Remote Procedure Call

- **O aumento de complexidade das aplicações, torna desejável um paradigma que permite software distribuído ser programado de maneira similar a aplicações convencionais que executam em um único processador.**
- **O modelo Remote Procedure Call (RPC) fornece tal abstração. Neste modelo, IPC acontece como uma chamada de procedimento, função, que é familiar para os programadores de aplicação.**
- A chamada remota segue o modelo da chamada local sendo que o procedimento chamado executa em um processo diferente normalmente em um computador diferente.

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.

No nível de RPC o serviço fornecido por um servidor pode ser visto como um modulo com uma interface que exporta a funcionalidade apropriada. Ex: um serviço de arquivo (`read, write, open, close,etc`).

Comunicação em Sistemas Distribuídos

Chamada Remota de Procedimento - RPC

Características:

1. RPC segue o modelo de chamada local de procedimento, sintática e semântica.

2. Especifica parâmetros de entrada e saída.

3. Equivalência total em parâmetros por valor.

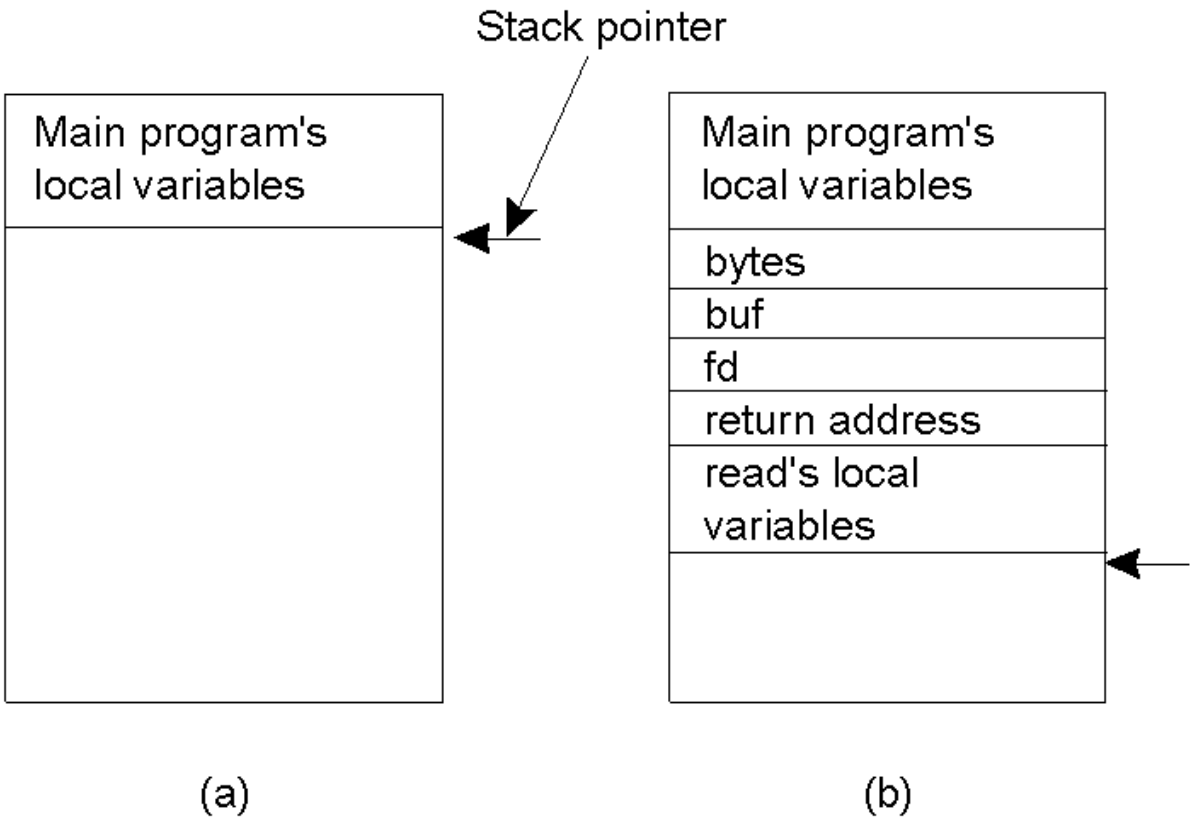
4. Executada em ambiente diferente do chamador.

