Remote Procedure Call (RPC)

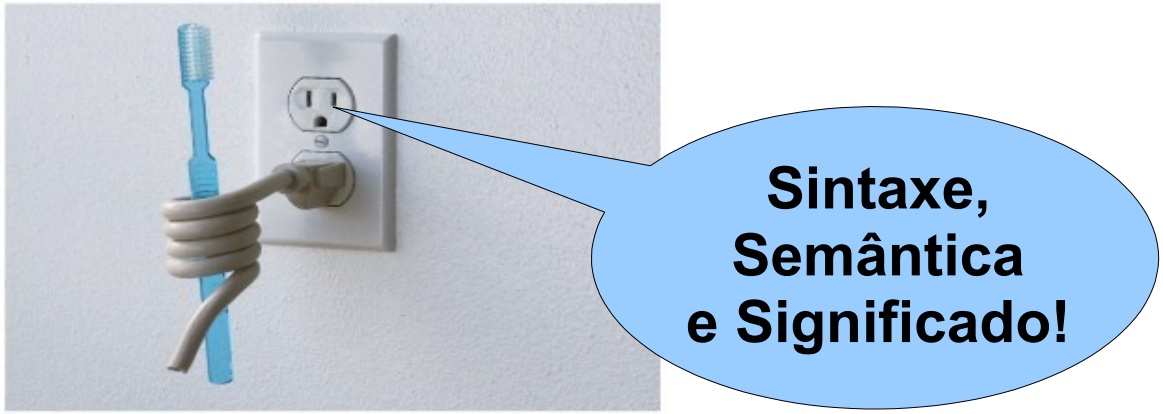
Referências:

• Seção 4.2 - STEEN, V. M. TANENBAUM, A. S. , “Sistemas Distribuídos: princípios e paradigmas”, 2a. Edição, 2007.

• Seção 5.3 - Coulouris et al., “Sistema Distribuídos: conceitos e projeto”, 4a. Edição, 2007.

Remote Procedure Call (RPC):introdução

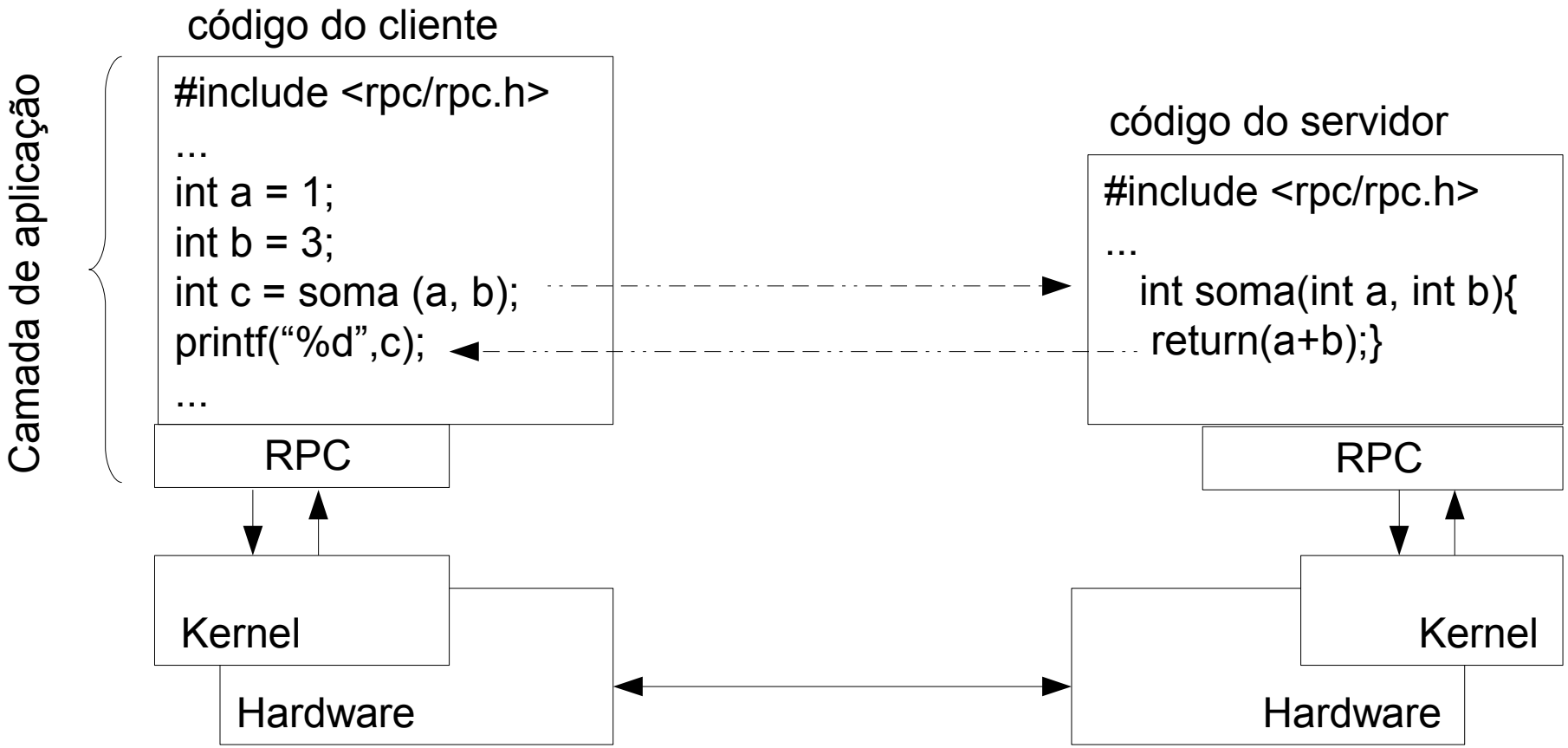
- Pouca Transparência
  Sockets → troca de um fluxo não-estruturado de bytes entre processos



