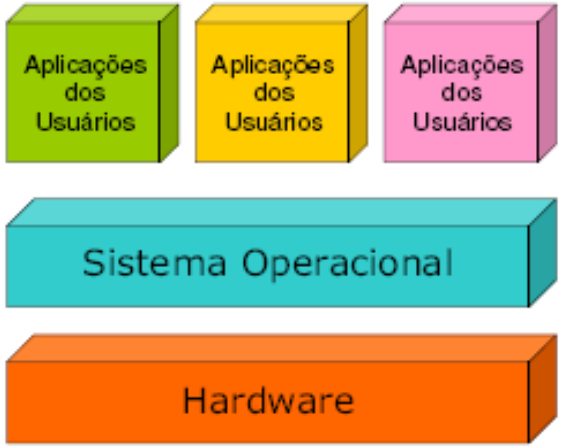
- Problemas que podem ocorrer na transmissão das mensagens:
  - Interceptação
  - Atraso
  - Confirmação de recepção
  - Sincronismo
  - Lealdade

O Problema dos Generais Bizantinos
Problema com 2 Generais

- Um número ilimitado de mensagens é necessário para que as duas divisões do Exército Vermelho cheguem a um consenso sobre o momento do ataque ou o ataque nunca acontece.
- Em resumo, é impossível resolver este problema satisfatoriamente sob as condições impostas.

Conceitos de Sistema Operacional

- Definição:
  - Camada de software colocada entre o hardware e os programas que executam as tarefas para os usuários



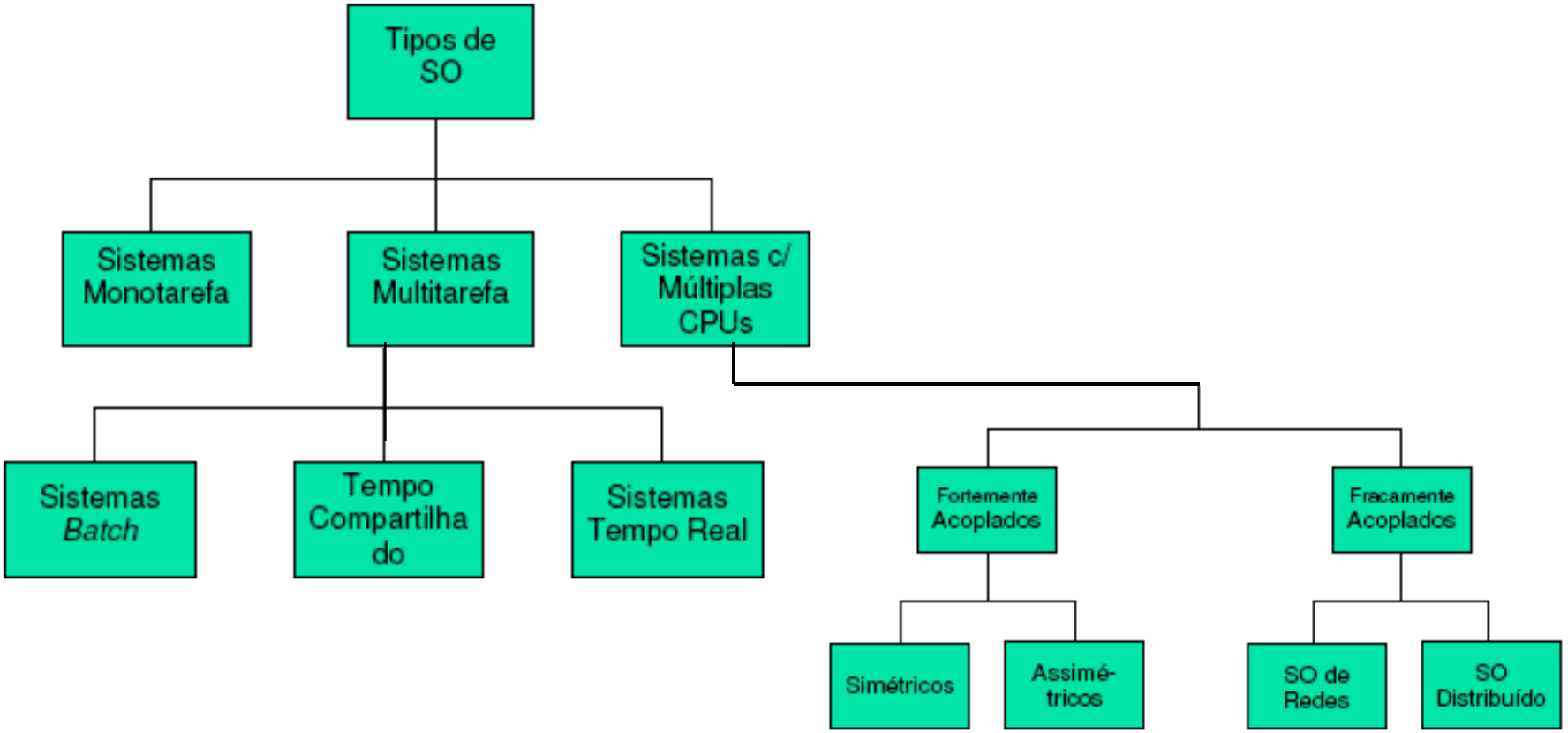
Conceitos de Sistema Operacional

- Segundo Tanenbaum, o Sistema Operacional realiza, basicamente, duas funções:
  - **estender a máquina** e
  - **gerenciar recursos**.
- Estas funções permitem que o computador seja utilizado com:
  - **Eficiência**
    - Distribui os recursos do sistema entre os programas. Por exemplo: tempo de processador, espaço na memória principal, impressora, espaço em disco, acesso a disco, etc.
    - Melhora a utilização dos recursos do sistema.
  - **Conveniência**
    - Omite dos usuários os detalhes de acesso ao hardware.
    - Diminui o tempo necessário para a construção de programas.

Conceitos de Sistema Operacional

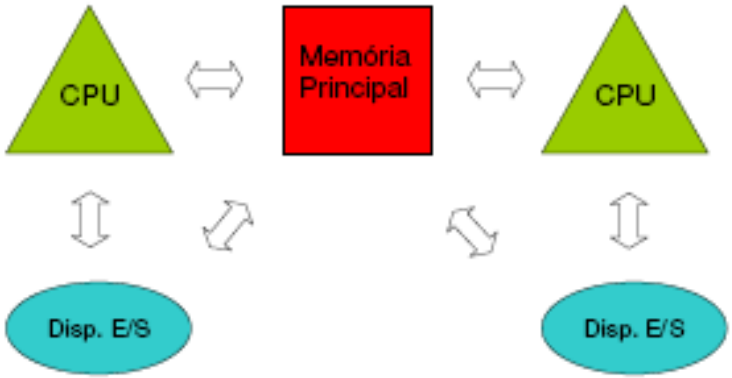
- Tipos de serviço oferecidos:
  - Sistema de arquivo
  - Gerência de memória
  - Gerência de processos
  - Gerência de E/S
  - Manutenção das informações do sistema
  - Segurança e detecção de erros

