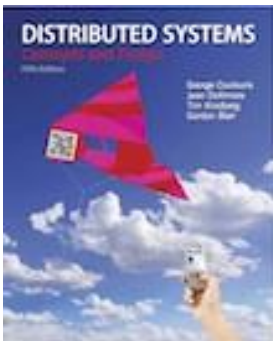


# Slides for Chapter 2: Architectural Models

---



From Coulouris, Dollimore, Kindberg and Blair  
**Distributed Systems:  
Concepts and Design**  
Edition 5, © Addison-Wesley 2012

# Dificuldades e ameaças para sistemas distribuídos

## *Pra começo de conversa*

- **Variados modos de uso**
  - Sujeitos à ampla variação de carga de trabalho (ex: acessos a servidores WEB,).
  - Conexão intermitente de dispositivos móveis
  - Requisitos de QoS
- **Variedade de ambientes de sistema**
  - Além de HW, SO heterogêneos
  - Taxas heterogêneas de redes de comunicação
  - SD de diferentes escalas
- **Problemas internos**
  - Relógios não sincronizados
  - Atualização conflitante de dados
  - Diferentes modos de falhas de HW e de SW
- **Ameaças externas**
  - Ataques à integridade e ao sigilo dos dados
  - Negação de serviço

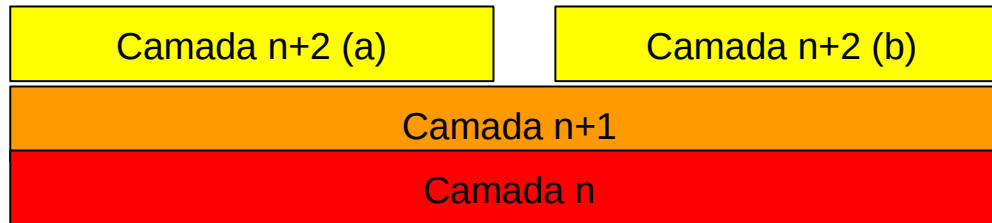
# Tipos de Modelos

- **Modelos Físicos**
  - Descrevem a composição de hardware de um sistema, computadores ou outros dispositivos computacionais, como dispositivos móveis, e suas redes de interconexões.
- **Modelos de Arquitetura**
  - Descrevem o sistema em termos das tarefas computacionais e de comunicação realizadas por seus elementos computacionais, ou seja, pelos seus computadores individuais ou clusters portados pelas interconexões de rede apropriadas
- **Modelos fundamentais**
  - De uma perspectiva abstrata, examinam os aspectos individuais de um sistema distribuído.
    - **Modelos de interação**, que consideram a estrutura e a ordenação da comunicação entre os elementos do sistema
    - **Modelos de falha**, que consideram as maneiras pelas quais um sistema pode deixar de funcionar corretamente.
    - **Modelos de segurança**, que consideram como o sistema está protegido contra tentativas de interferência em seu funcionamento correto ou de roubo de seus dados.

# Modelos de Arquitetura

A arquitetura de um sistema é sua estrutura em termos de componentes especificados separadamente e de suas relações:

## *Interface de serviços*



Uma arquitetura bem projetada deve garantir que a estrutura atenda as demandas atuais e, provavelmente, as que forem impostas futuramente.

Originalmente tratava-se de uma visão em Camadas de Software, hoje a interface de serviços tem destaque.

# Modelos de Arquitetura

- **O Middleware não é mágico**

- Nem toda a abstração deve ser aceita como suficiente! Ou seja, a aplicação deve se precaver de falhas.
- Exemplo: para um sistema de emails, um mecanismo básico de TCP pode resolver a comunicação. E se for transferido, como anexo, um arquivo muito grande? Podem haver falhas? TCP tem um nível de tolerância, mas é razoável considerá-la também na camada aplicativo.