5. Não tem acesso a recursos do chamador como:

    - globais

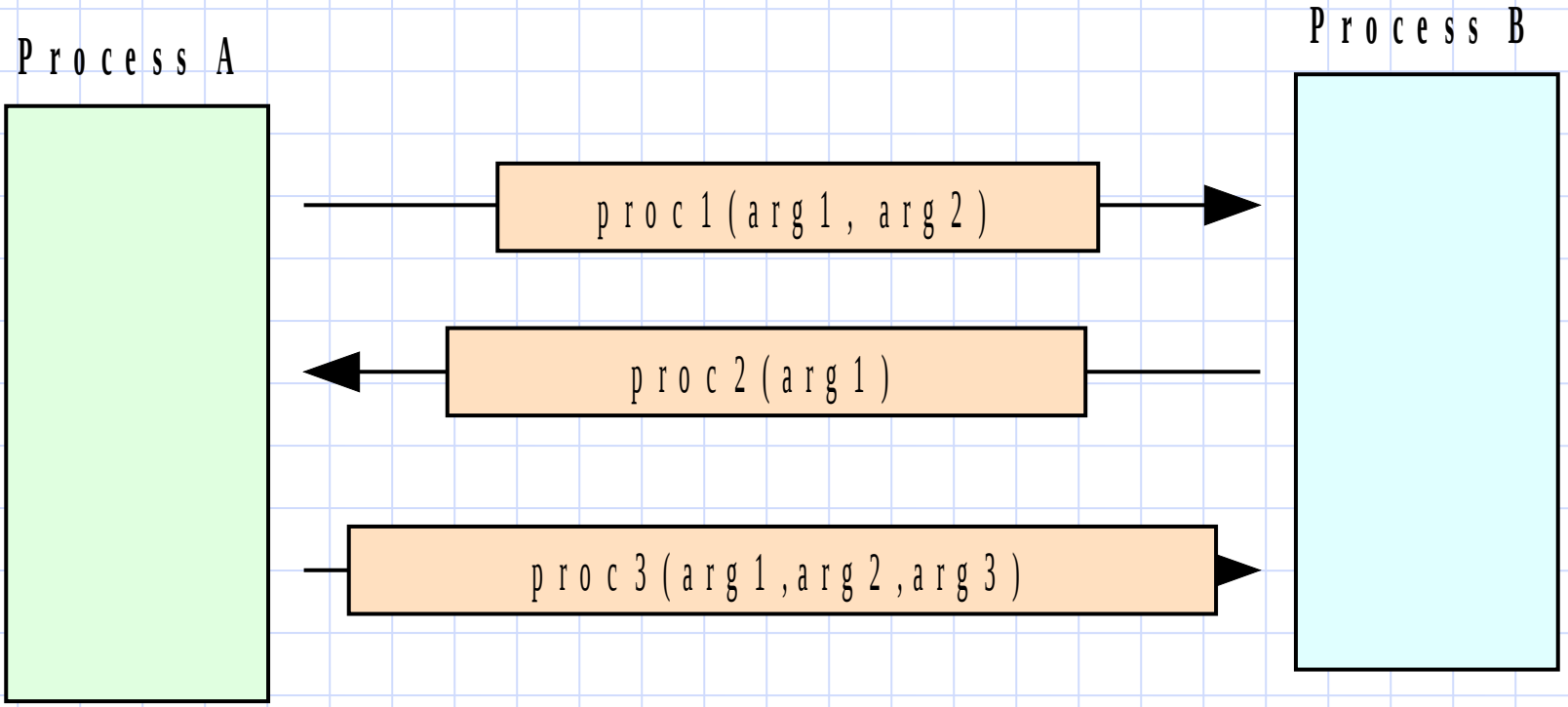
    - endereços enviados pelo chamador

Chamada de Procedimento Convencional



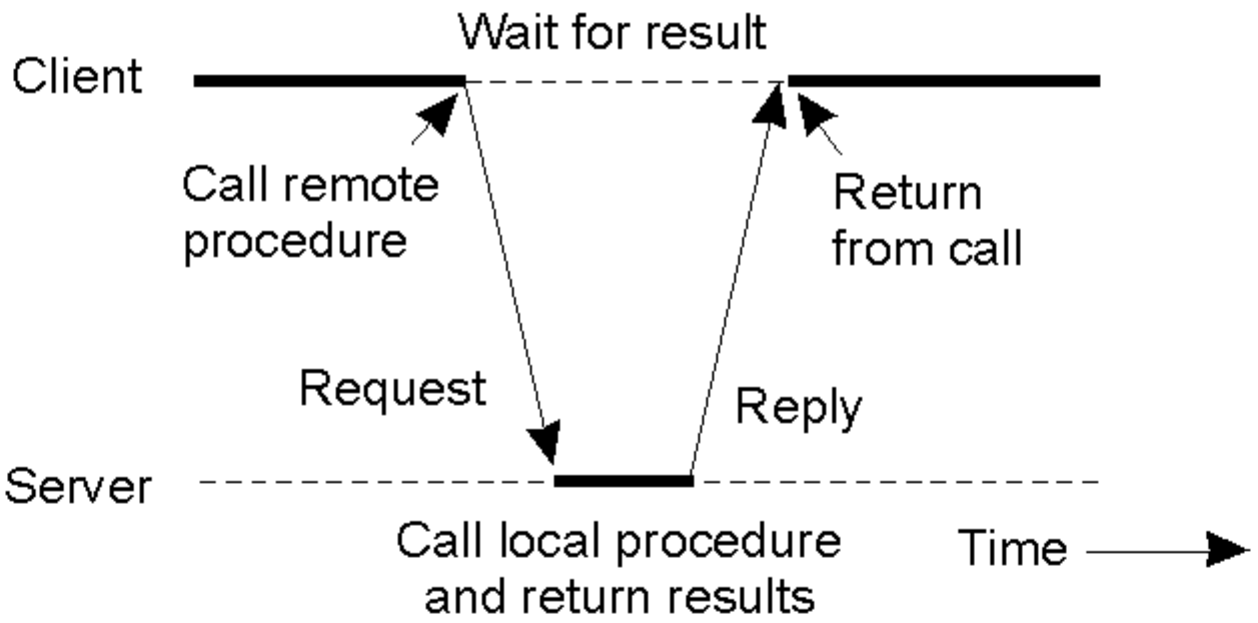
a) Passagem de parâmetro chamada local: pilha antes da chamada
b) pilha enquanto procedimento está ativo

Remote Procedure Call - 2



Stubs Cliente e Servidor

- Princípio do RPC entre um programa cliente e servidor.

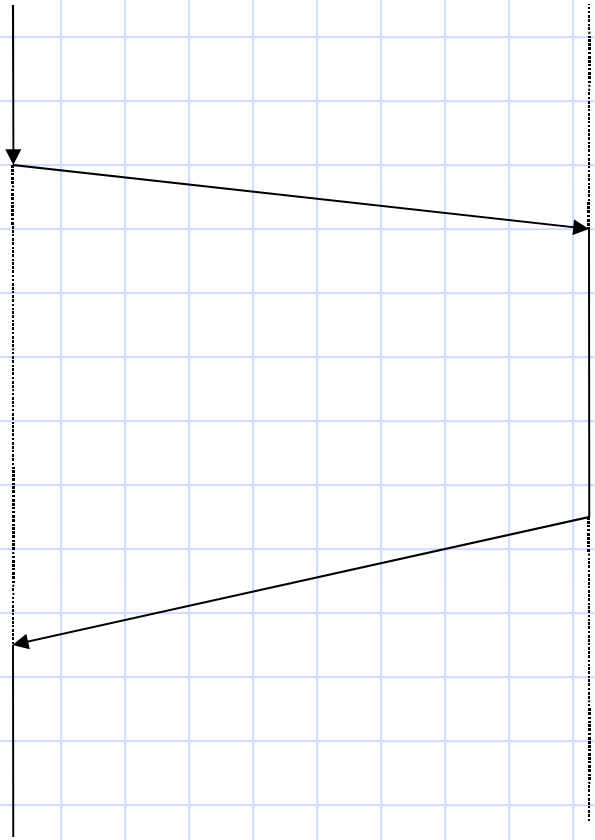


Comunicação em Sistemas Distribuídos

Chamada Remota de Procedimento - RPC

Máquina A

Máquina B



Chama procedimento e espera resposta

read(f,&buff)

Recebe requisição e inicia execução

Execução do procedimento

Envia resposta

Retorna execução

Comunicação em Sistemas Distribuídos

Chamada Remota de Procedimento - RPC

O mecanismo de RPC é transparente para o usuário:

- transparência sintática

- transparência semântica

chamador é suspenso, passa parâmetros e recebe resultados.

Conseguir a mesma semântica é praticamente impossível:

- execução em espaço de endereçamento disjunto

- vulnerabilidade a falhas

- consome mais tempo

Comunicação em Sistemas Distribuídos

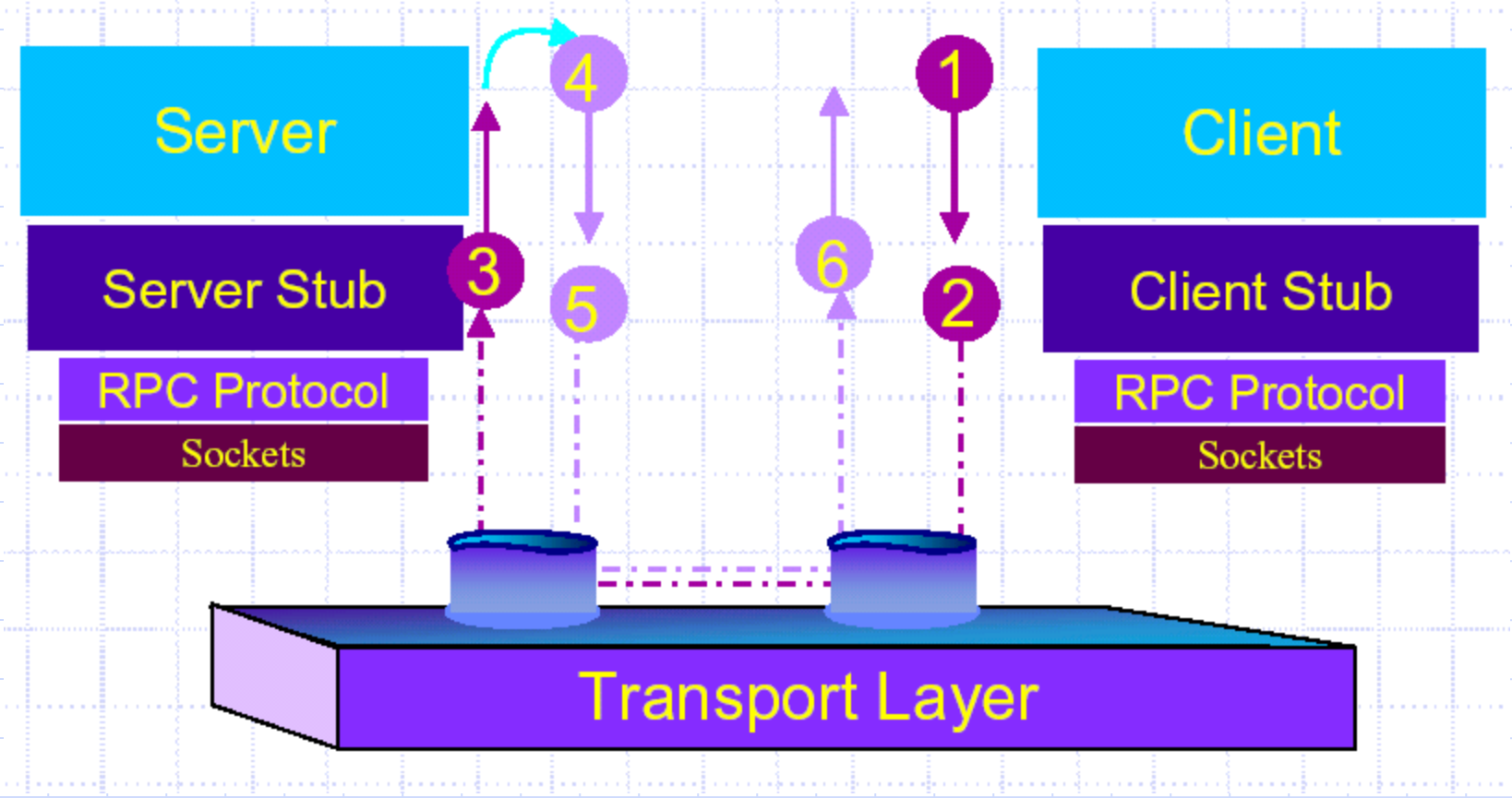
Chamada Remota de Procedimento - RPC

Para conseguir a transparência de semântica o mecanismo de RPC é baseado no conceito de *stubs*, que permite a abstração da chamada.

A implementação de um mecanismo RPC (componentes):

1. Cliente
2. *Stub* cliente
3. RPC runtime
4. *Stub* servidor
5. Servidor

Passos em RPC



Passos em RPC

1. Cliente chama stub cliente
2. Stub client constrói mensagem, chama SO local
   - SO cliente envia mensagem para SO remoto
   - SO remoto entrega mensagem para stub servidor
3. Stub servidor retira parametros, chama servidor
4. Servidor realiza trabalho, retorna resultado para stub
5. Stub servidor coloca na mensagem, chama SO local
   - SO servidor envia mensagem para SO cliente
   - SO cliente entrega mensagem para stub cliente
6. Stub retira resultado, retorna para cliente

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).

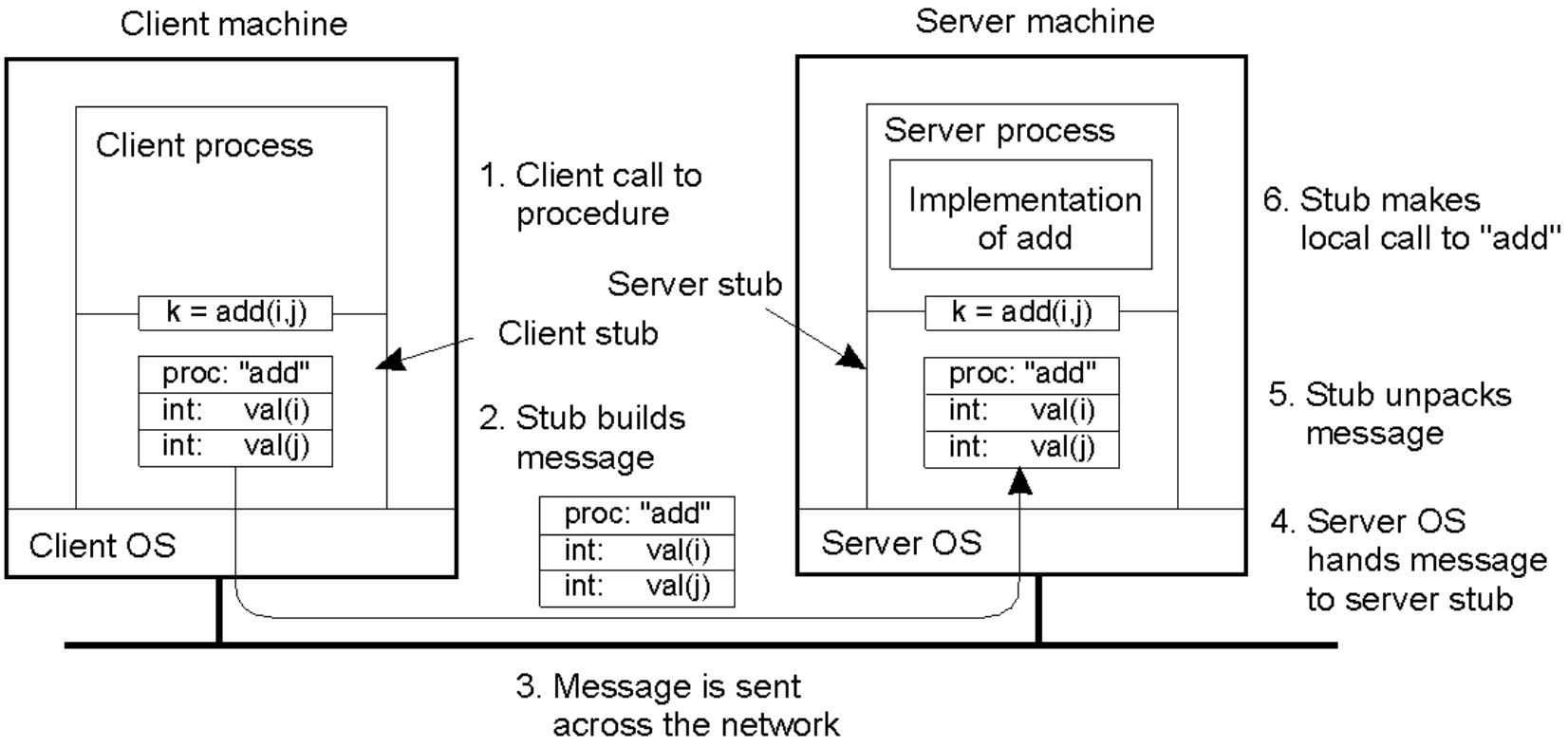
- Passagem de Pointers é um problema

  - proibir

  - copiar só quando necessário

Passando Parâmetros

- Trocando parâmetros em RPC



Passing Value Parameters (2)

a) Original message on the Pentium
b) The message after receipt on the SPARC
c) The message after being inverted. The little numbers in boxes indicate the address of each byte

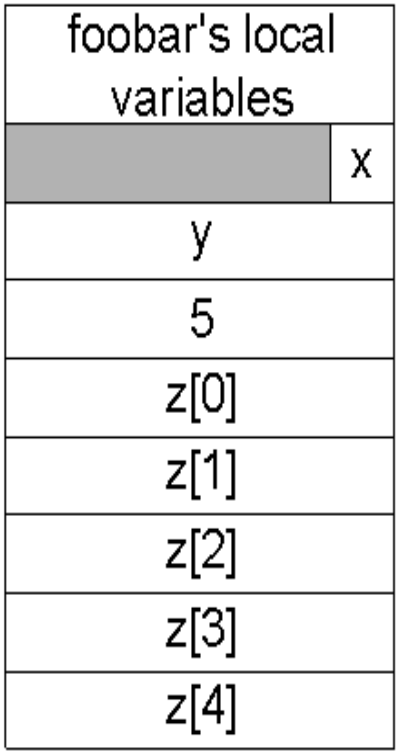
Parameter Specification and Stub Generation

a) A procedure

b) The corresponding message.

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

(a)



(b)

Chamada Remota de Procedimento (RPC)

Classes

(1) O mecanismo RPC está integrado com a linguagem de programação que inclui uma notação para a definição de interfaces.

