Universidade Estadual de Mato Grosso dos Sul
Curso de Ciência da Computação
Disciplina de Algoritmos Paralelos e Distribuídos

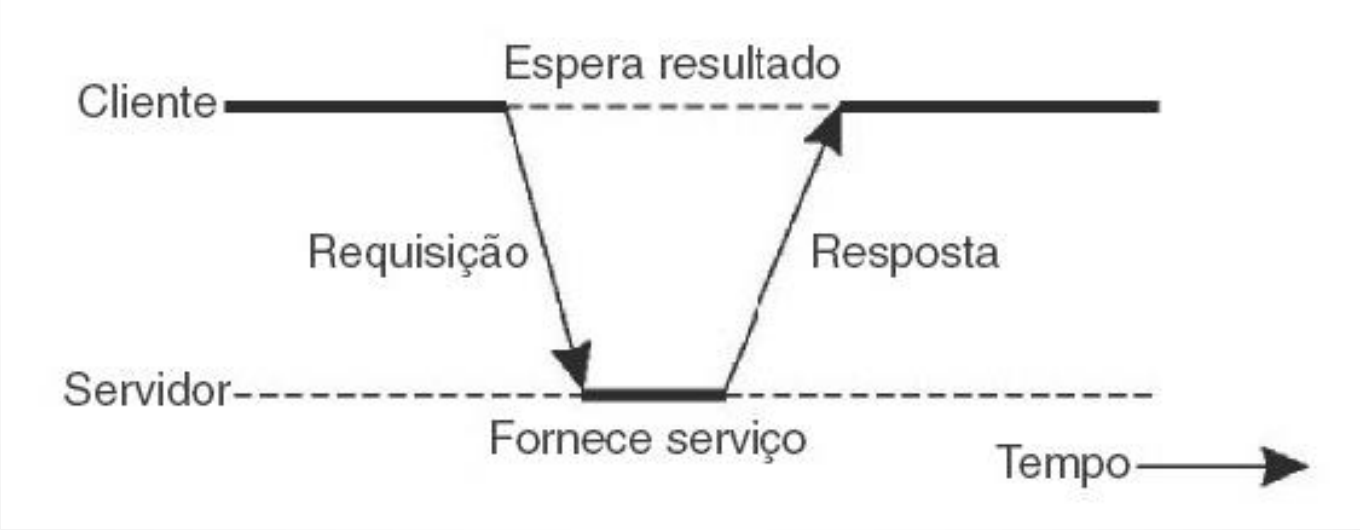
Aula 4 – Comunicação: Sockets, Tipos e RPC, Comunicação Orientada a Mensagem, Orientação Orientada a Fluxo e Multicast.

Comunicação entre Processos

- Ponto Central de qualquer SD.
- Como ocorre a troca de informações entre os processos de diferentes máquinas?
  - Comunicação – O que não é nada fácil!!
- Ideia: Obter modelos onde a complexidade da comunicação seja transparente para o desenvolvedor.

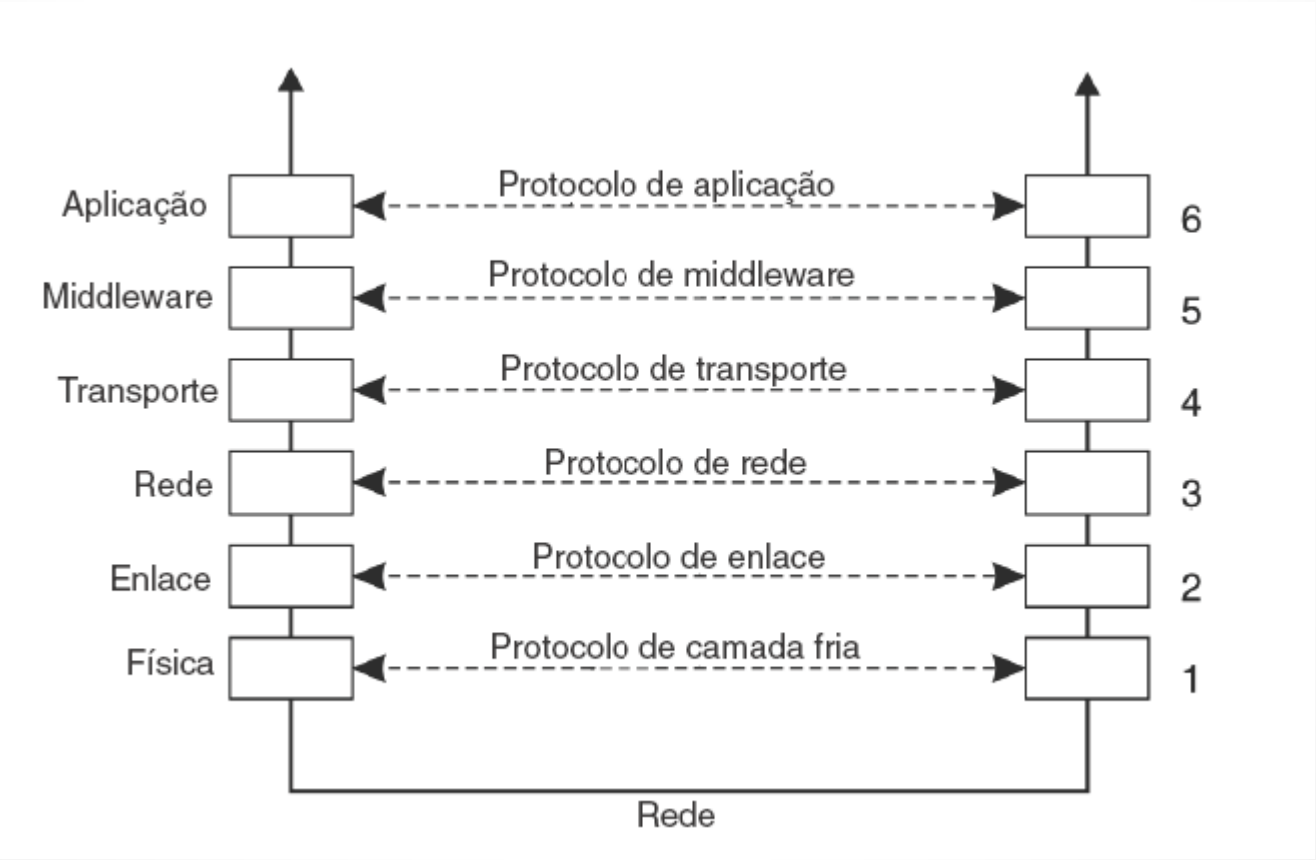
Modelo Cliente-Servidor

- Atores:
  - Servidores: fornecem um serviço específico.
    - Exemplo: servidor Web.
  - Clientes: Consomem um serviço fornecido por um determinado servidor.
- Modo Requisição – Resposta
  - Interação entre um cliente e um servidor.



