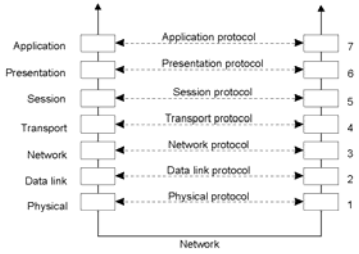
Comunicação

Tipos de redes

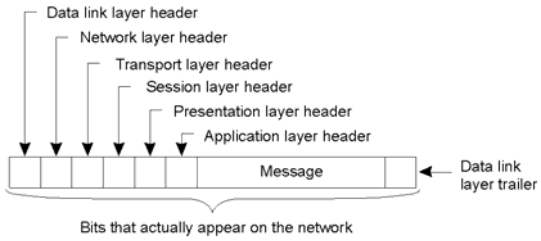
	Range	*Bandwidth (Mbps)*	*Latency (ms)*
LAN	1-2 kms	10-1000	1-10
WAN	worldwide	0.010-600	100-500
MAN	2-50 kms	1-150	10
Wireless LAN	0.15-1.5 km	2-11	5-20
Wireless WAN	worldwide	0.010-2	100-500
Internet	worldwide	0.010-2	100-500

Protocolos (1)



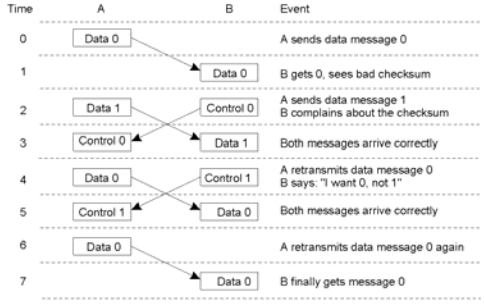
níveis, interfaces e protocolos no modelo OSI.
Protocolos orientados a conexão e sem conexão

Protocolos (2)

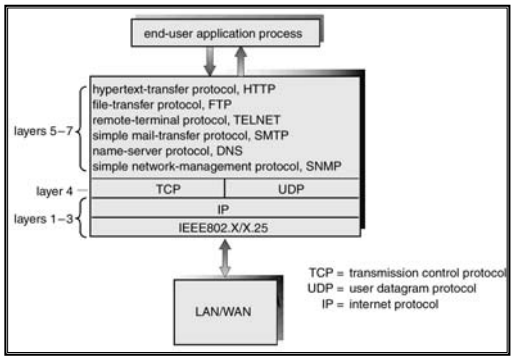


mensagem como aparece na rede.
Cada nível adiciona seu cabeçalho

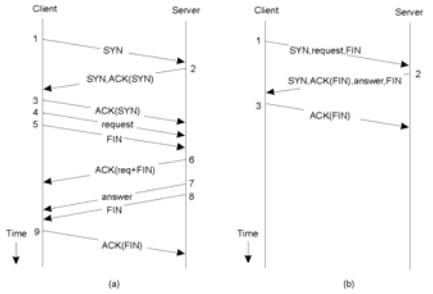
Nível de ligação



Tratativas entre recebedor e enviante no nível de ligação.

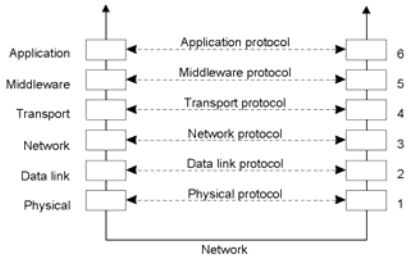


TCP Cliente-Servidor



- operação normal de TCP.
- TCP Transacional.

Protocolos Middleware



modelo de referência adaptado para comunicação em rede.
Protocolos para suportar serviços de Middleware

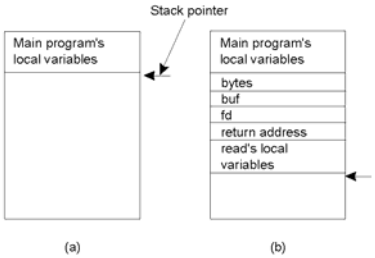
Comunicação em Sistemas Distribuídos

Chamada Remota de Procedimento - RPC

O mecanismo de RPC integra o protocolo RR usado para comunicação Cliente/Servidor com as linguagens de programação convencionais permitindo clientes comunicar com servidores através de chamadas de procedimentos.