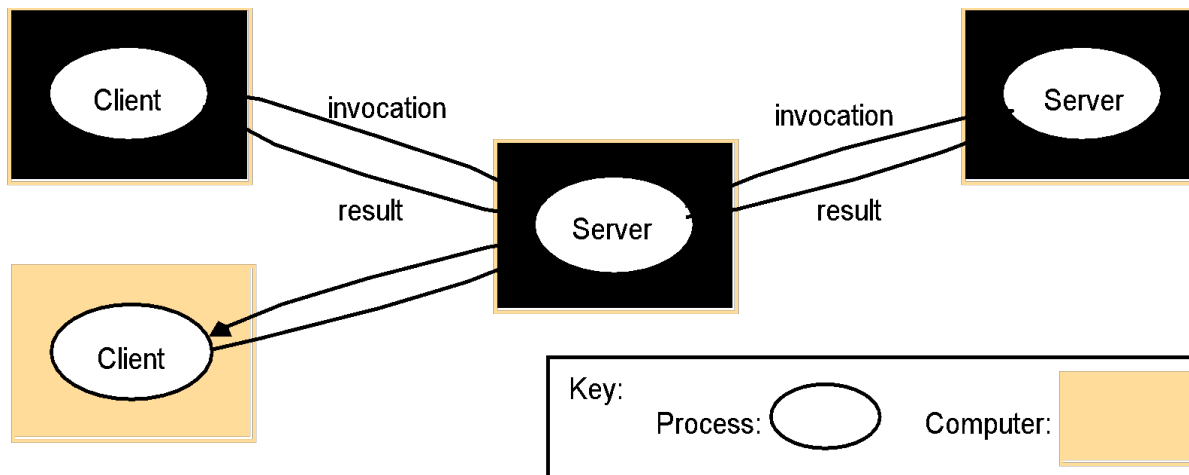
*“Algumas funções relacionadas à comunicação podem ser completa e corretamente implementadas apenas com o conhecimento e a ajuda do aplicativo que está nos pontos extremos de um sistema de comunicação. Portanto, fornecer uma função como um recurso próprio de um sistema de comunicação nem sempre é sensato.” [Saltzer et al. 1984]*

- O ponto principal é que o comportamento correto em programas distribuídos depende de verificações, mecanismos de correção de erro e medidas de segurança em muitos níveis.

# Arquitetura Cliente-Servidor

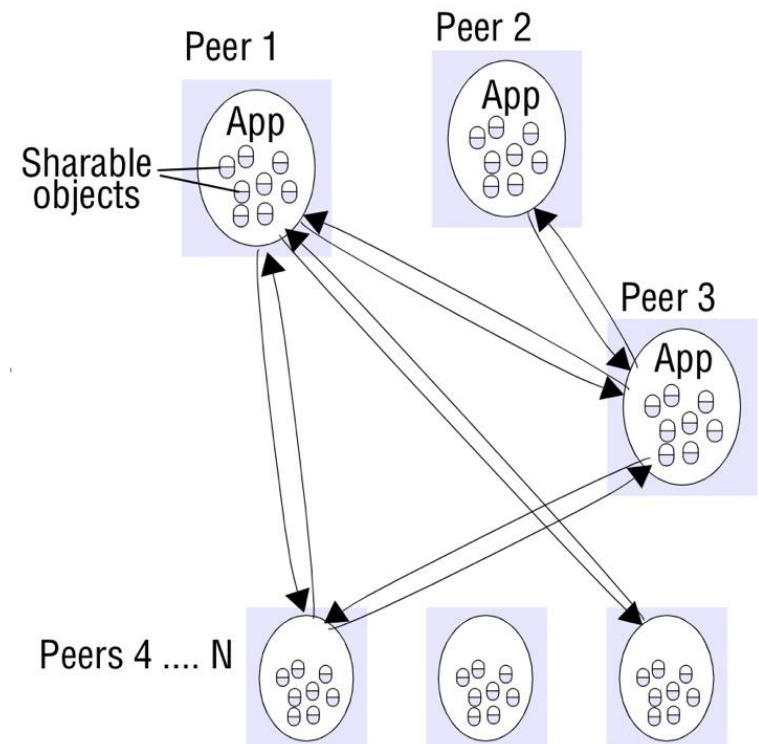
- **Papéis claros**

- Clientes requisitam serviços e/ou recursos oferecidos/compartilhados pelo(s) Servidor(es)
- Um Servidor pode ser cliente de outro(s) Servidor(es)



# Arquitetura Peer-to-peer

- **Mesmos papéis**
  - Os pares executam o mesmo programa e oferecem a mesma interface de serviços
  - O segredo do sucesso: a aplicação oferecida dá suporte ao próprio mecanismo de pares
- **Funciona, pois**
  - A tecnologia atual permite ter computadores pessoais potentes, capazes de alta carga de processamento, ligados em redes de comunicação suficientemente rápidas.