Camada de Middleware

- Modelo de referência para comunicação em rede.



Sockets

- Meio pelo qual as diferentes máquinas trocam informações.
- Socket: É o ponto final de uma comunicação full-duplex entre dois processos.
  - Em uma analogia, o Processo seria a casa e o Socket seria a Porta
  - Socket: Porta entre o processo de aplicação e o protocolo de transporte.
- As informações são *strings* de *bytes* sem significado aparente.
- Não existe a transparência de distribuição: toda a comunicação é explícita através de funções send e *receive*.
  - Funções mais complexas devem ser feitas na camada de aplicação.

Podemos ter comunicação de alto nível independente da Aplicação?

- Sim!
  - Middleware de comunicação!
- São três os tipos:
  1. Chamada de Procedimento Remoto.
  2. Comunicação Orientada a Mensagens.
  3. Comunicação Orientada a Fluxo.

Tipos de Comunicação (Middleware)

- Persistência
  - Persistente: Mensagem é armazenada pelo middleware de comunicação durante o tempo que for necessário para entregá-la ao receptor.
  - Transiente: Mensagem é armazenada somente durante o tempo em que a aplicação remetente e a aplicação receptora estiverem executando.

Tipos de Comunicação (Middleware)

- Sincronização
  - Assíncrona: Remetente continua sua execução imediatamente após ter apresentado sua mensagem para a transmissão.
  - Síncrona: Remetente é bloqueado até saber se sua requisição foi aceita.
    - O middleware será encarregado da transmissão.
      - Requisição deverá ser entregue ao receptor.
      - Aguardar a resposta do receptor.

Tipos de Comunicação (Middleware)

- Granularidade
  - Discreta: Partes se comunicam por mensagens e cada mensagem forma uma unidade de informação completa.
  - Fluxo: Várias mensagens, sendo que as mensagens estão relacionadas uma com as outras pela ordem ou pela relação temporal.

Chamada de Procedimento Remoto

- Segundo Birrel e Nelson (1984), a chamada de procedimento remoto permite a processos chamar procedimentos localizados em outras máquinas.

- E o lado do Desenvolvedor?

  - Não precisará mais se preocupar com os detalhes de implementação de rede (chega de sockets).

Complicações do RPC

- A arquitetura de duas máquinas podem ser diferentes.
- Espaço de endereçamento diversos
- Passagem de parâmetros.

Como funciona o RPC?

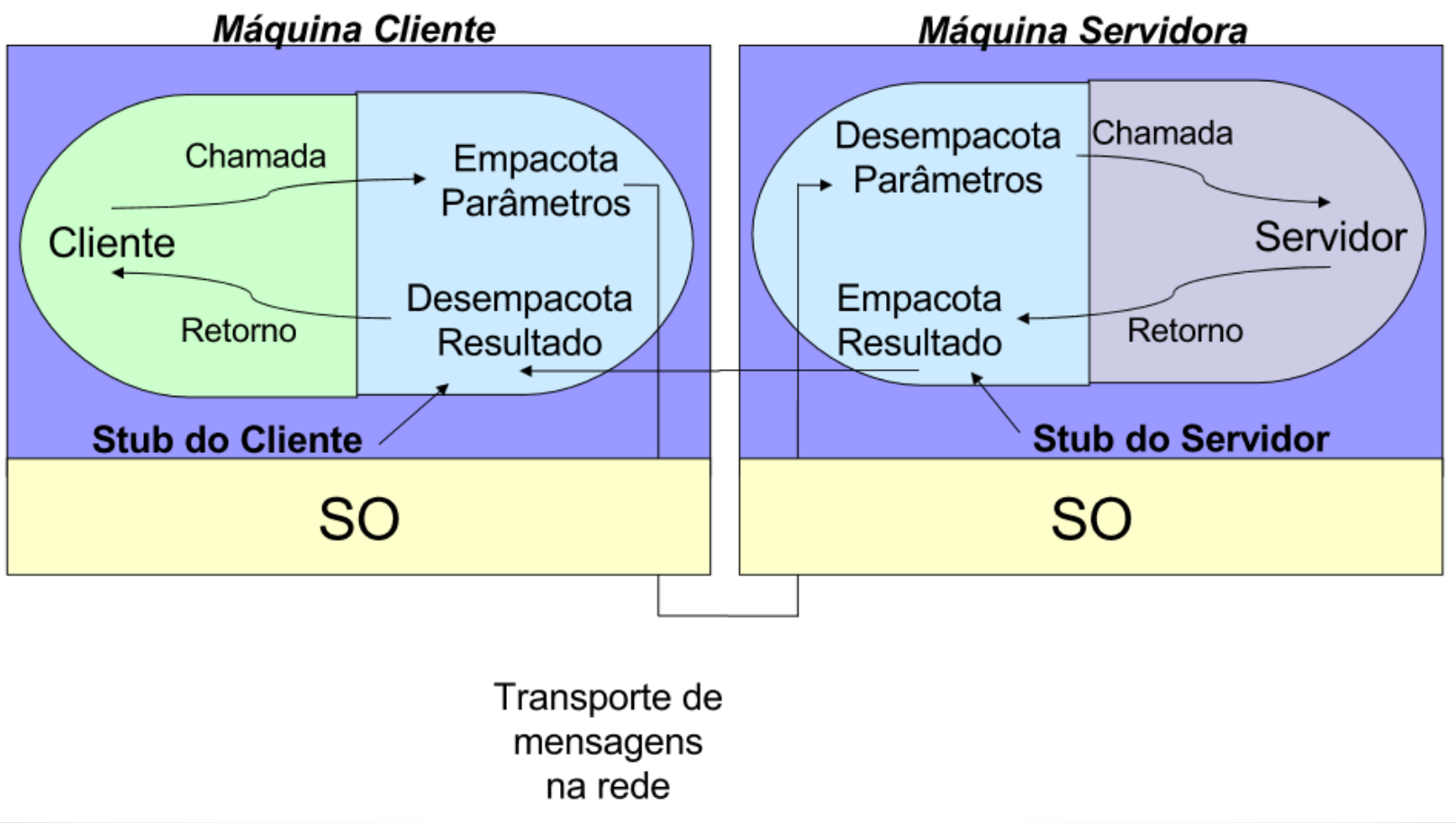
- A ideia é fazer com que uma chamada de procedimento remoto pareça com uma chamada local (transparência).