- construções da linguagem, exceções, usadas para tratar RPC (Argus)

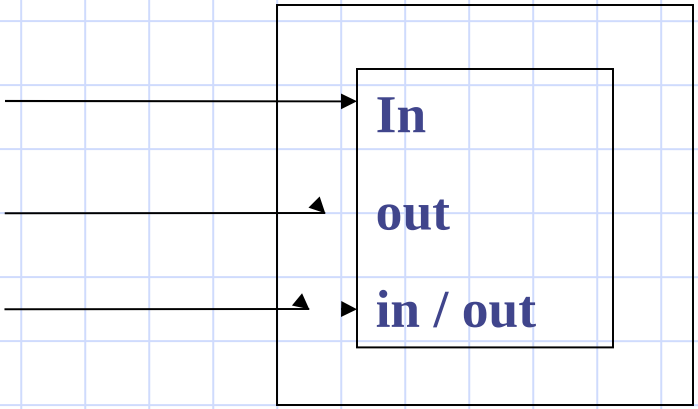
(2) Uma linguagem de propósito especial IDL (Interface Definition Language - Sun RPC) é usada para descrever as interfaces entre clientes e servidores.

- não está amarrada a nenhum ambiente de linguagem em particular.

Chamada Remota de Procedimento (RPC)

Linguagem de Definição de Interface (IDL)

- Especifica aquelas características do Servidor visíveis ao cliente: descreve os procedimentos do servidor, seus parâmetros, tipos e quando são de *entrada*, *saída*, ou *entrada/saída*.
- Compilador IDL pode gerar os *stubs* cliente e servidor de forma automática a partir da especificação formal do servidor.



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)

Semântica da Chamada

Fornecer diferentes níveis de garantia de entrega:

1. Sem garantias

   Se a resposta não foi recebida depois de um `timeout` e não existe `retransmissões`, não existe garantias da execução do procedimento remoto:

   - `Requisição foi perdida`
   - `Servidor quebrou`
   - `Resposta foi perdida`

Chamada Remota de Procedimento (RPC)

Semântica da Chamada

2. Pelo menos uma vez (`AtLeast Once`)

Retransmite requisições sem filtrar possíveis duplicatas.

- O Cliente pode receber uma resposta ou ser informado de que provavelmente o servidor falhou.
- O Cliente não saberá quantas vezes o procedimento remoto foi chamado.

Chamada Remota de Procedimento (RPC)

Semântica da Chamada

3. No máximo uma vez (At Most Once)

Possibilita filtragem de possíveis duplicatas e retransmissão de respostas sem re-executar as operações.

- Algumas operações podem ter efeito errado se elas são executadas mais de uma vez.

  Ex. uma operação para retirar R$100 da conta corrente deveria ser executada apenas uma vez (não idempotente). Executar exatamente uma vez.

Chamada Remota de Procedimento (RPC)

Semântica RPC na presença de falhas

Enquanto cliente e servidor funcionam bem RPC é bastante útil. No caso de erros as diferenças entre chamada remota e local não são tão fácil de mascarar.

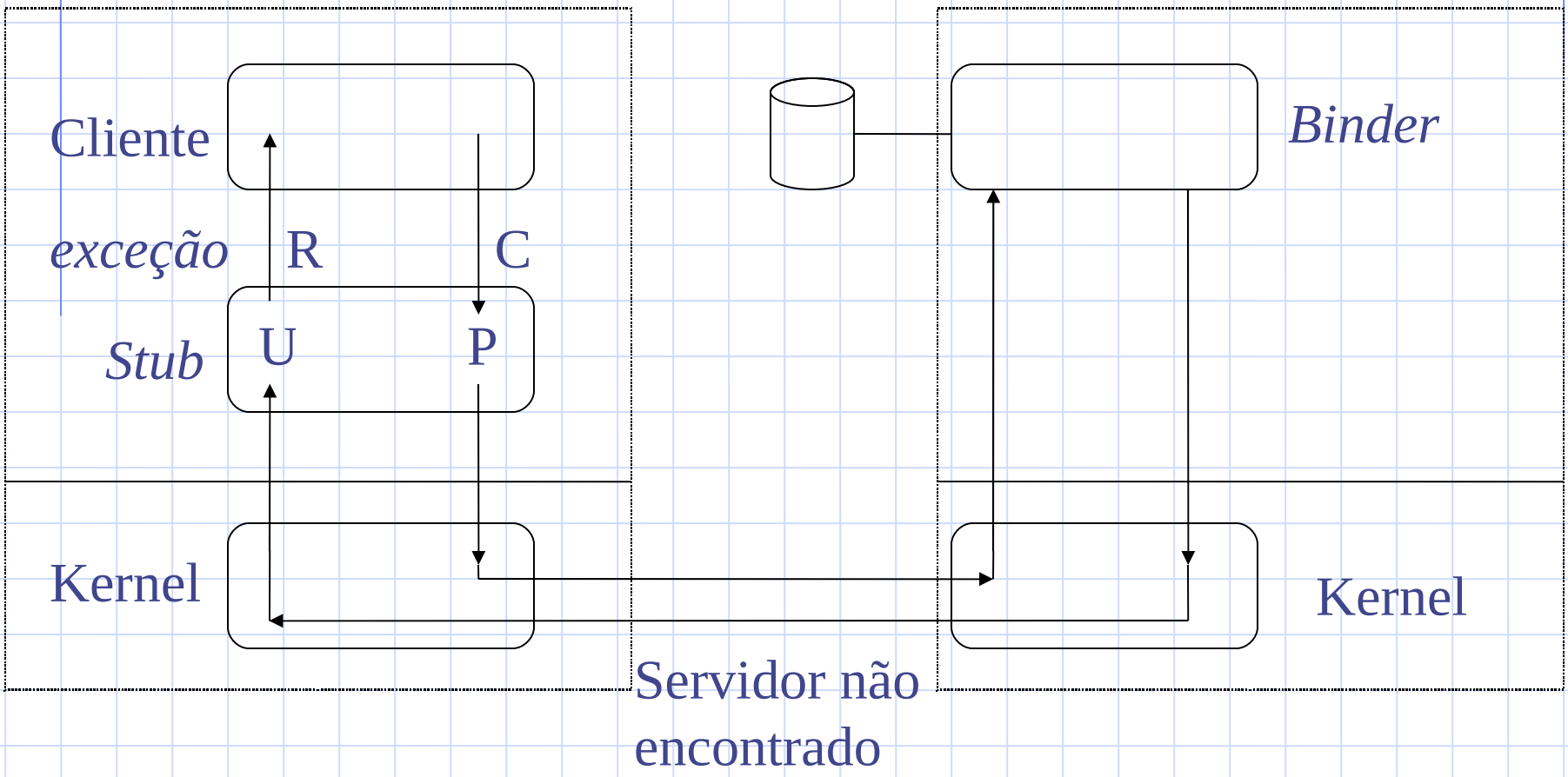
Classes de falhas:

1. O cliente não consegue localizar o servidor
2. A mensagem de requisição (C-S) foi perdida
3. A mensagem de resposta (S-C) foi perdida
4. O servidor quebra depois de receber uma requisição
5. O cliente quebra depois de enviar uma requisição.

