Aula 1
Sistemas Distribuídos:
Introdução e modelos de comunicação

Prof. Rafael Guimarães
FAESA

O que é um Sistema Distribuído?

- Sistema composto por diferentes elementos de software que executam em nós (hardware) espalhados pela rede e que utilizam essa rede para comunicação entre eles
- Quase todos os sistemas hoje em dia são Distribuídos!
- Difusão da Internet colaborou muito para essa realidade

Alguns conceitos

- Desejável que haja **acoplamento fraco** entre os elementos do sistema distribuído
    - Quanto menos conversarem (menos mensagens, com menos frequência) melhor
- **Ausência de relógio global**: cada nó tem seu próprio relógio, não há uma referência única de tempo! Pode ser necessária sincronização.

Ausência de relógio global

- Relógios de Lamport (1978)
  - Não sincroniza relógios, ordena eventos
- NTP (*Network Time Protocol*)
  - Sincroniza relógios segundo uma base de tempo comum (servidor NTP)
  - Referência: http://ntp.br

Network Time Protocol

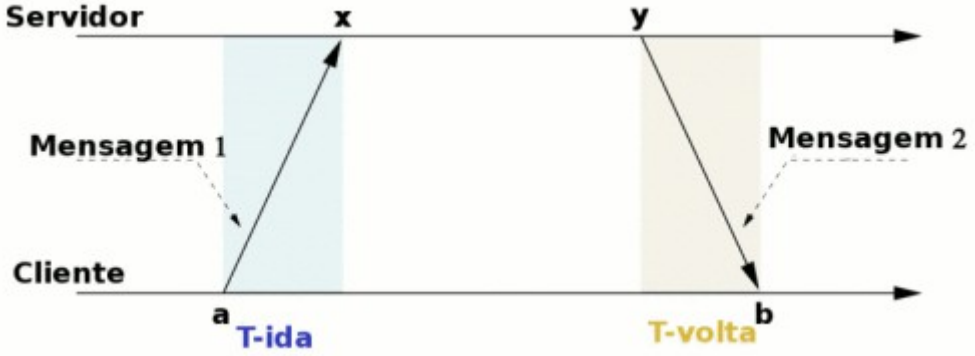
- Algumas implementações do SNTP (*Simplified NTP*) apenas consultam a hora no servidor e ajustam a hora local
  - NTP tradicional é mais complexo que isso
  - O atraso de rede entre a resposta o servidor e a recepção do cliente pode variar e não é levado em consideração no caso do SNTP
  - Como ser mais preciso?

Network Time Protocol

- No NTP, o cliente verifica seu relógio (hora **a**)
- Envia requisição de hora ao servidor, informando essa hora **a**
- O servidor recebe essa requisição na hora **x**, processa, e responde na hora **y**
- A resposta chega ao cliente na hora **b**

a e **b** estão na mesma referência de tempo

x e **y** estão na mesma referência de tempo

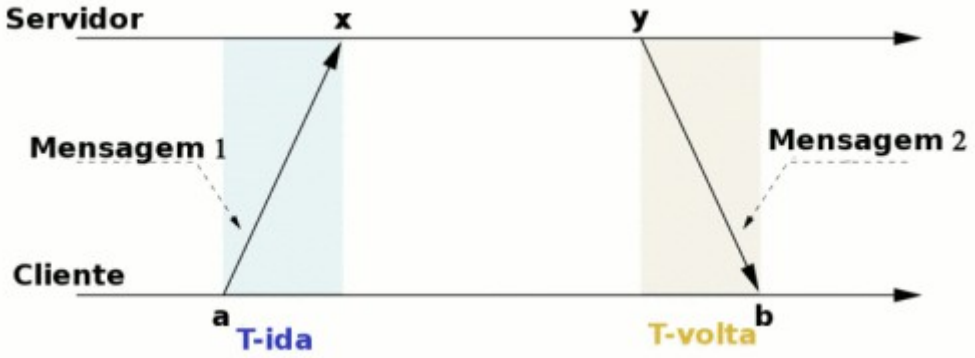


Network Time Protocol

- O tempo total de ida e volta pode ser calculado pelo tempo desde a transmissão da requisição até a chegada da resposta, descontando o tempo de processamento do servidor:
  - RTT (atraso ida e volta) = (b-a) - (y-x)