# O problema da alocação dos serviços

---

Mapeamento dos serviços sobre os recursos físicos. Deve considerar as características da aplicação:

- Padrão de comunicação
- Confiabilidade
- Qualidade de comunicação
- Carga do nó

Estratégias básicas:

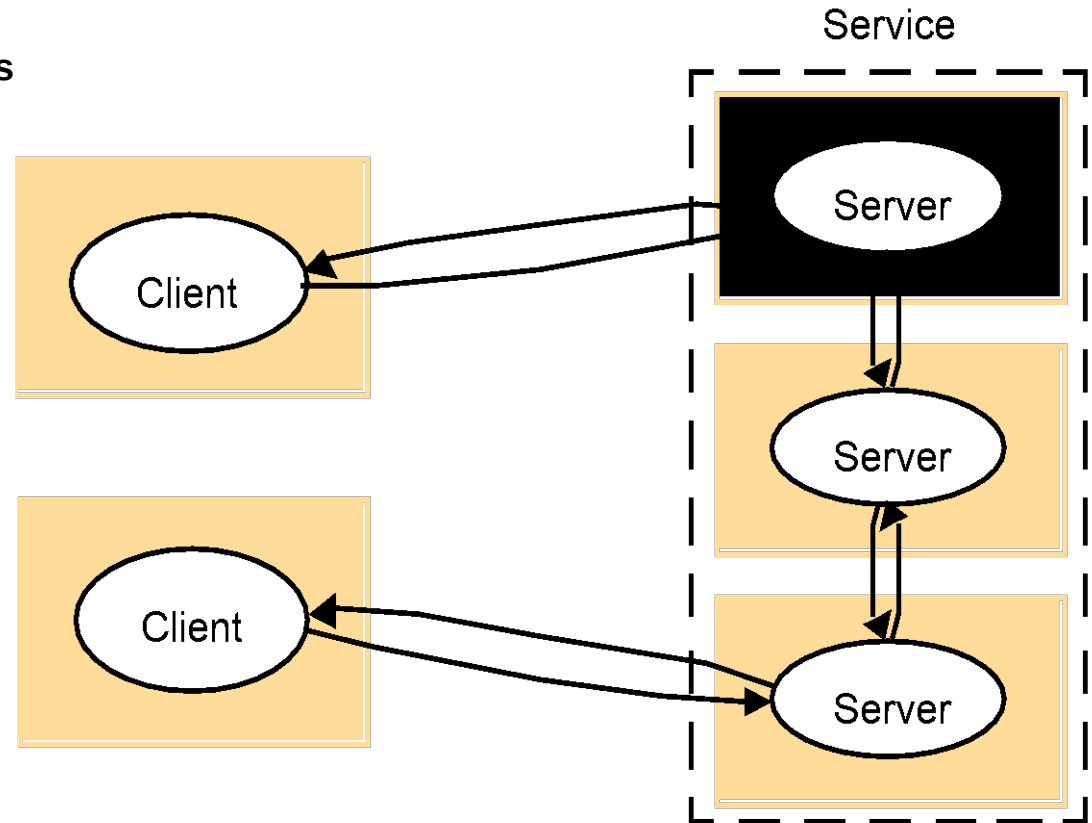
- Servidores múltiplos
- Caching
- Código móvel
- Agentes móveis



## O problema da alocação dos serviços

### Múltiplos servidores

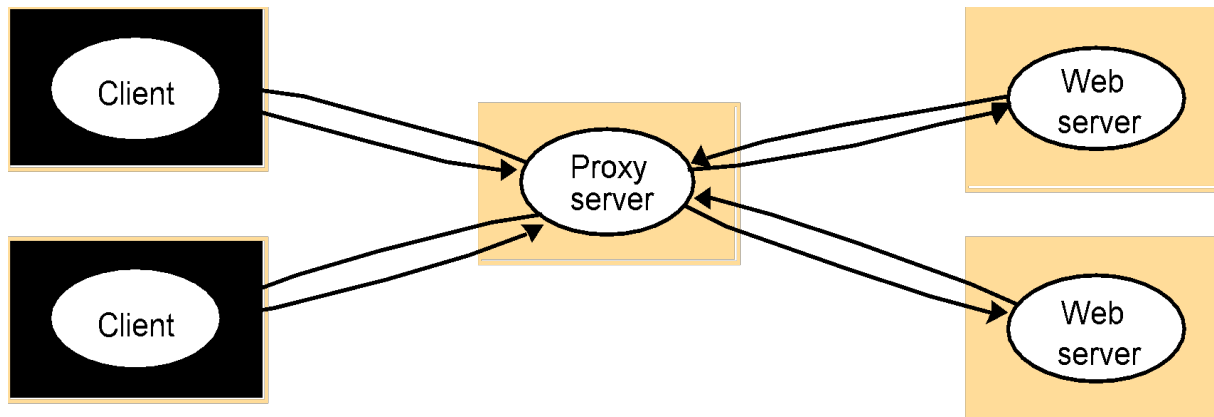
Os servidores interagem para atender os clientes. Os serviços podem ser replicados ou cada servidor pode oferecer um subconjunto de serviços.