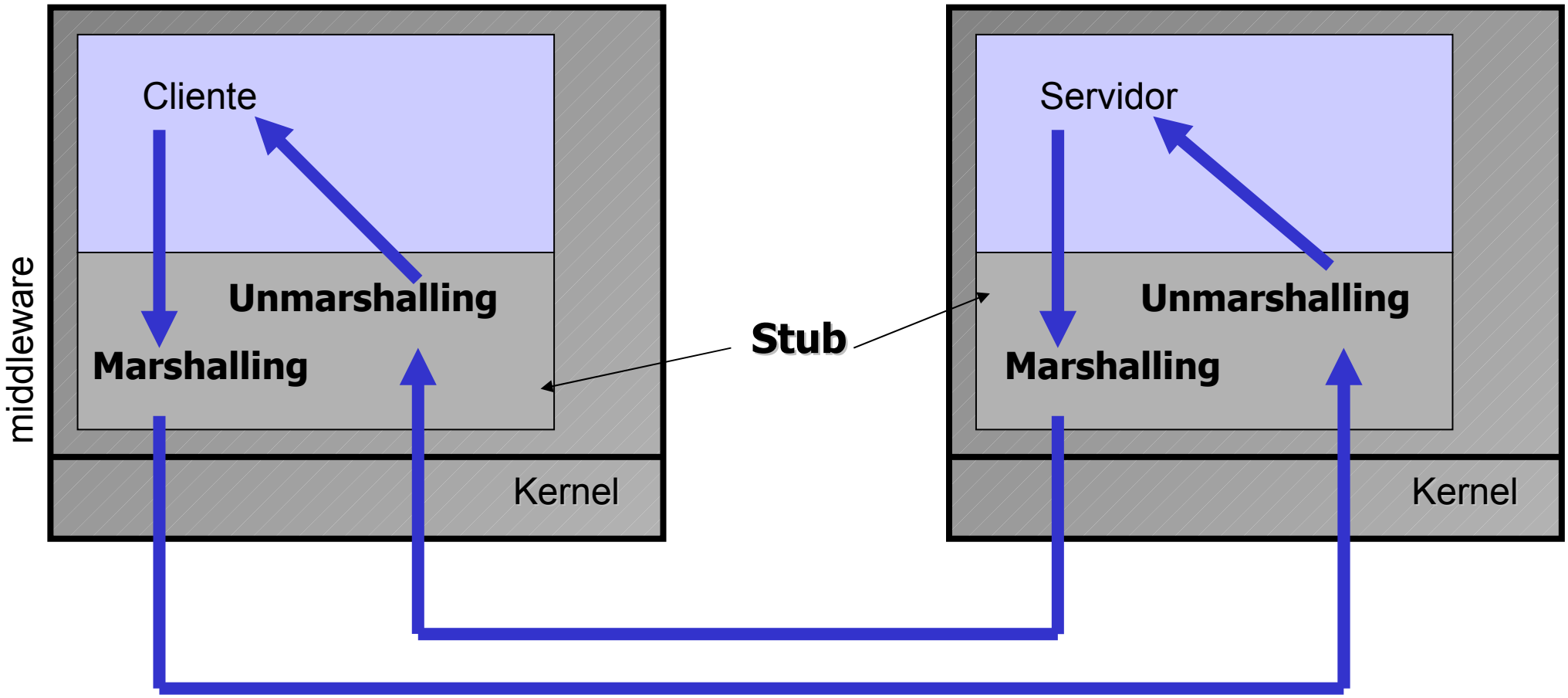
- Solução:
  - Trabalho de Birrel e Nelson 1984
  - **Proposta**: converter chamada local em chamada remota
  - Como?
    - Um mecanismo para transferência de **controle** e de **dados**

“RPC são chamadas a procedimentos que estão fora do espaço de endereçamento corrente”

- Facilitar o desenvolvimento de aplicações distribuídas → **mais transparência** → programadores com foco no negócio!