Classificação de Sistema Operacional



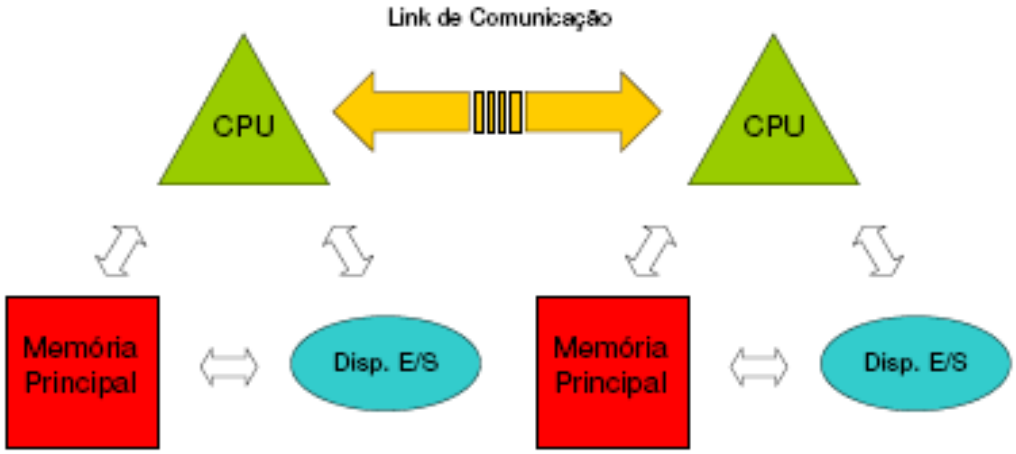
Classificação de Sistema Operacional

- Fortemente Acoplados:
  - Sistemas Timesharing Multiprocessadores
  - Compartilham uma única memória física e dispositivos de E/S



Classificação de Sistema Operacional

- Fracamente Acoplados:
  - Existência de dois ou mais sistemas computacionais conectados através de linhas de comunicação
  - Cada um trabalha de forma independente, possuindo seu próprio S.O. e gerenciando seus próprios recursos



Comparação entre 3 Sistemas Operacionais

	S.O. Rede	S.O. Distribuído	S.O. Multiprocessador
Se parece com um uniprocessador virtual?	Não	Sim	Sim
Tem o mesmo S.O?	Não	Sim	Sim
Número de cópidas de S.O.	N	N	1
Comunicação	Arquivos compartilhados	Mensagens	Memória Compartilhada
Uma única fila de execução?	Usualmente Não	Sim	Sim

Sistemas Distribuídos

- Definição:
  - Segundo Tanenbaum, um Sistema Distribuído é uma coleção de computadores independentes que aparenta aos usuários como se fosse um único computador
  - Dois aspectos da definição:
    - Hardware: as máquinas são autônomas
    - Software: os usuários vêem o sistema como uma única máquina
  - A definição de Tanenbaum é abrangente porém deve-se considerar também tolerância a falhas

Sistemas Distribuídos
Vantagens e Desvantagens

- Vantagens:
  - Extensibilidade:
    - O sistema pode crescer gradativamente
    - Novos softwares podem ser instalados gradativamente
    - Manutenção facilitada
  - Compartilhamento de Recursos
    - Recursos de alto custo podem ser melhor utilizados.
    - Servidores de arquivo.
    - Servidores de e-mail
    - Servidores correio
    - etc.

Sistemas Distribuídos
Vantagens e Desvantagens

- Vantagens (continuação)
  - Replicação
    - Diversas cópias de um mesmo arquivo podem ser armazenadas em locais distintos aumentando a confiabilidade.
  - Disponibilidade
    - Falha de um elemento não interrompe a operação do sistema global.
  - Mobilidade dos Usuários
    - Dependendo do modelo arquitetural usado, usuários ficam livres para acessar o sistema de diversos pontos.

Sistemas Distribuídos
Vantagens e Desvantagens

- Desvantagens:
  - Gerenciamento:
    - Dependendo do modelo usado, cada usuário pode necessitar ser um gerente;
    - Execução de backup é mais complexa;
    - Alocação de tempo de processamento;
    - Manutenção / evolução do software
  - Desempenho / Confiabilidade:
    - Dependência da rede utilizada

Sistemas Distribuídos
Vantagens e Desvantagens

- Desvantagens (continuação)
  - Segurança:
    - Para aumentar a flexibilidade do sistema existem clientes com acesso à comunicação básica e isto enfraquece a segurança;
    - Via de regra, qualquer informação colocada na rede deixa de ser “segredo”;
    - Existem mecanismos para melhorar a segurança.
  - Complexidade:
    - Conjunto de componentes tende a aumentar a dificuldade.

Sistemas Distribuídos
Características de Projeto

- Conceito chave: transparência
- A transparência pode ser entendida de duas formas:
  - Transparência de recursos físicos
    - Esconder dos usuários a localização dos diversos recursos
  - Transparência para os sitemas
    - A interface para as chamadas de sistema deve ser projetada considerando a existência de vários computadores que estão “visíveis” para os programas

Sistemas Distribuídos
Características de Projeto

- Tipos de transparência:
  - Transparência de Acesso: acesso a objetos locais ou remotos de maneira uniforme
  - Transparência de Localização: o usuário não deve se preocupar sobre onde estão os recursos de que necessita
  - Transparência de Falha: garante que em caso de falha de um subsistema, as aplicações continuam disponíveis sem interrupção
  - Transparência de Replicação: permite a duplicação de informação objetivando aumentar a disponibilidade e o desempenho do sistema, de forma sincronizada e consistente
  - Transparência no Desempenho: oferece aos usuários tempos de resposta independentes de alterações na estrutura do sistema ou na sua carga

Sistemas Distribuídos
Características de Projeto

- Tipos de Transparências (continuação):
  - Transparência de Concorrência: permite que vários processos sejam executados paralelamente
  - Transparência de Paralelismo: possibilita a execução de uma aplicação paralela como se fosse num sistema fortemente acoplado
  - Transparência de Migração: permite que os recursos sejam movidos para outro sistema sem afetar os usuários e as aplicações
  - Transparência de Escalabilidade: permite que o sistema cresça sem a necessidade de alterar aplicações e algoritmos

Sistemas Distribuídos
Características de Projeto

- Flexibilidade
  - É importante que o sistema seja flexível pois o desenvolvimento dos SD ainda está no princípio
  - Decisões de projeto que agora parecem razoáveis podem se mostrar erradas no futuro

Sistemas Distribuídos
Características de Projeto

- Confiabilidade
  - Idéia básica de que se uma máquina deixa de funcionar por algum motivo, outra máquina assume seus serviços
  - Para que um sistema seja confiável, deve prover *Tolerância a Falhas*
    - Hardware: fácil de se obter utilizando componentes redundantes, tais como RAID, vários processadores, memória com detecção e correção de erros, etc.
    - Software: replicação, porém, quanto maior o número de cópias de software, maior a probabilidade de inconsistência

Sistemas Distribuídos
Características de Projeto

- Desempenho
  - Uma aplicação executada em um Sistema Distribuído não deve ter um desempenho pior do que quando executada num sistema centralizado
  - Medidas de desempenho:
    - Tempo de execução;
    - Throughput (número de aplicações por unidade de tempo)
    - Capacidade consumida da rede

