Comunicação em Sistemas Distribuídos

Bruno M. Carvalho
Sala: 3B2
Horário: 35T34

Comunicação em Sistemas Distribuídos

- Protocolos – regras que os processos que estão se comunicando tem de seguir
- Protocolos de níveis
- O modelo cliente-servidor
- Remote Procedure Call (RPC)
- Comunicação de grupos

Protocolos de Camadas

- Como não existe memória compartilhada, toda a comunicação em SDs acontece através de troca de mensagens
- Qual o significado dos bits enviados? Qual a voltagem usada para sinalizar 0 e 1? Como se detecta o bit final da mensagem, ou que uma mensagem foi danificada ou perdida?
- A International Standards Organization (ISO) desenvolveu um modelo de referência para interconexão de sistemas abertos (OSI)

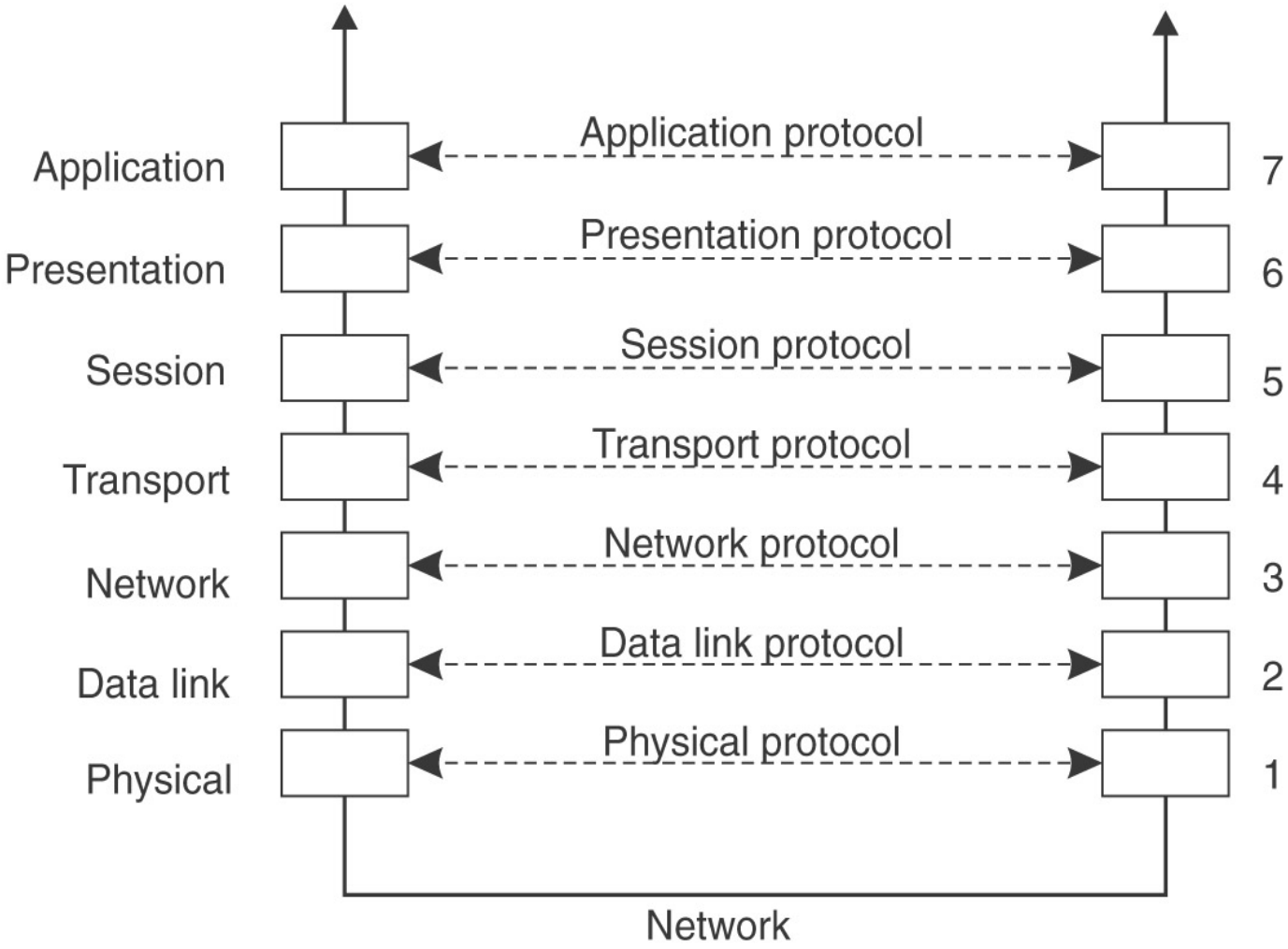
Protocolos de Camadas

- Um sistema aberto pode se comunicar com qualquer outro sistema aberto utilizando os protocolos do modelo OSI em 1983
- Modelo abstrato de redes
- Para que um grupo de computadores se comuniquem em uma rede, todos devem usar os mesmos protocolos de comunicação
- Protocolos orientados a conexão (ligação telefônica)
- Protocolos sem conexão (carta)

