Sistemas Distribuídos

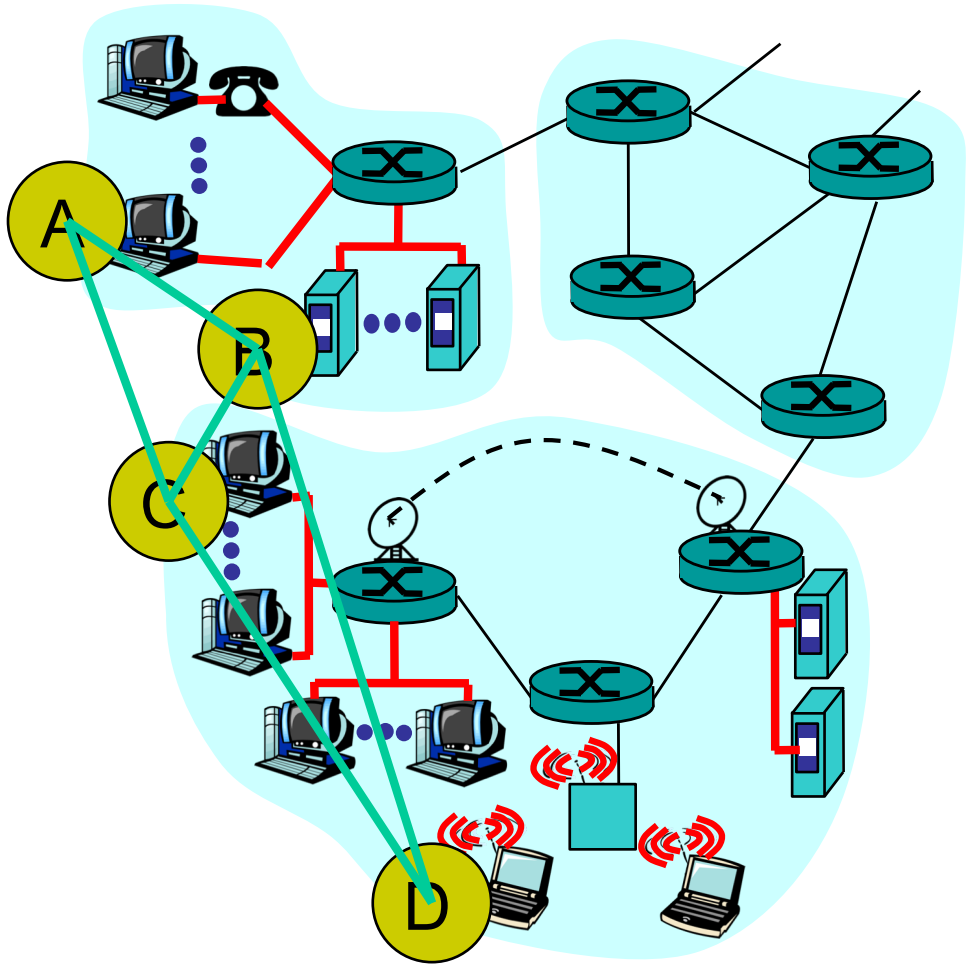
Introdução

Centro de Informática U·F·P·E

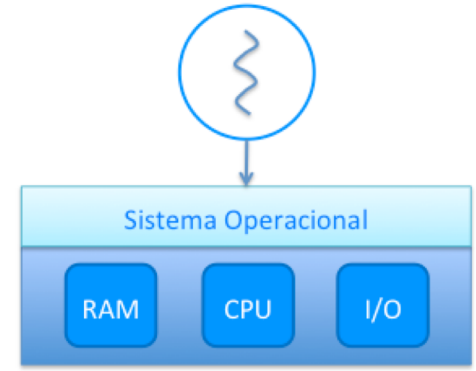
Motivação

Dividindo para conquistar!!!

- Programa distribuído
- Componentes interligados (**comunicação**)
- Processamento (**computação**) distribuído ou paralelo

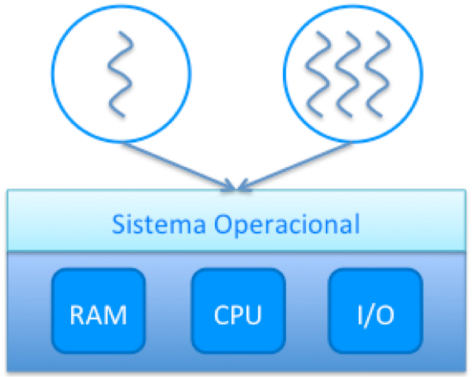


Evolução da Computação

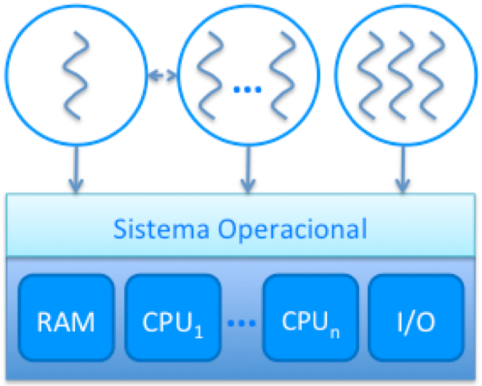


Computação Centralizada

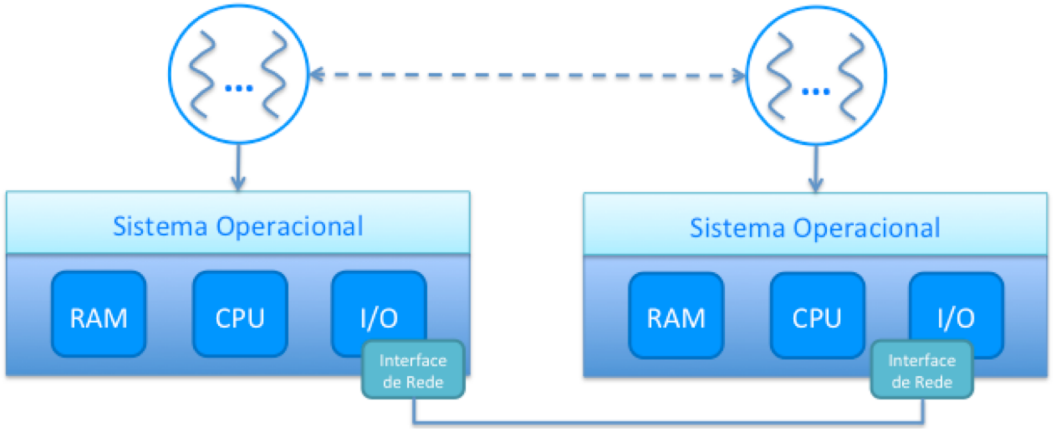
Ou qualquer combinação entre elas !!!