Chamada Remota de Procedimento (RPC)

- Semântica RPC na presença de falhas

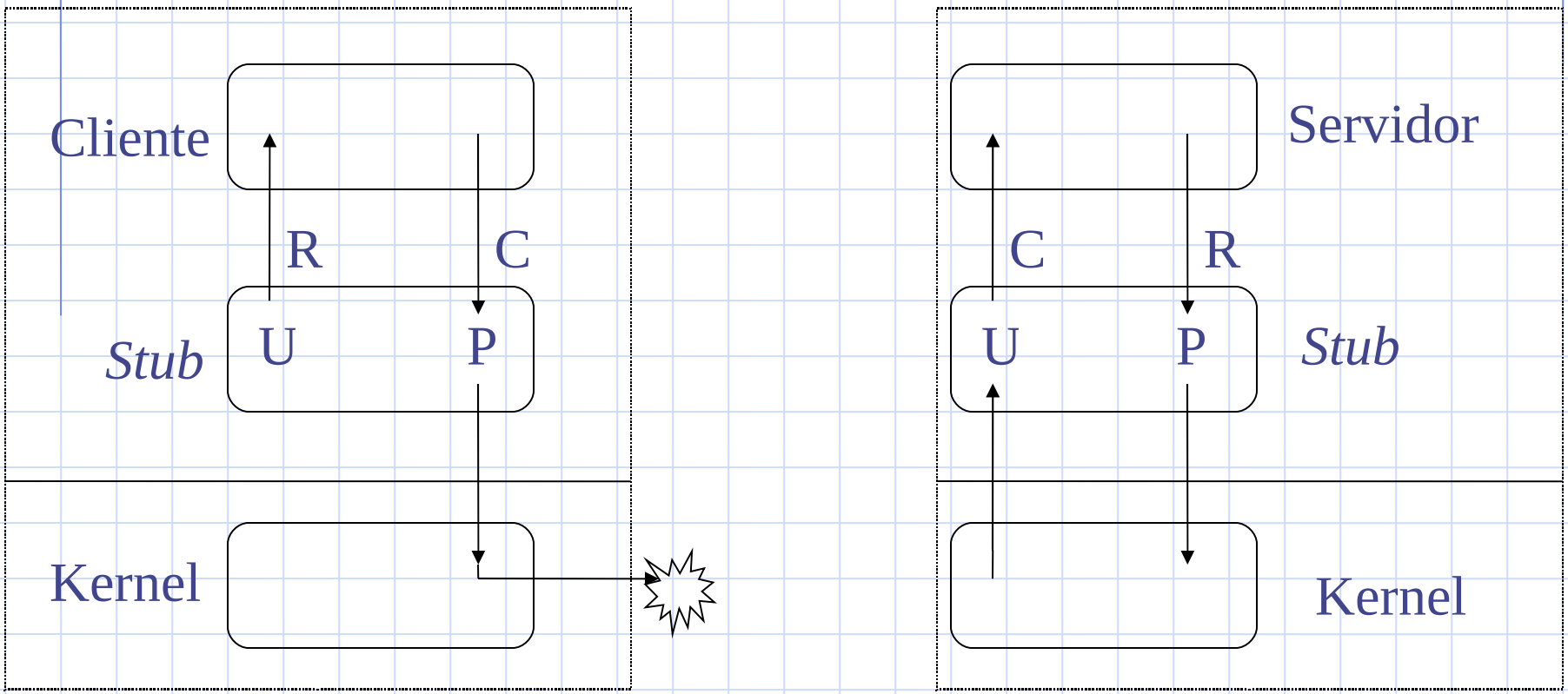
Servidor não localizado



Chamada Remota de Procedimento (RPC)

Semântica RPC na presença de falhas

Requisição perdida

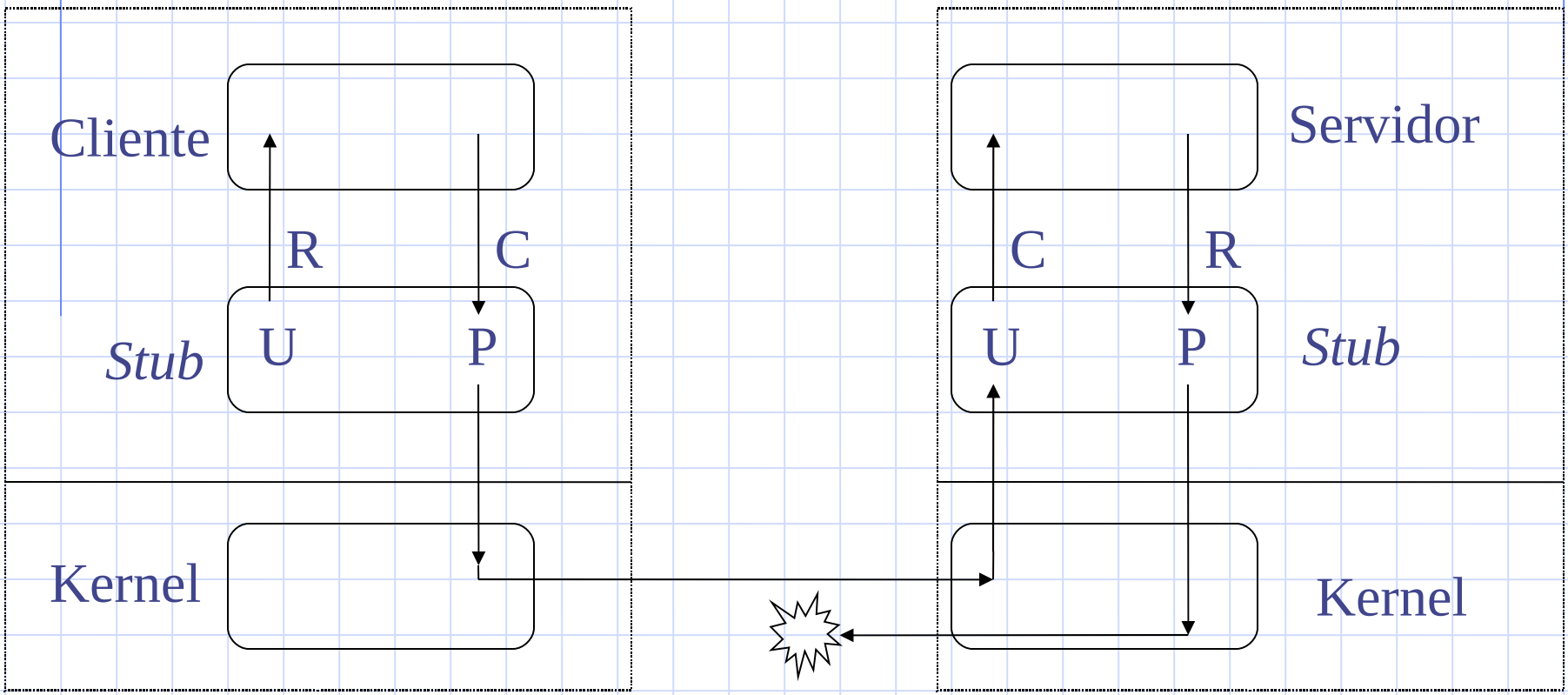


Stub Cliente usa *timeout* e retransmissão

Chamada Remota de Procedimento (RPC)