- A transparência é conseguida com o uso de *stubs* (apêndices).

  - *Stub* do Cliente: É responsável por empacotar os parâmetros em uma mensagem e enviar a mensagem para a máquina do servidor.
    - Quando a resposta chega, o resultado é copiado para o cliente .
  - *Stub* do Servidor: É o responsável por desempacotar os parâmetros, chamar o procedimento do servidor e retornar a resposta para a máquina do cliente.

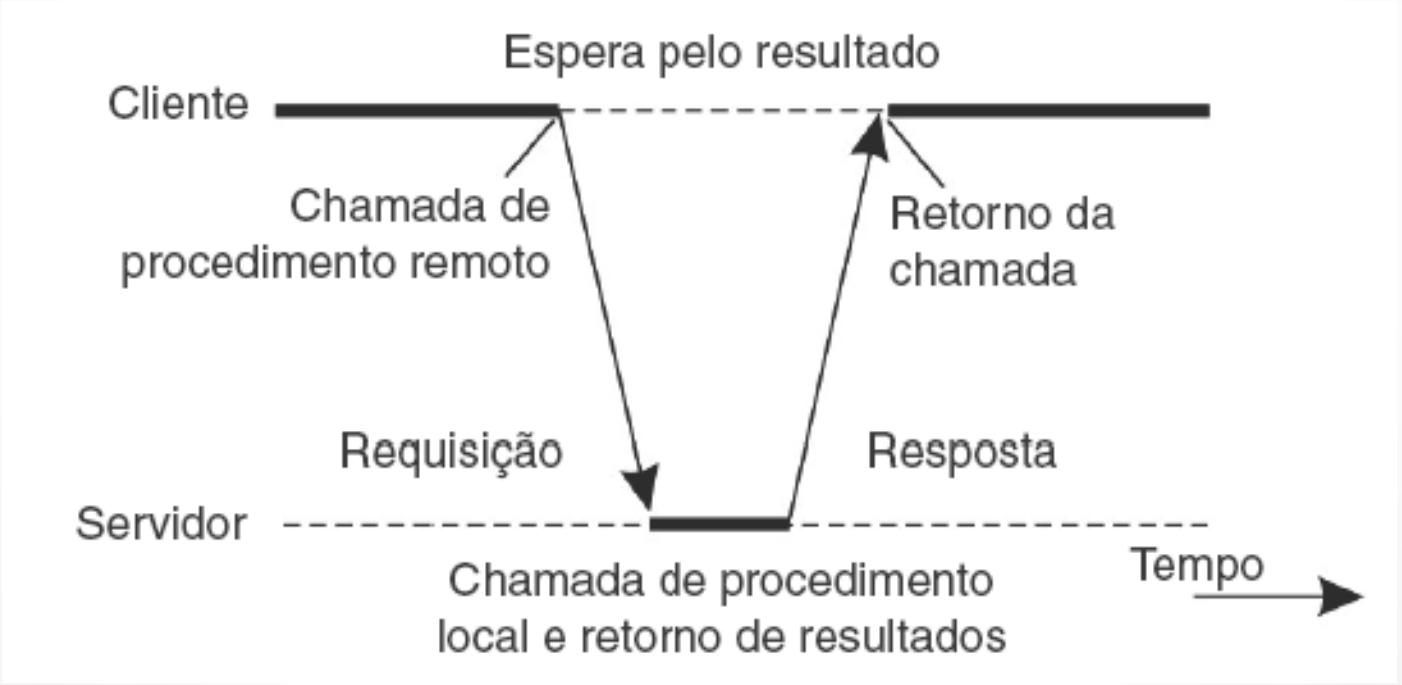
Funcionamento do RPC

- Exemplo:



Funcionamento do RPC

- Princípio de RPC entre um programa cliente e um programa servidor.



Passos executados em um RPC

1. Procedimento do lado cliente chama o *stub* cliente de modo usual.
2. O *stub* do cliente constrói uma mensagem e chama o SO.
3. O SO envia uma mensagem para o SO remoto.
4. O SO remoto repassa a mensagem para o *stub* do servidor.
5. O *stub* do servidor desempacota os parâmetros e chama o procedimento no lado servidor.
6. O procedimento servidor executa e retorna o resultado.
7. O *stub* do lado servidor empacota o resultado em uma mensagem e chama o SO.
8. O SO remoto envia uma mensagem para o SO da máquina cliente.
9. O SO do lado cliente passa a mensagem para o *stub* cliente.
10. *Stub* cliente desempacota o resultado, repassando-o para o ciente.