## O problema da alocação dos serviços

### Caching

Permite que dados sejam armazenados próximos aos clientes, como o serviço de proxy da figura.

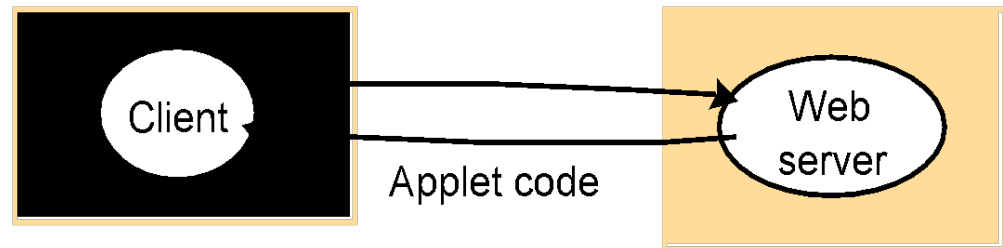


# O problema da alocação dos serviços

## Código móvel

Oferecem uma maior responsividade na interação com o usuário. Exemplo da figura: um applet é baixado e executado localmente, conversando com o servidor.

a) client request results in the downloading of applet code



b) client interacts with the applet



# Padrões arquiteturais

---

Define o modelo de organização da arquitetura de u,m SD

- **Múltiplas camadas**
- ***Tiered***
  - ***2-Tiers***
  - ***3-Tiers***
- ***Thin clients***
- ***Proxy***
- ***Brokerage***
- ***Reflexivo***

# Padrões Arquiteturais

- **Múltiplas camadas**
  - Organização vertical onde uma camada abstrai as inferiores
  - Cada camada oferece um serviço para a camada superior e utiliza os serviços da camada inferior
  - **Plataforma**
    - Camadas de HW e SW de mais baixo nível que fornecem serviços para as camadas superiores oferecendo interfaces de programação do sistema para facilitar a comunicação e a coordenação entre processos. Exemplo: X86/Windows, x86/Linux...
  - **Middleware**
    - Camada de SW cujo objetivo é o de mascarar a heterogeneidade e fornecer um modelo de programação conveniente.
    - Fornece blocos básicos para construção de componentes de SW colaborativos.
    - Simplifica o processo de construção de SW oferecendo camadas de abstração
    - Exemplos:
      - RPC, RMI, Corba, DCom ....

