Remote Procedure Call (RPC)

Prof. Marcelo Veiga Neves
marcelo.neves@pucrs.br

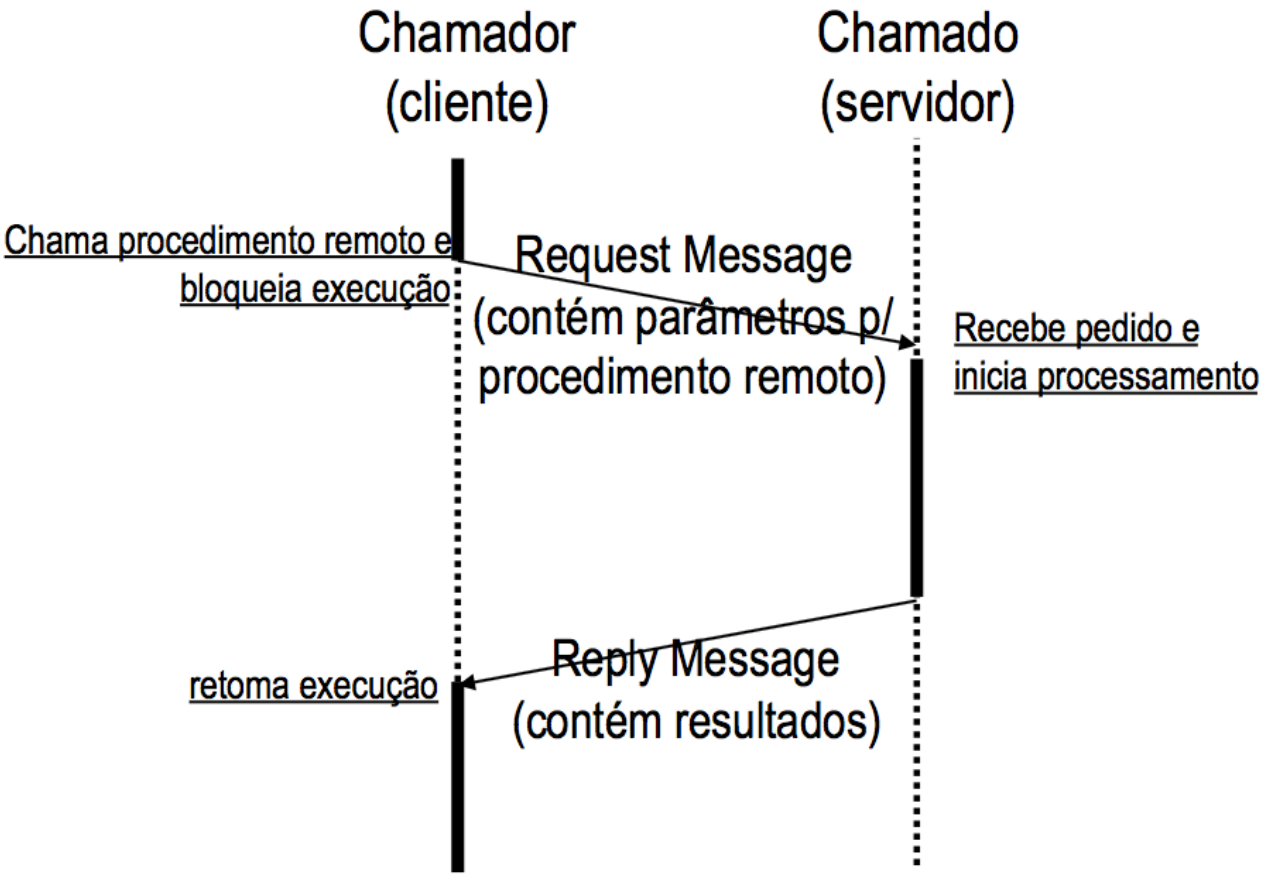
Comunicação entre Processos

- Troca de mensagens:
  - cada aplicação tem um protocolo específico
    - formato de mensagens; forma de tratamento de erros;
    - ex.: servidor de operações matemáticas
      - mensagens: operandos e operação
  - construção de outro cliente: tem que conhecer estes detalhes
  - Problema: necessidade de um protocolo genérico para IPC para o projeto de aplicações distribuídas