Protocolos de Camadas

- Divisão da comunicação em 7 camadas
- Processo cria mensagem que ao passar pelas várias camadas de protocolos é partida e tem cabeçalhos adicionados a ela
- Divisão em camadas torna sua implementação mais flexível, facilitando atualizações e correções
- Redes não necessitam implementar todas as camadas
- Cada camada efetua função bem definida, e elas são definidas para minimizar comunicação entre elas
- Não detalha serviços

Protocolos de Camadas



Estrutura

- Camada presta serviços para camada superior
- Camada usa serviços da camada inferior
- Camadas de mesmo nível “comunicam-se”
- Uma camada apenas toma conhecimento da camada inferior
- Interação entre camadas feita por serviços
- Divisão de tarefas
- Facilita abstração

Camada Física

- Transmissão de sequências de bits sobre meio físico
- Especifica
  - voltagens ecorrentes
  - tempos
  - conectores e pinagens
  - meio físico utilizado
  - aspectos eletrônicos e mecânicos
- Domínio da engenharia eletrônica
- Não trata de correção de erros na transmissão

Camada de Enlace

- Organiza sequências de bits em conjuntos de bits chamados *frames*
- Reconhece início e fim de frames
- Detecta perdas de frames e requisita retransmissão

Camada de Rede

- Encaminha informação da origem para o destino (roteamento)
- Controla fluxo de transmissão entre sub-redes (controle de congestão)
- Funções de contabilização
- Estabelece esquema único de endereçamento independente da sub-rede utilizada
- Permite conexão de sub-redes heterogêneas

Camada de Transporte

- Divide e reagrupa a informação binária em pacotes
- Garante a sequência dos pacotes
- Assegura a conexão confiável entre origem e destino da comunicação
- Primeira camada que estabelece comunicação origem-destino

Camada de Sessão

- Gerencia sessões de comunicação
- Sessão é uma comunicação que necessita armazenar estados
- Estados são armazenados para permitir re-estabelecimento da comunicação em caso de queda da comunicação
- Ex: Retomar transferências de arquivos

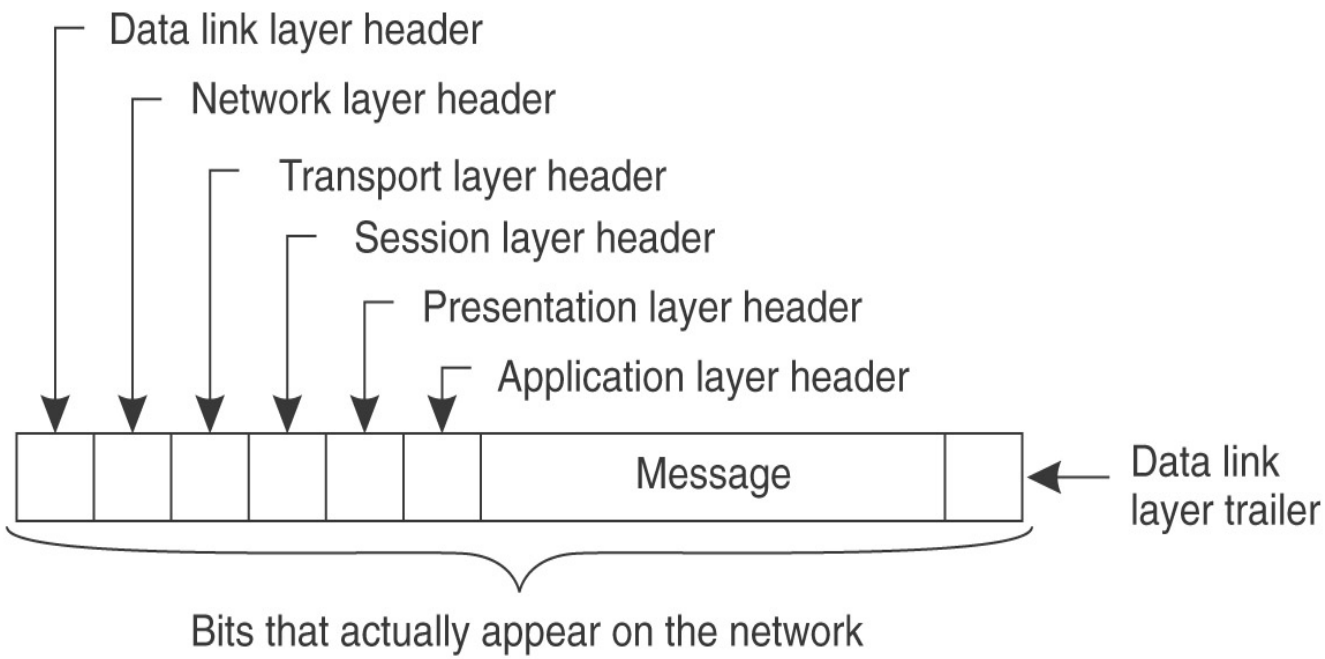
Camada de Apresentação

- Trata da representação dos dados em alto nível
- Adoção de sistema padronizado de representação de caracteres
- Adoção de códigos de representação numérica padrão
- Compressão de dados
- Codificação de dados