a e **b** estão na mesma referência de tempo

x e **y** estão na mesma referência de tempo

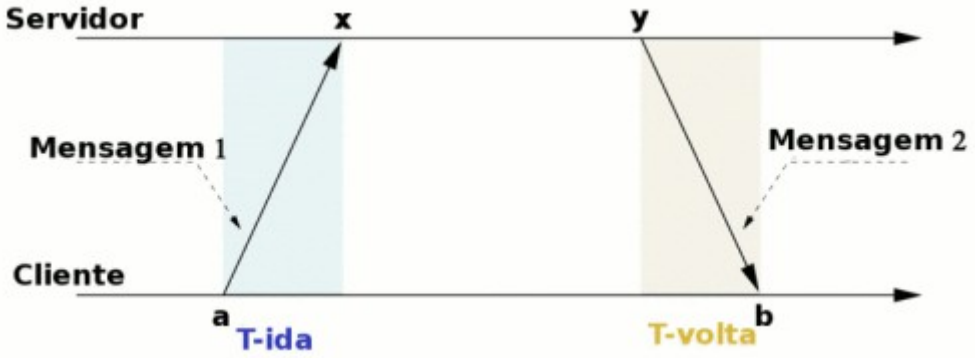


Network Time Protocol

- Assumindo que o tempo de ida é igual ao tempo de volta:
  - Deslocamento do relógio = RTT / 2

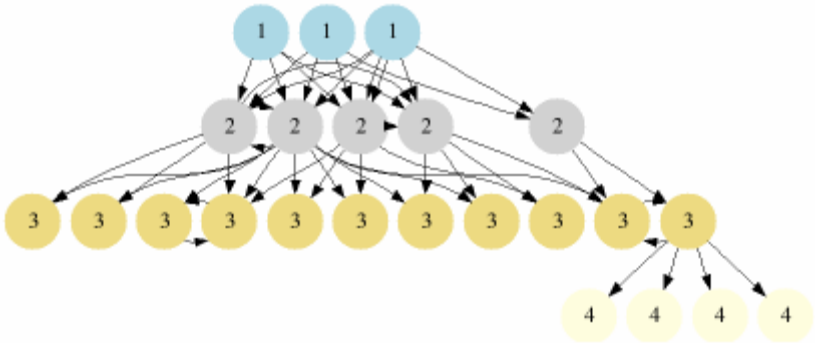
a e **b** estão na mesma referência de tempo

x e **y** estão na mesma referência de tempo



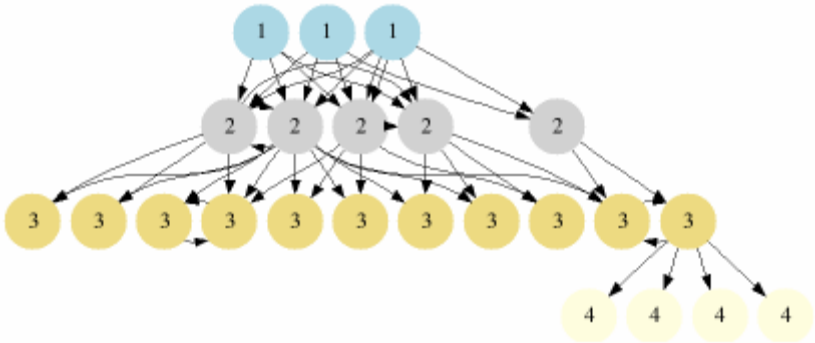
Network Time Protocol

- Servidores NTP são organizados em hierarquia
  - Divididos em camadas (estratos): 0 a 16
  - 0 geralmente é a refência de tempo (GPS ou relógio atômico)
  - 1 a 15 são servidores cada vez mais distantes da referência original
  - 16 é um servidor inoperante



Network Time Protocol

- Quanto mais perto da raiz, maior a exatidão do tempo
- Geralmente a diferença entre os estratos não é tão expressiva
- Melhor escolher um servidor mais próximo
  - menos atraso: menos incerteza



O que pode nos levar a construir um Sistema Distribuído?