# Padrões Arquiteturais

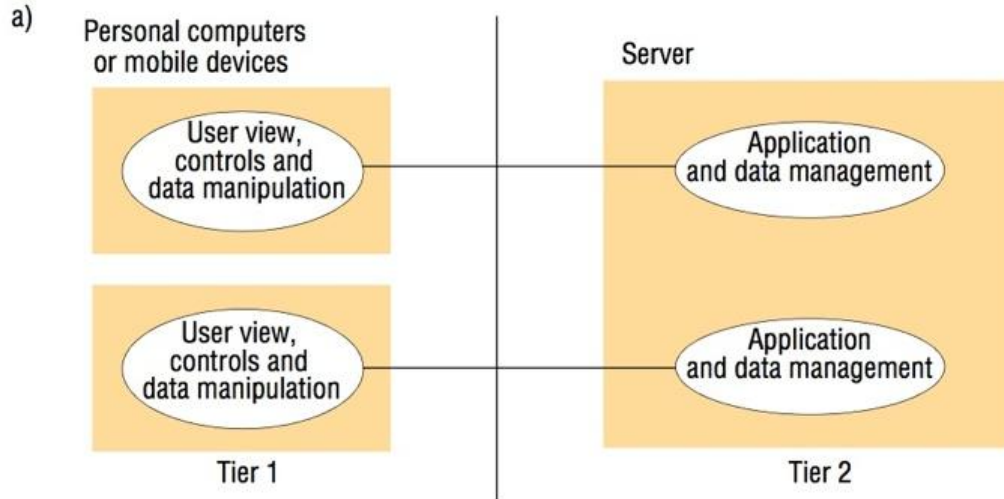
- Múltiplas camadas



# Padrões Arquiteturais

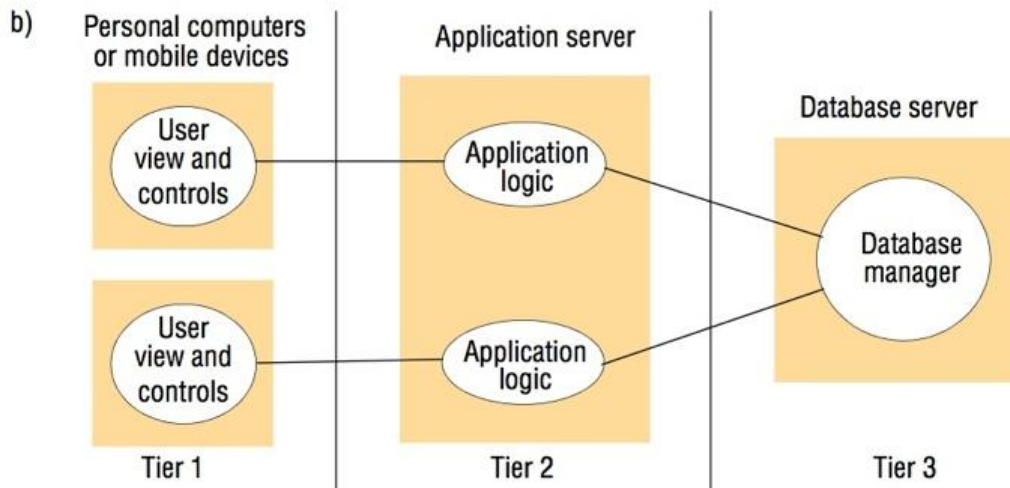
- **Tiered**
  - Organização complementar à arquitetura de camadas, dando papéis (funcionalidades) a serem cumpridos pelas camadas
  - Decomposição funcional de uma aplicação:
    - Apresentação lógica
      - Interação com o usuário e atualização da visão do usuário pela aplicação
    - Aplicação lógica
      - Trata-se do processamento propriamente dito, ou seja, a manipulação dos dados
    - Dados lógicos
      - Tipicamente uma base de dados com as informações consideradas
- Organização 2-tiers e 3-tiers diferem na forma como estas funcionalidades se encontram divididas entre os processos que compõem a aplicação

# Padrões Arquiteturais



## 2-tiers

- Baixas latências nos pedidos de serviço
- Porém sobrecarga no servidor



## 3-tiers

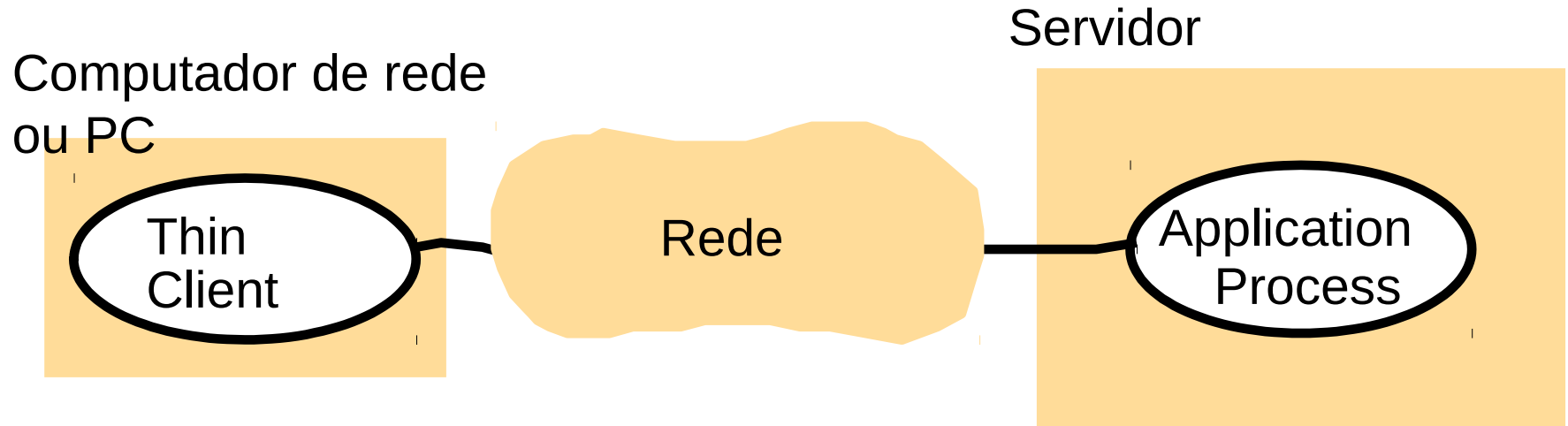
- Manutenção mais simples e maior potencial de adaptação
- Porém maior complexidade na gerência de dois níveis de servidores e latências de comunicação distintas