RPC – Passagem de Parâmetros

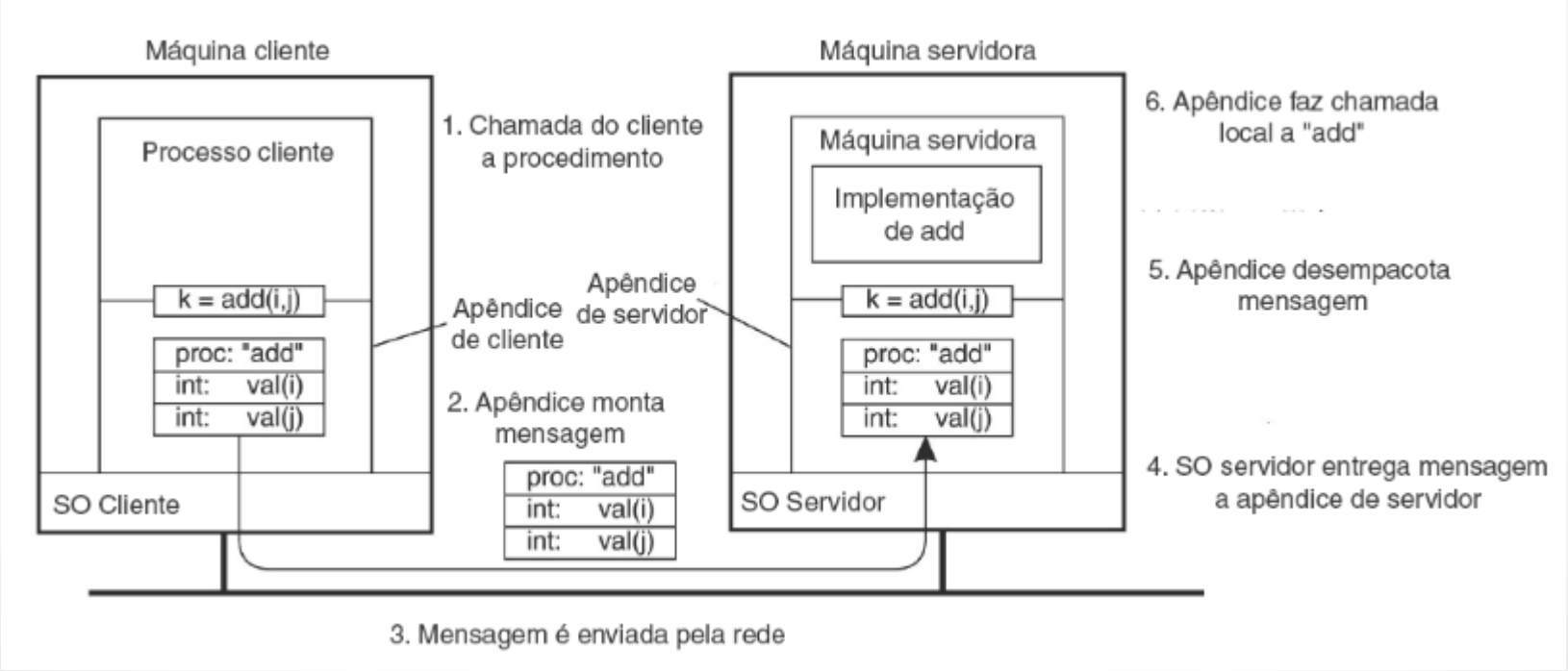
- A função do *stub* do lado cliente é pegar seus parâmetros, empacotá-los em uma mensagem e enviá-los ao *stub* do servidor (montagem de parâmetros).

- O que acontece quando as arquiteturas são diferentes?

  - Como proceder com a passagem de ponteiros (diferente espaço de endereçamento)?

RPC – Passagem de Parâmetros por Valor

- Etapas envolvidas para fazer um cálculo remoto por meio de um RPC.



RPC – Passagem de Parâmetros por Valor

- A chamada do procedimento remoto *add* somente funcionará se as máquinas do cliente e do servidor forem idênticas.

- Problemas:
  - Representação diferente para caracteres.
  - Ordenação de bytes.

- Solução:
  - O cliente "fala" qual o seu tipo e a conversão é feita pelo servidor por meio de pré-processamento quando os tipos forem diferentes.

RPC – Passagem de Parâmetros por Referência

- Problemas e Dificuldades:
  - Um ponteiro é significativo somente dentro do espaço de endereço do processo no qual está sendo usado.
  - Read(fd, buf, nbytes) → executado no servidor de arquivos.
    - fd → inteiro que indica um arquivo.
    - buf → endereço de vetores de caracteres.
    - nbytes → total de bytes a serem lidos.

- Mecanismo de passagem de parâmetros → copiar/restaurar.
  - A variável é copiada na pilha do cliente (passagem de parâmetro por valor).
  - Variável é manipulada no servidor.
  - Valor de retorno sobrescreve o valor original na pilha do cliente.

RPC – Passagem de Parâmetros por Referência

- No caso do *read, o stub* do cliente "sabe" que o segundo parâmetro aponta para um conjunto de caracteres.

- Suponha que o cliente saiba o tamanho do vetor .

- Solução:
  - Copiar o vetor para a mensagem e enviar ao servidor.
  - *Stub* do servidor, chama o servidor com um ponteiro para este vetor.
  - Modificação feita pelo servidor é armazenada diretamente no vetor que está no *stub*.
  - Ao enviar o vetor de volta ao *stub* do cliente, o vetor é copiado de volta ao cliente.

RPC – Linguagem de Programação de Interface - IDL

- Interface consiste em um conjunto de procedimentos que podem ser chamados por um cliente e que são implementados por um servidor.
- A utilização de uma interface simplifica consideravelmente as aplicações cliente-servidor baseadas em RPCs.
- Gera completamente *stubs* de cliente e servidor.
- Todos os sistemas de middleware baseados em RPC oferecem uma IDL para suportar o desenvolvimento de uma aplicação.

Resumo RPC

- Permite a um cliente o Acesso a um serviço remoto por meio de uma simples chamada a um procedimento local.
- Possibilita que programas clientes sejam escritos de modo simples.
- Pode localizar automaticamente o servidor correto.
- Estabelece a comunicação entre software cliente e software servidor.