Semântica RPC na presença de falhas

Resposta perdida

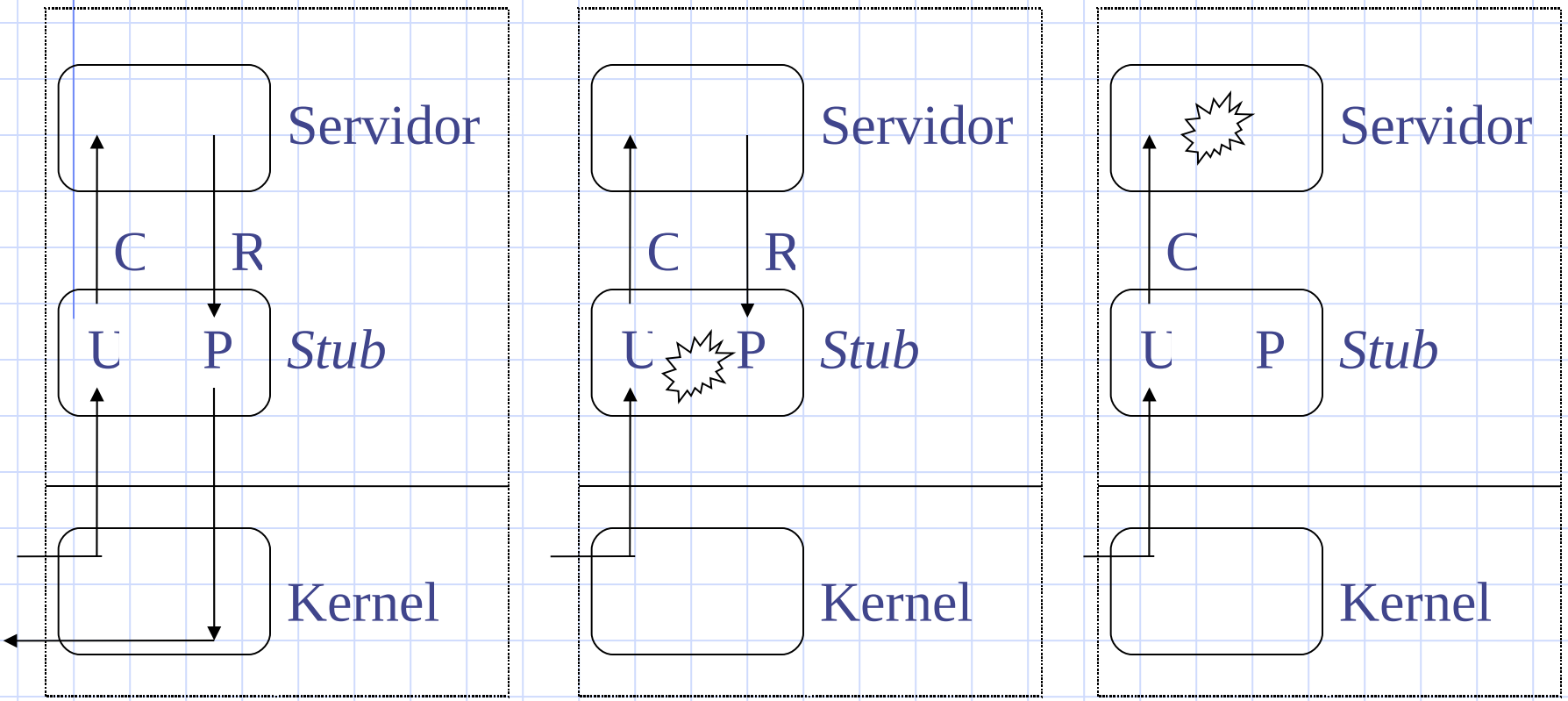


Retransmissão pode causar problemas se não for *idempotente*

Chamada Remota de Procedimento (RPC)

Semântica RPC na presença de falhas

Servidor Quebra



Normal

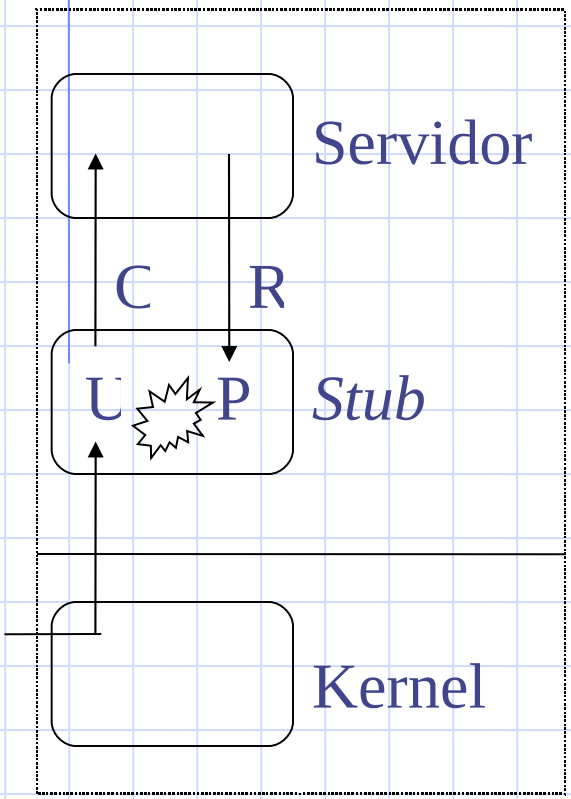
Quebra depois da execução

Quebra antes da execução

Chamada Remota de Procedimento (RPC)

Semântica RPC na presença de falhas

Servidor Quebra



Quebra depois da execução

1. *At least once*: continua tentando até conseguir realizar o RPC (receber resposta). Cuidado com requisições não *idempotentes*.

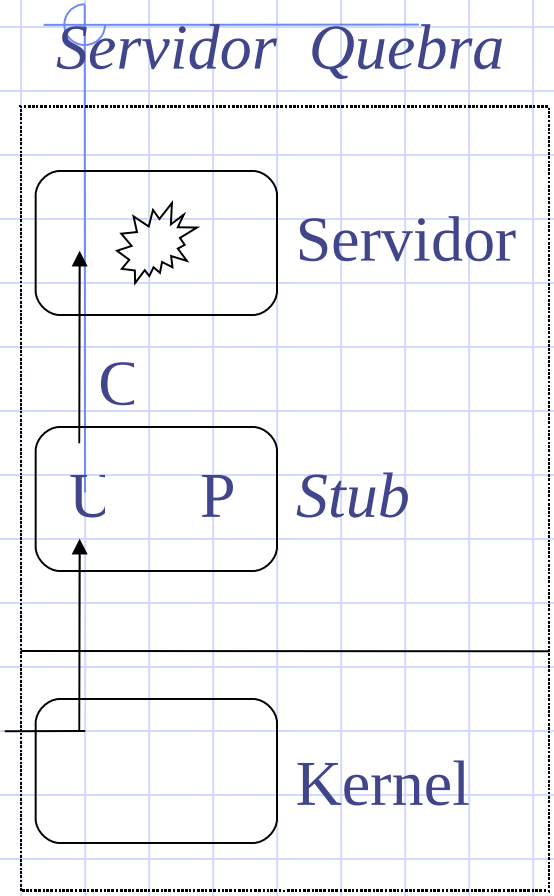
2. *At most once*: desiste imediatamente e informa a falha no RPC. Funciona bem neste caso.

3. Não garante nada.

Melhor: *exactly once*

Chamada Remota de Procedimento (RPC)

Semântica RPC na presença de falhas



Quebra antes
da execução

1. *At least once*: continua tentando até conseguir realizar o RPC (receber resposta). Funciona bem neste caso.