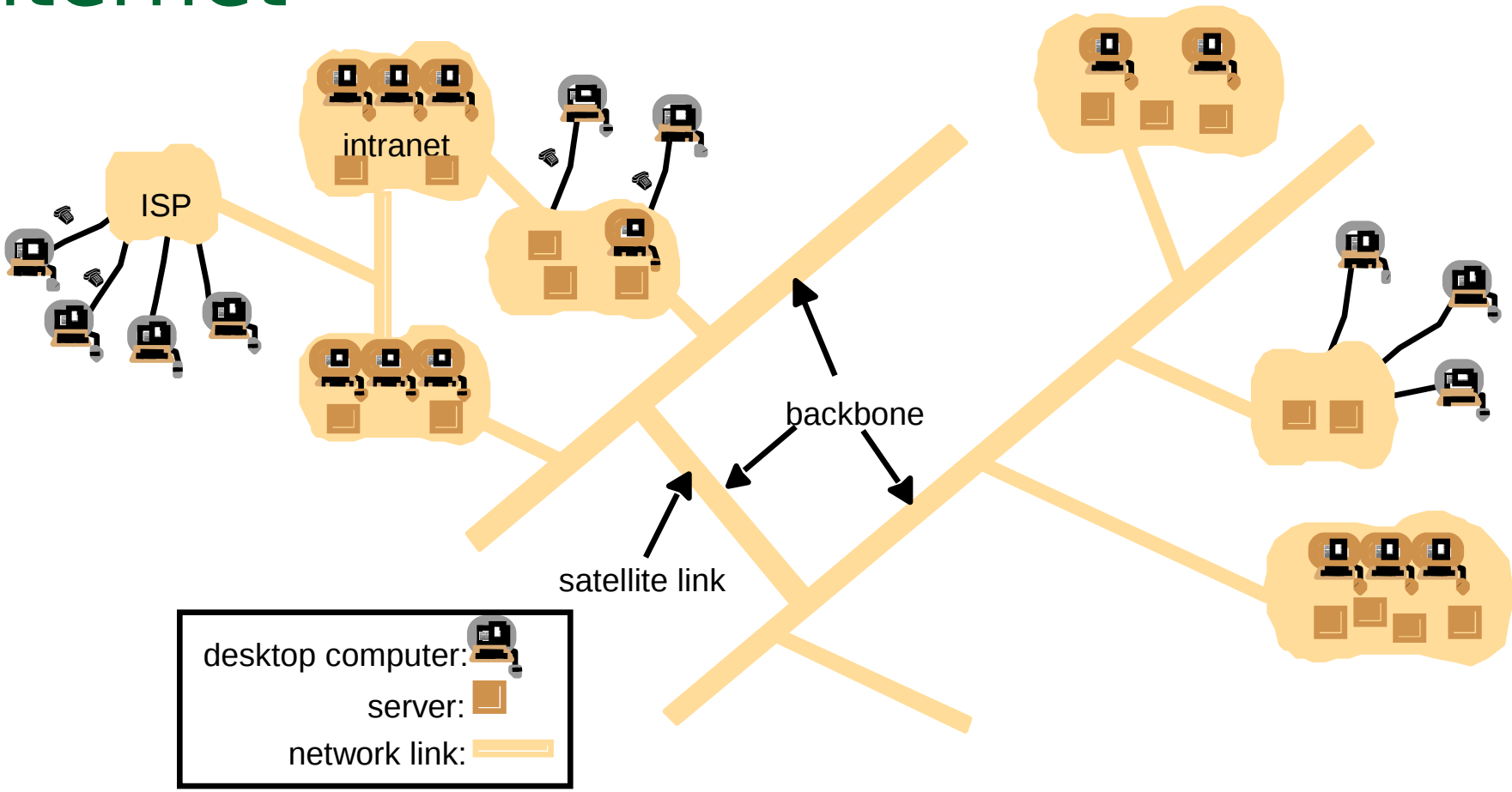
Sistemas Distribuídos
Características de Projeto

- Desempenho
  - Comunicação é relativamente lenta
  - Uma boa alternativa para se conseguir um bom desempenho é desenvolver aplicações com granulosidade grossa (grandes computações, poucas interações, poucos dados) ao invés de granulosidade fina (grande número de pequenas computações, altamente interativas umas com as outras)

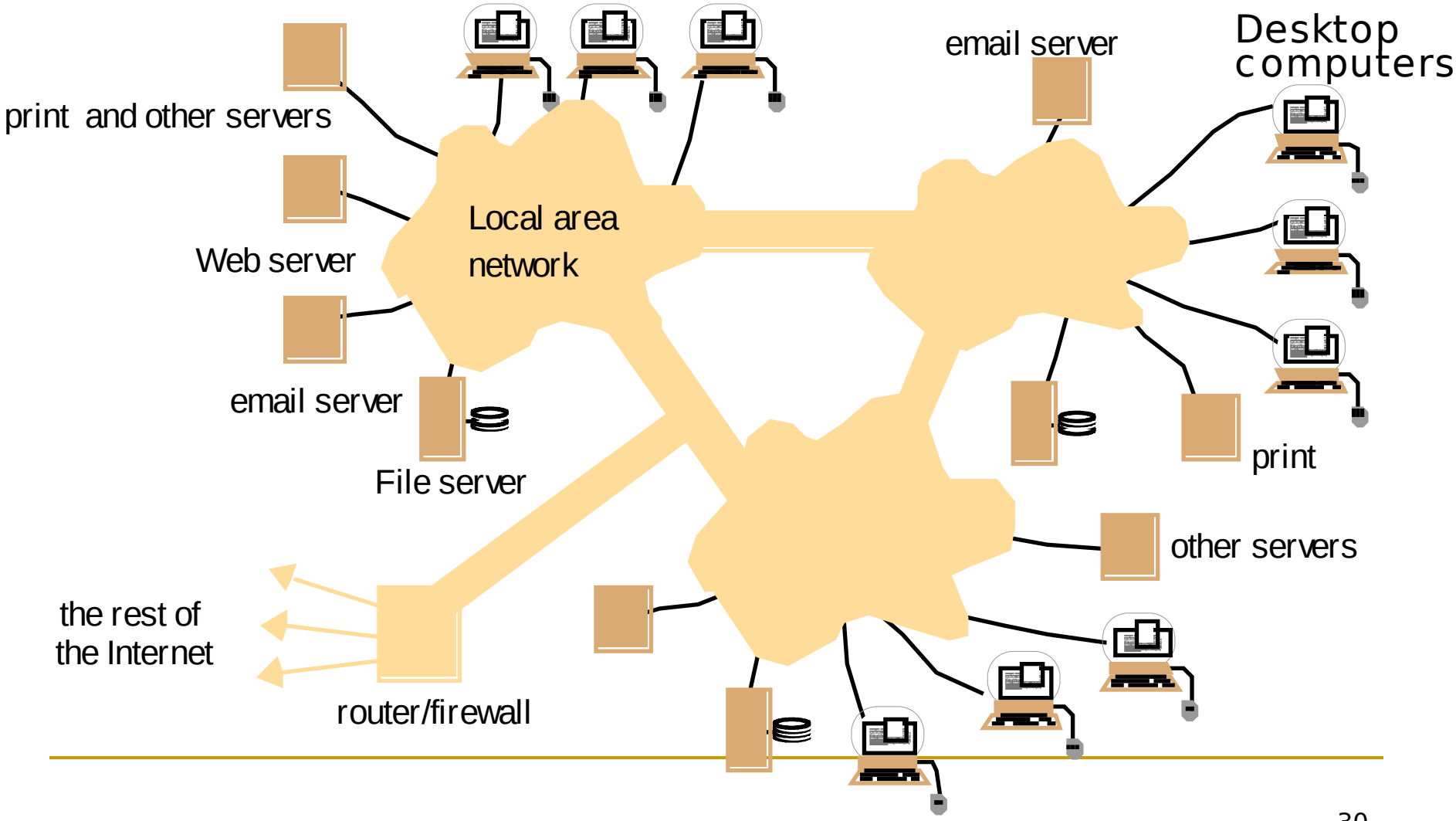
Sistemas Distribuídos
Características de Projeto