# Padrões Arquiteturais

## Thin Client

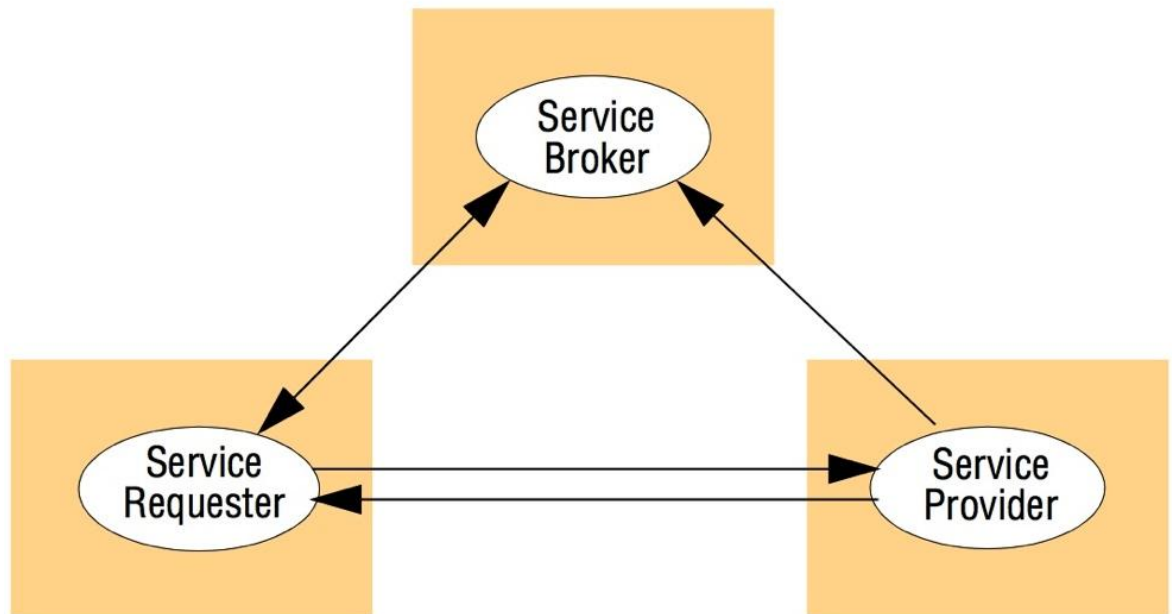
- O dispositivo cliente não possui muitos recursos computacionais, basicamente opera como uma interface para um serviço em rede (nuvem)



# Padrões arquiteturais

## Brokerage

- Um broker oferece uma abstração para acesso a um conjunto de serviços complexos fornecidos por dois ou mais servidores



# Padrões arquiteturais

---

## Proxy

- Oferece um serviço de transparência de localização (usado em RPC e RMI). Pode ser utilizado para replicação e caching

## Reflexão

- Permite o que o nome mesmo diz, reflete sobre a estrutura do sistema e/ou reflete para a localização desejada, promovendo dinamicidade na estrutura