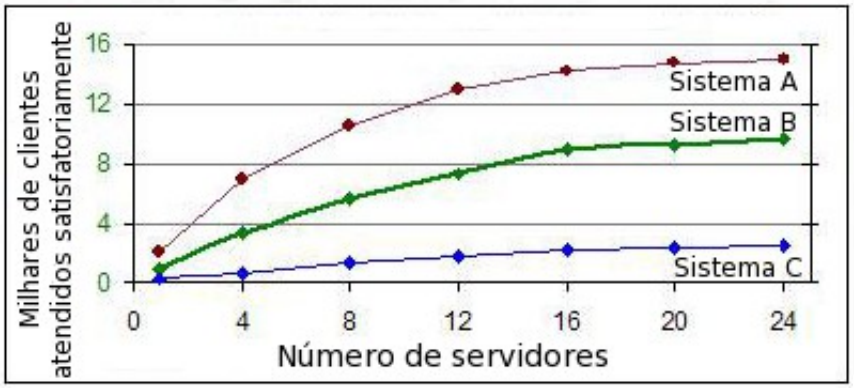
- **Desempenho**
  - Duas cabeças pensam melhor do que uma
- **Tolerância a falhas**
  - Redundância é a palavra-chave
- **Composição de serviços heterogêneos**
  - Reaproveitamento de componentes distribuídos que já oferecem certos serviços

Desafios na construção de Sistemas Distribuídos

- **Heterogeneidade**
  - Diferentes hardwares, softwares e implementações de um mesmo serviço
- **Abertura**
  - Possibilidade de extensão ou reimplementação sem impedimentos legais ou técnicos
- **Segurança**
  - Operações seguraças sobre os elementos do sistema