Concorrência

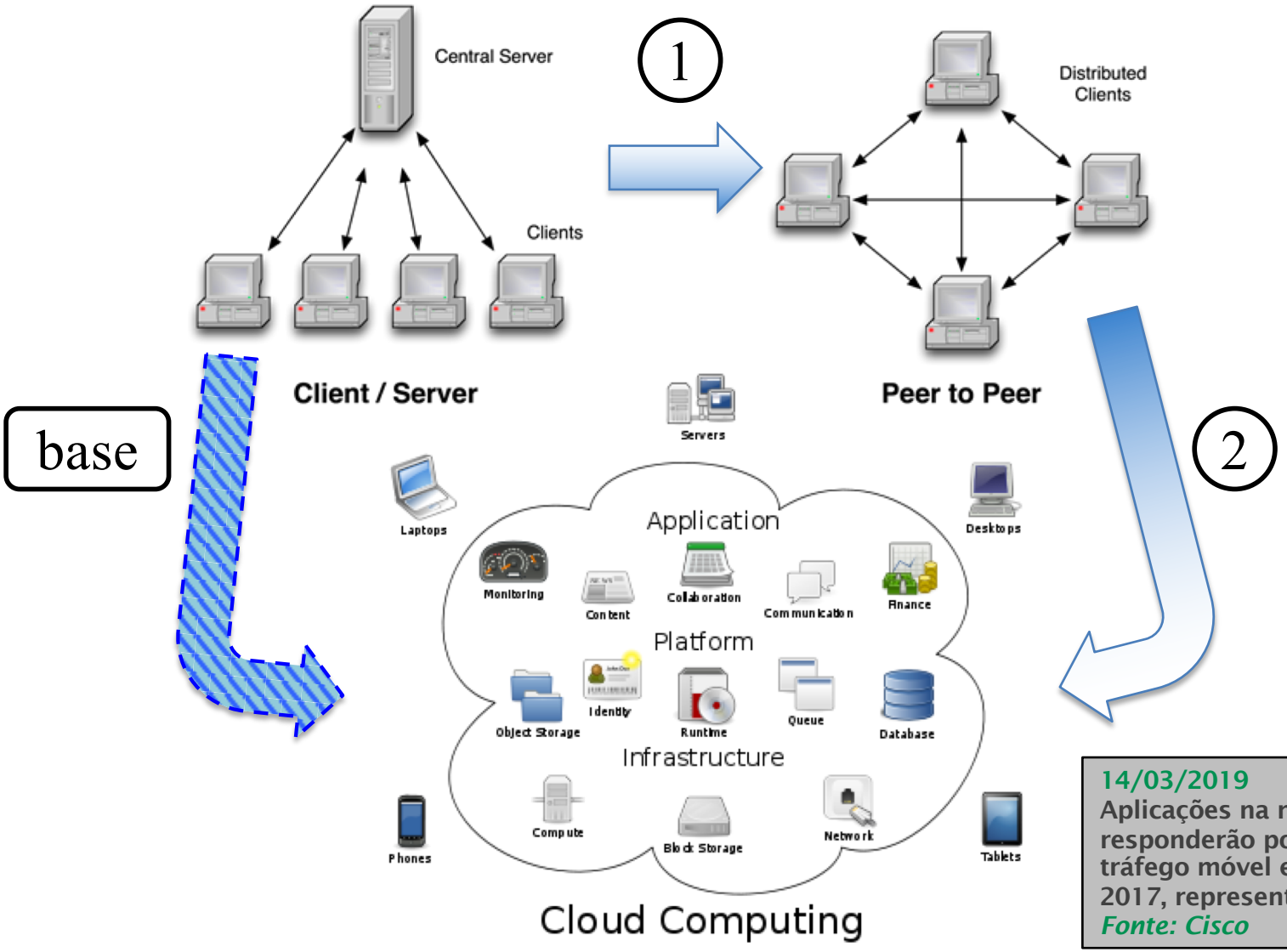


Computação Paralela



Computação Distribuída

Uma história acelerada da evolução da Computação Distribuída



14/03/2019
Aplicações na nuvem responderão por 94% do tráfego móvel em 2022. Em 2017, representaram 86%
Fonte: Cisco

Potencial para ser mais poderoso do que um sistema centralizado, convencional

- Pode ser mais **confiável**: toda função pode ser replicada
  - Quando um processador falha, outro pode continuar o trabalho
  - Se um disco dá *crash*, arquivos gravados também em outros discos não são perdidos
- Várias computações podem ser realizadas em **paralelo**: um sistema distribuído pode realizar mais na mesma quantidade de tempo

Pode-se considerar **tolerância a falha** e possibilidade de **paralelismo** como as propriedades fundamentais de um sistema distribuído

Definições

- Lamport: “Um sistema distribuído é aquele que faz você parar de ter o trabalho realizado quando uma máquina da qual você nunca ouviu falar falha”

- Mais seriamente, Tanenbaum e van Renesse (1985):

Um sistema (operacional) distribuído é aquele que aparece para os usuários como um sistema (operacional) centralizado ordinário, mas que executa em múltiplas CPUs independentes.
O conceito chave é **transparência**, ou seja, o uso de múltiplos processadores deve ser invisível (transparente) para o usuário.
Pode-se dizer que o sistema é visto como um “uniprocessador virtual”, e não como uma coleção de máquinas distintas.

Tipos de Transparência

- **Localização**: esconde onde o recurso está localizado
- **Acesso**: operações idênticas para acesso local e remoto
- Migração: esconde que um recurso pode se mover para outra localização
- Relocação: esconde que um recurso pode ser movido para outra localização enquanto está em uso
- Concorrência: compartilhamento de recursos sem interferência entre processos concorrentes
- Falha: esconde a falha e recuperação de um recurso
- Replicação: esconde de usuários ou programadores de aplicação a existência de réplicas de recursos

Definições mais recentes

- "Coleção de computadores independentes que aparecem para os usuários do sistema como um único computador." (**Tanenbaum & van Steen**)

- "Um sistema em que componentes de *hardware* e *software* localizados em computadores em rede se comunicam e coordenam suas ações por passagem de mensagens." (**Coulouris** *et al*)

- "Uma coleção de elementos de processamento interconectados, tanto logicamente como fisicamente, para execução cooperativa de programas de aplicação com o controle geral dos recursos centralizado." (**M. Eckhouse**)

- Vários componentes
- Conectados via uma rede
- Compartilhando recursos
- Transparência

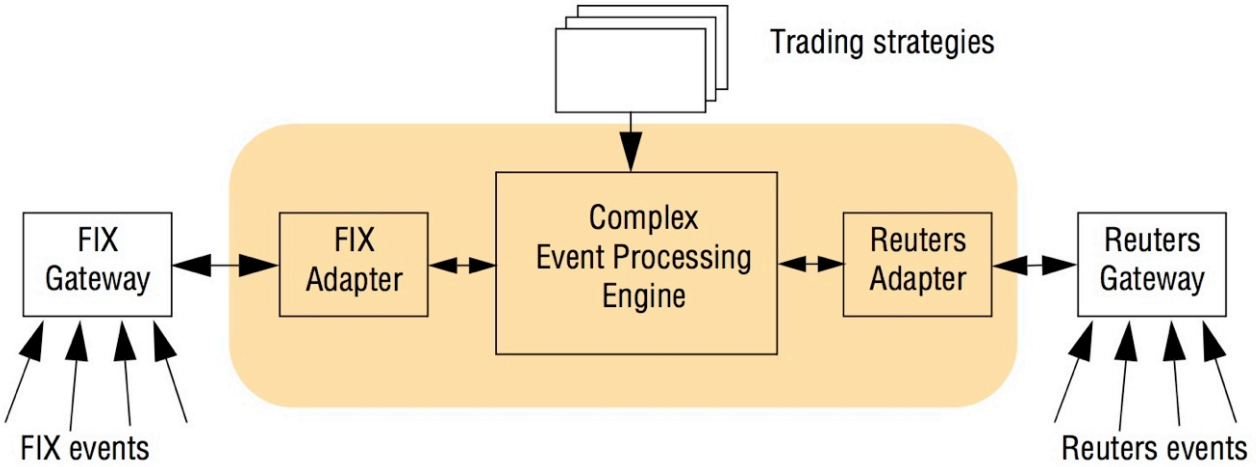
Por que construir SDs?

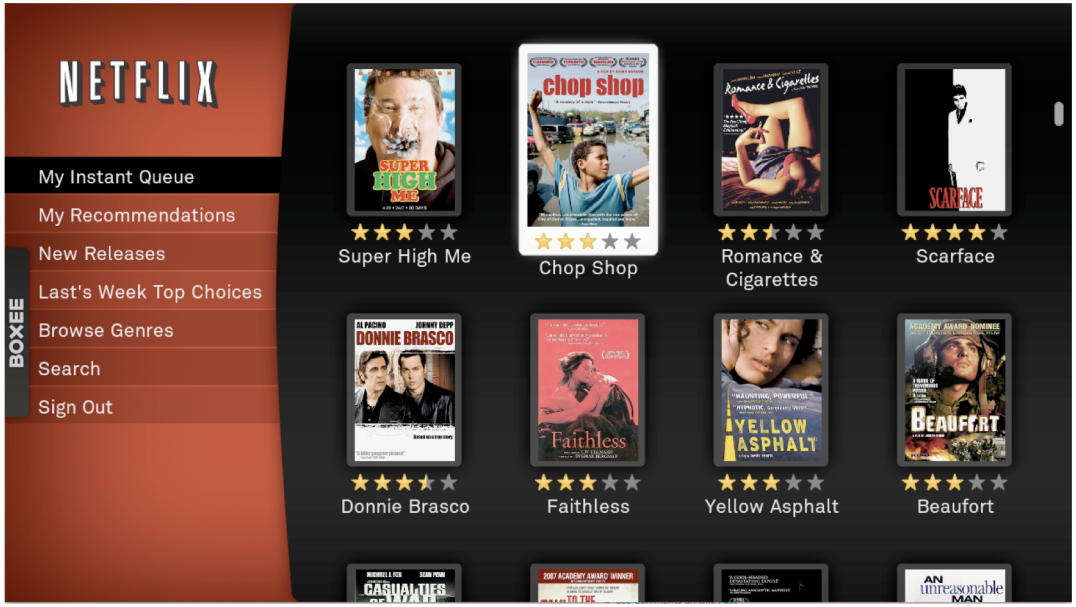
- Pessoas são distribuídas, informação é distribuída
  - Desejo de comunicar e compartilhar informações e recursos
- Relação desempenho/custo
- Modularidade
- Expansibilidade
  - Sistemas distribuídos são capazes de crescimento incremental
- Disponibilidade
  - SDs têm capacidade de replicação e redundância