- Escalabilidade
  - Alguns sistemas podem funcionar bem com algumas centenas de processadores mas falharem com milhares de processadores
  - Os sistemas devem ser projetados para permitir que novos componentes ou CPUs sejam adicionadas ao sistema causando um impacto proporcional no desempenho. Em particular, se deseja que o desempenho do sistema melhore e não o contrário.
  - A escalabilidade pode se tornar um problema pois muitos recursos não podem ser replicados, ou não foram projetados considerando esta característica.

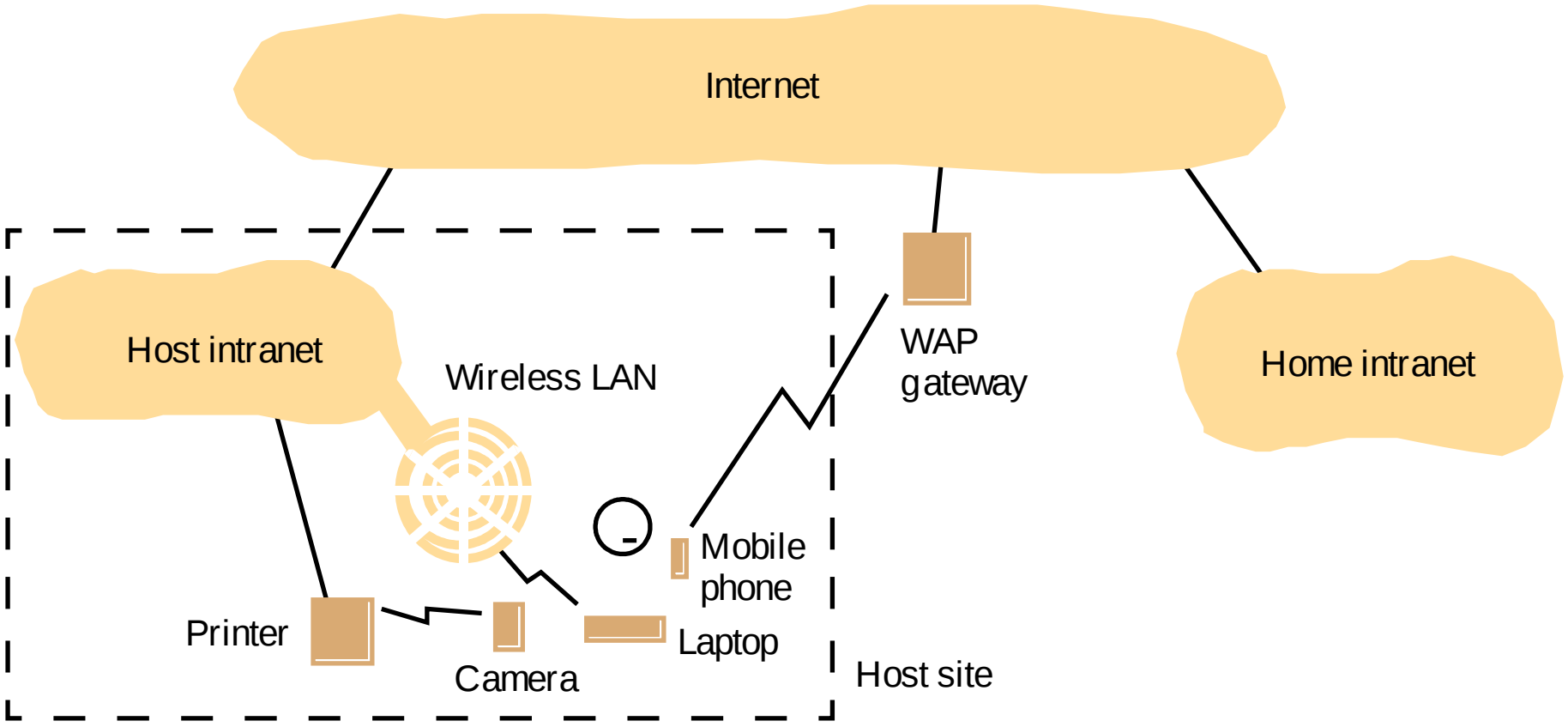
Exemplos de Sistemas Distribuídos
Internet



Exemplos de Sistemas Distribuídos Intranet



Exemplos de Sistemas Distribuídos
Dispositivos portáteis e de mão



SSC150 – Sistemas Computacionais Distribuídos

Modelos Arquiteturais

Alguns desafios de Sistemas Distribuídos

- Grande variedade de modelos para se usar
- Grande variedade de ambientes
- Problemas internos
  - Relógio
  - Atualizações
  - Falhas
- Problemas externos
  - Ataques