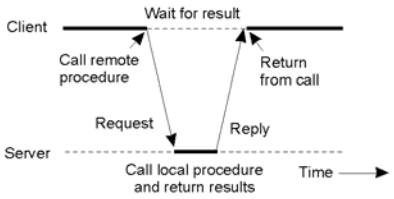
A chamada remota segue o modelo da chamada local sendo que o procedimento chamado executa em um processo diferente normalmente em um computador diferente.

Chamada de Procedimento Convencional



- Passagem de parâmetro chamada local: pilha antes da chamada
- pilha enquanto procedimento está ativo

Stubs Cliente e Servidor

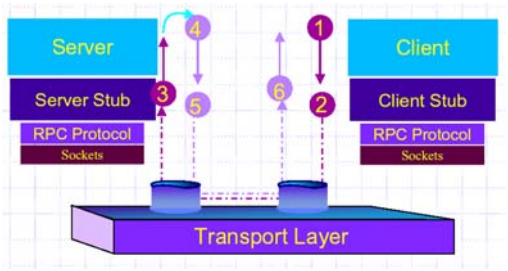


Princípio do RPC entre um programa cliente e servidor.

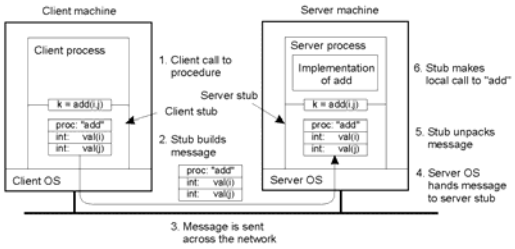
Passos em RPC

- Cliente chama stub cliente
- Stub client constrói mensagem, chama SO local
  - SO cliente envia mensagem para SO remoto
  - SO remoto entrega mensagem para stub servidor
- Stub servidor retira parametros, chama servidor
- Servidor realiza trabalho, retorna resultado para stub
- Stub servidor coloca na mensagem, chama SO local
  - SO servidor envia mensagem para SO cliente
  - SO cliente entrega mensagem para stub cliente
- Stub retira resultado, retorna para cliente

Passos em RPC



Passando Parâmetros (1)



Trocando parâmetros em RPC

Passando Parâmetros (2)

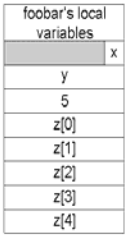
- mensagem original no Pentium
- mensagem depois de recebida em SPARC
- mensagem depois de invertida

Especificação de parâmetros e geração de stubs

- procedimento
- Mensagem correspondente.

```
foobar( char x; float y; int z[5] )
{
  ....
}
```

(a)



(b)

Chamada Remota de Procedimento (RPC)
Parâmetros

A escolha da semântica de passagem de parâmetros é crucial para o projeto de um mecanismo de RPC :

- Call-by-Value : cópia dos valores na mensagem

- Call-by-Reference , Pointers: cópia da estrutura de dados (array) na mensagem e refaz na volta, no cliente (call-by-copy/restore).

- Passagem de Pointers é um problema

  - proibir

  - copiar só quando necessário

Chamada Remota de Procedimento (RPC)
Especificação de um servidor de arquivos

Specification of file_server *version* 2.0

long read (*in char* fname[n_size], *out char* buffer [b_size],
in long bytes, *in long* position);

long write(*in char* fname[n_size], *in char* buffer [b_size],
in long bytes, *in long* position);

int create(*in char* fname[n_size], *in int* mode);

int delete(*in char* fname[n_size]);