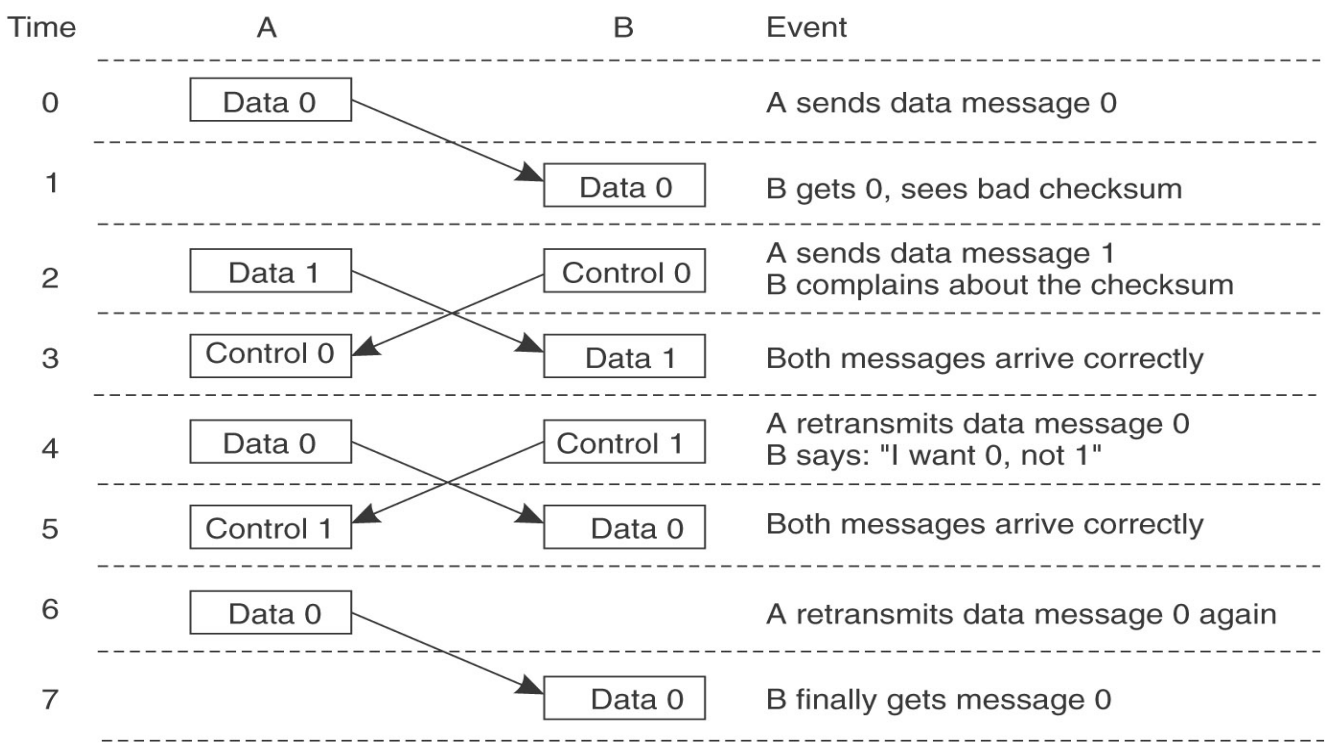
Camada de Aplicação

- Aplicações que oferecem os serviços ao usuário final
- Unificação de sistemas de arquivos e diretórios
- Correio eletrônico
- Login remoto
- Transferência de arquivos
- Execução remota

Mensagem na Rede



Comunicação a Nível de Enlace



O Modelo Cliente-Servidor

- O modelo diz como as máquinas podem se comunicar, mas qual a estrutura do SD?
- Classificar máquinas em servidores, que proveêm serviços para as máquinas cliente
- Máquina podem executar múltiplos processos clientes e/ou servidores
- Elimina overhead implementando um protocolo request/reply (camadas 1, 2 e 5)
- Simplicidade e eficiência

Endereçamento

- Como enviar a mensagem a máquina servidor e depois ao processo servidor?
- Endereço deve incluir também o número do processo (codificado no programa dos clientes)
- Elimina ambiguidade, não necessita de um coordenador global gerando números de processos, mas não é transparente
- O que fazer quando o servidor cai? Substituição de máquinas não funciona