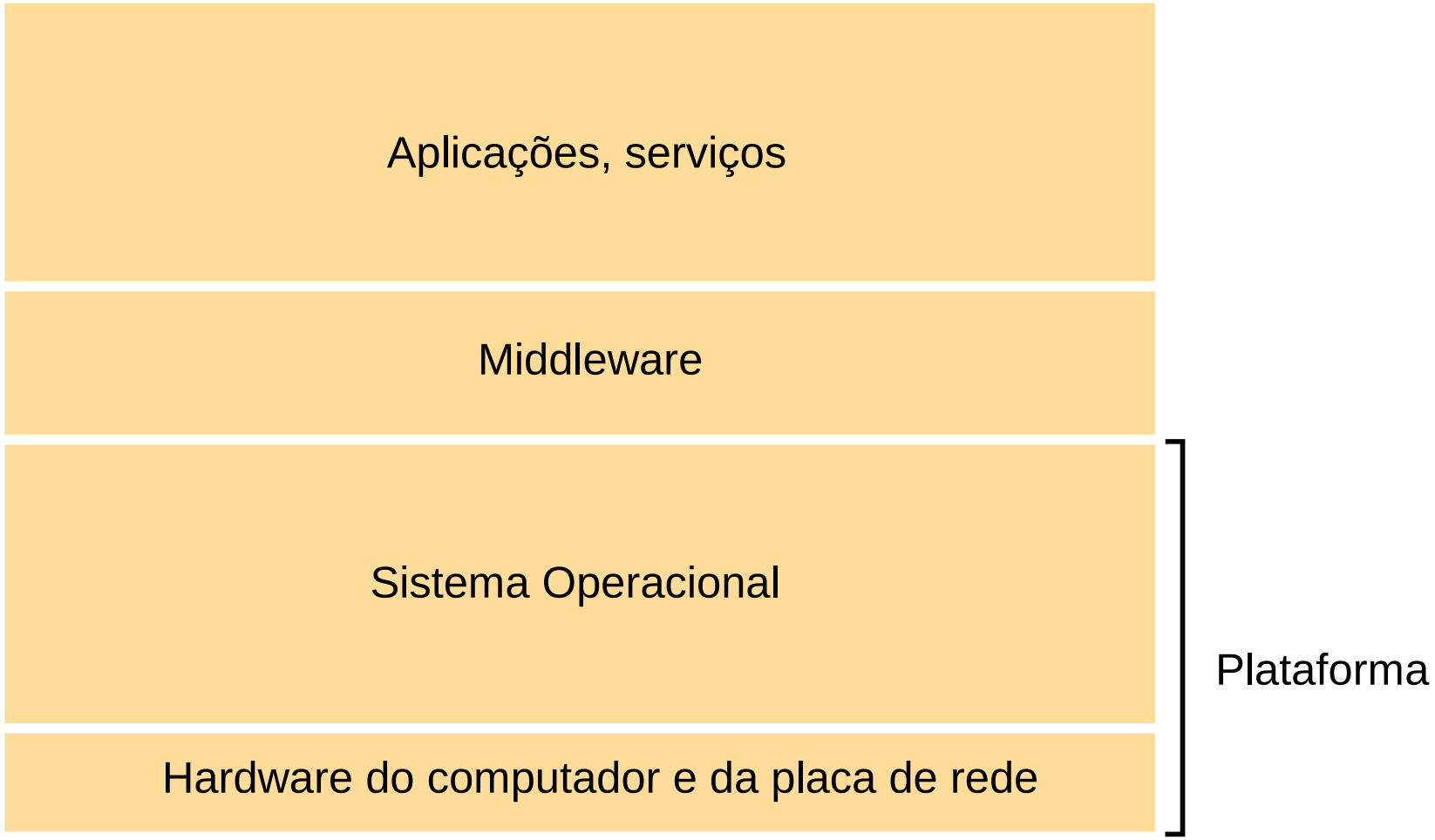
Modelos Arquiteturais

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

Modelos Arquiteturais

- Processos
  - Servidor
  - Cliente
  - Par (*Peer*)
- Identificação da responsabilidade de cada um

Modelo em camadas



Modelo 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 em camadas

- *Middleware*
  - Camada de software que tem por objetivo mascarar a heterogeneidade e fornecer um modelo de programação conveniente para os programadores das aplicações
  - 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

Modelo em camadas

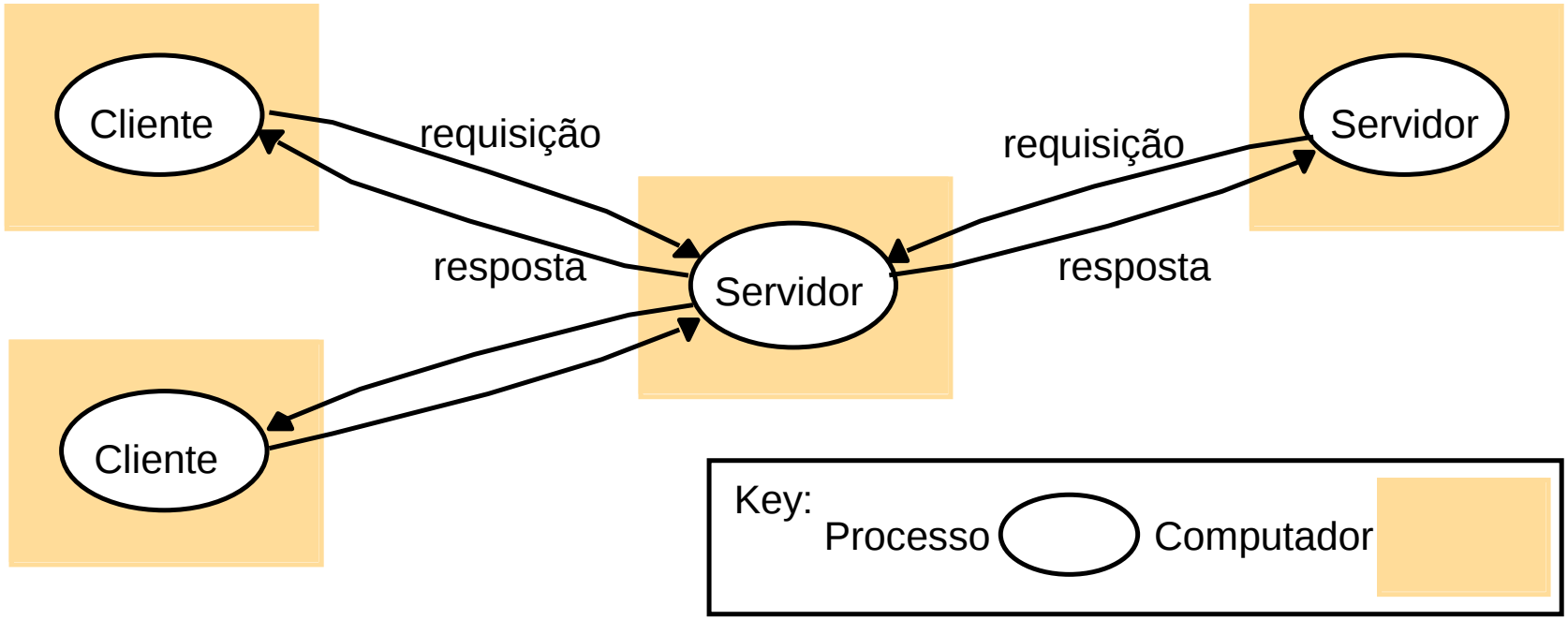
- *Middleware*
  - Simplifica as atividades de comunicação de programas aplicativos por meio do suporte de abstrações como por exemplo:
    - Invocação de métodos remotos
    - Comunicação entre grupo de processos
    - Notificação de eventos
    - Particionamento, posicionamento e recuperação de objetos de dados compartilhados entre computadores
    - Replicação de objetos de dados compartilhados
    - Transmissão de dados multimídia em tempo real

Modelo em camadas

- *Middleware*
  - Exemplos:
    - CORBA;
    - RMI Java;
    - Serviços Web (*Web Services*)
    - DCOM (*Distributed Component Object Model*) da Microsoft
    - RM-ODP (*Reference Model for Open Distributed Processing*) do ISO/ITU-T

Arquiteturas de sistema Cliente/Servidor

- Processos (servidor e cliente) com funções diferentes
- Baseada em Requisição/Resposta (*Request*/*Reply*)
- Servidores podem ser clientes de outros servidores