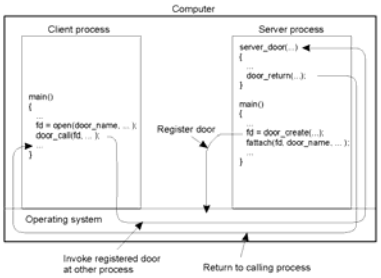
end_specification;

Chamada Remota de Procedimento (RPC)
Modelos Extendidos

RPC se tornou um padrão de fato para comunicação em sistemas distribuídos. Extensões ao modelo original foram propostas para solucionar alguns problemas:

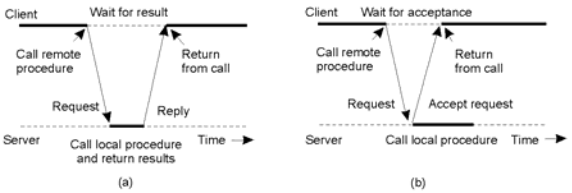
- RPC leve
- RPC Assíncrona

Doors



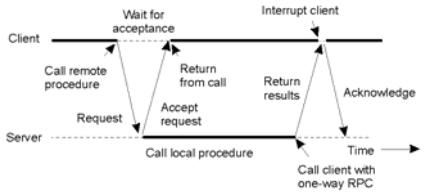
O principio de uso de doors como mecanismo de IPC.

RPC Assíncrona (1)



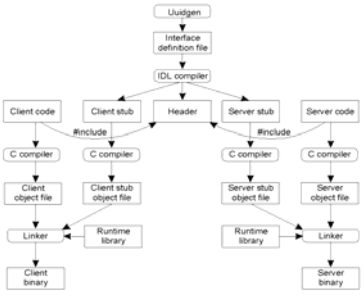
- interconexão entre cliente e servidor RPC tradicional
- interação usando RPC assíncrona

RPC Assíncrona (2)

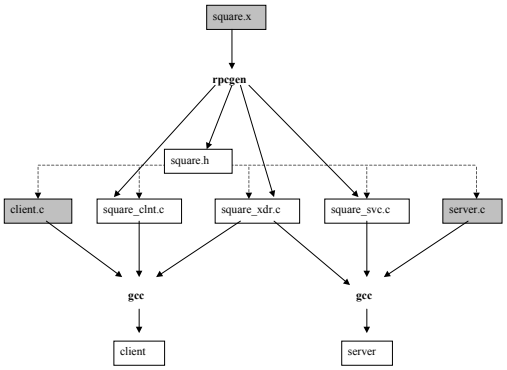


cliente e servidor interagindo com RPC assíncrona

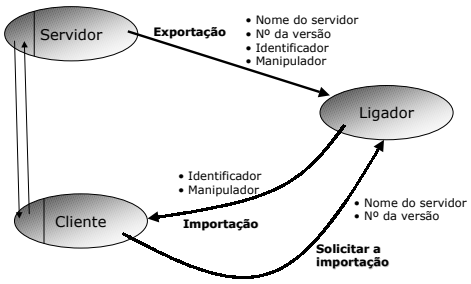
Programa Cliente / Servidor



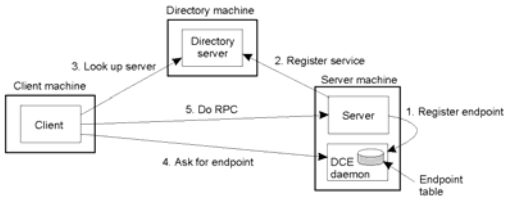
Passos escrevendo cliente e servidor em DCE RPC.



Ligação Cliente Servidor



Ligação Cliente Servidor



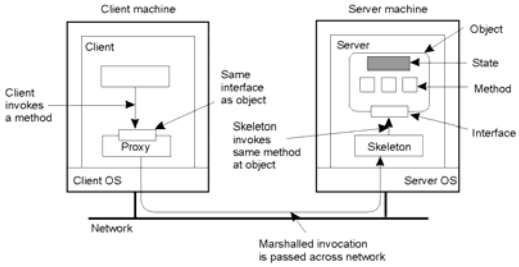
C/S binding em DCE.