Desafios na construção de Sistemas Distribuídos

- **Escalabilidade**
  - Relação Recursos x Capacidade razoavelmente linear

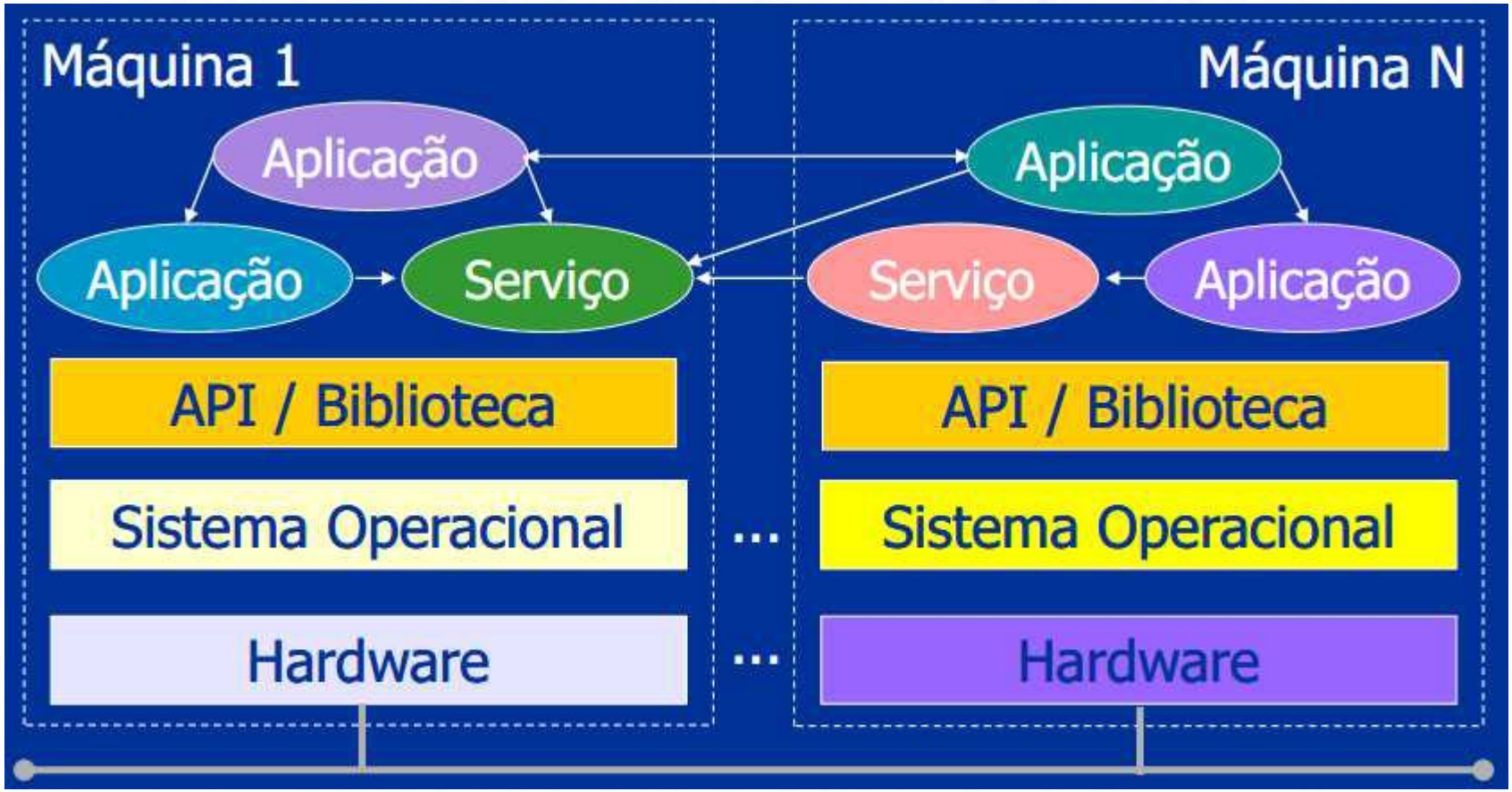


- **Tolerância a falhas**
  - Detecção, mascaramento e recuperação de falhas
- **Transparência**
  - Percepção do sistema como peça única

Suporte

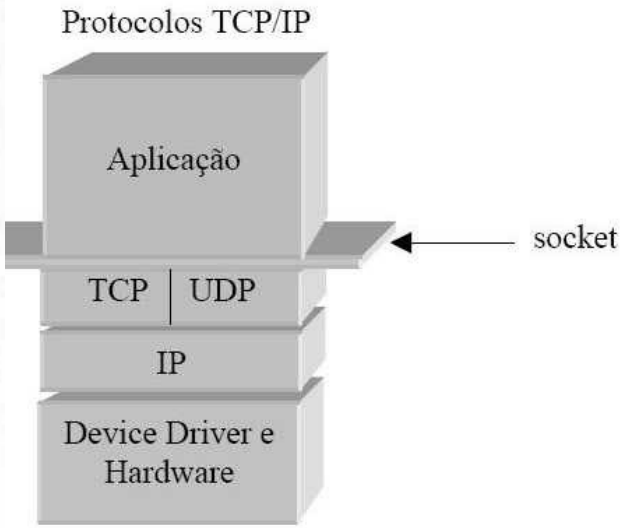
- Para vencer esses desafios, há soluções para suportar o desenvolvimento de SD:
  - **API e Bibliotecas**
    - Rotinas de comunicação entre processos distribuídos (Ex: sockets)
  - **Middlewares**
    - Suporte para criar/executar sistemas distribuídos (Ex: Corba, DCOM)
  - **Servidores de aplicação**
    - Permitem o acesso a aplicações via rede (Ex: Tomcat, JBoss, Glassfish)