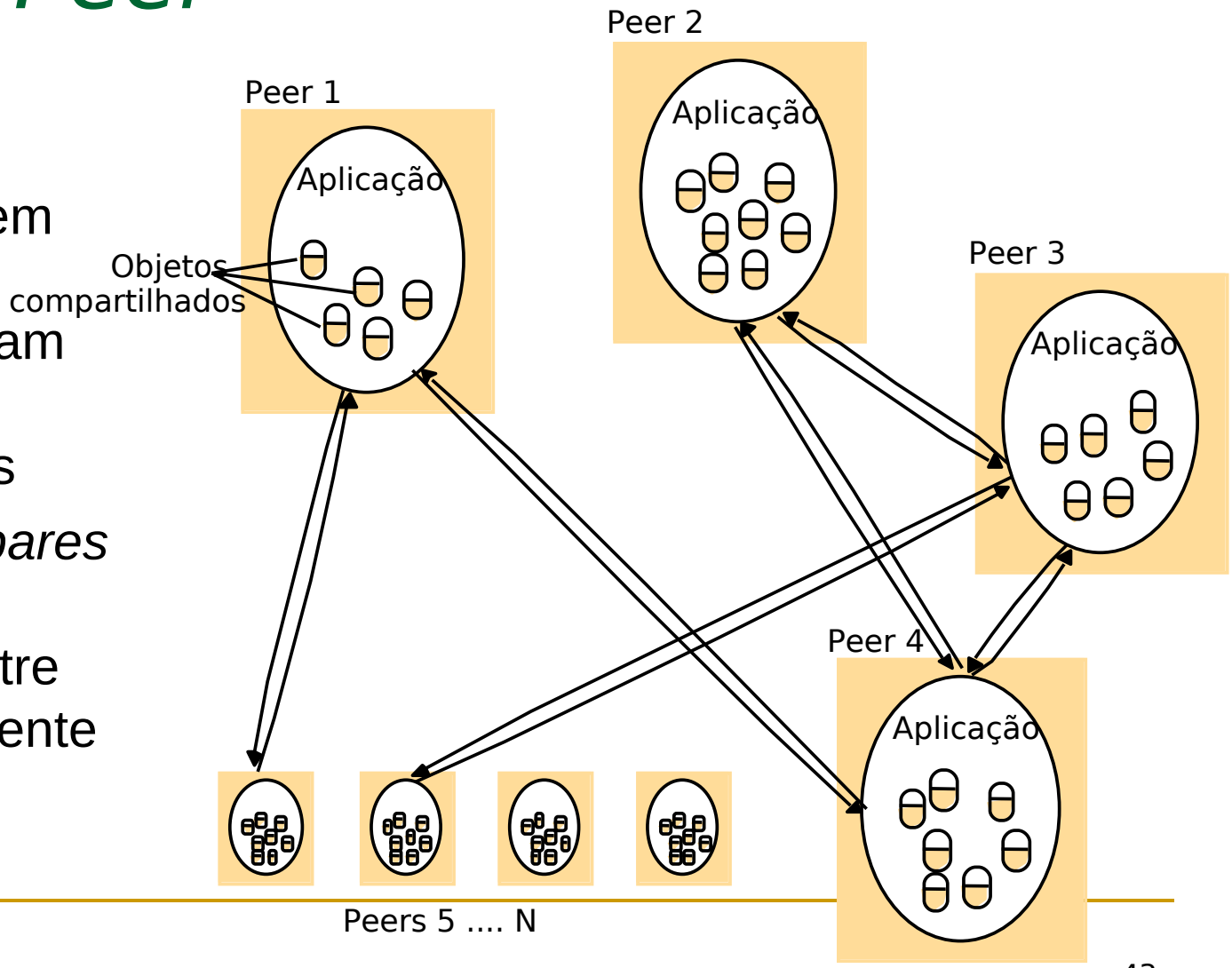


Arquitetura Cliente/Servidor Exemplo

- Servidor Web
  - Servidor de navegadores (*browsers*)
  - Cliente de um servidor de arquivos local que gerencia os arquivos das páginas Web
  - Cliente de serviço de DNS (*Domain Name Server*), que mapeia nomes do domínio Internet a endereços de rede (IP)

Arquiteturas de sistema *Peer-to-Peer*

- Todos os processos envolvidos em uma tarefa desempenham funções semelhantes
- Processos *pares* (*peer*), sem distinção entre processo cliente ou servidor



Arquitetura Peer-to-Peer Exemplo

- Skype
  - Diretório de usuários descentralizado global
    - Normalmente: diretório central com registro do nome do usuário e IP onde esse usuário está conectado
    - Tecnologia de Índice Global (GI)
      - Rede de diversas camadas
      - Supernós se comunicam de maneira que cada nó da rede tem conhecimento total de todos os usuários e recursos disponíveis
  - Problema ocorrido em 16/08/07:
    - http://blogs.technet.com/fcima/archive/2007/08/25/skype-

Modelos fundamentais

- Independente de ser cliente/servidor ou *peer-to-peer*, todos os modelos possuem características comuns:
  - São constituídos de processos
  - Esses processos se comunicam através do envio de mensagens através de uma rede de comunicação

Modelos fundamentais

- Um modelo de sistema deve tratar as seguintes questões:
  - Quais são as principais entidades presentes no sistema?
  - Como elas interagem?
  - Quais são as características que afetam seu comportamento individual e coletivo?

Modelos fundamentais

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

Modelos de Interação

- Os sistemas distribuídos são compostos de muitos processos que interagem através de troca de mensagens
- Cada processo possui seu próprio estado que só pode ser acessado através de uma interface pública
- Dois fatores significantes que afetam a interação entre os processos são:
  - O desempenho da comunicação é normalmente um fator limitante
  - Não existe a noção de tempo global

Modelos de Interação

- Desempenho dos canais de comunicação:
  - Latência
    - Atraso entre o início da transmissão da mensagem para um processo e o início da recepção dessa mensagem pelo outro processo
    - Inclui:
      - Tempo de transmissão
      - Atraso para acessar a rede
      - Tempo de processamento do sistema operacional

Modelos de Interação

- Desempenho dos canais de comunicação:
  - Largura de banda (*bandwidth*)
    - Quantidade total de dados que podem ser transmitidos em um dado tempo
  - *Jitter*
    - Variação do tempo que se leva para entregar uma série de mensagens (muito importante para dados de tempo real)

Modelos de Interação

- Temporização
  - Todos os computadores possuem um relógio interno
  - Os processos podem anexar “marcas de tempo” (*timestamps*), mas isso possui uso limitado quando os processos estão em máquinas diferentes
  - Taxa de variação do *clock* é a quantidade de tempo que um computador varia em relação a um tempo de referência

Modelos de Interação

- Sincronização
  - Sistemas distribuídos síncronos definem limites
    - O tempo de execução de cada passo de um processo possui limites inferiores e superiores
    - Cada mensagem transmitida sobre um canal é recebida em um tempo limite conhecido
    - Cada processo possui um relógio local cuja variação em relação ao tempo real é conhecida
  - Definir esses limites pode ser difícil, mas os benefícios são muitos