Invocação Remota de Métodos

RPC tornando-se padrão de comunicação e tecnologia baseada em objetos apresentando aspectos importantes com relação a adaptabilidade, porque não aplicar o princípio RPC em objetos.

- encapsulamento de operações e dados (objetos)
- operações (métodos) acessadas via interfaces
- objeto servidor (coleção de objetos)
- stub cliente (proxy) implementa interface
- stub servidor (skeleton)

Objetos Distribuídos



Organização comum de um objeto remoto com proxy no lado cliente.

Objetos Distribuídos

- Objetos tempo de compilação: objetos nível de linguagem, a partir dos quais proxy e skeleton são automaticamente gerados.
- Objetos tempo de execução: implementados em qualquer linguagem, necessitam utilização de um adaptador de objeto que faz com que a implementação pareça como um objeto.

- Objeto transiente: existe apenas em virtude de um servidor, se o servidor termina o objeto tambem.
- Objeto persistente: existe independente de um servidor, se o servidor termina o estado do objeto e código permanecem (passivos) no disco.

Ligação Cliente-Objeto

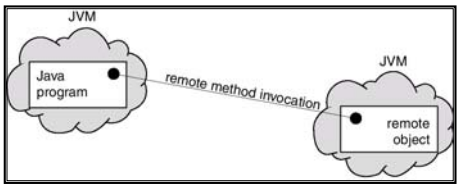
- Referência a Objeto : ter uma referência ao objeto permite um cliente se ligar a um objeto:
  - Referência denota servidor, objeto e protocolo de comunicação
  - Cliente carrega código stub associado
  - Stub é instanciado e inicializado para objeto específico

- Formas de Ligação:
  - Implícito: invoca métodos diretamente no objeto referenciado.
  - Explícito: cliente deve primeiro explicitamente ligar-se ao objeto antes de invoca-lo.

Remote Method Invocation

Remote Method Invocation (RMI) é um mecanismo próprio de Java, similar a RPC.

RMI permite um programa Java em uma máquina invocar um método remoto em um objeto remoto.

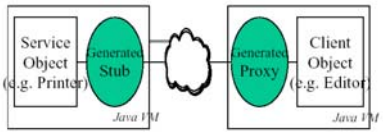


Remote Method Invocation

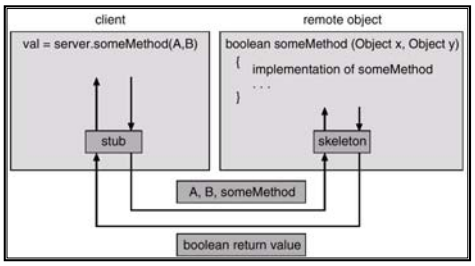
Remote Method Invocation (RMI) é o mecanismo Java, que gera as classes proxy de forma automática.

O usuário codifica os objetos cliente e serviço.

O compilador RMI gera o código responsável pela comunicação na rede.

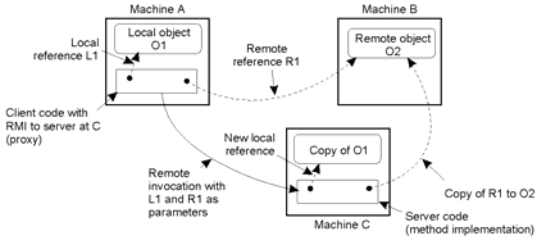


Marshalling dos Parâmetros



http://java.sun.com/docs/books/tutorial/rmi/overview.html

Passagem de Parâmetros



Passando um objeto por referência ou por valor.

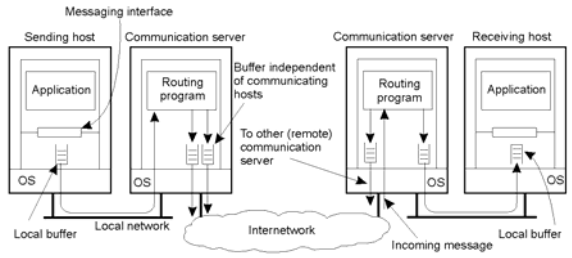
Comunicação orientada a mensagens

- Comunicação Síncrona X Assíncrona.
- Sistema de Fila de Mensagens.
- Conversores de Mensagens (Message Brokers).