Endereçamento

- Cada processo pode ter seu próprio endereço. Necessita de um alocador de endereços de processos centralizados. Não tem boa escalabilidade
- Processos escolhem seus próprios endereços aleatoriamente em um grande espaço de números.
- Boa escalabilidade. Processos localizam outros através de mensagens de broadcast. Cache de endereços
- Mensagens extras são colocadas na rede

Endereçamento

- Outra opção é o uso de servidores de nomes. Referências a processos servidores são feitas usando-se nomes ASCII e um servidor de nomes realiza a tradução
- Caches de endereços podem ser usadas
- Requer um componente centralizado
- Servidor de nomes pode ser replicado, mas cuidados devem ser tomados para manter a consistência entre servidores

Primitivas Bloqueadas X Não-Bloqueadas

- Nas primitivas bloqueadas (síncronas), o processo é bloqueado enquanto uma mensagem é enviada ou recebida
- Nas primitivas não-bloqueadas (assíncronas), o processo é desbloqueado antes do envio da mensagem
- Processo continua sua execução em paralelo com o envio da mensagem
- Processo que envia a mensagem não pode usar o buffer até que a mensagem seja enviada. Como saber que a mensagem foi enviada?

Primitivas Bloqueadas X Não-Bloqueadas

- Cópia da mensagem para um buffer do kernel (aparenta ser uma primitiva bloqueada para o processo emissor)
- Cópia extra é “desperdiçada”, e pode reduzir a performance do sistema
- Outra solução é se interromper o emissor quando o envio da mensagem for completado, liberando o buffer para o processo emissor
- Método eficiente que permite o maior grau de paralelismo entre as opções apresentadas.

Primitivas Bloqueadas X Não-Bloqueadas

- Complica programação e depuração de programas
- Análise semelhante com os processor receptores
- Uso de timeouts (temporizadores) para se evitar que processos fiquem bloqueados para sempre no caso de falha de transmissão

Primitivas Bufferizadas X Não-Bufferizadas

- receive(addr,&m) – processo diz ao núcleo de sua máquina que está esperando uma mensagem endereçada a addr e que deve colocá-la em m
- receive tem de ser executado antes do recebimento da mensagem (não-bufferizadas), senão o núcleo não sabe a quem entregar a mensagem
- O núcleo pode descartar a mensagem, que provavelmente será enviada novamente e talvez receive tenha sido executada

Primitivas Bufferizadas X Não-Bufferizadas

- Após várias tentativas cliente pode pensar que o servidor pifou ou endereço é inválido. Esse problema é agravado no caso do servidor estar com muitas requisições
- Núcleo pode armazenar mensagens temporariamente e iniciar temporizadores. Uso de caixas-postais (bufferizadas)
- O problema se repete quando caixas-postais enchem
- Emissor bloqueado até receber ACK do servidor. Caso a caixa-postal esteja cheia o emissor é suspenso pelo escalonador

Primitivas Confiáveis X Não-Confiáveis

- Definem se o sistema garante ou não a entrega de mensagens
- Definir semântica de send como não-confiável. implementação de comunicação confiável é tarefa dos usuários
- Núcleo da máquina receptora envia ACK de volta ao núcleo da máquina emissora. Transparente para os processos
- Reposta do servidor serve como um ACK. Pode-se implementar o envio de um ACK do núcleo da máquina emissora para o núcleo da máquina receptora

Implementando o Modelo Cliente-Servidor

- Escolha dos tipos de primitivas de comunicação
- Numeração de mensagens para auxiliar na correção de erros na transmissão
- ACKs de pacotes individuais (mais mensagens, menos retransmissão) ou de mensagens inteiras (menos mensagens, recuperação mais complexa)
- Escolha vai depender da taxa de erro da rede
- Exemplos de tipos de pacotes: REQ, REP, ACK, AYA, IAA, TA, AU

Implementando o Modelo Cliente-Servidor

- Pacote AYA serve para diferenciar entre uma requisição que está demorando para ser atendida e um servidor que pifou
- TA pode ser usado para indicar que a caixa-postal do servidor está cheia mas o endereço está correto enquanto que AU indica que o endereço está errado

Chamadas Remotas de Procedimentos (RPC)

- É um protocolo para chamadas de procedimento remoto, composto por camadas apoiadas sobre um ambiente de computação distribuído (DCE-Distributed Computing Environment)
- Segue um conjunto de regras que especificam meios de codificar e decodificar a informação transmitida entre duas aplicações.
- Auxilia os programadores a projetar e entender programas distribuídos com mais facilidade porque relaciona comunicação cliente-servidor a chamadas de procedimento convencional.