Por que construir... (cont)

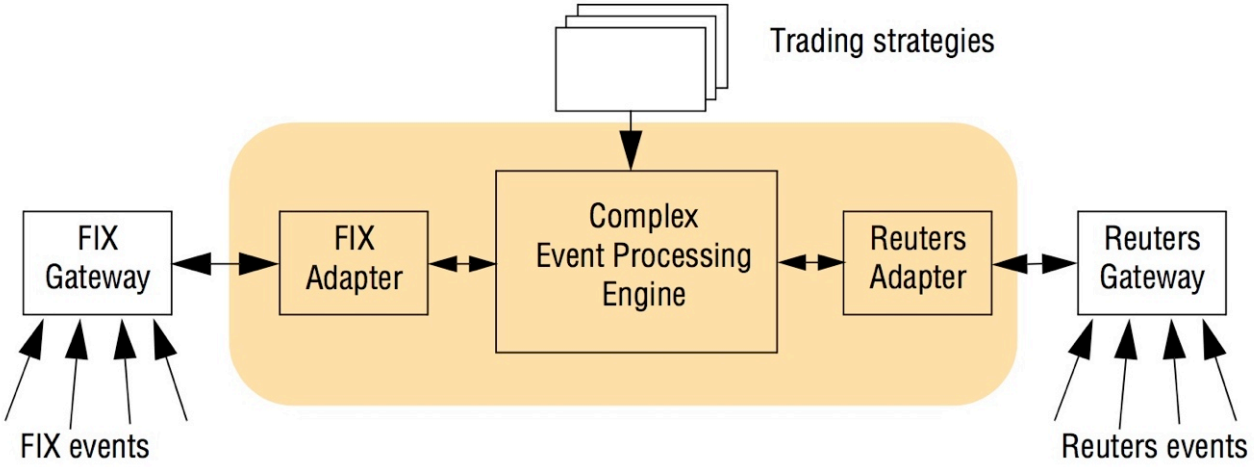
- Escalabilidade
  - Idealmente, sistemas distribuídos não devem ter qualquer componente centralizado (cuja capacidade impõe limites para o tamanho máximo de um sistema), tal que a restrição ao crescimento não deve existir
- Confiabilidade
  - Disponibilidade é apenas um aspecto de confiabilidade
  - O sistema deve ser capaz de se recuperar de falhas

Dois exemplos de sistemas distribuídos





EVENTOS



- Acesso em tempo real a **muitas fontes de informação** (ex. Reuters, FIX – Financial Information eXchange), interessantes para **muitos clientes**
- Comunicação e processamento de itens de interesse – **eventos**

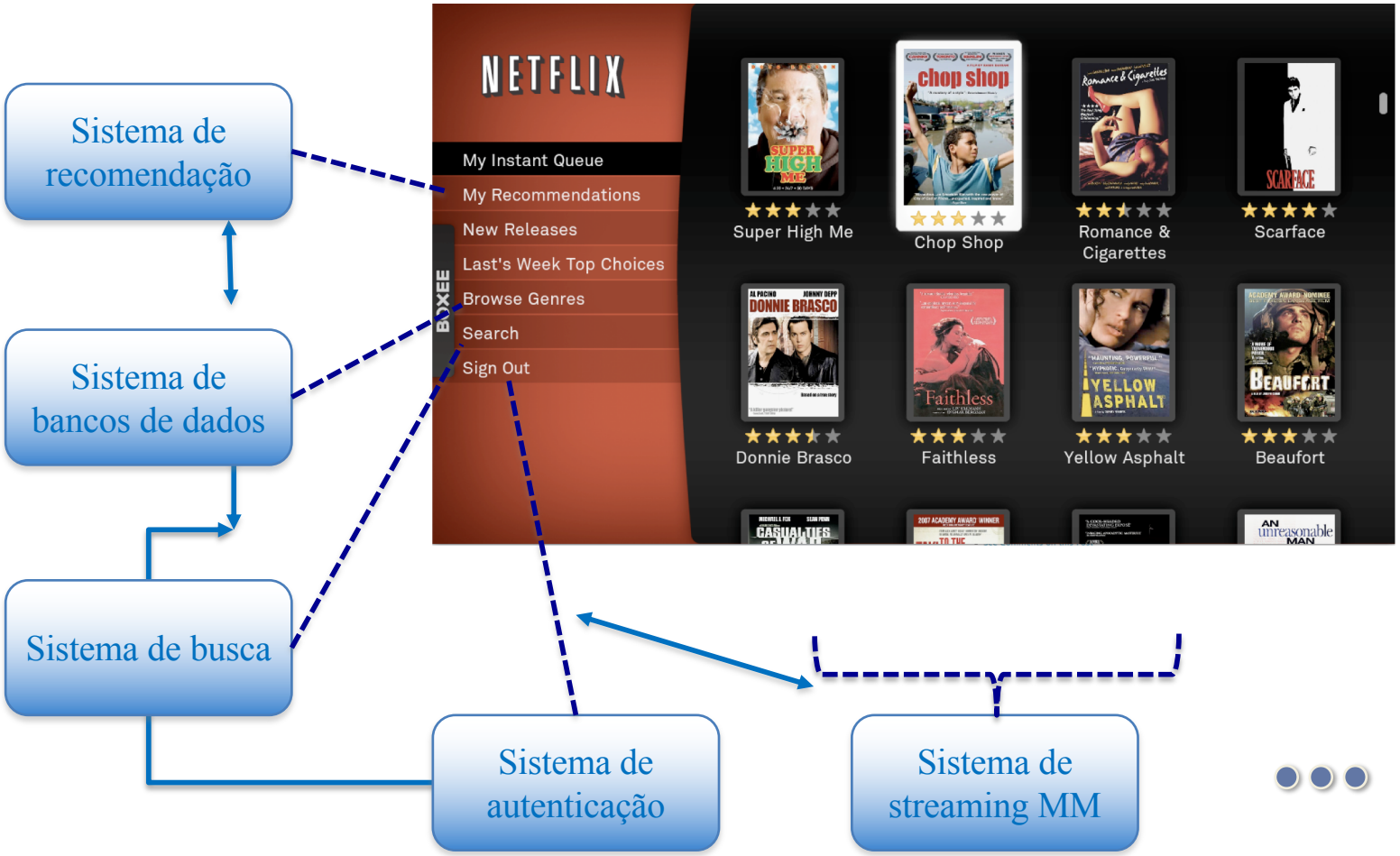
- **Sistemas distribuídos baseados em eventos**

Transparência de distribuição

```
WHEN
        MSFT price moves outside 2% of MSFT Moving Average
FOLLOWED-BY (
        MyBasket moves up by 0.5%
        AND
                HPQ's price moves up by 5%
                OR
                MSFT's price moves down by 2%
        )
)
ALL WITHIN
        any 2 minute time period
THEN
        BUY MSFT
        SELL HPQ
```

- O usuário que escreveu este script não precisa ter noção da ocorrência distribuída de eventos
  - Dados e processamento distribuídos