Comunicação orientada a mensagens

- Chamada de procedimento remoto e invocação remota de objetos contribuem para esconder a comunicação em sistemas distribuídos, geralmente baseado no modelo C-S (síncrono).
- Algumas vezes nenhum destes mecanismos é adequado.
  - Não podemos assumir que o lado recebedor está executando qdo requisição é emitida.
  - Natureza síncrona de RPC e RMI não é adequada, cliente não pode fazer outra coisa enquanto espera.
- Algumas vezes o mecanismo adequado é mensagens, ou seja, o modelo síncrono não é apropriado (mail, news).

Comunicação Orientada a Mensagens (1)



Organização geral de um sistema de comunicação no qual nodos (hosts) estão conectados através de uma rede

Comunicação Orientada a Mensagens (2)

- Aplicações executadas no host, ambiente oferece interface para comunicação
  - processos enviam mensagens (colocadas em filas) uns para os outros
  - enviante não precisa esperar por resposta imediata, pode fazer outras coisas
  - Ambiente prove tolerância a falhas
- Exemplo: correio eletrônico
  - host executa agente do usuário (aplicação de e-mail)
  - cada host está conectado a 1 servidor
  - interface no host usuário para agente enviar msg para dest.
  - Agente submete msg, host manda para servidor local
  - servidor verifica destino, envia para servidor alvo
    - armazena buffer do receptor
  - senão armazena msg no local

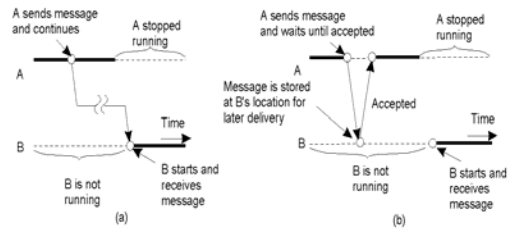
Persistência e Sincronismo em Comunicação (1)

- **Comunicação Persistente**
  - mensagem submetida para transmissão é armazenada no sistema de comunicação até entregar para o recebedor
  - aplicação enviante não precisa continuar executando
  - aplicação recebedora não precisa estar executando na sub.
- **Comunicação Transiente**
  - mensagem é armazenada apenas enquanto as aplicações enviante e recebedora estão executando
  - se não é possível entregar msg ao próximo servidor msg descartada
  - serviço de transporte, se roteador não pode entregar msg para próximo (destino), descarta msg

Persistência e Sincronismo em Comunicação (2)

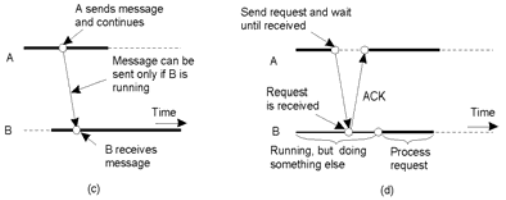
- **Comunicação assíncrona**
  - enviante continua depois de submeter msg
  - msg armazenada em buffer local no enviante ou no servidor de comunicação
- **Comunicação síncrona**
  - o enviante é bloqueado até sua msg ser armazenada em buffer local no recebedor ou realmente entregue
  - a forma mais forte é qdo o enviante fica bloqueado até o recebedor processar a msg
- Várias combinações destes tipos de comunicação

Persistência e Sincronismo em Comunicação (3)



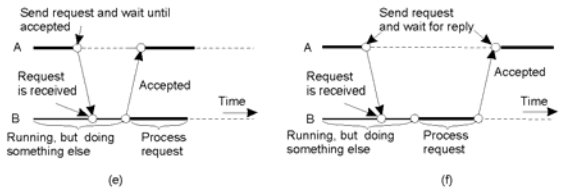
- Comunicação persistente assíncrona
- Comunicação persistente síncrona