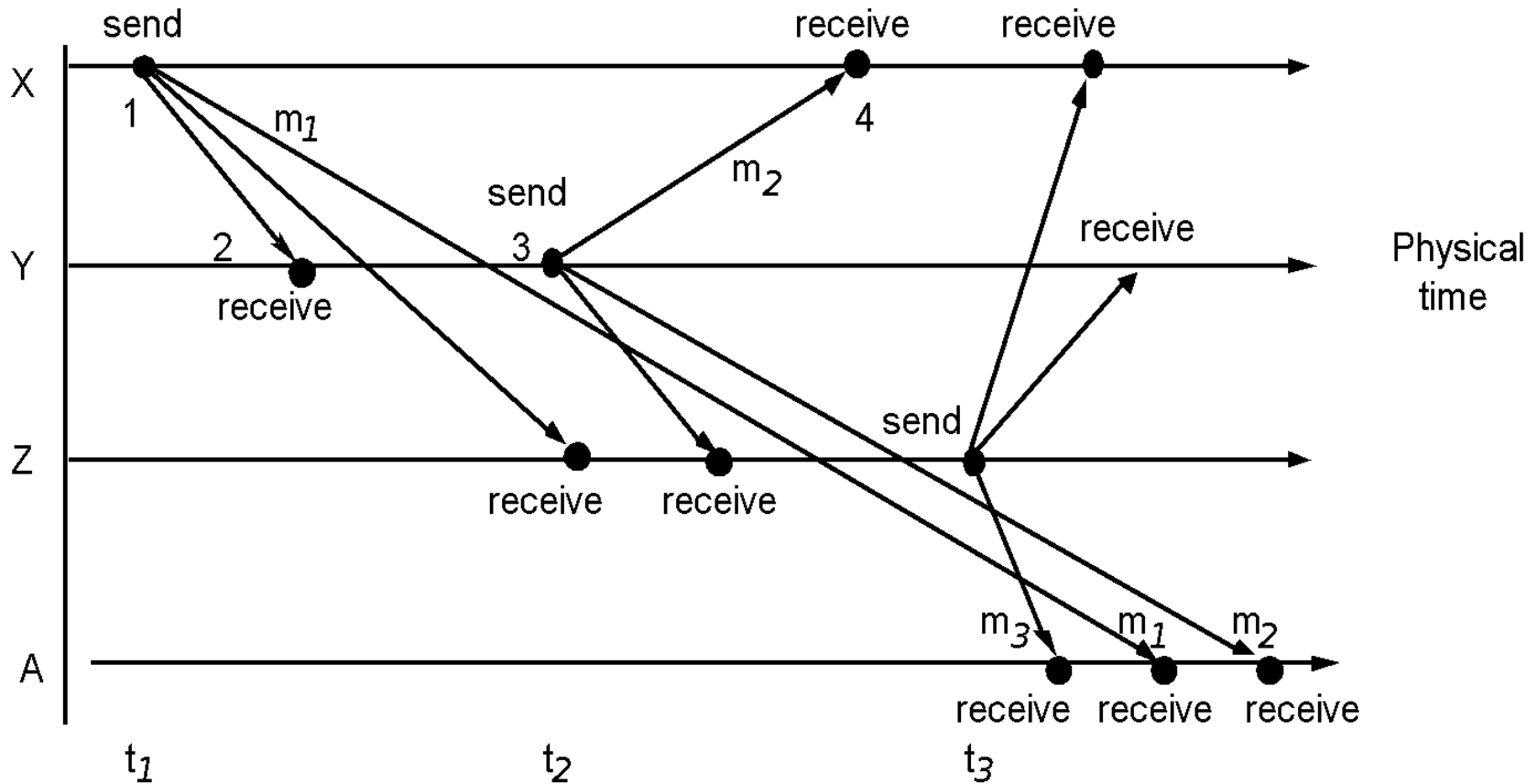
# Modelos fundamentais

Permitem entender o comportamento do SD em termos de suas propriedades

- **Interação**
  - Os processos comunicam, qual a influência das latências de comunicação? Qual a noção de tempo?
- **Falhas**
  - Ocorrem. Quais são suportadas, quais não? Quais as consequências?
- **Segurança**
  - SDs são naturalmente expostos, quais os níveis de segurança previstos?