Sistemas distribuídos: multimídia



Complexidade

- Complexidade limita o que pode ser construído
- Schroeder chama os problemas causados pela complexidade de **problemas de sistema**:
  - Interconexão: um grande número de problemas de sistemas acontece quando componentes que antes operavam independentemente são interconectados
  - Interferência: dois componentes de um sistema, cada um com comportamento razoável quando observados em isolamento, podem exibir comportamento indesejável quando combinados
  - Propagação de efeito: “efeito cascata” de falhas pode derrubar um sistema inteiro se não houver cuidados no projeto

Problemas de sistema (cont)

- Efeitos de escala: um sistema que funciona bem com 10 nós pode falhar se crescer para centenas de nós
- Falha parcial
  - Grande diferencial de sistemas distribuídos em relação a sistemas centralizados, tradicionais
  - Fonte considerável de complexidade no projeto de aplicações tolerantes a falhas

Requisitos não-/funcionais como fonte de complexidade

- Sistemas distribuídos são complexos porque o que eles têm que fazer é complexo
- Exemplos:
  1. Gerenciamento do escalonamento de trens em uma rede em que passageiros têm que trocar de trens para chegar em seus destinos – **o problema da sincronização**
  2. **Sistema de arquivos distribuídos**
     - É preciso prever aspectos como autenticação, controle de acesso, controle de concorrência etc.
     - Complexidade ainda maior quando há os requisitos de alta disponibilidade e tolerância a falhas
       - Aspectos como mecanismos de localização de arquivo, coordenação de estado de servidor replicado, mecanismos de recuperação de falhas parciais etc.

Necessidade econômica como fonte de complexidade

- A solução simples nem sempre pode ser usada – às vezes é cara demais!
- Exemplo:
  - Em uma rede de longa distância, interconectar todos os pontos (nós) seria a solução mais simples, porém extremamente cara!
  - A solução mais barata é fazer uma rede em que todos os nós são alcançáveis, porém não necessariamente de forma direta
    - O custo dessa solução é mais baixo, porém a complexidade é bastante aumentada, pois são necessários
      - algoritmos de roteamento,
      - "buferização" para gerenciar o tráfego multiplexado,
      - mecanismos de controle de fluxo para prevenir congestionamento etc.

Sistemas Distribuídos: Recapitulando

- Do ponto de vista do usuário e do programador de aplicação, é preciso **transparência**
- **Tolerância a falha** e possibilidade de **paralelismo** são propriedades fundamentais de um sistema distribuído
- **Falha parcial** é um grande diferencial de sistemas distribuídos em relação a sistemas centralizados/tradicionais, mas também é fonte considerável de complexidade
- Do ponto de vista de **software**, é preciso agregar outras funcionalidades às que os sistemas operacionais convencionais oferecem para dar suporte adequado a sistemas distribuídos

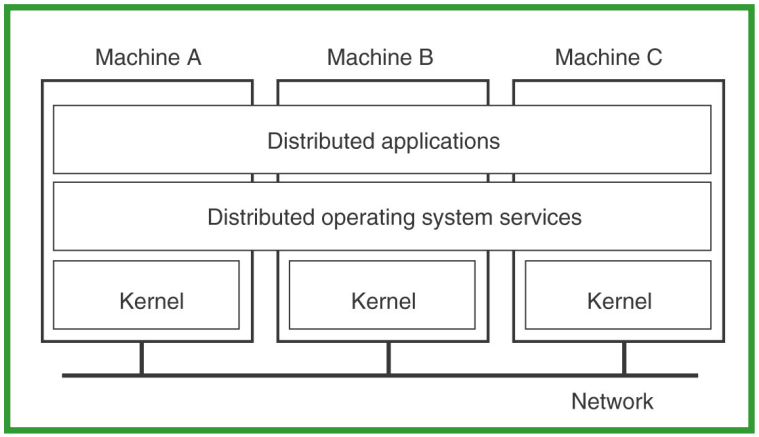
Desafios de SDs

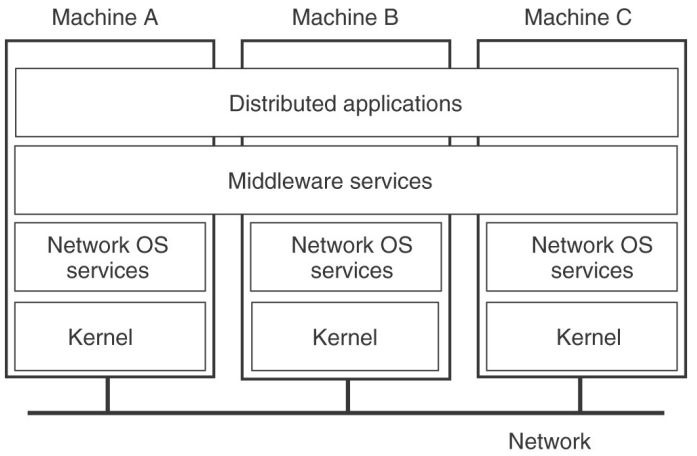
- Heterogeneidade
- Transparência
- Tolerância a Falhas
- Segurança
- Escalabilidade
- Concorrência
- Abertura

Infraestruturas de Software para Sistemas Distribuídos

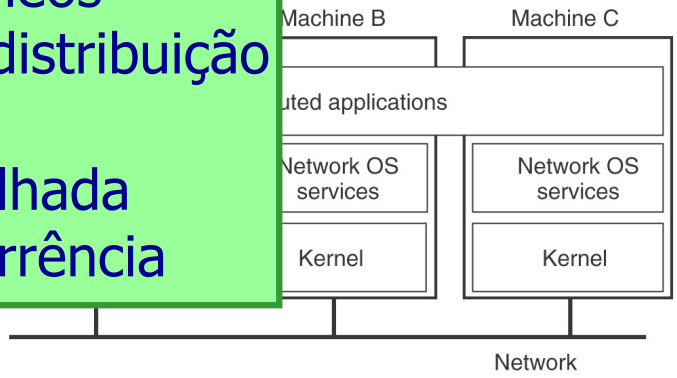
- Sistemas Operacionais Distribuídos
- Sistemas Operacionais de Rede
- Middleware