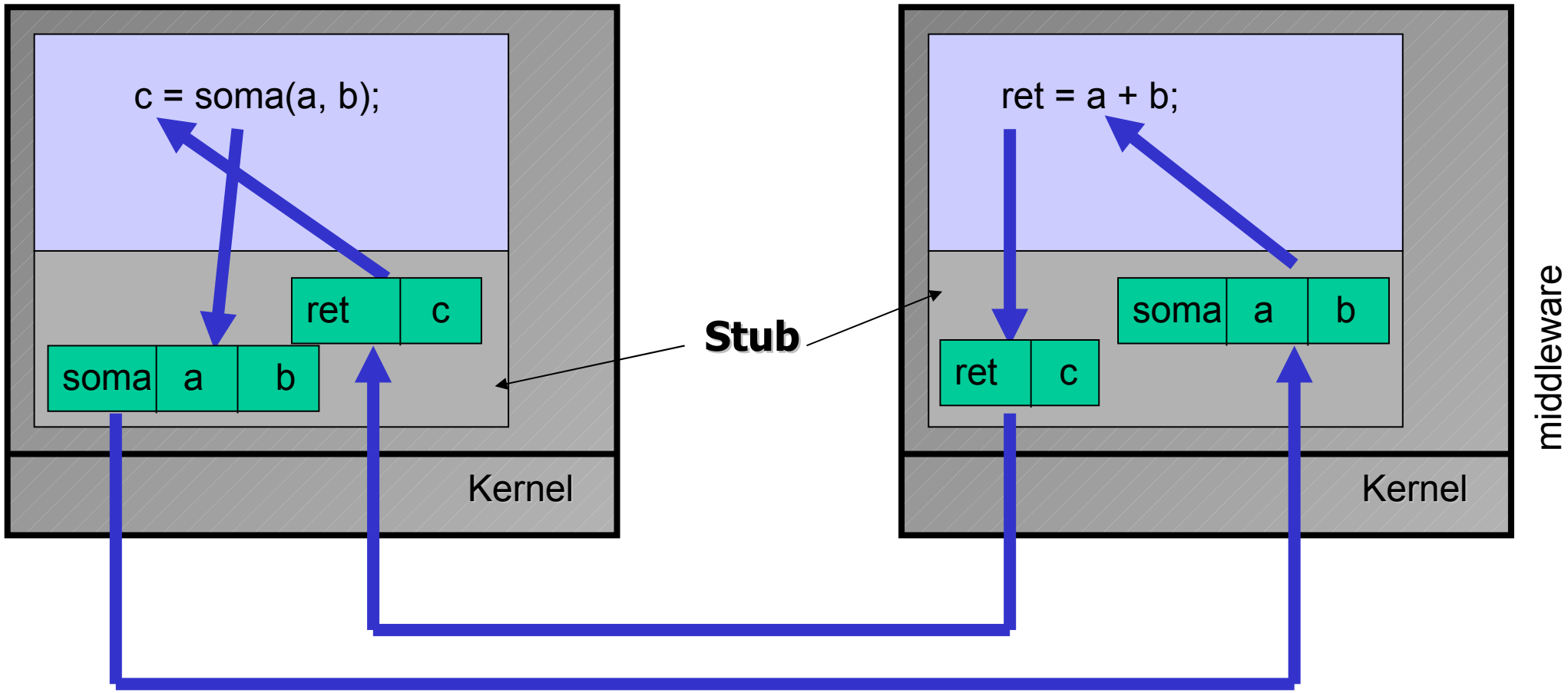


Remote Procedure Call (RPC): introdução



Usa-se uma IDL (*Interface Definition Language*) para especificar as chamadas remotas. Um compilador traduz IDL em Stubs

Remote Procedure Call (RPC): introdução



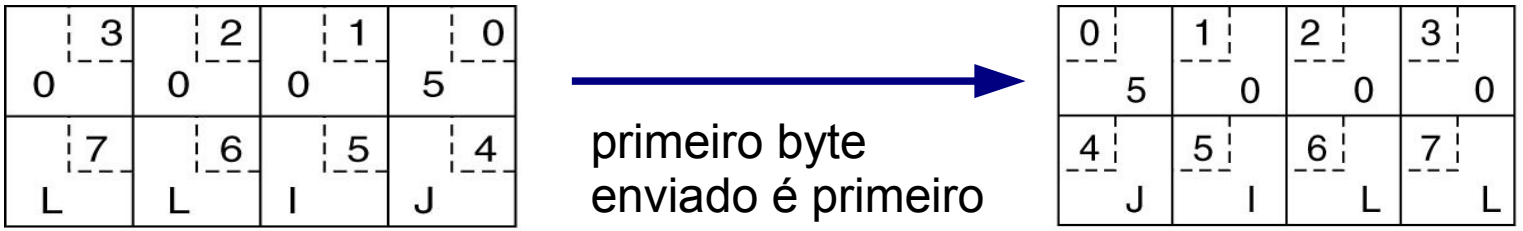
Remote Procedure Call (RPC): resumo

1. o procedimento do cliente chama o *stub* (ou apêndice) do cliente;
2. o *stub* do cliente constrói uma mensagem e envia um aviso ao Kernel;
3. o kernel envia a mensagem ao kernel remoto;
4. o kernel remoto entrega a mensagem ao *stub* do servidor;
5. o *stub* do servidor desempacota os parâmetros e chama o servidor;
6. o servidor realiza o trabalho e retorna o resultado para o *buffer* dentro do *stub*;
7. o *stub* do servidor empacota tais resultados em uma mensagem e emite um aviso para o kernel;
8. o kernel remoto envia uma mensagem para o kernel do cliente;
9 o kernel do cliente entrega a mensagem ao *stub* do cliente;
10. o *stub* desempacota os resultados e os fornece ao cliente.

Representação de parâmetros

- Cada máquina tem sua própria representação para números, caracteres e outros itens de dados
  - Ex. Mainframes IBM usam código de caracteres EBCDIC; PC usam ASCII;
  - Algumas máquinas usam complemento de um equanto outras usam complemento de dois para inteiros;
  - Máquinas Intel Pentium numeram seus bytes da esquerda para a direita → ***little endian;*** Máquinas Sun SPARC muneram da direita para a esquerda → ***big endian***

Int v=5;
String s="JILL"



Intel **SPARC**

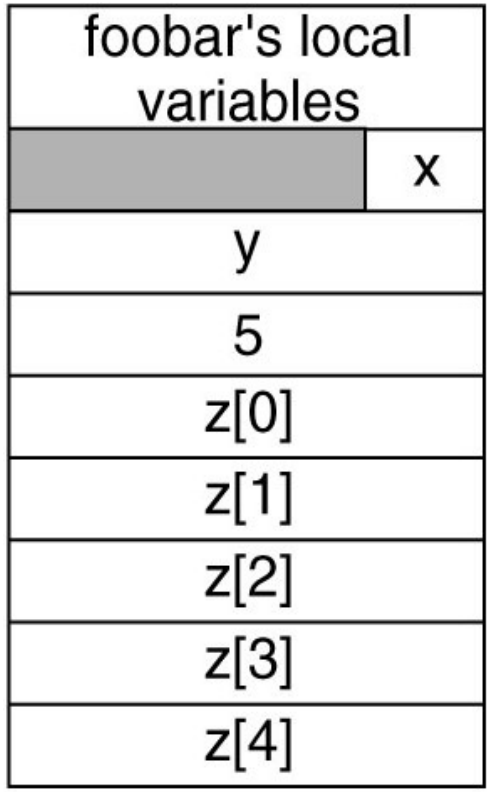
O que pode acontecer caso não seja feita uma conversão dos dados recebidos?