# Modelos fundamentais

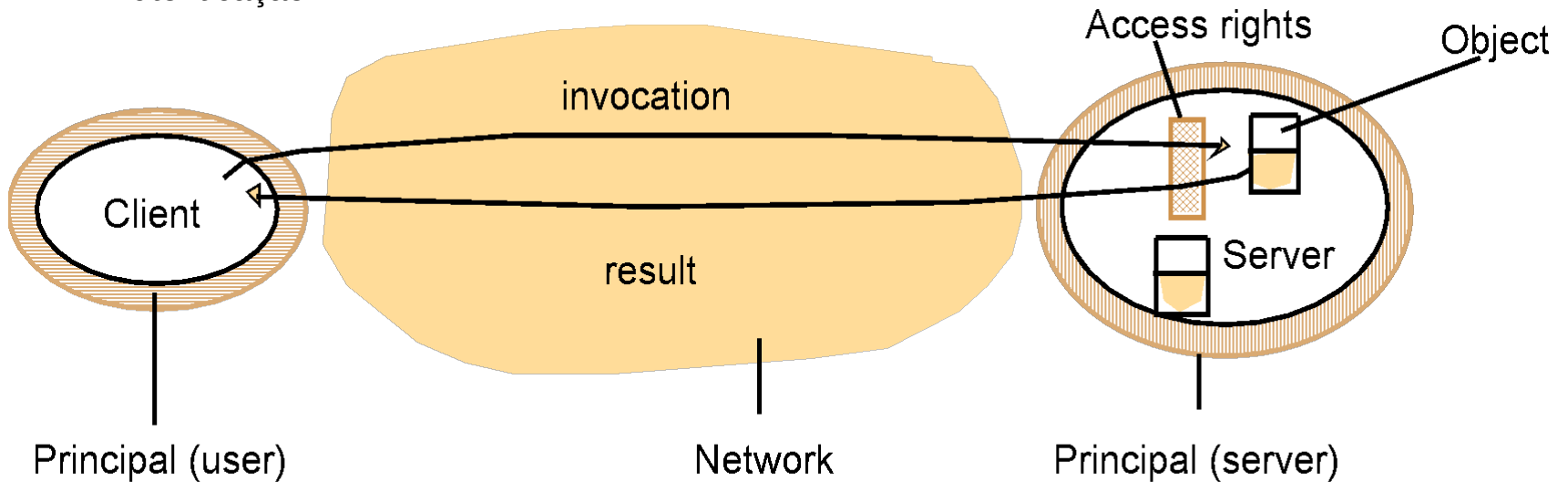
## Interação



# Modelos fundamentais

## Segurança

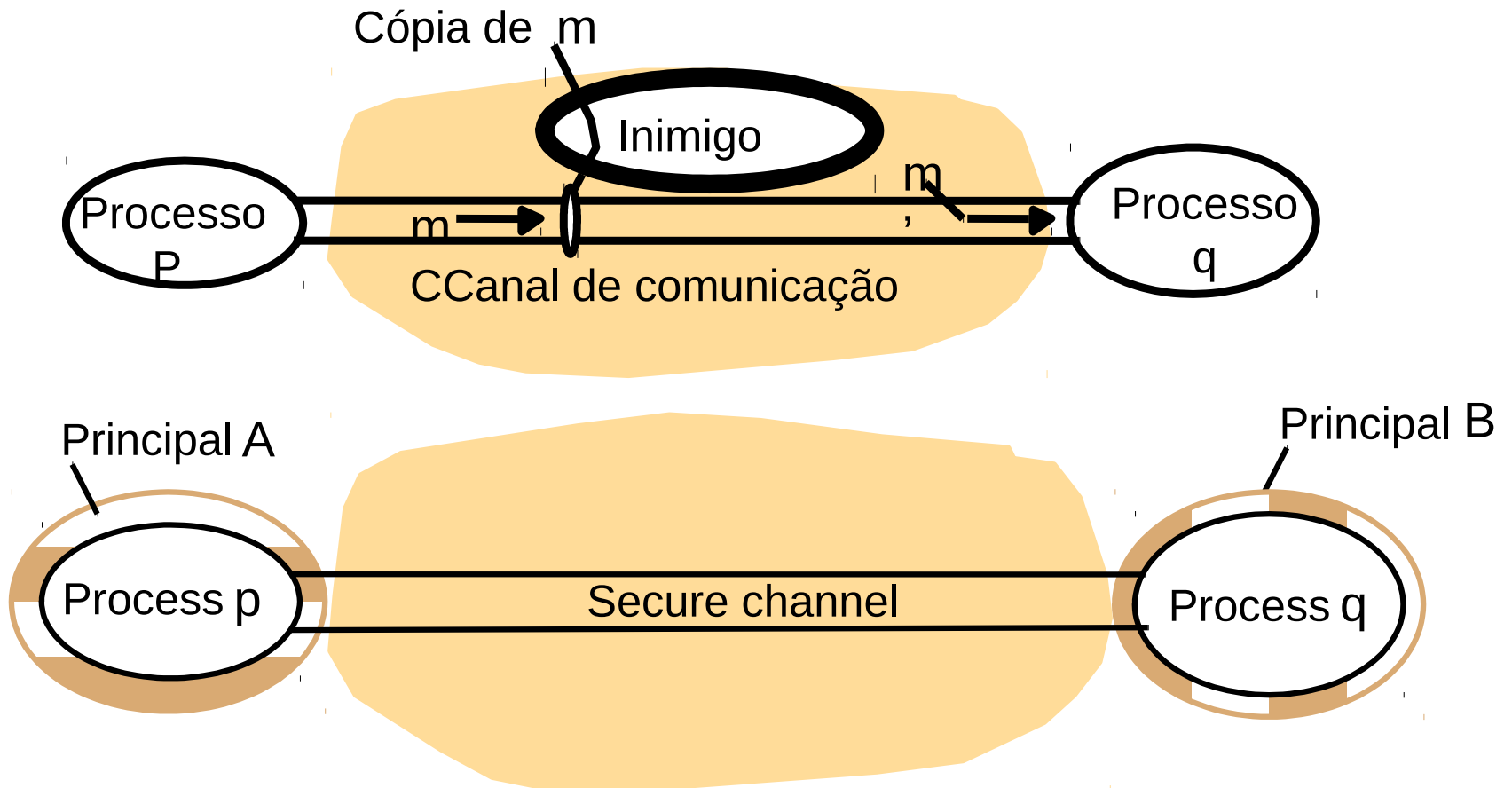
- Autenticação



# Modelos fundamentais

## Segurança

- Criptografia



Ler capítulo 1 e 2.