Comunicação Orientada a Mensagem

- Como mecanismos de comunicação, RPC e RMI podem ser inadequados.
  - O que acontece caso não seja possível considerar que o receptor esteja sempre "vivo"?
  - Como contornar estas limitações?
    - Mensagens

Interface de Troca de Mensagens (MPI)

- Com multicomputadores de alto desempenho, desenvolvedores começaram a procurar primitivas orientadas a mensagem.
- Objetivo: Escrever com facilidade aplicações de alta eficiência.
- Necessidade de independência de hardware e de plataforma.
- Considerando como um padrão de troca de mensagens para escrever programas paralelos a serem executados em clusters.
- Comunicação Transiente → Mensagem é armazenada no sistema enquanto remetente e receptor estiverem ativos.

Middleware Orientado a Mensagem (MOM)

- Conhecidos como sistemas de enfileiramento de mensagens.
- Suporte para comunicação assíncrona persistente.
- Capacidade de armazenamento de médio prazo para as mensagens trocadas.
- Ideia: Aplicações se comunicam retirando e colocando mensagens em filas específicas.
- A mensagem é eventualmente entregue ao receptor.

MOM - Exemplo

- Consulta que abranja vários bancos de dados pode ser repartida em subconsultas que são repassadas para banco de dados individuais.

- Sistemas de enfileiramento de mensagens ajudam fornecendo meios básicos para empacotar cada subconsulta em uma mensagem e roteá-la até o banco de dados adequado.

- Aplicações: E-mail, fluxo de trabalho, groupware, processamento em lotes, integração de BD.

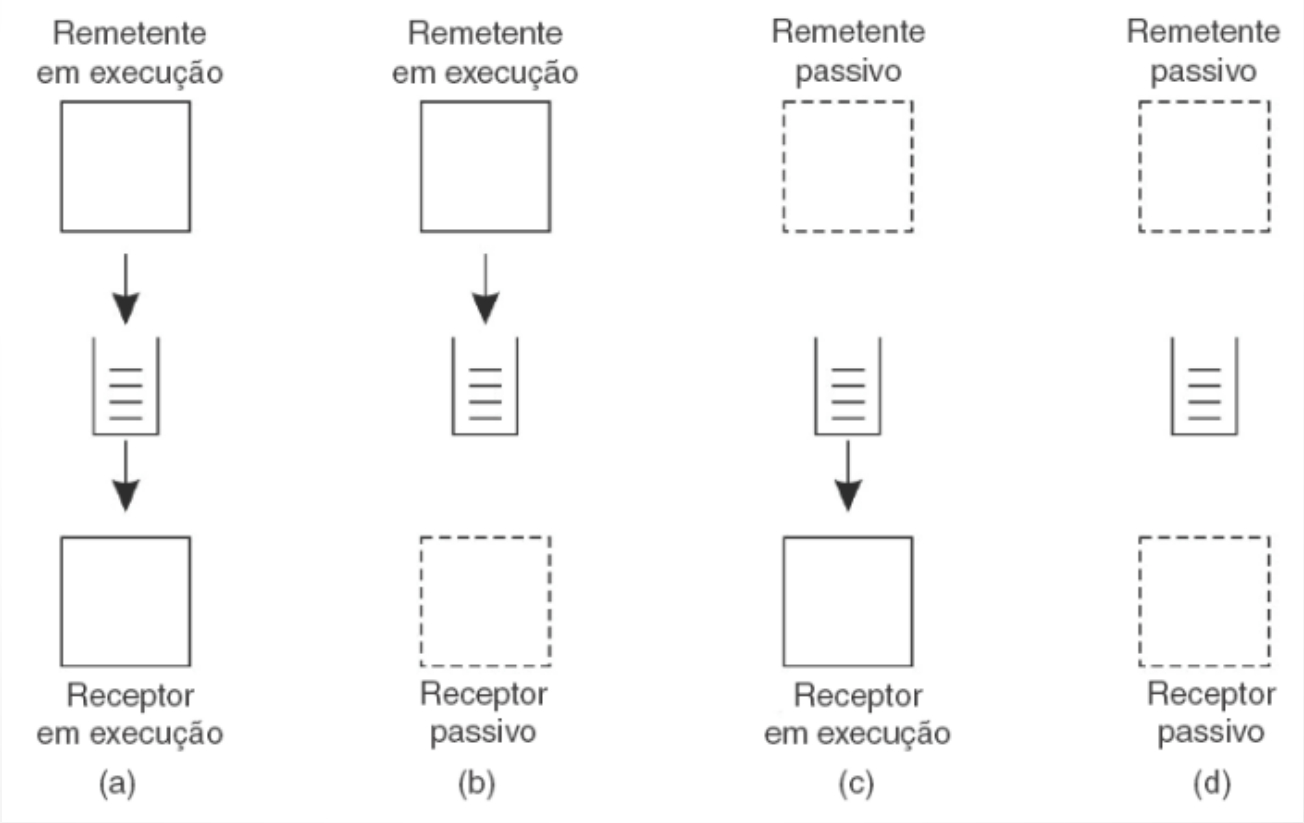
Como funciona o MOM?

1. Aplicações se comunicam inserindo mensagens em filas específicas.
2. Mensagens são repassadas por uma série de servidores de comunicação.
3. Mensagens são entregues ao destinatário, mesmo que ele não esteja em funcionamento quando a mensagem foi enviada.

   - Remetente e Receptor podem executar em completa independência.

Estados do Remetente e do Receptor - MOM

- Quatro combinações para comunicações fracamente acopladas que utilizam filas.



Características das Mensagens - MOM

- As mensagens podem conter qualquer tipo de dado.
- As mensagens devem ser adequadamente endereçadas.
- O endereçamento é feito com o fornecimento de um nome exclusivo da fila destinatária no âmbito do sistema.