Chamadas Remotas de Procedimentos (RPC)

- O paradigma do RPC é orientado a aplicação, e não ao protocolo de comunicação
- Permite ao programador projetar um programa convencional que solucione o problema, e então dividir o programa em procedimentos que podem ser executadas em vários computadores.
- Uma mensagem enviada por um cliente a um servidor corresponde a uma "chamada" de um procedimento remoto, e uma resposta do servidor ao cliente corresponde a um "retorno" de uma chamada de procedimento.

Chamadas Remotas de Procedimentos (RPC)

- Uma operação possui uma assinatura (o nome da operação), seus parâmetros de entrada, resultados e exceções que podem acontecer
- A assinatura de uma operação é encapsulada em uma estrutura chamada IDL (Interface Definition Language), responsável pela especificação de características do procedimento fornecido pelo servidor ao seu cliente
- Chamar um procedimento que está localizado em um sistema remoto requer especificar qual sistema contatar, como codificar os parâmetros, como receber a resposta e como decodificar a resposta para utilização em um sistema específico

Chamadas Remotas de Procedimentos (RPC)

- RPC possibilita a comunicação entre máquinas com diferentes SOs e/ou configurações de hardware, pois a mensagem transferida é escrita em uma estrutura de dados padronizada
- Máquina com menor capacidade de processamento pode requisitar serviços para outra mais rápida
- Exemplo - servidores de arquivos
- Permite que qualquer programador com conhecimento em programação estruturada seja capaz de desenvolver aplicações distribuídas sem grandes dificuldades

Chamadas Remotas de Procedimentos (RPC)

- Detalhes sobre a localização do servidor e os protocolos de transporte subjacentes são totalmente invisíveis ao programador
- A comunicação por RPC pode ser em chamadas locais ou remotas
- O objetivo da RPC é manter a transparência da execução, fazendo com que chamadas remotas se pareçam com chamadas locais
- Uso de um stub no cliente e um skeleton no servidor. O skeleton e o stub são procedimentos auxiliares, gerados pelo compilador RPC.

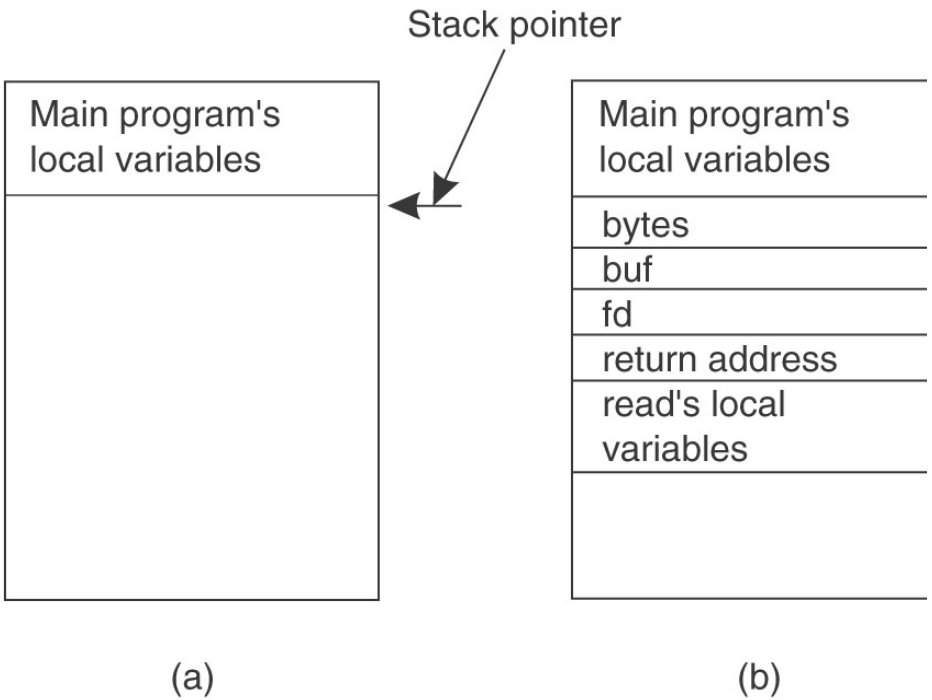
Chamadas Remotas de Procedimentos (RPC)