Infra-estruturas para SDs: diferenças de objetivos e abstrações

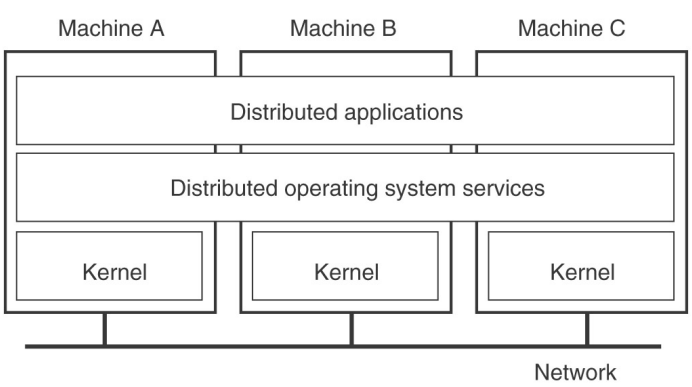


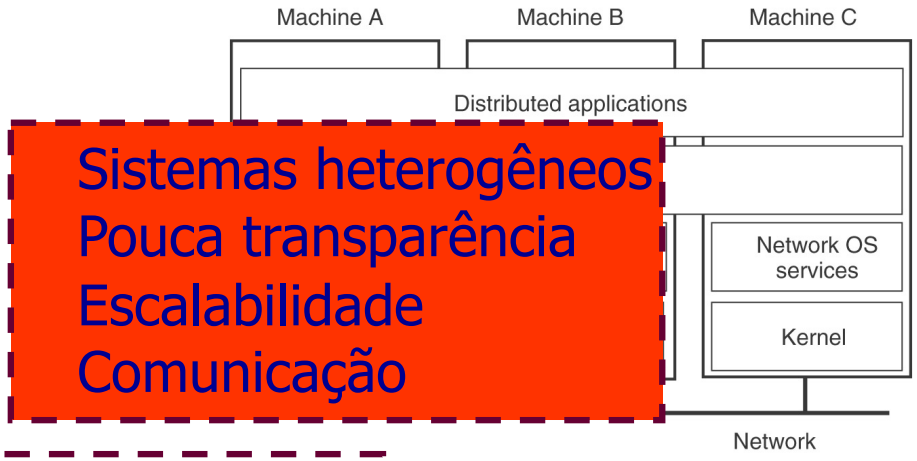


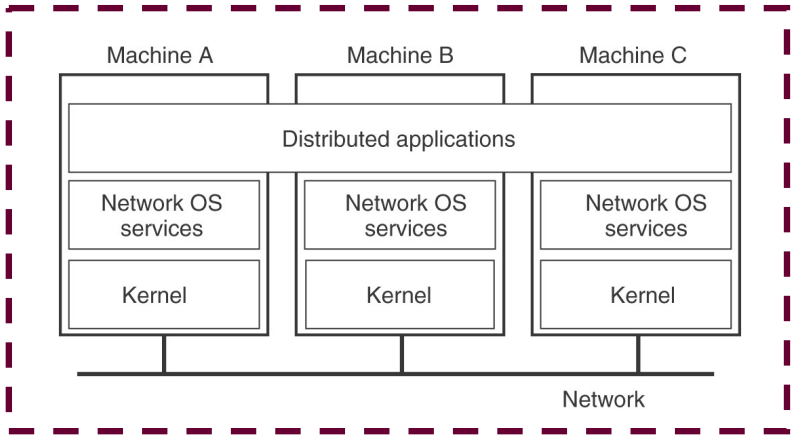
Sistemas homogêneos
Transparência de distribuição
Alto desempenho
Memória compartilhada
Controle de concorrência



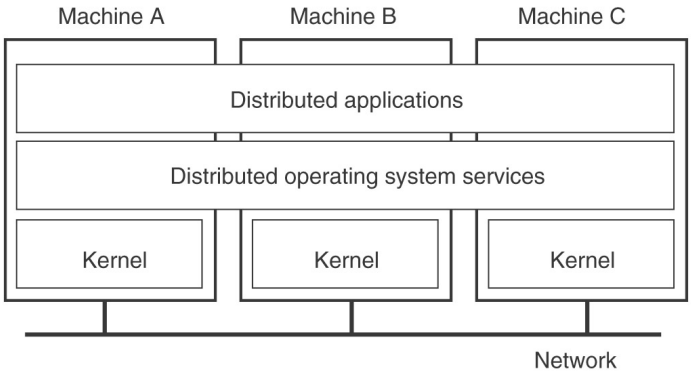
Infra-estruturas para SDs: diferenças de objetivos e abstrações

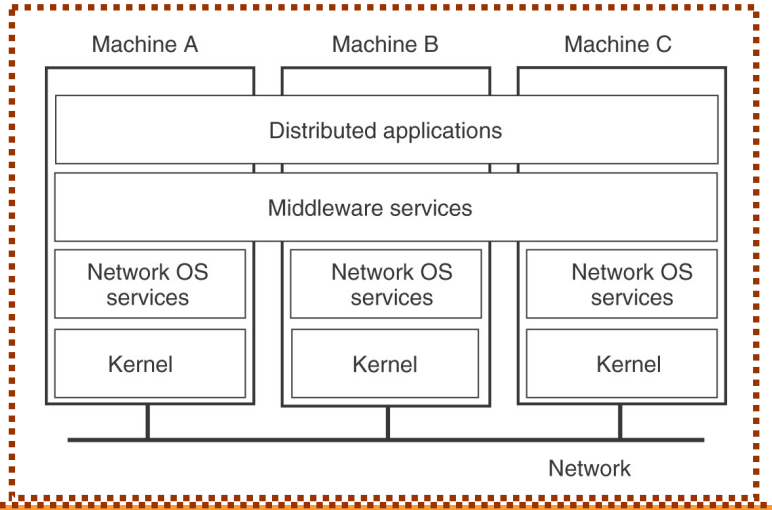


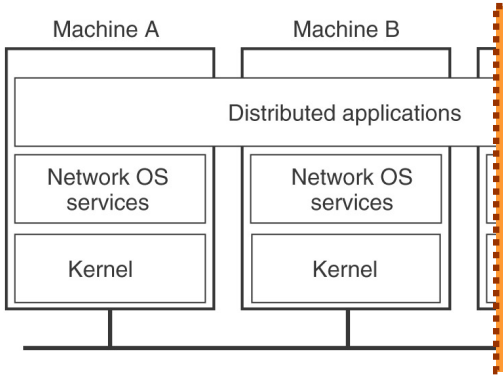




Infra-estruturas para SDs: diferenças de objetivos e abstrações

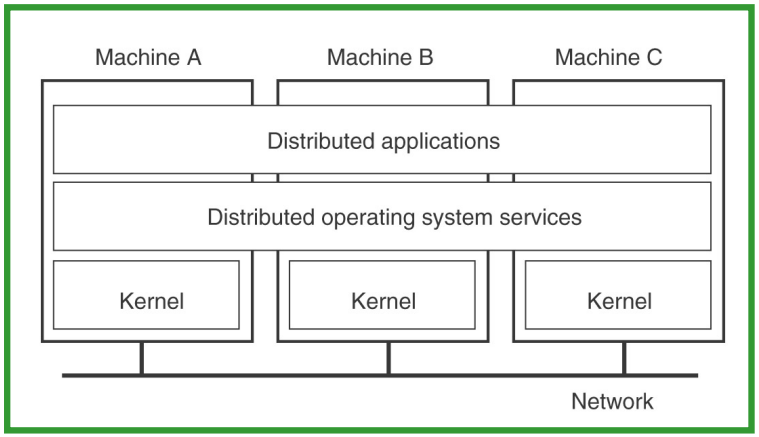




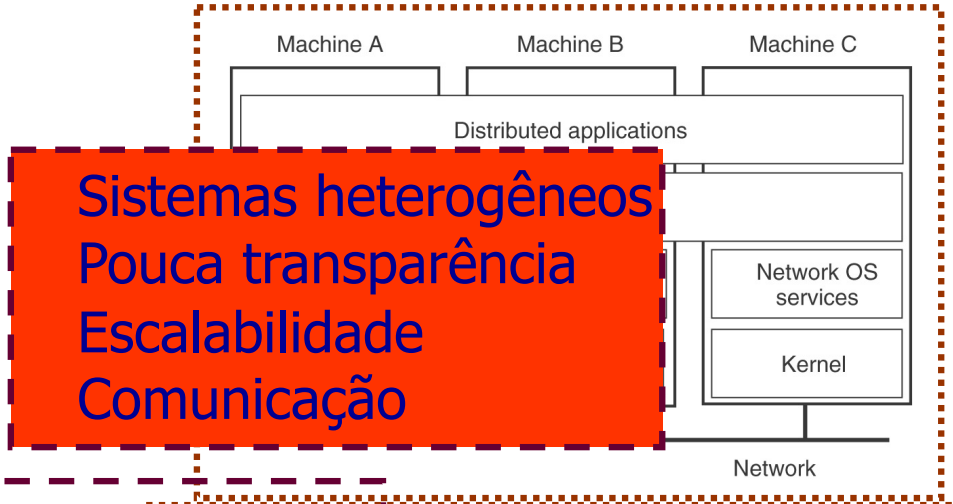


- Sistemas heterogêneos
- Transparência de distribuição e comunicação
- Serviços
- Abertura

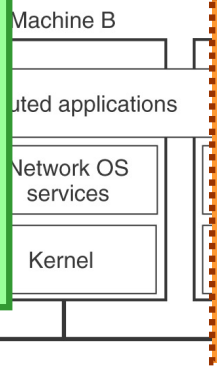
Infra-estruturas para SDs: diferenças de objetivos e abstrações



Sistemas homogêneos
Transparência de distribuição
Alto desempenho
Memória compartilhada
Controle de concorrência