Passagem de parâmetros

- Por valor: sem problema
- Por referência: C é estritamente necessário

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

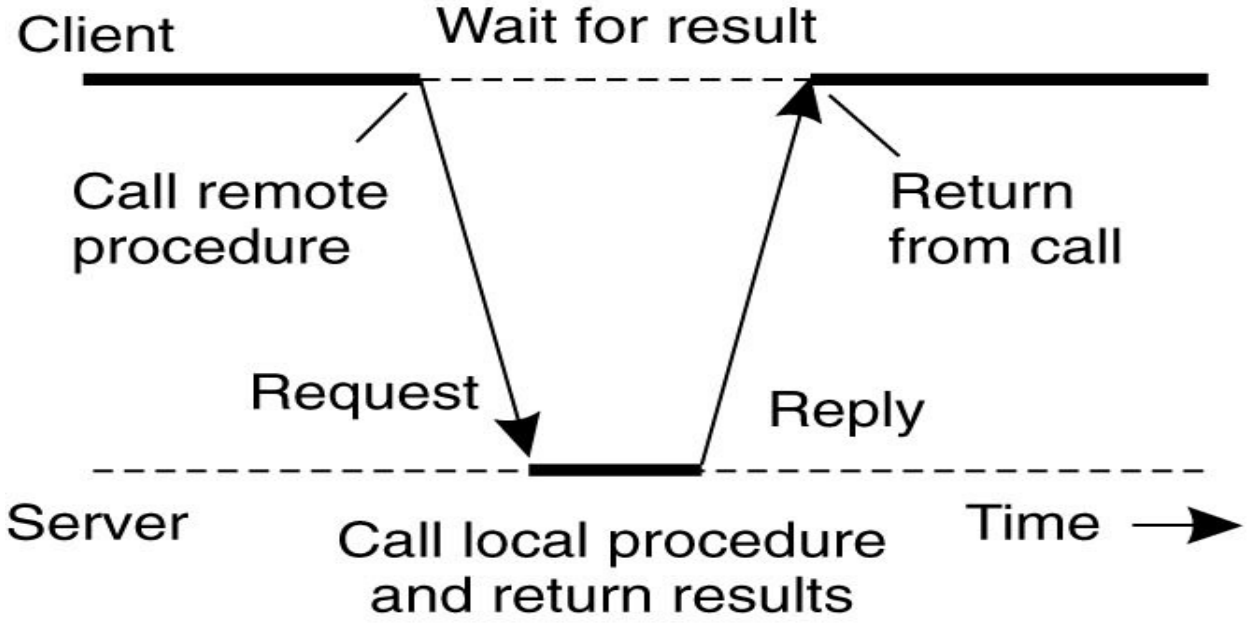
- **Solução:** Cópia/restauração
  - Melhorias podem ser implementadas
    - Ex. parâmetro de entrada ou de saída?



Remote Procedure Call (RPC): sincronismo

RPC Bloqueante (ou Síncrona): ocorre na maioria das implementações → cliente espera

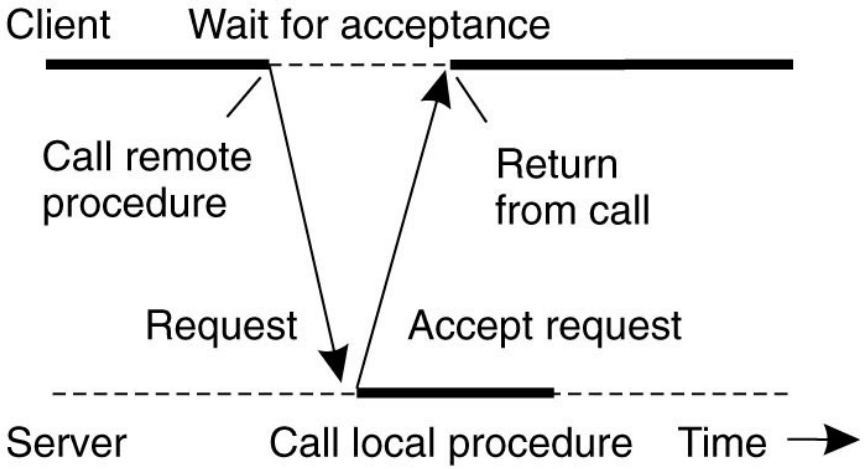
- **Variante**: **RPC deferida** → o cliente sonda o servidor para ver se os resultados estão disponíveis, em vez de aguardá-lo.

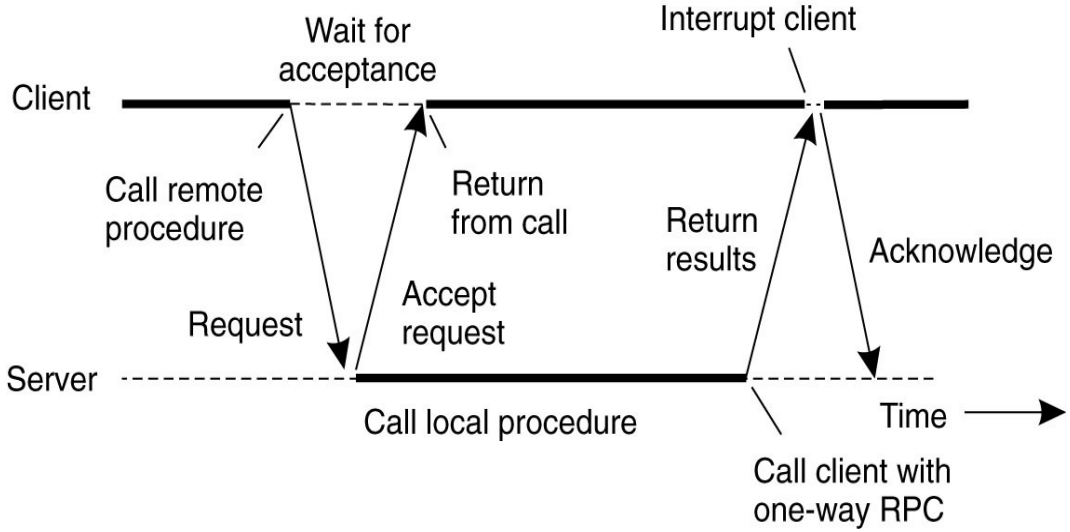


Remote Procedure Call (RPC): sincronismo

RPC Não-Bloqueante (ou Assíncrona): cliente **pode** continuar o fluxo de execução

- Ex. Servidores X-11 tratam RPCs de forma assíncrona
  - Clientes atualizam poucas informações por vez
  - Sem necessidade de confirmação imediata da atualização





Variante: RPC de uma via → cliente continua executando
Imediatamente após enviar a requisição ao servidor, não aguarda “acceptance”

Principais Semânticas:

- No máximo uma vez (*at-most-once*)
  - cliente não repete a operação
  - *default* do DCE
- No mínimo uma vez *(at-least-once)*
  - cliente retransmite a requisição quantas vezes forem necessárias
  - aplicável a operações **idempotentes**
- Há outras semânticas...
  - Ex. fazer *broadcast* da requisição para todas as máquinas presentes na rede local

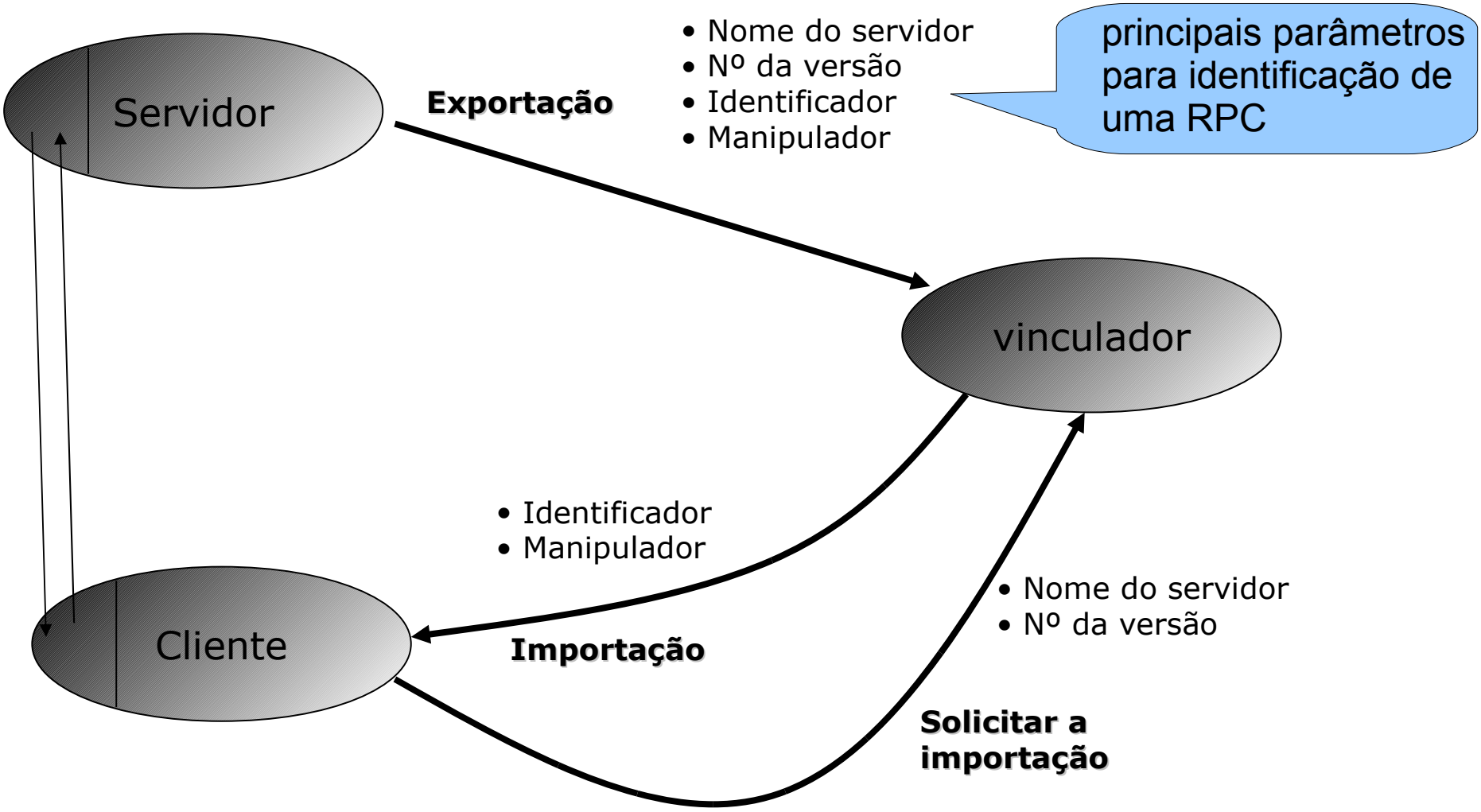
Como localizar o servidor?

- Codificar no hardware
  - Muito inflexível, embora seja transparente
- Codificar no código cliente/servidor (**vinculação estática**)
  - Ex. *bind()* de IP/PORTA na programação com *Sockets*
  - Pouco transparente e de difícil modificação
- Servidor de servidores (**vinculação dinâmica**)
  - Vinculação em tempo de execução