Persistência e Sincronismo em Comunicação (4)



- Comunicação transiente assíncrona - UDP
- Comunicação transiente síncrona, baseada na recepção

Persistência e Sincronismo em Comunicação (5)



- Comunicação transiente síncrona, baseada na entrega
- Comunicação transiente síncrona, baseada na resposta - RPC

Comunicação Transiente orientada a mensagem

Sockets

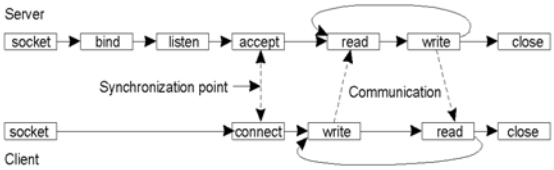
Interface Passagem de Mensagem (MPI)

Sockets (1)

Primitive	Meaning
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Primitivas Socket para TCP/IP.

Sockets (2)



Comunicação orientada a conexão usando sockets.

Message-Passing Interface (MPI)

Primitiva	Significado
MPI_bsend	Anexa mensagem de saída à um buffer local de envio
MPI_send	Envia uma mensagem e espera até copiar para buffer local ou remoto
MPI_ssend	Envia uma mensagem e espera até recebedor iniciar
MPI_sendrecv	Envia uma mensagem e espera resposta
MPI_isend	Passa referência para mensagem de saída e continua
MPI_issend	Passa referência para mensagem de saída e espera até recebedor iniciar
MPI_recv	Recebe uma mensagem; bloqueia se não tem
MPI_irecv	Verifica se existe mensagem, mas sem bloqueio

Algumas primitivas de MPI.

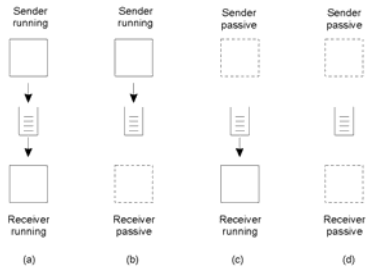
Comunicação Persistente Orientada a Mensagem

Sistemas de filas de mensagens, fornece suporte para comunicação persistente assíncrona, enviante e receptor não precisam estar ativos.

Diferente de sockets e MPI, suportam transferência de msg que podem levar minutos.

Abordagem geral

Modelo Fila de Mensagens (1)



Quatro combinações para comunicação fracamente acoplada usando filas.

Modelo Fila de Mensagens (2)

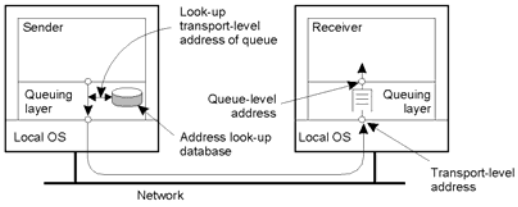
Primitive	Meaning
Put	Anexa uma mensagem em uma fila especificada
Get	Bloqueia até que a fila não esteja vazia e remove a primeira mensagem
Poll	Verifica uma fila determinada e remove mensagem se tiver. Sem bloqueio.
Notify	Instala um tratador para ser chamado quando uma mensagem é colocada na fila.

Interface básica para uma fila em um sistema de fila de mensagens.

Arquitetura Geral de um Sistema de Fila de Mensagens (1)

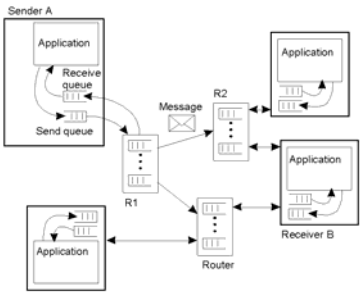
- Mensagens colocadas em *filas fontes* (local, LAN)
- Mensagens lidas de filas locais
- Mensagens contém endereço destino
- Coleção de filas distribuída através de várias máquinas
- Sistema tem mapeamento de filas para localizações na rede (DNS)
- Filas são gerenciadas por *gerentes de fila*
- *Gerentes de fila* interagem com a aplicação, podem funcionar como roteadores (serviço de nomes, segurança, tolerância a falhas, multicast)

Arquitetura Geral de um Sistema de Fila de Mensagens (2)



Relação entre endereçamento nível de fila e nível de rede.