Sistemas heterogêneos
Pouca transparência
Escalabilidade
Comunicação



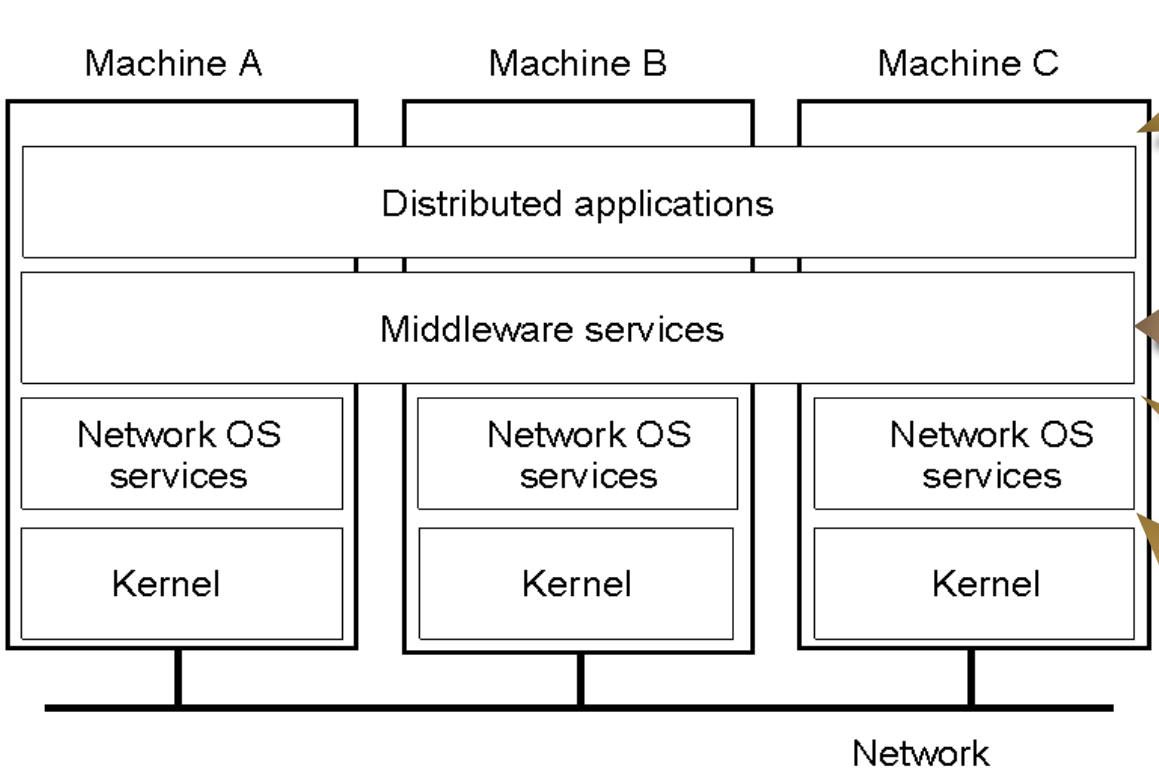
Sistemas heterogêneos
Transparência de distribuição e comunicação
Serviços
Abertura

Próximas Datas

14/11, qui	Aula normal
19/11, ter	Aula prática: JavaRMI e OpenMP – Lab G2
21/11, qui	**Revisão 1o. EE – D002**
26/11, ter	**Exercício** de RPC/RMI – Lab G2
28/11, qui	**3o. EE**: Sist. Arquivos, E/S, Sist. Distribuídos – D002
05/12, qui	**Revisão de Avaliações/Notas – Sala C-125**
10/12, ter	**PROVA FINAL**

Middleware

Necessidades: quem atende o quê?



- Lógica do negócio

- Integração de Sistemas Heterogêneos
- Interoperabilidade destes sistemas
- Portabilidade

- Comunicação

Uso e gerência de recursos de hardware:

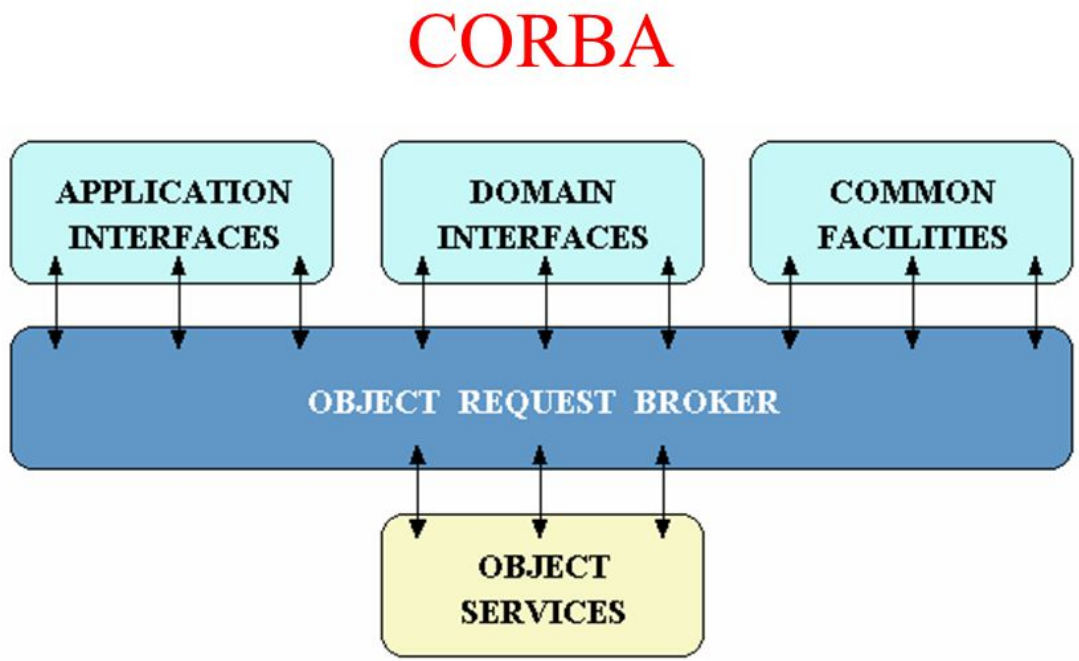
CPU
Memória
I/O

Middleware: definição

- um conjunto reusável, expansível de serviços e funções …
- que são comumente necessários por parte de várias aplicações distribuídas …
- para funcionarem bem em um ambiente de rede

Middleware: definição

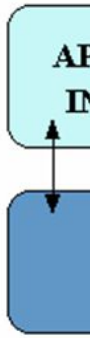
- um conjunto reusável, expansível de serviços