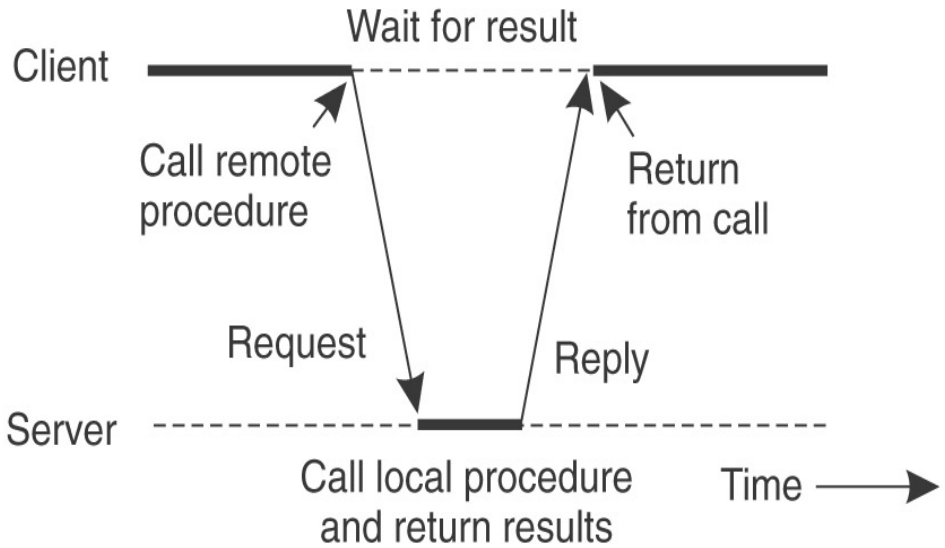
- O stub faz com que uma chamada remota do cliente, que é executada na máquina servidora, se pareça com uma chamada local feita a um procedimento dentro do mesma máquina do cliente.
- Essa função trata da conversão dos dados para uma representação externa dos dados, envia uma mensagem de solicitação, recebe a resposta e converte o retorno da representação externa para o tipo de dado especificado.

Chamadas Remotas de Procedimentos (RPC)

- O skeleton faz com que uma requisição de serviço de um cliente, que é recebido de outra máquina, se pareça como uma chamada de requisição local
- Servidor se registra em um binder (programa que conhece a localização do servidor, e faz a conexão do cliente e do servidor com o serviço de nomes)
- Para invocar um procedimento, o cliente faz uma solicitação para o ‘client stub’, que transporta a mensagem para o servidor, que por sua vez entrega a mensagem ao ‘server stub’

Chamadas Remotas de Procedimentos (RPC)