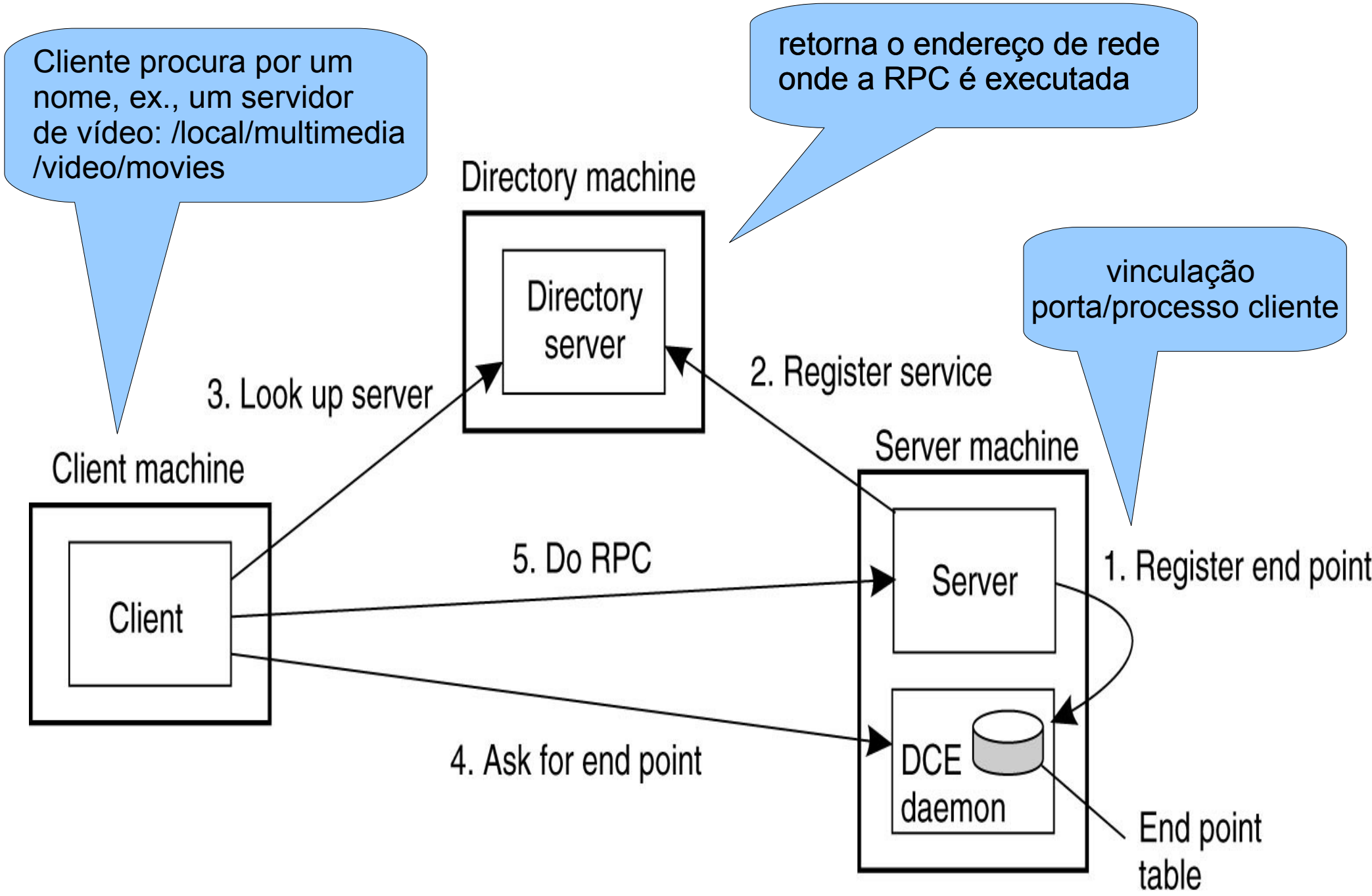
BINDER (ou vinculador)

- um simples tradutor ou um servidor de diretório sofisticado
  - Maiores detalhes em Serviços de Nomes...
- Objetivo: manter e disponibilizar os registros dos servidores no sistema