OMG Reference Model architecture

Middleware: definição

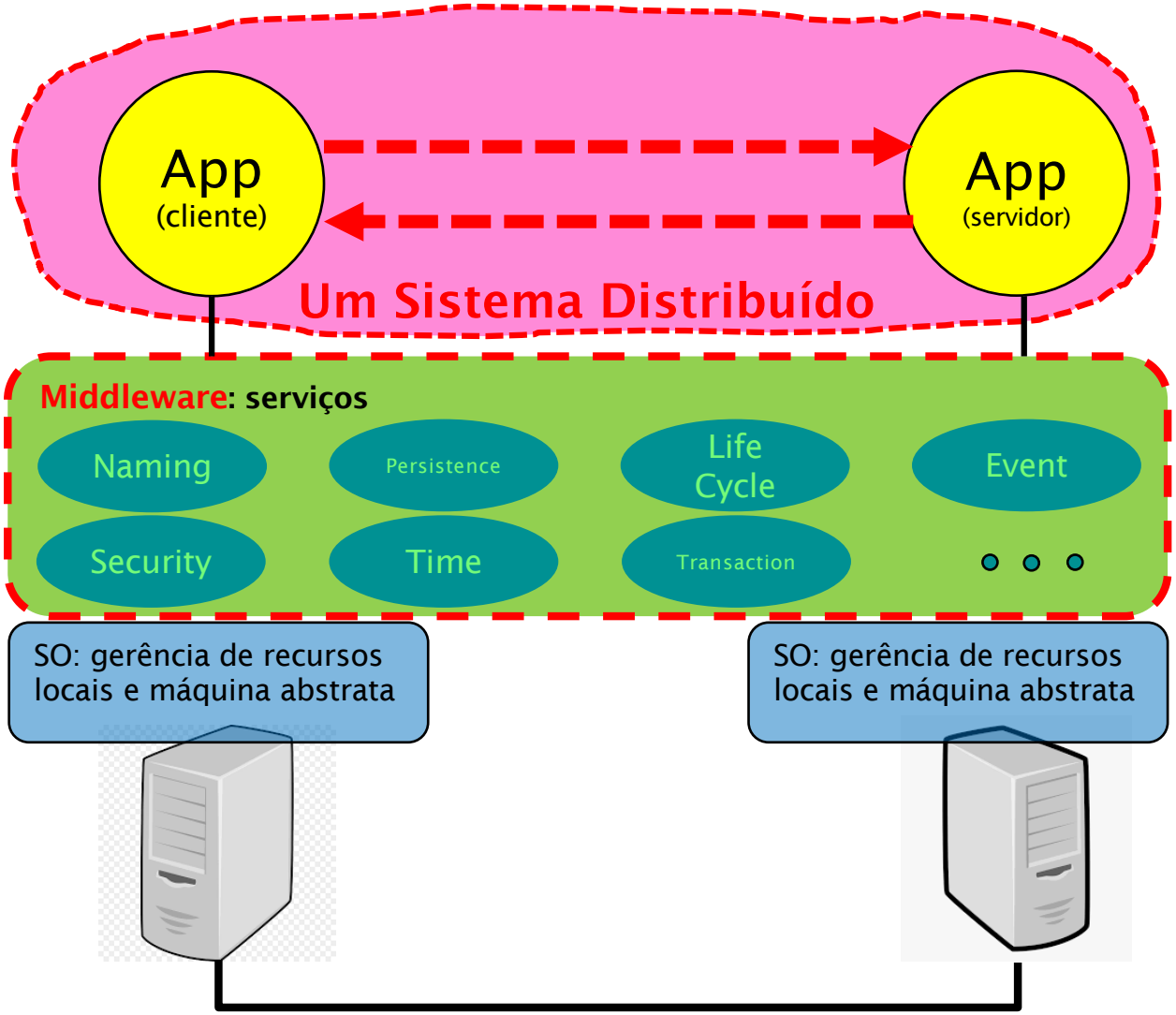
- um conjunto reusável, expansível de serviços



Service	Description
Life Cycle Service	Support for the creation, copying, moving and deleting of objects.
Persistence	Common interface for storing / retrieving objects from storage.
Naming Services	Primary object location service, mapping of human readable names to object references.
Event Service	Event notification mechanism (event supplier and consumer objects). Push (publish - subscribe) and pull (=polling) models.
Concurrency Control	Resource locking, transaction locks.
Transaction Service	Flat and nested transactions (sub-transactions).
Relationship Services	Explicit definition of relationships of objects.
Query Service	Mapping CORBA objects to relational DBs (a kind of ORM).
Licensing Service	Controlled access to objects.
Property Service	Association of properties to objects at runtime (properties are not part of IDL interface).
Time Service	Synchronization of clocks on different hosts.
Security Service	Authentication and propagation of credentials, encryption.

OMG Reference Model architecture

Middleware: definição



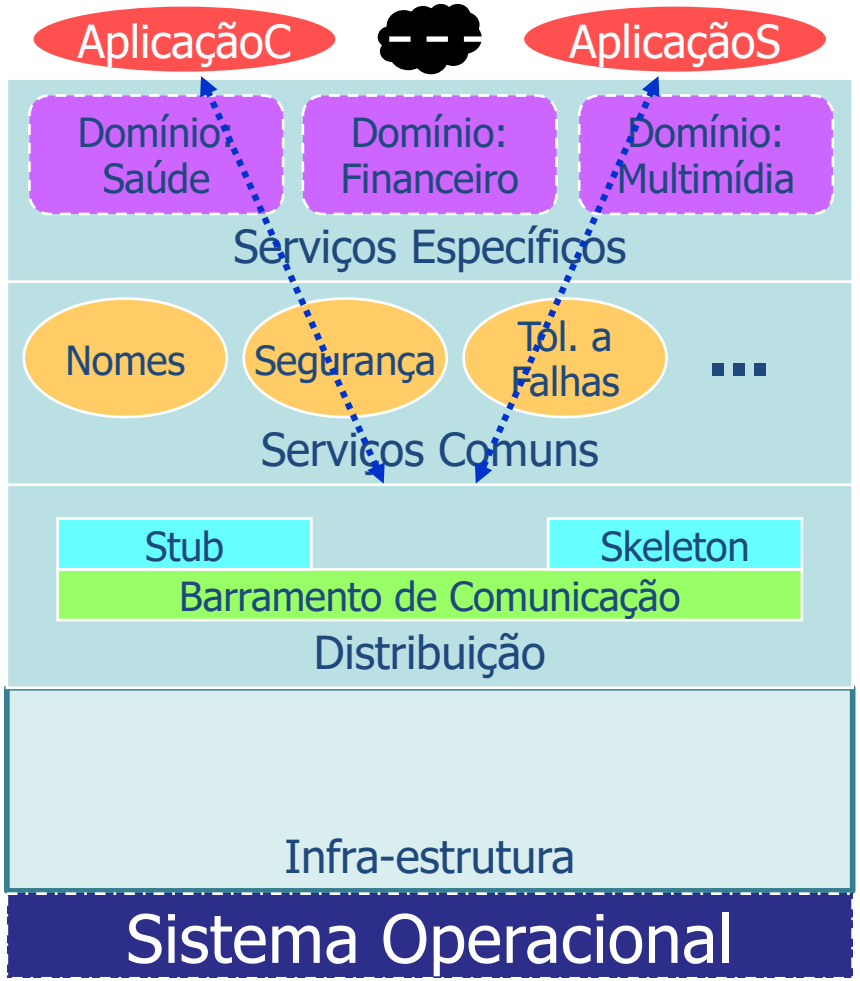
Infra-Estrutura de SW para SDs

Middleware: Arquitetura Básica em Camadas

Middleware: coleção de serviços (serviços de middleware) fornecidos através de interfaces padrões (APIs) – visão "unificada" de redes e engenharia de software, respectivamente
+ formado sobre camadas de infra-estrutura

modelos de programação, onde a comunicação é abstraída, por ex. – ORB CORBA, incluindo RPC

abstrai as peculiaridades dos sistemas operacionais, encapsulando e melhorando os mecanismos de concorrência, por ex. – ex. JVM