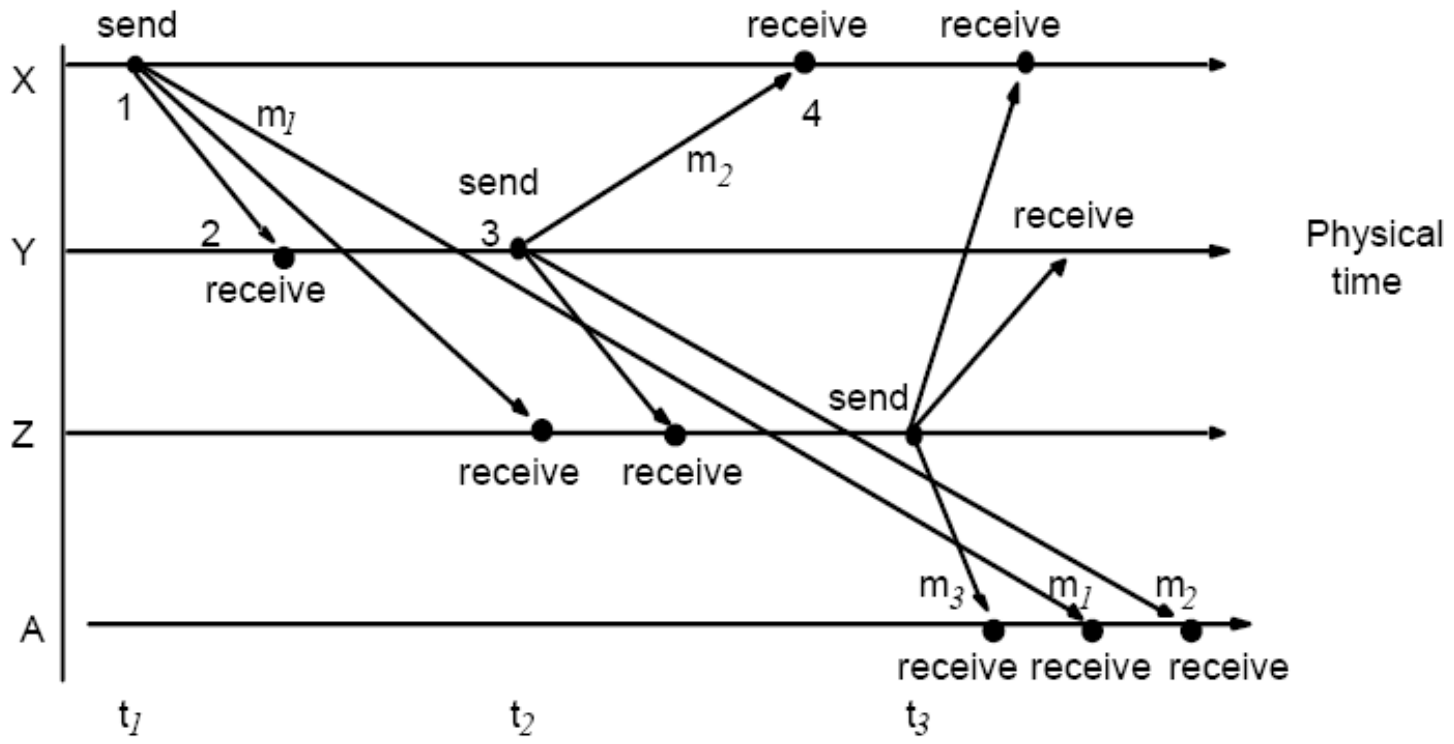
Modelos de Interação

- Sincronização
  - Sistemas distribuídos assíncronos não possuem limites em relação a:
    - Velocidade de execução do processo
    - Atrasos na transmissão da mensagem
    - Taxas de variação do relógio
  - A Internet é o melhor exemplo de um sistema assíncrono
  - Impossível de se sincronizar os relógios

Modelos de Interação

- Ordenação dos eventos
  - Interesse em saber a ordem dos eventos ao invés do tempo exato de acontecimento deles
    - Antes
    - Depois
    - Concorrentemente
  - Considere um e-mail:
    - X envia uma mensagem com o *subject*: Reunião
    - Tanto Y como Z respondem com o *subject*: Re: Reunião
    - Usuário A pode receber as mensagens na ordem errada

Modelos de Interação



Modelos de Interação

- Se não é possível sincronizar os relógios, então Lamport definiu o *tempo lógico* – a ordem dos eventos pode ser inferida sem referência ao tempo real

Modelo de Falha

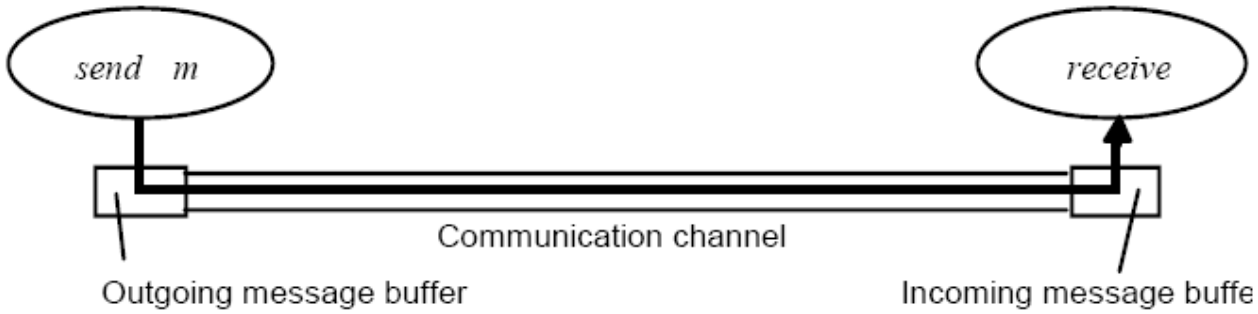
- Processos podem falhar
- Canais de comunicação podem falhar
- O modelo de falha define de que maneira as falhas podem ocorrer para que se tenha um entendimento dos efeitos da falha

Modelo de Falha

- Falhas por omissão:
  - Processos ou canais de comunicação deixam de executar as ações que deveriam
  - Falhas por omissão de processo:
    - Programa travou
    - Detectado através de *timeouts*
      - Pode indicar somente que o sistema está lento caso seja um sistema distribuído assíncrono

Modelo de Falha

- Falhas por omissão:
  - Falha por omissão do canal de comunicação
    - O canal de comunicação produzirá uma omissão por falha se a mensagem não for transportada de um *buffer* para outro



Modelo de Falha

- Falha arbitrária ou fala bizantina:
  - Pior semântica de falha possível
  - Omite arbitrariamente passos desejados do processamento ou efetua processamento indesejado
  - Exemplos:
    - Conteúdo da mensagem pode ser corrompido
    - Mensagens inexistentes podem ser enviadas
    - Mensagens reais podem ser entregues mais de uma vez

Modelo de Falha – Classes de Falha
Falhas por omissão e arbitrárias

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process / channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Modelo de Falha – Classes de Falha
Falhas de temporização