Remote Procedure Call (RPC): vinculação dinâmica

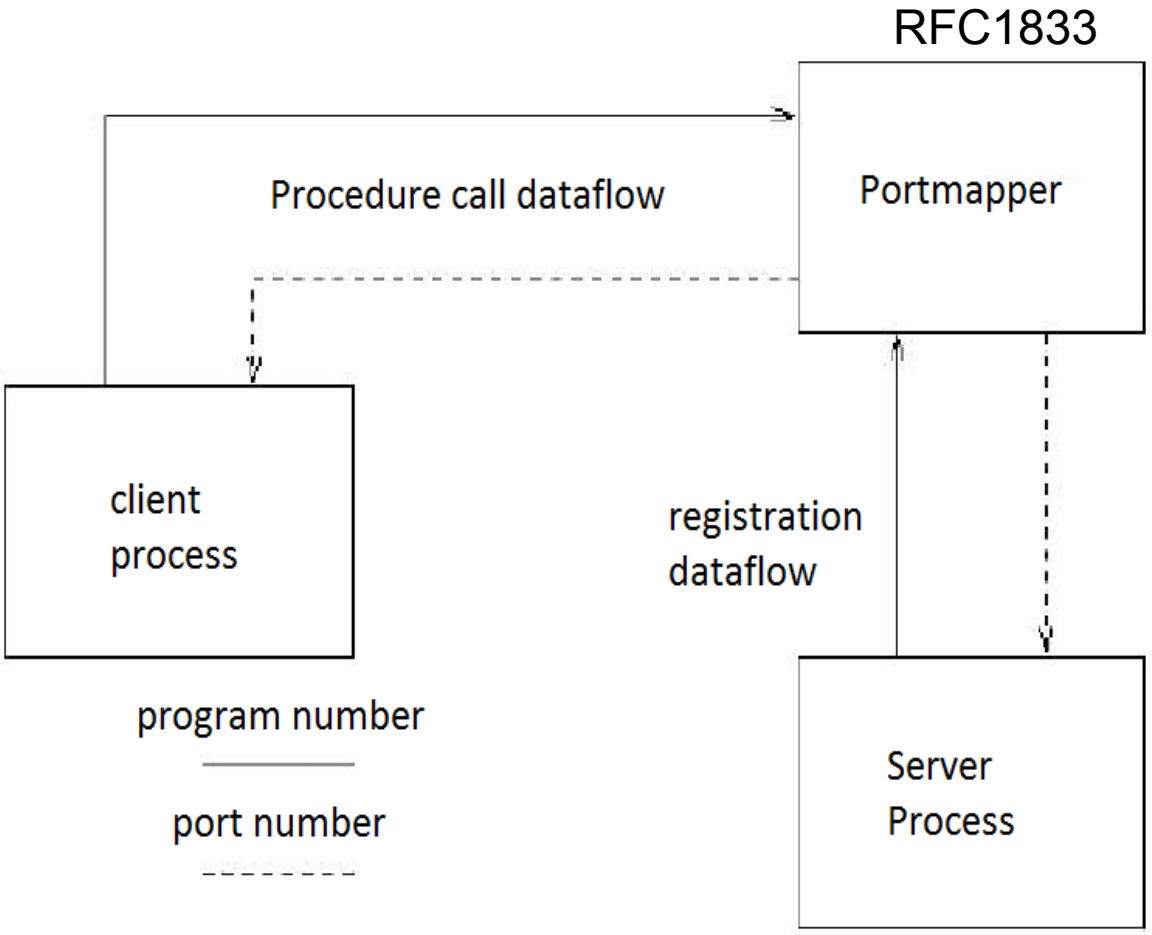


Remote Procedure Call (RPC): exemplo DCE



Remote Procedure Call (RPC): Sun RPC

An RPC service is identified by its RPC ***program number, version number, and the transport address*** *where it may be reached.*




```
$ rpcinfo -p
 program vers proto   port
  100000    2   tcp    111  portmapper
  100000    2   udp    111  portmapper
  100003    2   udp   2049  nfs
  100003    3   udp   2049  nfs
  100003    4   udp   2049  nfs
  100003    2   tcp   2049  nfs
  100003    3   tcp   2049  nfs
  100003    4   tcp   2049  nfs
  100024    1   udp  32770  status
  100021    1   udp  32770  nlockmgr
  100021    3   udp  32770  nlockmgr
  100021    4   udp  32770  nlockmgr
  100024    1   tcp  32769  status
  100021    1   tcp  32769  nlockmgr
  100021    3   tcp  32769  nlockmgr
  100021    4   tcp  32769  nlockmgr
  100005    1   udp    644  mountd
  100005    1   tcp    645  mountd
  100005    2   udp    644  mountd
  100005    2   tcp    645  mountd
  100005    3   udp    644  mountd
  100005    3   tcp    645  mountd
```

Também chamada de Open Network Computing (ONC RPC)

Remote Procedure Call (RPC): Sun RPC

- O vinculador (*port mapper*) é executado em cada computador num endereço de porta previamente definido

- Cada instância do mapeador de portas registra o **número de programa, o número da versão e o número da porta** para cada um dos serviços em execução, no momento em que cada um é iniciado
  - Quando um serviço tem **várias instâncias** executadas em diferentes máquinas, elas podem usar diferentes números de porta. Isso exige do cliente o envio de **uma mensagem multicast** para todos os mapeadores envolvidos.
  -

- O cliente se conecta ao servidor por intermédio do mapeador de porta (executado na máquina do servidor), especificando o **número de programa e número de versão**

Remote Procedure Call (RPC): vinculador

- Tarefas adicionais que podem ser atribuídas ao vinculador:
  - Autenticação do usuário
  - Checar os servidores para detecção e/ou tratamento de falhas
    - Servidor pode notificar saída de alguém do S.D.
- Desafios:
  - Sobrecarga (importação/exportação)
  - Vinculador pode virar gargalo
  - Replicá-lo!
    - Comunicação adicional para consistência

Aplicações com RPC

Etapas:

1 Definir interface (usando uma IDL)

2 Gerar *stubs*

3 Codificar Servidor

4 Codificar Cliente

5 Gerar todos os programas objeto (.o)

6 Linkar *stubs* ao cliente e ao servidor

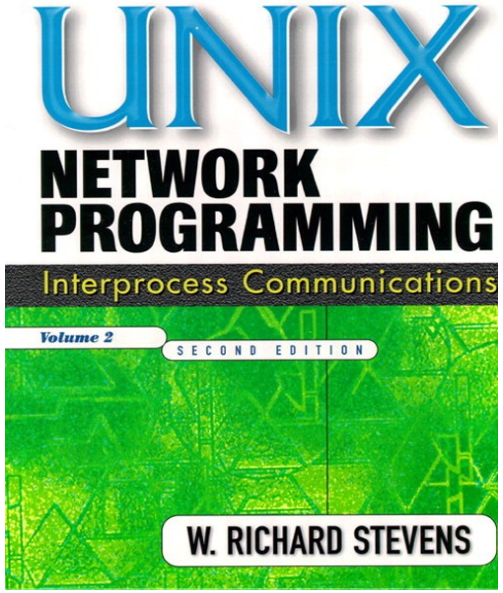
/*para gerar executável*/

7 Executar servidor; executar cliente

RPC para soma e subtração de dois inteiros

- Operaçõs básicas : soma (add) e subtrair (sub) dois números inteiros
- retorno: um inteiro com o valor do resultado

Implementação da Sun RPC para Windows: *http://gnuwin32.sourceforge.net/packages/sunrpc.htm*



Boa referência!

Remote Procedure Call (RPC): exemplo1

simp.x

A linguagem XDR da Sun, originalmente projetada para especificar representações externas de dados, foi estendida para se tornar uma linguagem de definição de interface (IDL)

```
#define VERSION_NUMBER 1

struct operands {
      int x;
      int y;
};

program SIMP_PROG {
   version SIMP_VERSION {
     int ADD(operands) = 1;
     int SUB(operands) = 2;
   } = VERSION_NUMBER;
} = 555555555;


#define SIMP_PROG 555555555
```

Define os tipos de dados utilizados como parâmetros

Identificadores usados pela RPC: **Número de Programa** e **Número de Versão**.
Número de versão muda quando uma assinatura de procedimento é alterada.
Assinatura: sintaxe + lista de parâmetros.
Cada operação da interface é identificada por um número.

Remote Procedure Call (RPC): número de programa

RFC1057

Program numbers are given out in groups of hexadecimal 20000000 (decimal 536870912) according to the following chart:

0 - 1fffffff defined by Sun

20000000 - 3fffffff defined by user

40000000 - 5fffffff transient

60000000 - 7fffffff reserved

80000000 - 9fffffff reserved

a0000000 - bfffffff reserved

c0000000 - dfffffff reserved

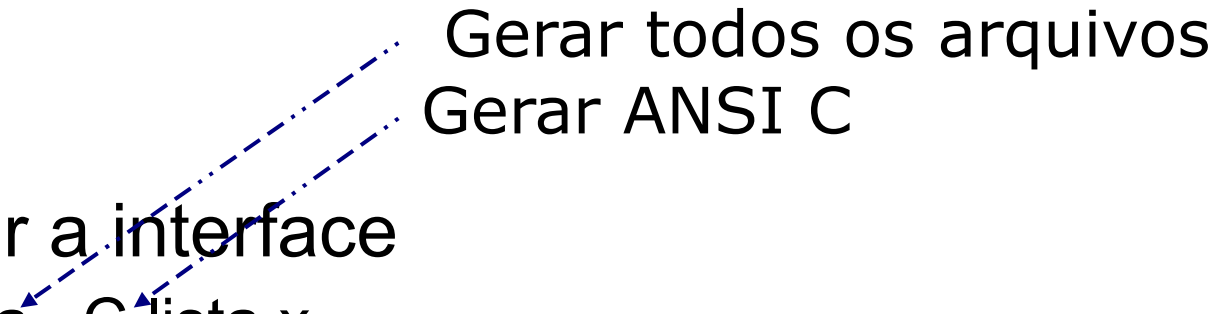
e0000000 - ffffffff reserved

The first group is a range of numbers administered by Sun Microsystems and should be identical for all sites