Remote Procedure Call (RPC)

- "Chamada remota de procedimento"
- Conceito de procedimento:
  - rotina que, ao ser chamada, recebe argumentos de entrada (passados através da pilha), executa algum processamento (ativado pela instrução CALL) e retorna valores (através de registradores)
- RPC permite executar procedimentos que estão em outras máquinas da rede
- Os argumentos de entrada e os valores a serem retornados são enviados e recebidos através de mensagens

Remote Procedure Call (RPC)



Remote Procedure Call (RPC)

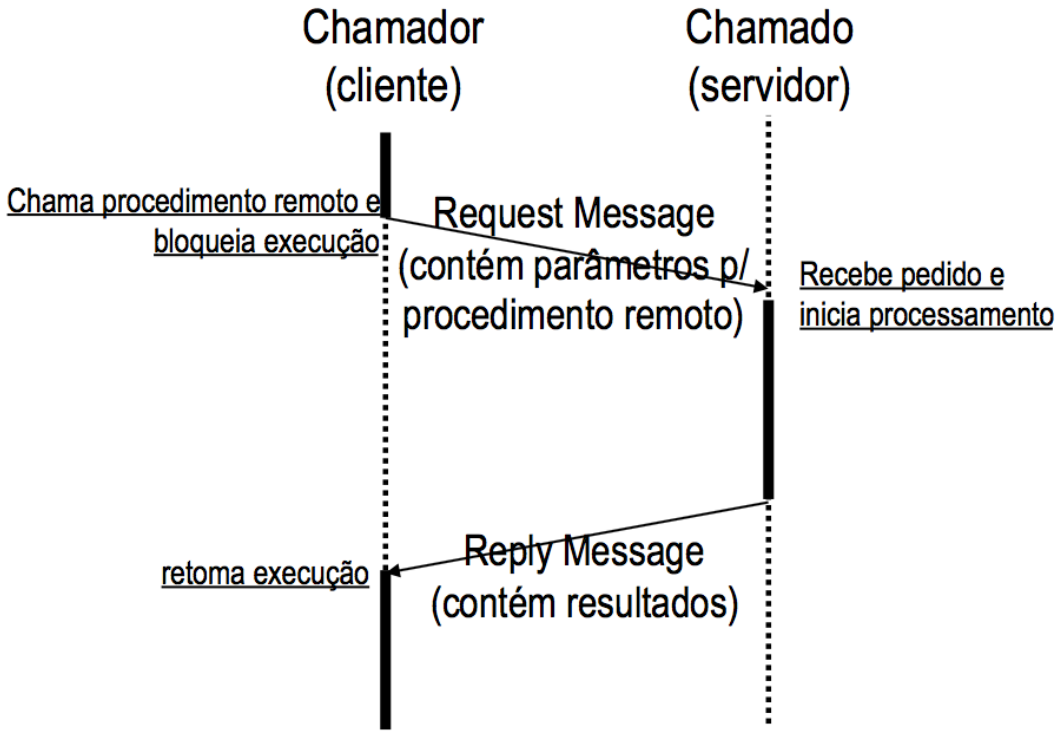
- Similar ao modelo de chamada de procedimentos usado para transferência de controle em um programa
- Chamador coloca argumentos para o procedimento em algum local especificado
- Controle é passado ao procedimento chamado
- Corpo do procedimento é executado
  - pode incluir cópia de parâmetros
- Após o final, o controle retorna ao ponto de chamada do procedimento, podendo envolver retorno de resultados
- Procedimento chamado pode estar na mesma ou em outra máquina
- Espaços de endereçamento separados
  - procedimento chamado não tem acesso a dados e variáveis do ambiente do chamador.