Arquitetura - MOM

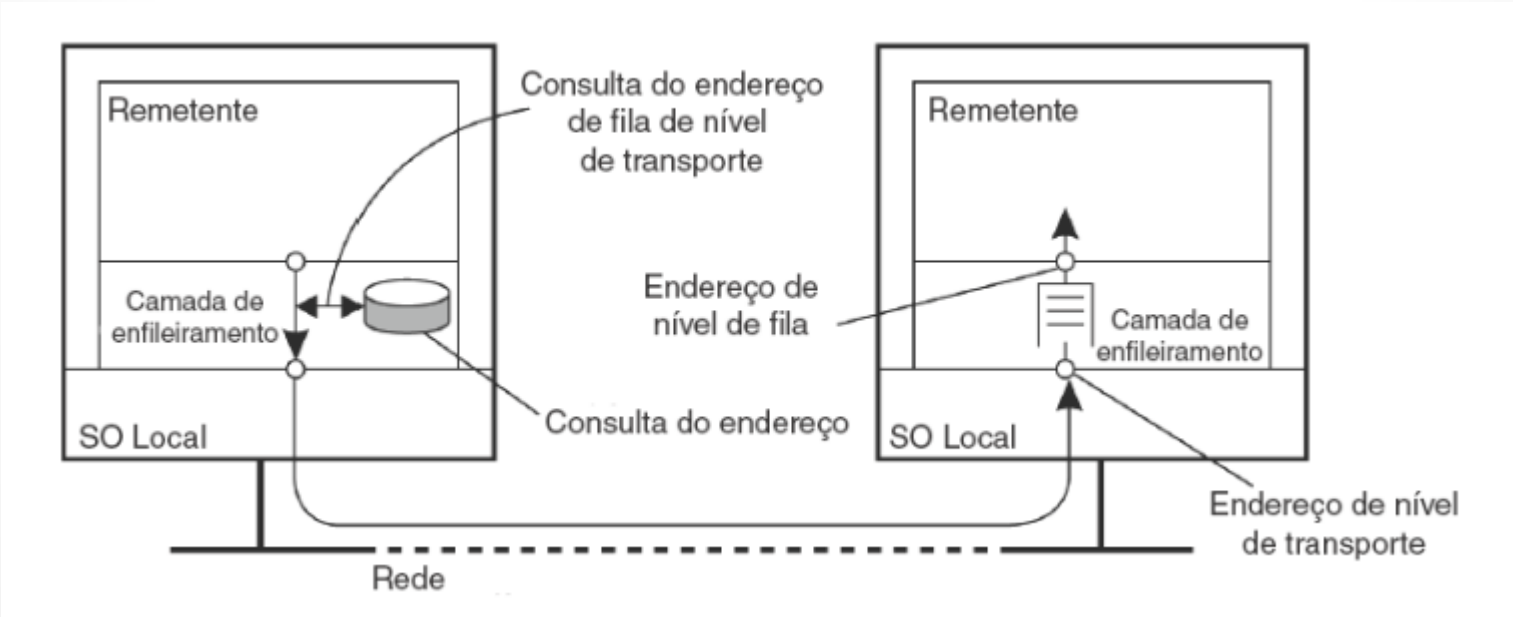
- Fila de Fonte
- Fila de Destino
- Gerenciadores de Fila
- Repassadores

Arquitetura - MOM

- Fonte de Fila: fila na qual o remetente envia a mensagem. Essas filas são filas locais do remetente.
- Fila de Destino: Uma mensagem colocada em uma fila contém a especificação de uma fila de destino para a qual ela deve ser transferida.
- Gerenciadores de Fila: Interage diretamente com a aplicação que esta enviando ou recebendo uma mensagem.
- Repassadores: Repassam mensagens que chegam para outros gerenciadores de fila.

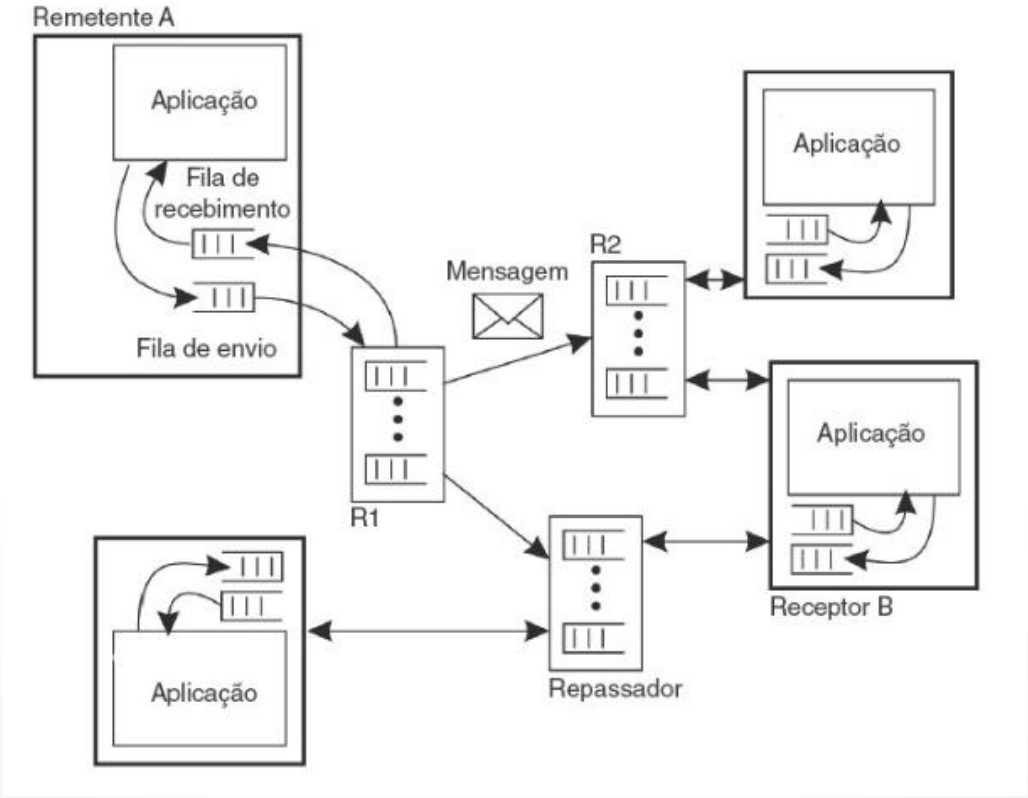
Arquitetura - MOM

- Relação entre endereçamento de nível de fila e endereçamento de nível de rede.