The second range is for applications peculiar to a particular site. This range is intended primarily ***for debugging new programs.***

The third group is for applications that ***generate program numbers dynamically***

Remote Procedure Call (RPC): compilação



1 - Compilar a interface

rpcgen –a –C lista.x

- Artefatos gerados:
  - *stubs :*
    - simp_clnt.c, simp._svc.c, simp_xdr.c, simp.h
    - *Makefile*

2 - Criar o cliente (client.c)

gcc -traditional client.c simp_clnt.c simp_xdr.c -o cliente

3 - Criar o servidor(server.c)

gcc -traditional server.c simp_svc.c simp_xdr.c -o servidor

simp_clnt.c → *stub* do cliente
simp_svc.c → *stub* do servidor
simp_xdr.c → para conversão da representação de dados
simp.h → encapsula os procedimentos remotos

Remote Procedure Call (RPC): execução

1 - Executar Servidor

./server

- Entretanto, se o vinculador não estiver operando...

```
aluno@LMC16:~/Área de Trabalho/RPC$ ./server
Cannot register service: RPC: Unable to receive; errno = Connection refused
unable to register (SIMP_PROG, SIMP_VERSION, udp).aluno@LMC16:~/Área de Trabalho
aluno@LMC16:~/Área de Trabalho/RPC$
```

Mensagem de erro. Comum para usuários do Ubuntu!

- **Solução:** instalar e iniciar o portmap. Para isso:

 # sudo apt-get install portmap

- Em seguida,

 # rpcinfo -p

2 - Executar o Cliente

.cliente localhost 2 2

endereço IP parâmetros

Uma lista encadeada

Operaçõs básicas : iniciar, inserir, imprimir, consultar, apagar

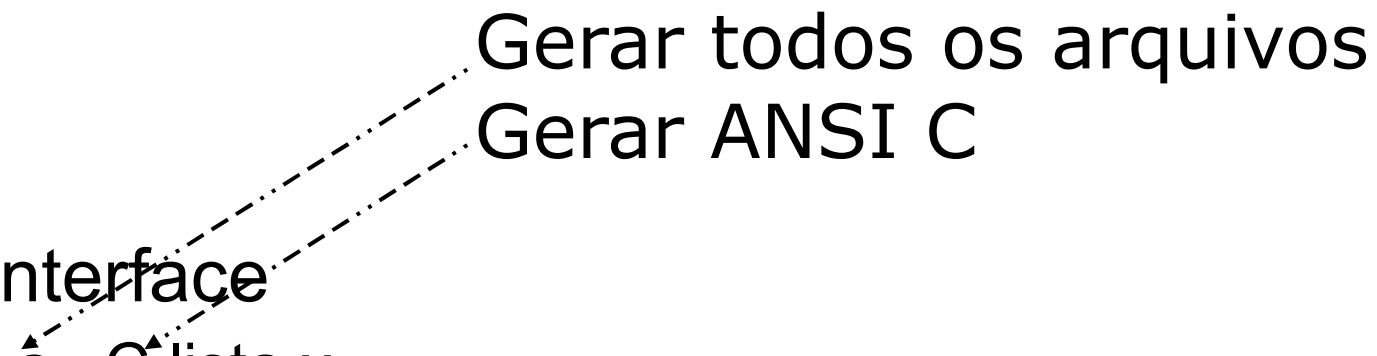
lista.x

```
 #define OK    0
 #define ERRO 1
struct no{
        int       dado;
        struct no * next;
       };
program LISTAPROG{
 version LISTAVER{
     int init ( int )=1;
     int insere ( int ) = 2;
     int deleta ( int ) =3;
     int consulta ( int ) = 4;
     int imprima ( int ) =5; } = 1;
                        } = 0x20000199;
```

Definição de Parâmetros e Mensagens
É permitido **apenas um parâmetro de entrada**. Portanto, os procedimentos que necessitam de vários parâmetros devem incluí-los como membros de uma única estrutura.

Número de Programa, Número de Versão e Identificação das operações.

Remote Procedure Call (RPC): exemplo2



- Compilar a interface
  - % rpcgen –a –C lista.x
  - Artefatos gerados:
    - *stubs :*
      - lista_clnt.c, lista._svc.c, lista_xdr.c, lista.h
      - *Makefile*
- Criar o cliente (gerado pelo *makefile*)
  - gcc -traditional cliente.c lista_clnt.c lista_xdr.c -o cliente
- Criar o servidor(gerado pelo *makefile*)
  - gcc -traditional servidor.c lista_svc.c lista_xdr.c -o servidor

Remote Procedure Call (RPC): exercícios

Para a lista de exercícios:

- do livro de Tanenbaum (cap.4): 6,7,8,9 e 14.

Para acrescentar 0,5 ponto à Primeira Nota:

- Modificar o exemplo SIMP_PROG para incluir duas novas operações: mult (identificador 3) e div (identificador 4). Estas operações devem implementar respectivamente a multiplicação e a divisão de dois inteiros.
- Incluir as chamadas para as operações supracitadas no código cliente e implementá-las no código do servidor.
- Construir um Makefile para compilação dos artefatos. Existem vários tutoriais na Internet. Alguns em português:
  - http://www.lcg.ufrj.br/Members/zezim/makefiles/
  - http://www.gacetadelinux.com/pr/lg/issue83/heriyanto.html