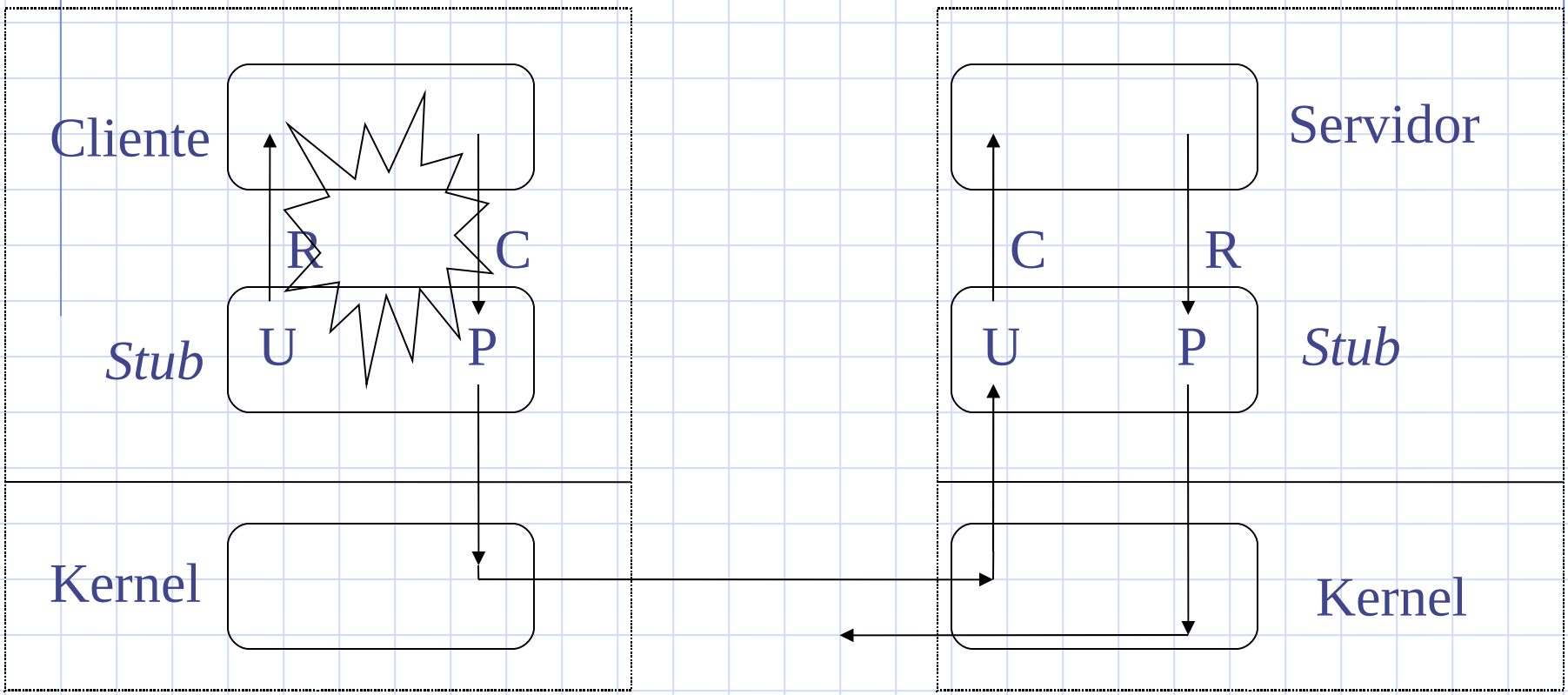
2. *At most once*: desiste imediatamente e informa a falha no RPC. Cliente precisa tentar de novo.

3. Não garante nada.

Chamada Remota de Procedimento (RPC)

Semântica RPC na presença de falhas

Cliente Quebra



Computação no Servidor fica *orfã*. Gasto de CPU, bloqueio de arquivos, etc.

Remote Procedure Call

- **RPC permite a construção de aplicações em rede, fornecendo uma abstração conveniente para ambos IPC e sincronização de eventos.**
- **Desde sua introdução nos 80s, o modelo RPC tem sido bastante usado em aplicações de rede.**
- **Duas importantes APIs para RPC.**
  - ***Open Network Computing Remote Procedure Call*, evoluido de RPC API originario da Sun Microsystems nos 80s.**
  - ***Open Group Distributed Computing Environment* (DCE) RPC.**
- **Ambas APIs fornecem uma ferramenta, *rpcgen*, para gerar as chamadas(stubs).**

SunRPC

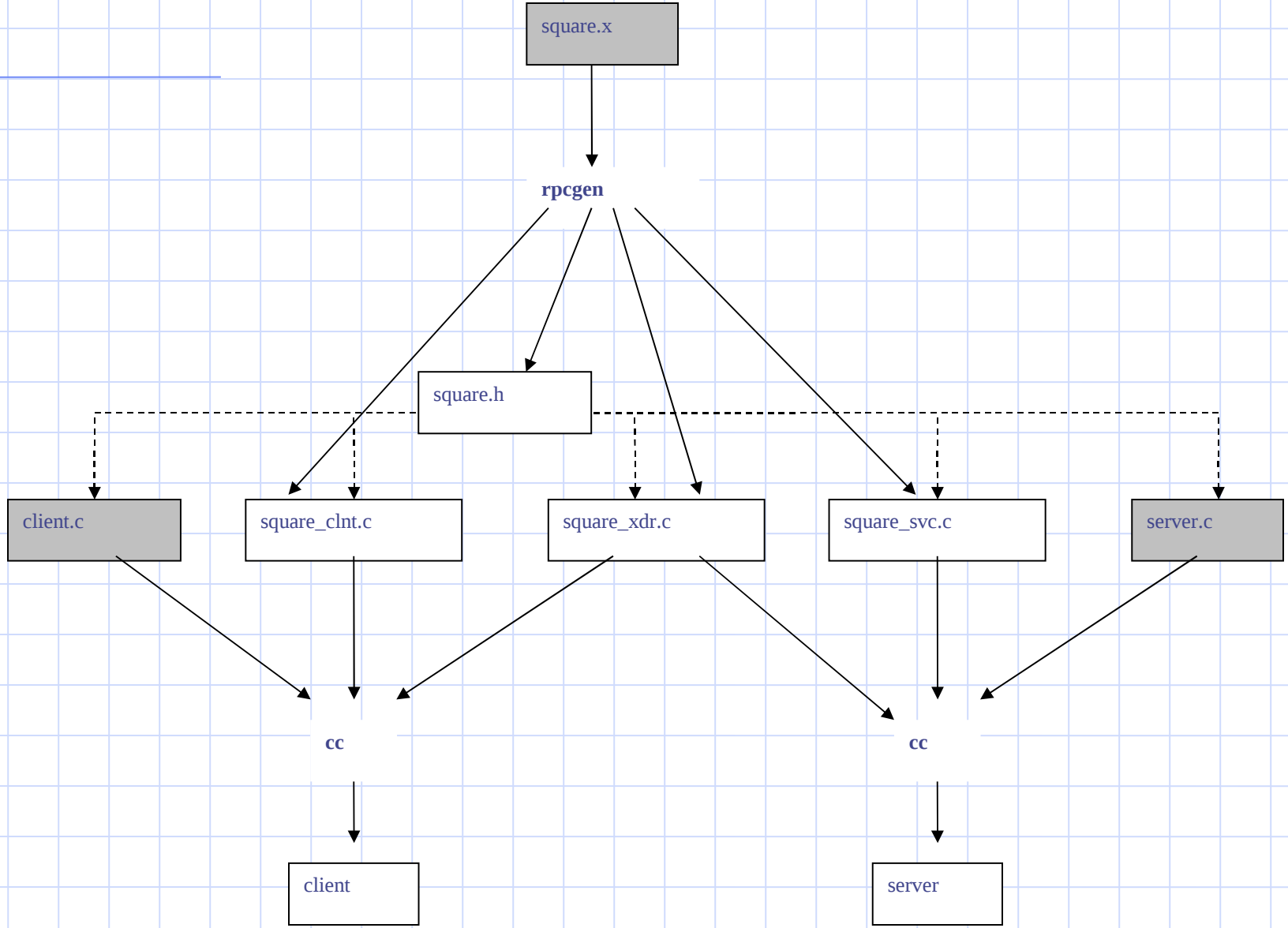
- Geração de Stubs
- Argumentos
- Semântica
- *Binding*
- Segurança
- RPC especial

Geração de Stubs

- Geração automática de stubs. Aplicação escrita em IDL, RPCL, extensão de XDR.
- O compilador *rpcgen* gera a partir de uma definição de interface:
  - arquivo *header* (tipos e constantes arq.h)
  - arquivo XDR (marshal, un.. arq_xdr.c)
  - stub cliente (procedures do cliente arq_clnt.c)
  - stub servidor (rotina principal, despachador, rotinas definidas na interface arq_svc.c)

Geração de Stubs

- Usando os arquivos gerados pelo *rpcgen* uma aplicação RPC é criada da seguinte forma:
  - O programador escreve o *cliente* e o *servidor* (os nomes remotos são os dos *stubs*)
  - O cliente é compilado gerando um objeto
  - O servidor é compilado gerando um objeto
  - O stub cliente e o XDR geram um objeto
  - O stub servidor e o XDR geram um objeto
  - gera executável cliente e servidor


square.x
rpcgen
square.h
client.c
square_clnt.c
square_xdr.c
square_svc.c
server.c
cc
cc
client
server

Argumentos

- Aceita 1 argumento de entrada : procedimentos com mais de 1 estabelecem estruturas.
- Sem argumentos : NULL deve ser passado
- RPC tem sempre 2 arg. : entrada, tratador(socket)
- O retorno é um único resultado. A variável de retorno tem que ser declarada como estática, ou o valor se torna indefinido.
- Marshaling usa eXternal Data Representation

Semântica

- Semântica suportada é *at-least-once* : depois de enviar uma mensagem de requisição o *runtime* espera resposta (*timeout*) antes de retransmitir. O numero de tentativas é o tempo total (25) dividido pelo *timeout* (5). Se neste tempo não vier resposta retorna um erro de *timeout*.