API e Bibliotecas



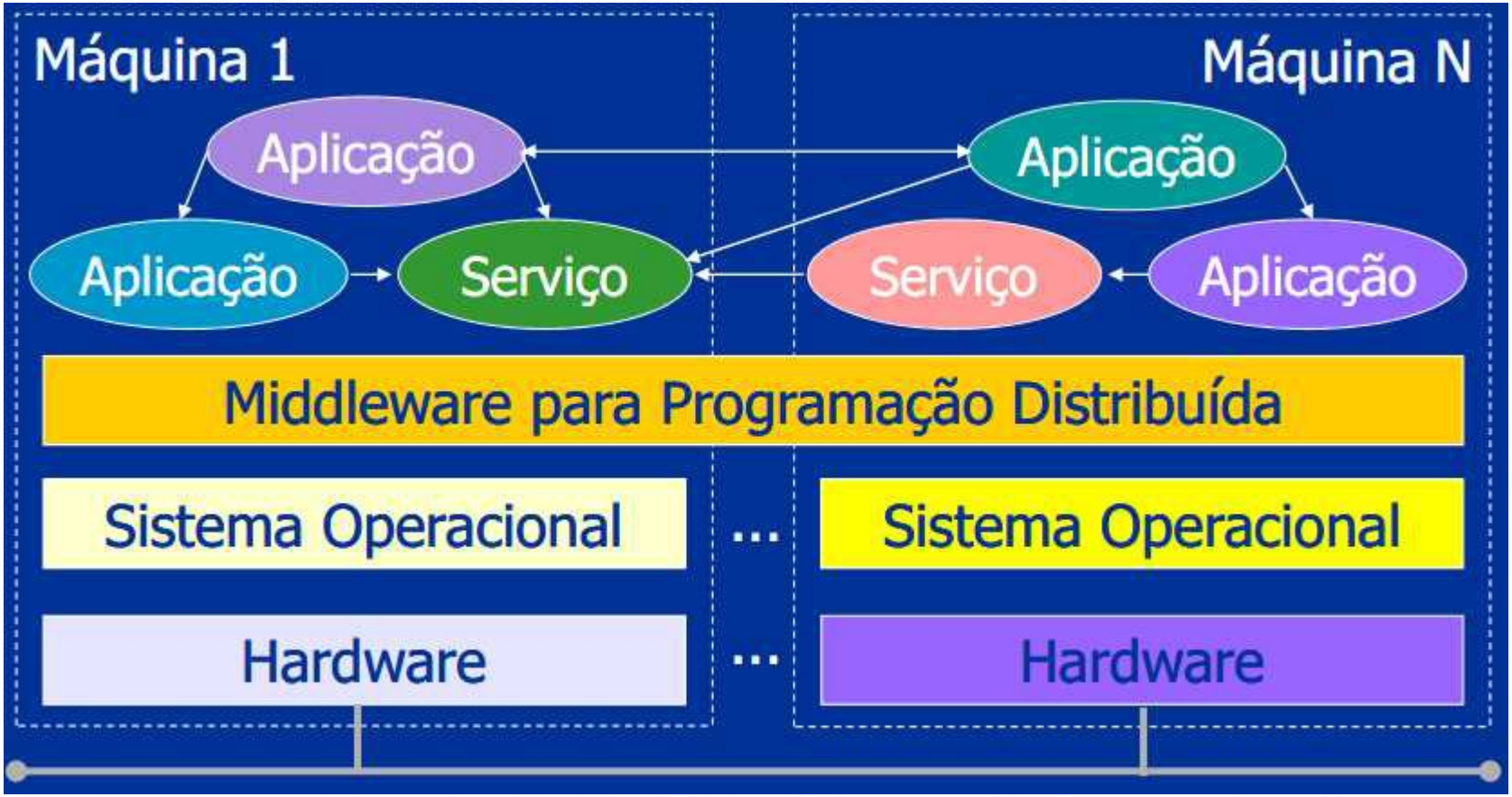
API e Bibliotecas

- Permitem que as aplicações troquem dados
- Fornecem primitivas de comunicação que podem ser utilizadas a partir do código
- Provêem acesso aos serviços de comunicação



- Sockets
- RPC
- RMI
- ...