Arquitetura - MOM

- Organização geral de um sistema de enfileiramento de mensagens com repassadores.



MOM - Repassadores

- Repassadores podem ajudar a construir sistemas escaláveis de gerenciamento de fila.
- Atualizações de remoção e adição de filas devem ser feitas somente nos repassadores.
- Gerenciadores de fila somente devem saber onde está o repassador mais próximo.

Enfileiramento de Mensagens *versus* E-mail

- Sistemas de E-mail
  - Requisitos de filtragem automática de mensagens.
  - Não precisa garantir a entrega de mensagens, prioridade das mensagens, facilidades de registro, balanceamento de carga, tolerância a falha.

- Sistemas de Enfileiramento de Mensagens
  - Fornece recursos mais amplos para tratamento de diversas aplicações diferentes.
  - Possibilitam comunicação persistente entre processos.
  - Manipula acesso a banco de dados.
  - Realiza cálculos.
  - Prioridade de Mensagens.

Comunicação Orientada a Fluxo

- Tipos de Fluxos:
  - Fluxos Simples: Sequência simples de dados.
    - Exemplo: a voz.
  - Fluxos Complexos: Consiste em vários fluxos simples relacionados denominados subfluxos.
    - Há uma relação temporal entre os subfluxos
    - Exemplo: Transmissão de um filme → Vídeo, som, legenda, entre outros.

Qualidade de Serviço (QoS)

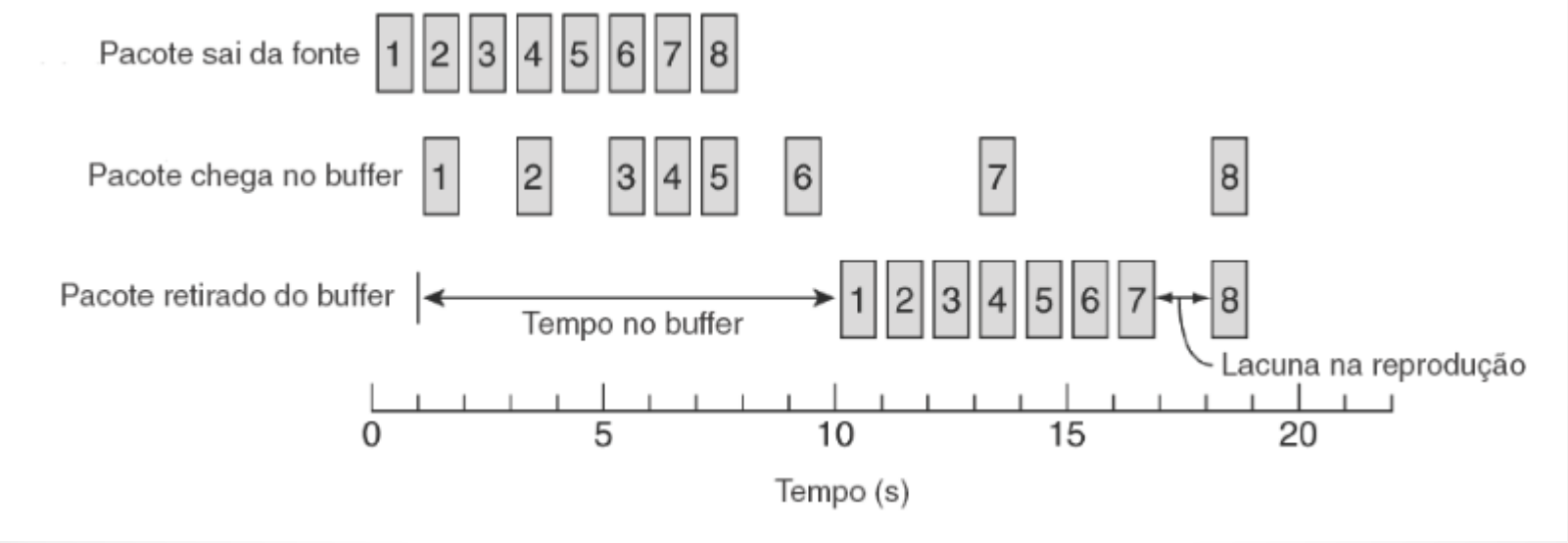
- Requisitos que descrevem o que é necessário para garantir que as relações temporais em um fluxo possam ser preservadas.

- Está relacionada com:
  - Pontualidade
  - Volume
  - Confiabilidade

- Sistemas Operacionais e redes não suportam QoS.
  - O serviço fornecido pelo IP é "*best effort*".

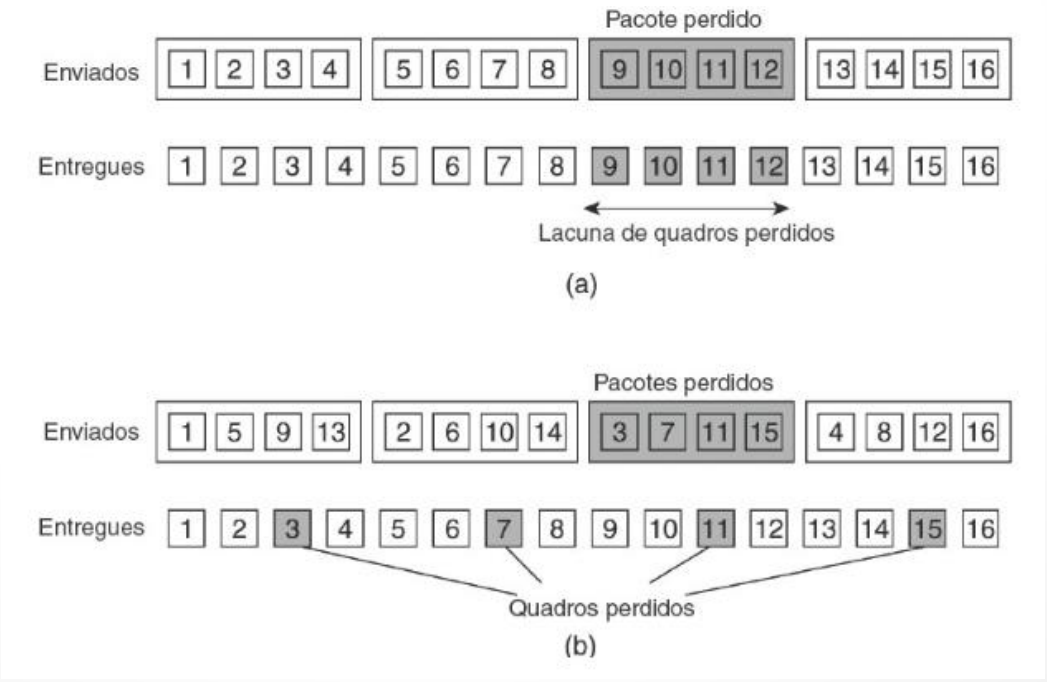
Como garantir QoS?