Binding

- O serviço de *binding* no SunRPC não é *networkwide*. Cada nodo tem um *binding* local (*portmapper*) que mantém um base dos serviços oferecidos. O servidor registra *pgr*, *ver* e *port* com o *portmapper* local. O cliente quando faz um RPC ele deve descobrir o *port* do servidor (*clnt_create*). O retorno deste processo é um tratador que é usado pelo cliente para comunicar diretamente com o servidor nas RPCs subsequentes.

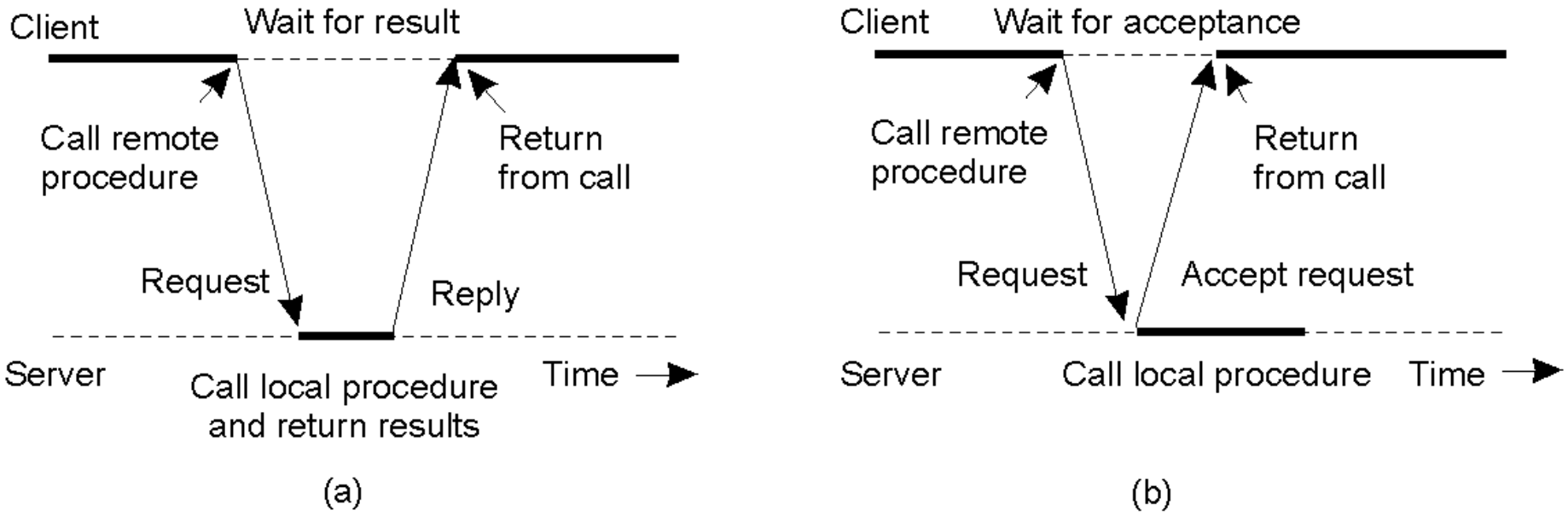
Segurança

- Sem Autenticação: nada é feito para verificar a autenticidade do cliente antes da execução do procedimento requisitado.
- Autenticação UNIX: cada mensagem carrega o *uid* e *gid* do usuário cliente.
- Autenticação DES (Data Encryption Standard): cada mensagem carrega o *netname* (criptografado) do usuário. O servidor decriptografa e decide a execução ou não *(Secure* RPC).

RPC especial

- RPC assíncrono - o valor *timeout* associado ao RPC é zerado e o servidor não gera resposta para a requisição.
- RPC callback - o cliente registra um serviço (prg) *callback* e envia para o servidor.
- RPC broadcast - é enviado para os *potmapper* de todos os nodos. O cliente recebe as respostas.
- RPC batch-mode - é realizado colocando em batch as chamadas de cliente que não precisam resposta e depois manda para o servidor.

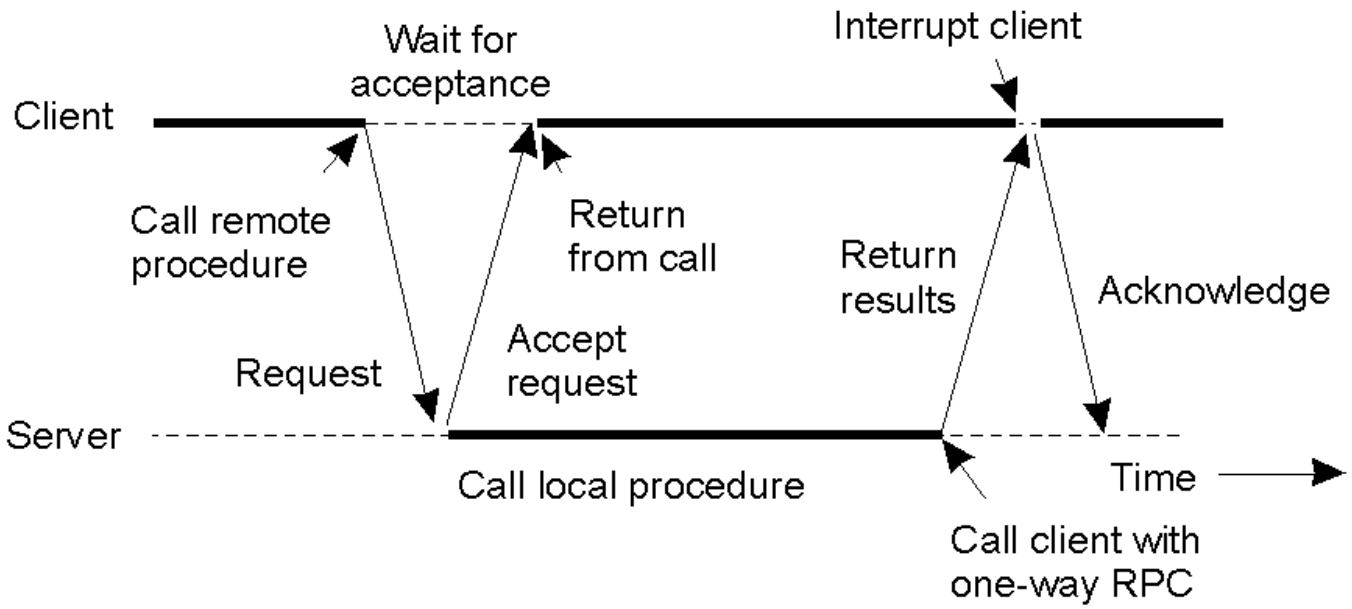
Asynchronous RPC (1)



a) The interconnection between client and server in a traditional RPC

b) The interaction using asynchronous RPC

Asynchronous RPC (2)



A client and server interacting through two asynchronous RPCs