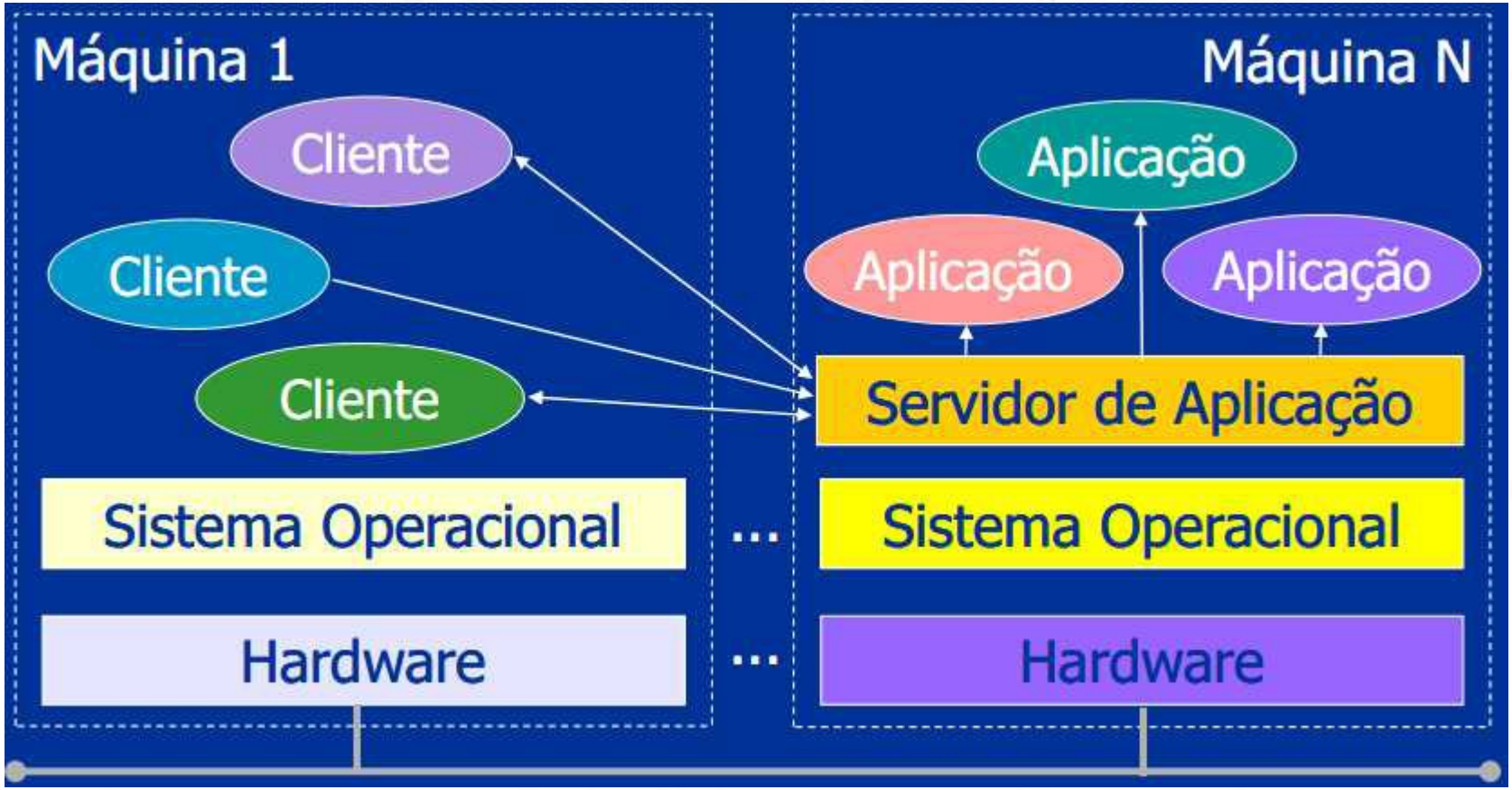
Middlewares



Middlewares

- Escondem as heterogeneidades da rede
- Oferecem serviços típicos de ambientes distribuídos funcionando como intermediário entre aplicações e SO
- São entidade a parte (não incorporados no código da aplicação distribuída)
- Transparência de acesso, migração, replicação etc
    - CORBA, DCOM, JMS ...

Servidores de aplicação



Servidores de aplicação

- Tipo de *middleware*
- Provê serviços às aplicações servidoras
- Bastante utilizado por simplificar processo de desenvolvimento
- Clientes não fazem uso direto
- Serviços de persistência, mensagens, tolerância a falhas etc

  - Tomcat, Glassfish,JBoss (RedHat)
    WebSphere (IBM), WebLogic (BEA) ...

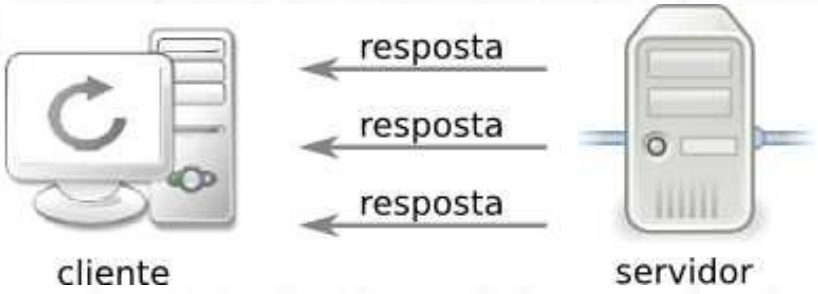
Modelos de comunicação

- A interação entre os elementos de um Sistema Distribuído independe do suporte de comunicação utilizado
- Pode ser de diversos tipos...
    - Cliente-Servidor
    - N-Camadas
    - Peer-to-Peer
    - Múltiplos Servidores