Arquitetura Geral de um Sistema de Fila de Mensagens (3)

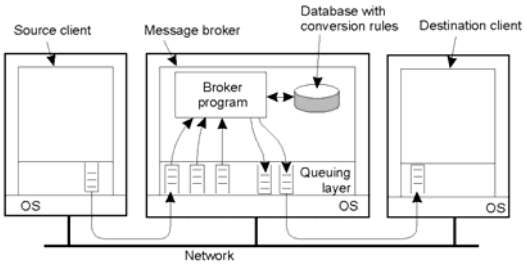


Organização geral de um sistema com roteadores.

Conversores de Mensagem

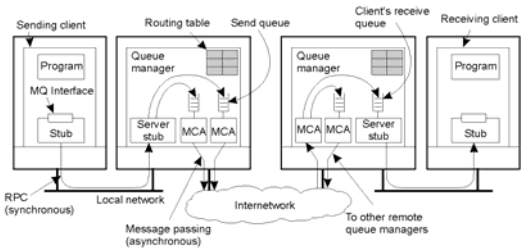
- Aplicação de sistema de mensagem integração de aplicações (novas,existentes) em único, coerente sistema de informações distribuído.
- Integração requer aplicações entendam mensagens recebidas, formatos iguais (enviante, receptor).
- Mesmo que alguns formatos comuns existam, a abordagem geral é aprender a conviver com formatos diferentes, tentar fornecer meios para conversão.
- Sistemas de filas de mensagens conversões são tratadas por nodos especiais na rede de filas, **message brokers**

Message Brokers



Organização geral de um conversor de mensagem em um sistema de fila de mensagem.

Exemplo: IBM MQSeries

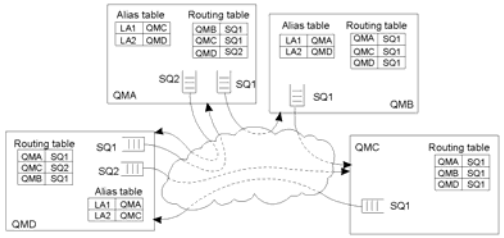


Channels

Attribute	Description
Transport type	Determines the transport protocol to be used
FIFO delivery	Indicates that messages are to be delivered in the order they are sent
Message length	Maximum length of a single message
Setup retry count	Specifies maximum number of retries to start up the remote MCA
Delivery retries	Maximum times MCA will try to put received message into queue

Some attributes associated with message channel agents.

Message Transfer (1)



The general organization of an MQSeries queuing network using routing tables and aliases.

Message Transfer (2)

Primitive	Description
MQopen	Open a (possibly remote) queue
MQclose	Close a queue
MQput	Put a message into an opened queue
MQget	Get a message from a (local) queue

Primitives available in an IBM MQSeries MQI

Comunicação orientada a stream

- Comunicação concentrado na troca de unidades de informação completas e mais ou menos independentes, sem importar que tempo ela se dá, ou seja, tempo não influi na correção.
- Existem formas de comunicação onde tempo é crucial, áudio.
- Facilidades que sistemas distribuídos oferecem para trocar informações dependentes do tempo, áudio e vídeo streams.
- Mídia contínua - relação temporal entre os diferentes itens de dados é fundamental para interpretação correta do significado. Considere movimento, uma série de imagens sendo mostradas com espaços uniformes (30-40mseg por imagem).
- Mídia discreta - relação temporal não é fundamental para interpretação correta.

Comunicação orientada a stream

• Capturar troca de informação dependente de tempo - *stream de dados (data stream)*, seqüência de dados, aplicado tanto a mídia contínua quanto discreta (pipes, TCP/IP).