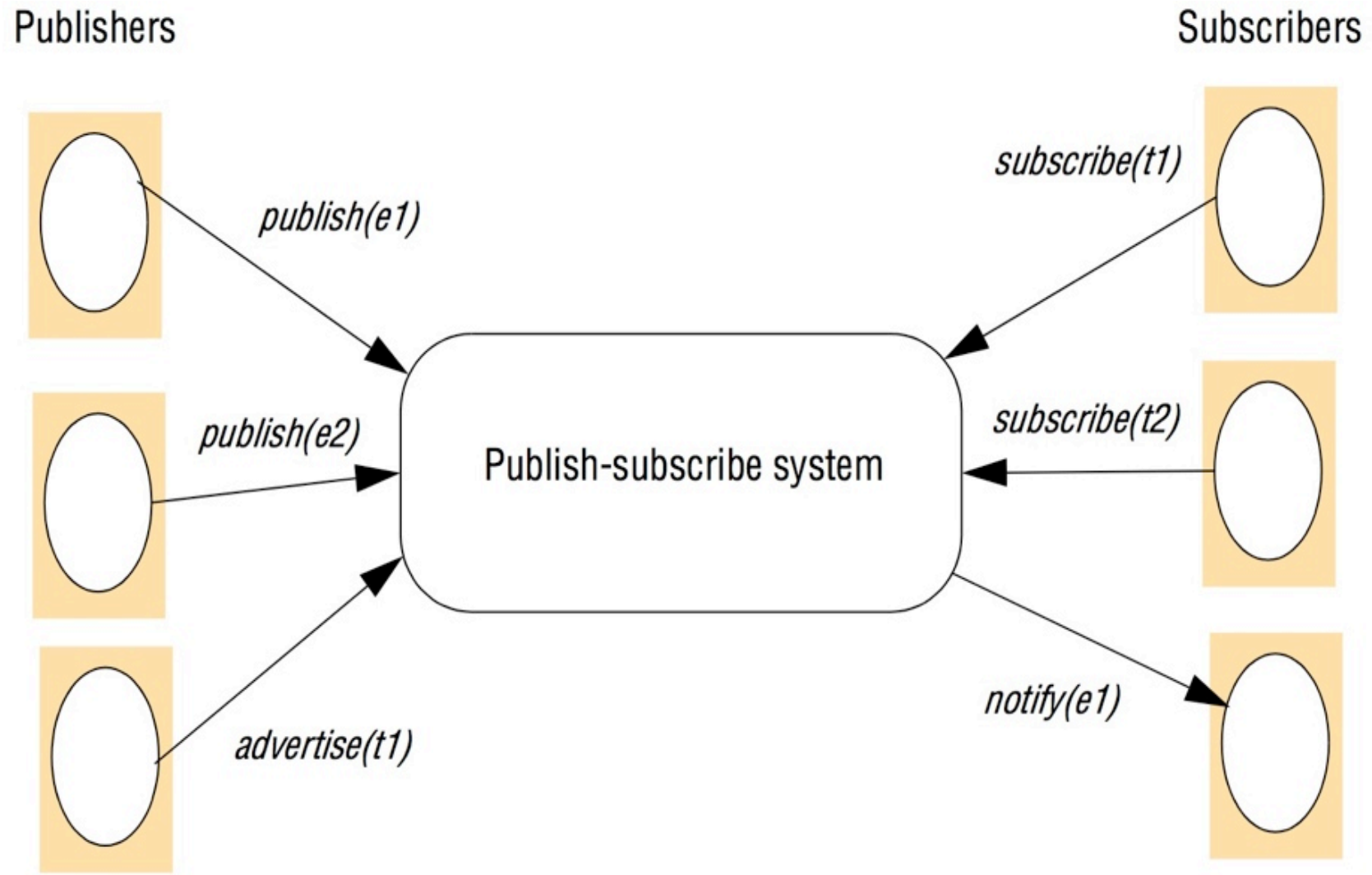
Middleware "Tradicional"

- Message-Oriented Middleware (MOM)
- Transaction Processing Monitors (TPMON)
  - Forte associação com acesso distribuído a BD
  - Propriedades "ACID"
- Remote Procedure Calls (RPC)
- Object-Oriented Middleware (ORB, ...)

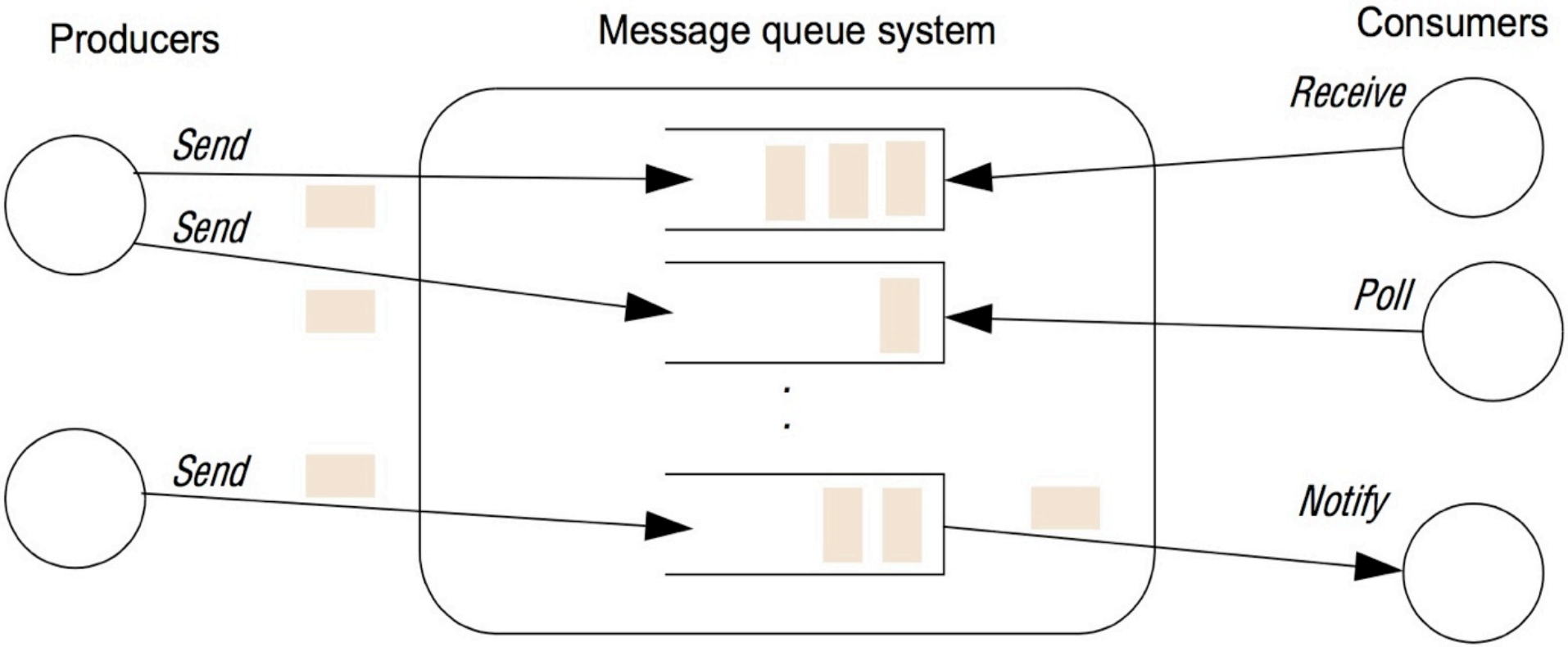
Middleware "Tradicional"

- Message-Oriented Middleware (MOM)
- Transaction Processing Monitors (TPMON)
  - Forte associação com acesso distribuído a BD
  - Propriedades "ACID"
- Remote Procedure Calls (RPC)
- Object-Oriented Middleware (ORB, ...)

O paradigma *publish-subscribe*

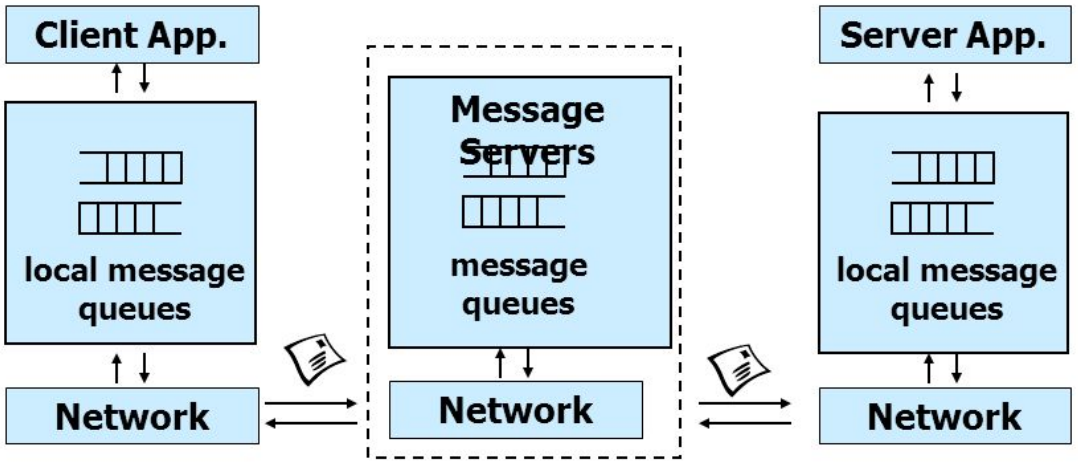


O paradigma de fila de mensagens



MOM

- Message servers desacoplam clientes e servidores de aplicação



Middleware "Tradicional"

- Message-Oriented Middleware (MOM)
- Transaction Processing Monitors (TPMON)
  - Forte associação com acesso distribuído a BD
  - Propriedades "ACID"
- Remote Procedure Calls (RPC)
- Object-Oriented Middleware (ORB, ...)

ACID

- **A**tomicidade
- **C**onsistência
- **I**solamento
- **D**urabilidade