Razões da Aceitação de RPC

- Sintaxe simples
- Semântica familiar - similar a chamadas locais a procedimentos
- Serviço tem interface bem definida
  - verificações são possíveis em tempo de compilação
- Eficiência
- Independência de localização: pode ser usado para comunicação entre processos locais ou remotos
- modelo cliente/servidor
  - um processo, ou um grupo de processos cooperantes, fornecem serviços
  - clientes fazem requisições
  - servidores ficam a espera de requisições, processam e dão a resposta

Modelo de Sincronismo

- Modelo básico de sincronismo
  - Totalmente síncrono
  - Somente um processo ativo em determinado tempo



Modelo de Sincronismo

- Outros modelos de sincronismo são possíveis
  - por exemplo, chamadas assíncronas (não bloqueantes) são possíveis
  - cliente pode processar enquanto espera resposta
  - servidor pode processar requisições em paralelo
    - ex.: lançar threads

Transparência

- Transparência sintática
  - chamada remota ter mesma sintaxe que chamada local
- Transparência semântica
  - aspectos de sincronização: OK
  - diferentes espaços de endereçamento:
    - não há sentido no uso de ponteiros, por exemplo
  - vulnerabilidade a falhas:
    - mais de uma máquina → tipos de falhas que não aconteceriam em local procedure call tem que ser tratados
  - latência da rede:
    - RPC consome muito mais tempo que chamada local: 100 a 1000 vezes mais tempo

Transparência

- RPC deve ser transparente para o usuário?

Transparência

- RPC deve ser transparente para o usuário?
  - Transparência semântica é impossível
  - Usuários/programadores tem os benefícios mas devem estar “cientes” de que um procedimento é remoto e dispor de mecanismos para tratamento de maneira dependente da aplicação de
    - atrasos demasiados
    - falhas

Implementação de RPC

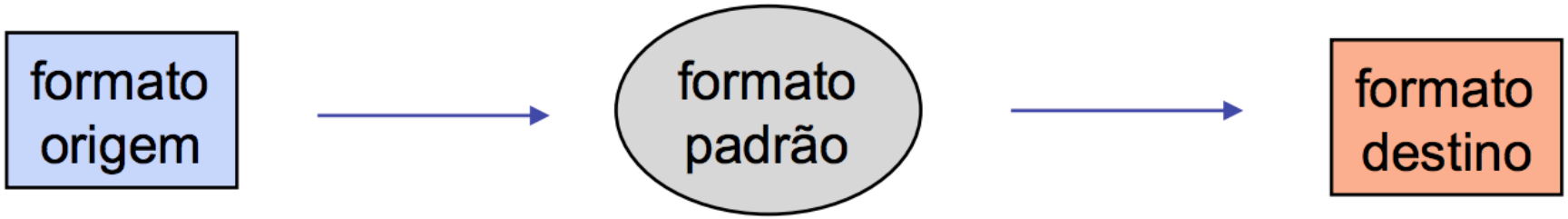
- Sun-RPC
- Originalmente criado para máquinas Sun nos anos 80
  - Como parte do sistema de arquivos NFS
  - Oferecido depois por diversos sistemas operacionais

Arquitetura Sun-RPC

- A arquitetura definida inclui:
  - uma linguagem para definição das interfaces (cabeçalhos de procedimentos, etc);
  - a ferramenta RPCGEN, que gera os stubs cliente e servidor automaticamente;
  - uma biblioteca RPC, que pode ser usada diretamente na construção de programas que não usem o RPCGEN;
  - protocolo de comunicação entre os stubs.
- utiliza TCP ou UDP

Tradução de dados

- Tradução entre formatos de dados:
  - utilização de uma representação padrão, XDR (eXternal Data Representation Standard).



  - conversão é especificada para um conjunto pré-definido de tipos de dados.