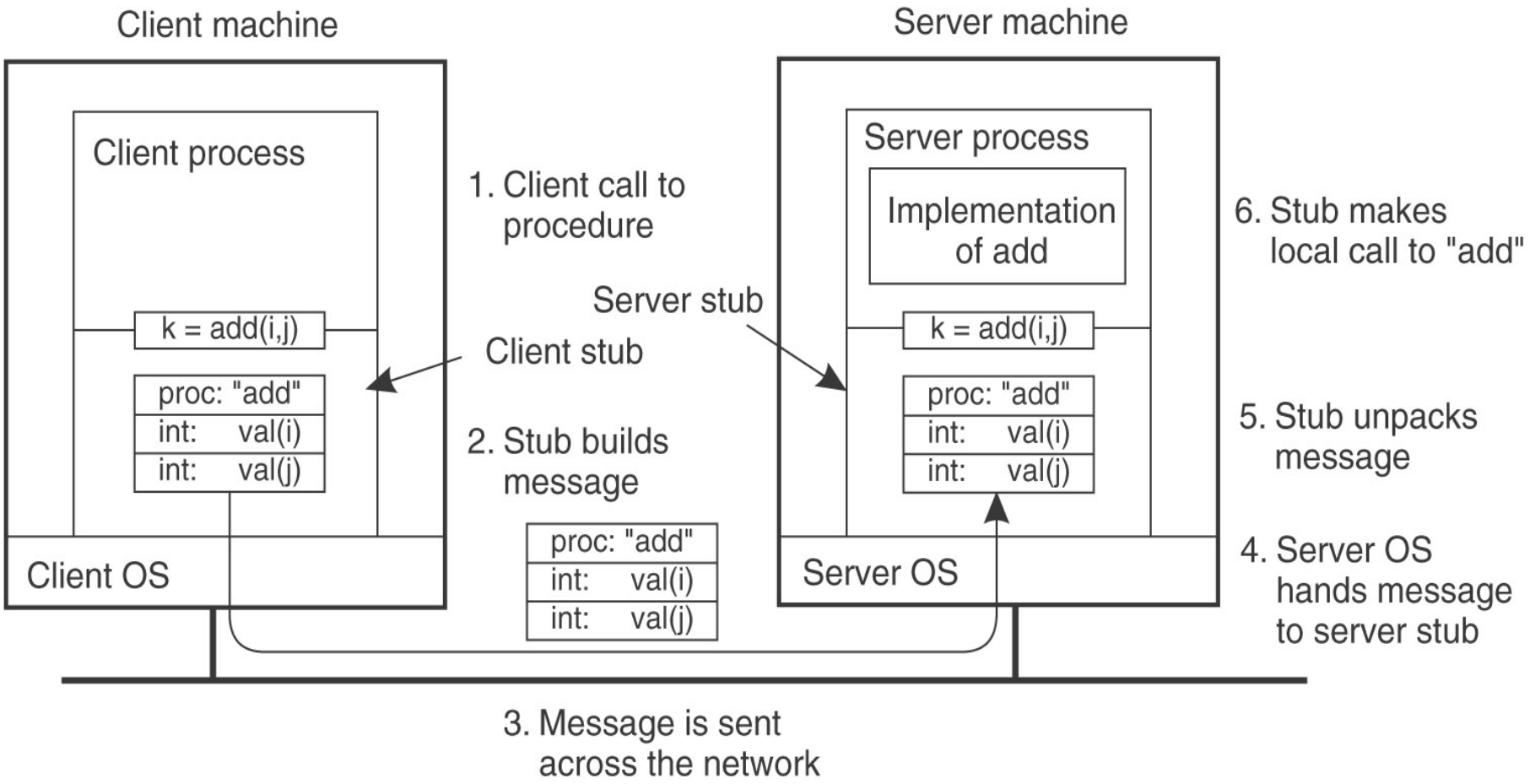




Chamadas Remotas de Procedimentos (RPC)

1. Procedimento cliente chama o stub do cliente (mesmo modo local ou remoto)
2. Stub do cliente monta mensagem e chama SO local
3. OS do cliente envia mensagem
4. OS remoto entrega mensagem ao stub do servidor
5. Stub do servidor desempacota parâmetros e chama o servidor
6. Servidor executa e retorna o resultado para o stub
7. Stub do servidor empacota mensagem e chama o SO local
8. SO do servidor envia mensagem para o SO do cliente
9. SO do Cliente entrega mensagem ao stub do cliente
10. Stub desempacota resultado e entrega ao cliente

Chamadas Remotas de Procedimentos (RPC)

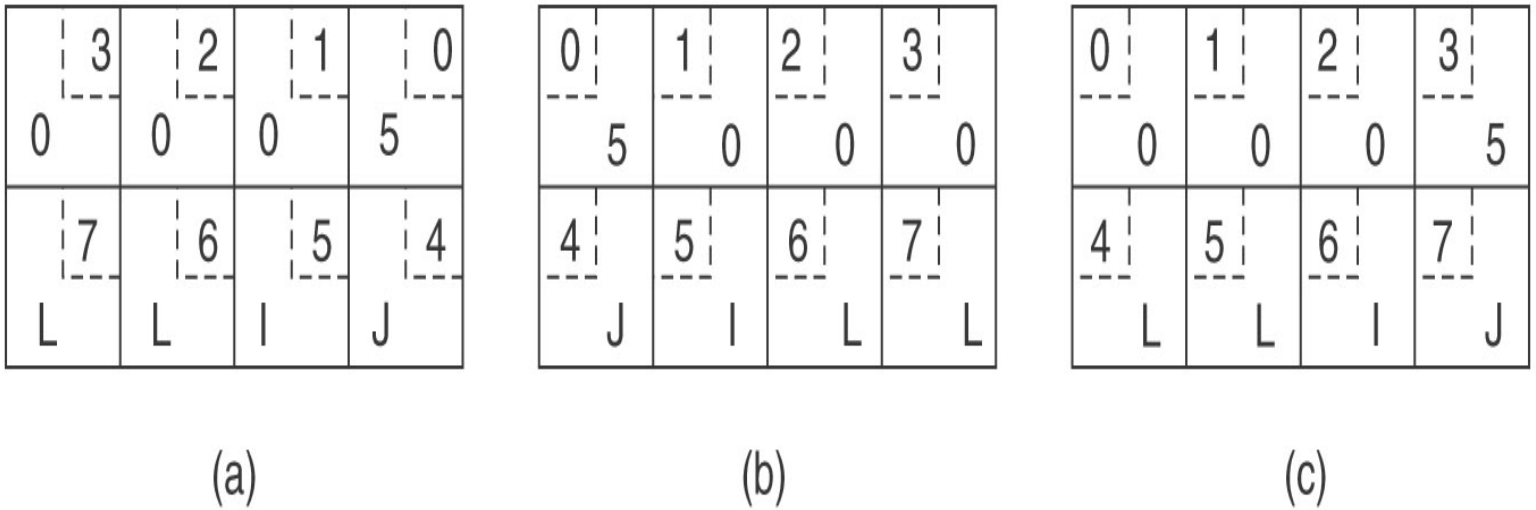


Binding Dinâmico

- Como o cliente localiza o servidor?
- A especificação formal do servidor (tipos dos parâmetros passados, nomes dos serviços, versão do servidor) é utilizada pelo gerador de stubs
- Ao iniciar sua execução, o servidor exporta a sua interface enviando uma mensagem ao binder (registro do servidor) contendo seu nome, versão, um ID e um handle
- Cliente envia uma mensagem ao binder solicitando um serviço de uma versão, e recebe um handle e ID casotal servidor exista no momento
- Flexível, mas gera mensagens extra e pode virar gargalo