- Aplicadas a sistemas distribuídos síncronos

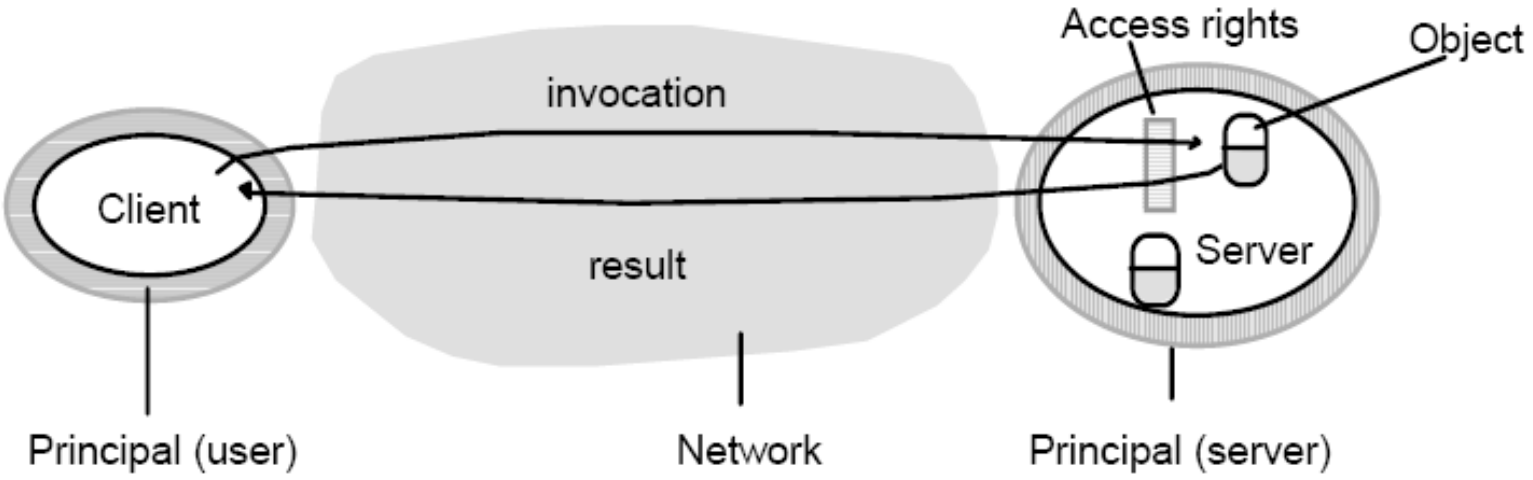
Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

Modelo de Segurança

- A segurança em um sistema distribuído pode ser obtida tornando seguros os processos e os canais de comunicação usados por suas interações e protegendo contra acesso não autorizado os objetos que encapsulam

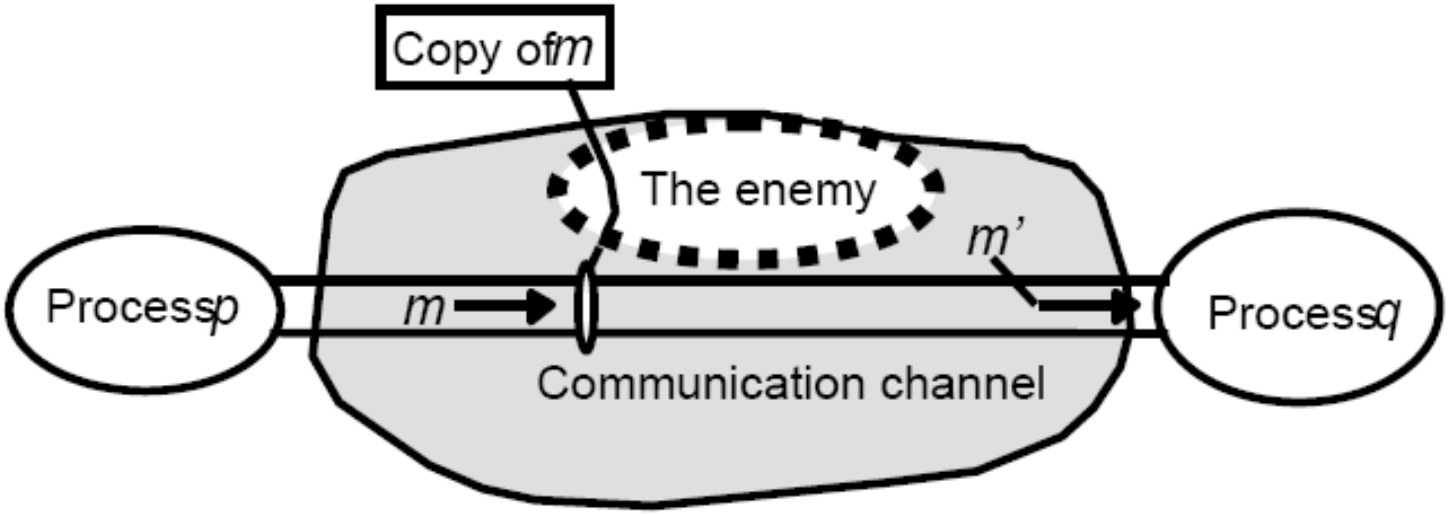
Modelo de Segurança

- Proteção de objetos
  - Direitos de acesso



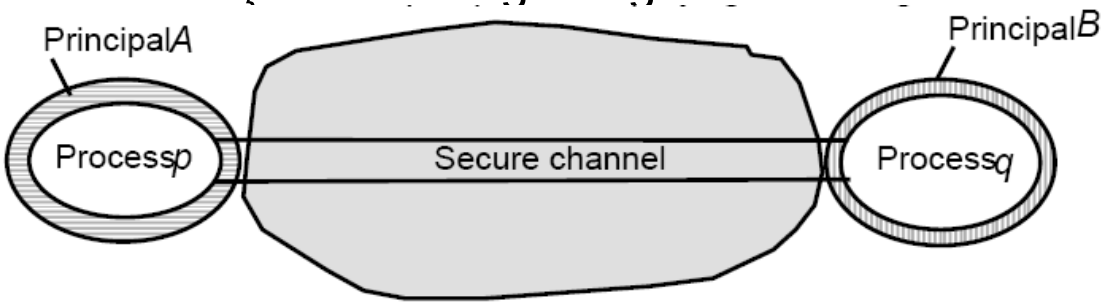
Modelo de Segurança

- Invasor
  - Capaz de enviar qualquer mensagem ou ler ou copiar qualquer mensagem entre dois processos



Modelo de Segurança

- Mecanismos de defesa
  - Criptografia
  - Autenticação
  - Um canal seguro possui as seguintes propriedades:
    - Cada processo conhece a identidade do principal em nome de quem o outro está executando
    - Garante a privacidade e a integridade
    - Inclui indicação de relógio lógico



Exercícios 1)

- Considere um servidor simples que responde a requisições dos clientes sem acessar outros servidores
- Explique porque normalmente não é possível estabelecer um limite do tempo que se leva para que tal servidor envie a resposta ao cliente
- O que precisaria ser feito para fazer com que o servidor seja capaz de executar as requisições em um tempo limite? Isso é uma opção prática?

Exercícios
2)

- Considere dois serviços de comunicação para uso em um sistema distribuído assíncrono. No serviço A, as mensagens podem ser perdidas, duplicadas ou chegar atrasadas e os *checksums* são aplicados somente ao cabeçalho. No serviço B, as mensagens pode ser perdidas, atrasadas ou entregues tão rapidamente que o receptor não consegue manipulá-las, mas elas são entregues em ordem e com o conteúdo correto.
- Descreva as classes de falhas de cada um dos serviços
- O serviço B pode ser descrito como um serviço de comunicação confiável?