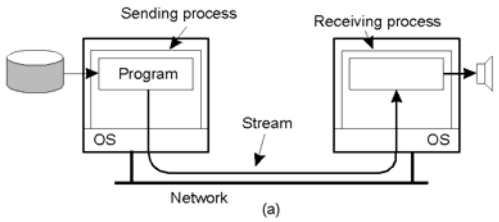
• Transmissão assíncrona, itens no stream são transmitidos um após outro sem restrições temporais, arquivo.

• Transmissão síncrona, existe um retardo máximo definido para cada unidade sem importar se a transferência é mais rápida, sensor de temperatura.

• Transmissão isócrona, é necessário que unidades sejam transferidas no tempo, sujeito a um máximo e mínimo retardo (jitter limitado).

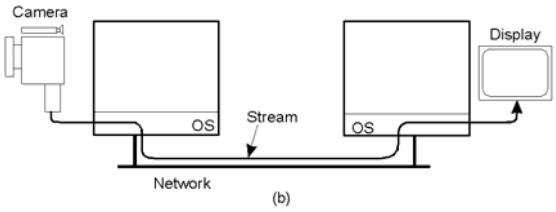
• Stream pode ser simples (mono) ou complexo (estéreo, filme)

Stream de dados (1)



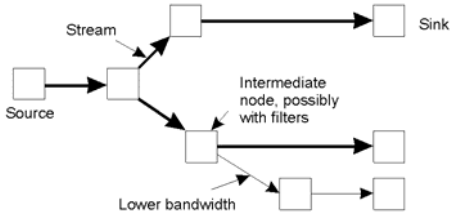
Preparação de um stream entre 2 processos através de uma rede.

Stream de dados (2)



Preparação de um stream direto entre 2 dispositivos através de uma rede.

Stream de dados (3)



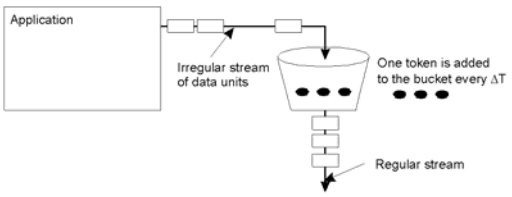
Um exemplo de multicasting um stream para vários receptores.

Especificando QoS (1)

Características da Entrada	Serviço necessário
• tamanho máximo unidade (bytes) • taxa de tokens (bytes/sec) • tamanho recipiente (bytes) • taxa máxima de transmisão (bytes/sec)	•Sensibilidade de perda (bytes) •intervalo de perda (µsec) •perda consecutiva (data units) •retardo mínimo notado (µsec) •jitter máximo (µsec) •Qualidade da garantia

Especificação de fluxo.

Especificando QoS (2)

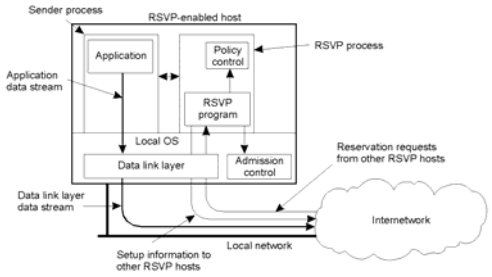


O princípio do algoritmo token bucket.

Estabelecendo um Stream

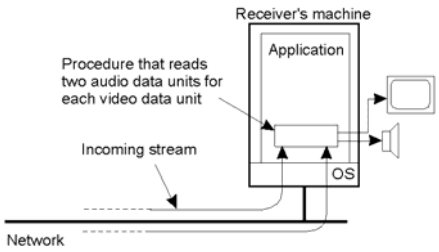
Após especificação o SD pode alocar recursos (banda, buffer, processamento) para estabelecer um stream que satisfaz os requisitos de QoS.

Estabelecendo um Stream



Organização básica de protocolo para reserva de recursos em um sistema distribuído - RSVP

Synchronization Mechanisms (1)



Synchronization Mechanisms (2)