- Bufferização para reduzir variância de atraso no receptor.



Como garantir QoS?

- Correção de erro de envio.
- Exemplo: Efeito da perda de pacotes em uma transmissão não intercalada em (a); e efeito da perda de pacotes em uma transmissão intercalada em (b).

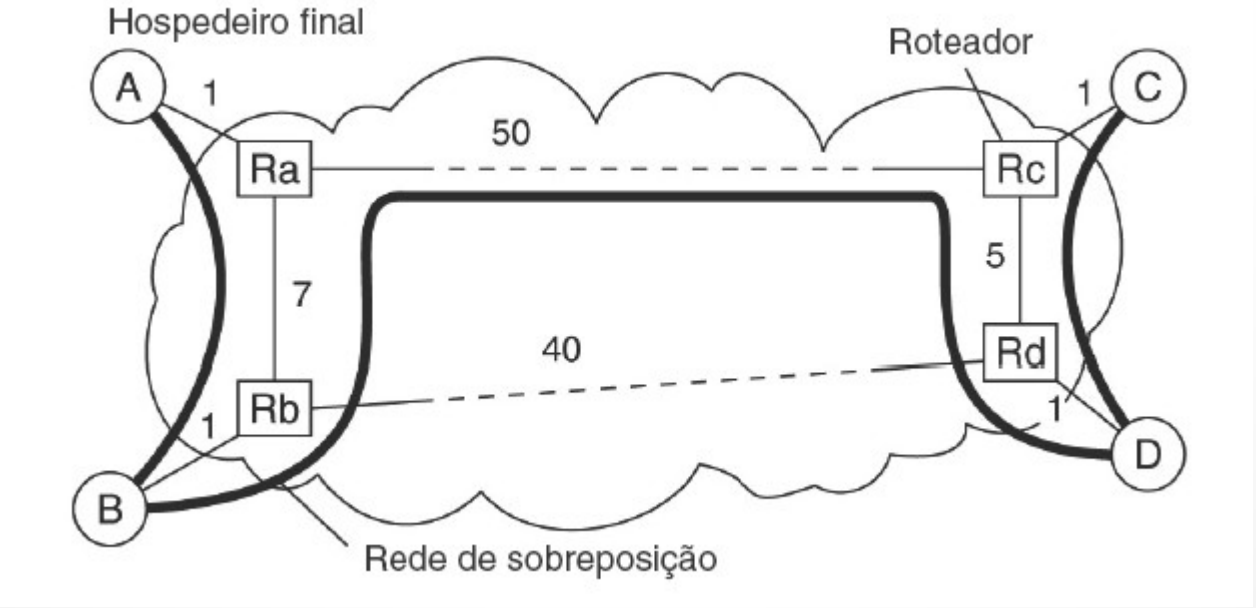


Comunicação MultiCast

- Multicast no nível de aplicação
  - Ideia básica: Os nós se organizam em uma rede de sobreposição que é usada para disseminar informações para seus membros.
  - Nós na rede de sobreposição podem cruzar vários enlaces físicos:
    - Neste caso, o roteamento pode não ser ótimo.
  - Construção do Overlay
    - Árvore: Único caminho entre cada par de nós.
    - Malha: Cada nó terá vários vizinhos, em geral, existem vários caminhos entre cada par de nós.

Multicast no Nível de Aplicação

- No caso da estrutura em árvore, o problema é construir uma árvore eficiente.
- Exemplo: Relação entre enlaces em uma rede de sobreposição e as rotas reais no nível de rede.



Qualidade na Estrutura

- Estresse de Enlace: Quantas vezes uma mensagem atravessa o mesmo enlace?
  - Exemplo: Uma mensagem de A a D atravessa Ra, Rb duas vezes.
- Penalidade de Atraso Relativo: Razão entre o atraso entre dois nós na sobreposição e o atraso que esses dois nós sofreriam na rede subjacente.
  - Exemplo: Mensagens de B a C possuem um atraso de 59 no overlay, mas de 47 no nível de rede, que gera uma penalidade de 1.255.
- Custo da Árvore: Parâmetro de medição global, relacionado com a minimização dos custos agregados de enlaces.
  - Exemplos: Atraso entre dois nós finais → *Spanning Trees*.

Disseminação de Dados – Algoritmos Epidêmicos

- Ideia básica: Propagar informações rapidamente entre um grande conjunto de nós usando somente informações locais.
- Não há nenhum componente que coordene a disseminação de informações.
- Atualizações para um item de dado específico são iniciadas em um único nó, evitando conflito de escrita.

Modelos de Disseminação de Dados

- Há três tipos de nós:
  - Infectado:
    - Se contiver dados que está disposto a espalhar para outros nós.
  - Suscetível:
    - Nó que ainda não tenha visto esses dados.
  - Removido:
    - Nó atualizado que não está disposto ou capacitado para propagar os dados.

Leitura, Estudo e Exercícios

- Este material tem sua fonte no livro:
  - Sistemas Distribuídos: Princípios e Paradigmas.
  - Andrew Tanenbaum e Marteen Steen
  - Capítulo 4

  - Distributed System – Concepts and Design
  - George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair.
  - Capítulos 4, 5 e 6.