Exemplo

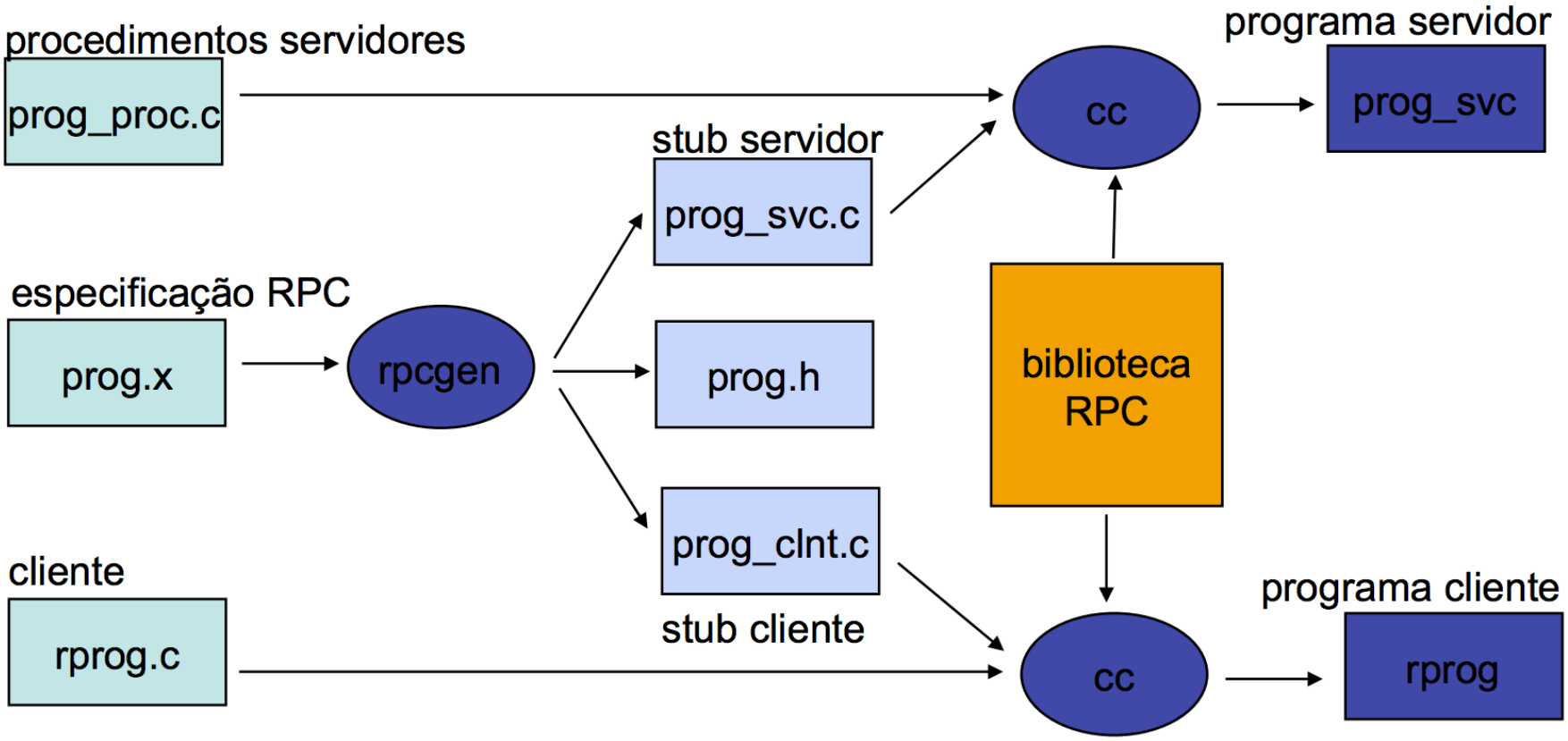
- Especificação da interface do serviço oferecido
- IDL (Interface Definition Language)

```
/* notas.x */

#define PROGRAM_NUMBER 0x20000000
#define VERSION_NUMBER 1

program NOTAS_PROG {                              /* Programa */
        version NOTAS_VERSION {                   /* Versao */
                double OBTEM_NOTA(string) = 1;    /* Funcao e numero */
        } = VERSION_NUMBER;                       /* Numero de versao */
} = PROGRAM_NUMBER;                               /* Numero de programa */
```