Cliente-Servidor

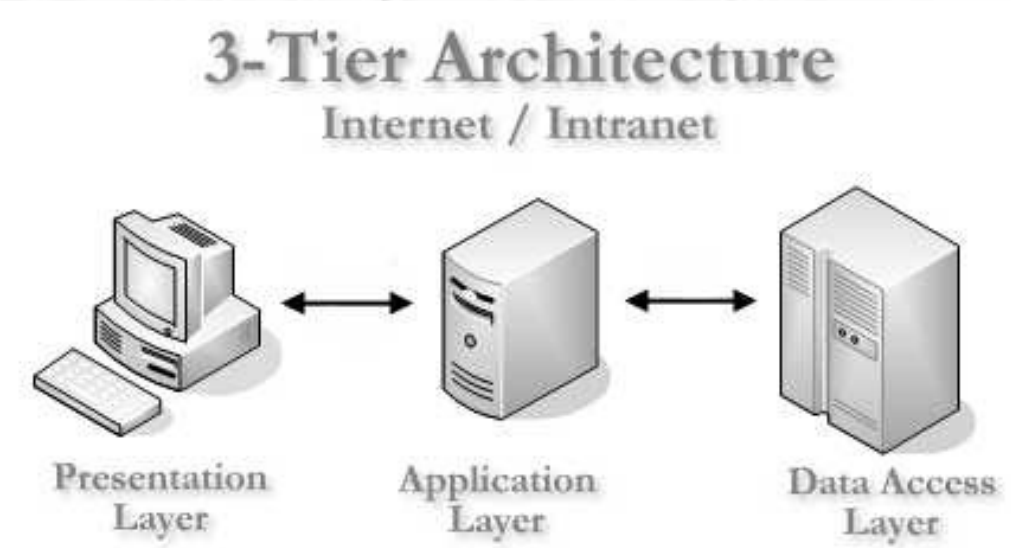
- Cliente ativo, servidor passivo

  - Variação: modelo *push*



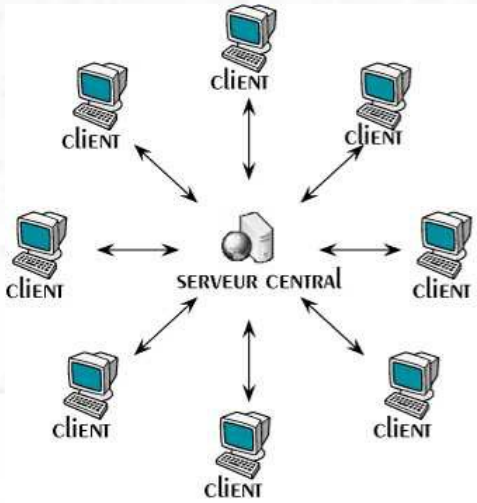
N-camadas

- Servidores intermediários são clientes de outros serviços
- Papéis bem definidos
- Cliente-servidor pode ser visto como caso particular

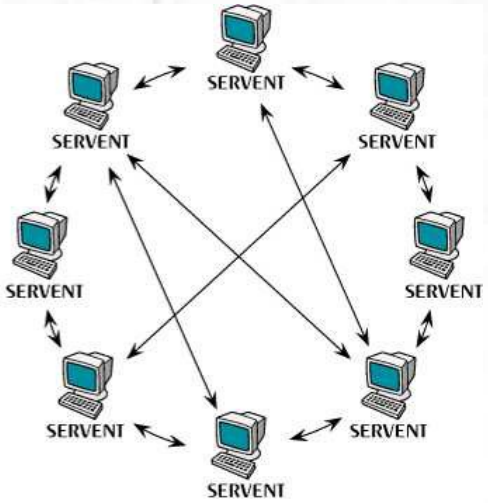


Peer-to-peer

- Todos são clientes e servidores de todos
- Os papéis não são definidos
- Muitas arquiteturas atuais são híbridas (cliente-servidor e peer-to-peer)



Architecture client-serveur



Architecture pair-à-pair

Múltiplos Servidores

- Diversos servidores se comportam como se fossem um grande servidor
- Não importa quem responde, importa que o cliente seja atendido