Chamadas Remotas de Procedimentos (RPC)

- Conversão de codificação de conteúdos das mensagens é transparente aos usuários e pode ser feita através de:
  - Formas Canônicas
  - Sinalização de formatos



Chamadas Remotas de Procedimentos (RPC)

- Um procedimento (a) e a mensagem correspondente (b)

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

(a)

foobar's local variables	
	x
y	
5	
z[0]	
z[1]	
z[2]	
z[3]	
z[4]	

(b)

Falhas em RPC

- Tipos de falha:
  1. Cliente não consegue localizar o servidor
  2. Mensagem de requisição do cliente para o servidor é perdida
  3. Mensagem de resposta do servidor para o cliente é perdida
  4. O servidor falha após receber a requisição
  5. O cliente falha após enviar a requisição

Falhas em RPC

- Tipo 1 – Cliente pode ter endereço errado do servidor. Variável global de erro ou exceções (nem todas as linguagens possuem exceções e acaba com a transparência do sistema)
- Tipo 2 – Uso de temporizadores
- Tipo 3 – Temporizadores poderiam fazer com que a requisição fosse enviada novamente, porém algumas operações não são idempotentes, isto é, não podem ser executadas mais que uma vez (transferência de fundos)
- Requisições sequenciadas e bits de retransmissão

Falhas em RPC

- Tipo 4 – Servidor pode falhar antes ou após processar requisição. Como diferenciar os dois casos?
- Opções são se garantir:
    - Ao menos uma vez
    - No máximo uma vez
    - Nada
- O ideal de executar exatamente uma vez não é possível em alguns casos

Falhas em RPC

- Tipo 5 – Cliente falha após enviar requisição, criando órfãos, que gastam ciclos de CPU, podem bloquear recursos, etc.
- Eliminação de órfãos através de :
  - Exterminação – log é escrito e após um reboot máquina solicita que os seus órfãos sejam eliminados
  - Reincarnação – Divide tempo em épocas sequenciais. Quando um cliente reinicializa, ele envia uma mensagem com uma nova época e todas as computações remotas são eliminadas

Falhas em RPC

  - Reincarnação branda – Após um broadcast de época, cada máquina procura localizar donos de computações remotas, e caso não ache, as elimina
  - Expiração – Cada RPC tem um tempo máximo para execução e é eliminado após passar do tempo alocado
- Problemas caso um órfão seja eliminado enquanto mantinha algum bloqueio de recursos ou tenha feito requisições ainda não atendidas para iniciar outros processos

Aspectos de Implementação

- Protocolos do RPC: orientados a conexão X não orientados a conexão (comunicação mais simples X comunicação mais eficiente)
- Protocolo específico para RPC (pode ser mais eficiente) ou geral, como IP (já existem implementações em vários sistemas)
- ACKs – Protocolos para-e-espera (stop-and-wait) ou de rajada (blast) enviam um ACK por pacote ou um por um grupo de pacotes, respectivamente
- Repetição seletiva pode ser implementada com protocolo de confirmação em rajada

Aspectos de Implementação

- Cópias – Algumas opções para diminuir o tempo com cópias em RPCs são
  - Uso de uma placa de rede com scather-gather, que pode copiar dados diretamente do stub do cliente, eliminando a cópia para o kernel
  - Caso o buffer do pacote ocupe uma página inteira e começe em uma ínicio de página e o buffer do stub do servidor também seja uma página inteira pode-se mapear a página no pacote para o stub. Não há ganhos em pequenas mensagens

Aspectos de Implementação

- Temporizadores
  - Raramente o valor máximo de um temporizador é atingido, mas ainda assim tem de ser incluidos e removidos de alguma lista
  - Valores máximos de temporizadores afetam a performance de algoritmos e não sua corretude
  - Podem ser armazenados em uma lista ligada ou na tabela de processos

Um Exemplo de RPC

- Sun-RPC: Sistema criado originalmente para máquinas Sun, mas que é atualmente utilizado por outros SOs
- A arquitetura definida inclui:
  - Uma linguagem para definição de interfaces (cabeçalhos de procedimentos, etc.)
  - A ferramenta RPCGEN, que gera os stubs do servidor e clientes automaticamente
  - Uma biblioteca RPC, que pode ser usada diretamente na construção de programas que não usem RPCGEN
  - O protocolo de comunicação entre os stubs

Um Exemplo de RPC

- Pode utilizar TCP ou UDP
- Tradução entre formatos de dados se dá através da utlização de uma representação padrão, a XDR (eXternal Data Representation Standard)
- A conversão é especificada para um conjunto pré-definido de tipos de dados