Funcionamento rpcgen



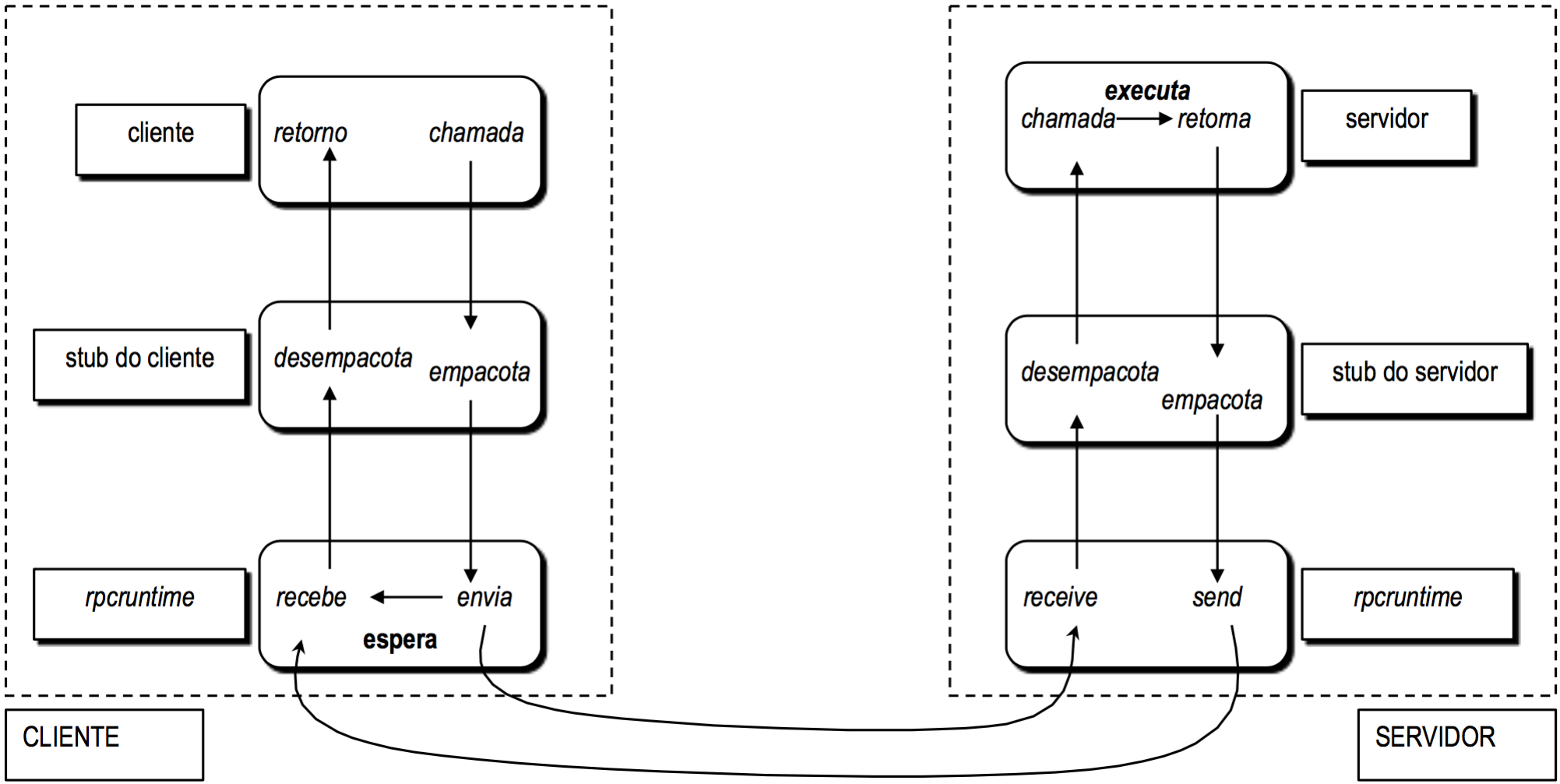
Exemplo

- Geração dos stubs:
  - rpcgen notas.x
  - Arquivos gerados:
    - notas_svc.c (servidor)
    - notas_clnt.c (cliente)
    - notas.h
- Necessário implementar o cliente e o servidor

Componetes Sun-RPC

- Componentes de uma aplicação usando RPC:
  - **Cliente:** faz a chamada ao procedimento remoto
  - ***Stub* do cliente:** faz a interface com o *runtime system* (esconde chamadas de "baixo nível" da aplicação)
  - **RPC *Runtime*:** comunicação entre dois computadores, esconde os detalhes da comunicação de rede
    - retransmissões, confirmações, etc ...
  - ***Stub* do servidor:** mesmo do *stub* do cliente
  - **Servidor**

Implementação de RPC



Exemplo

- Exemplo programa addsub

Referências

- Material baseado em slides dos Profs. Roland Teodorowitsch, Avelino Zorzo, Celso Costa, Fernando Dotti e Luiz Gustavo Fernandes