RPC - Remote Procedure Call

1 - Introdução ao RPC
2 - Objetivos do RPC
3 - Padronização do RPC
4 - Semântica de Execução
5 - Idéia Básica do Proc. RPC - Stub
6 - Identificação de Módulos e Procedimentos
7 - Cliente Remote Procedure Call
8 - XDR - Formatação no RPC
9 - Servidor Remote Procedure Call
10 - Servidor Portmapper

1 - Introdução ao RPC

- A Sun Microsystems desenvolveu o RPC na década de 80 com o objetivo de permitir o desenvolvimento de aplicações cliente/servidor sem necessidade de programar com interfaces sob a camada de transporte como, por exemplo, sockets.

- Este mecanismo de interação cliente/servidor serve de base para o desenvolvimento de serviços estendidos em sistemas operacionais de rede e encontra-se disponível em praticamente todos os sistemas operacionais derivados do UNIX.

... 1 - Introdução ao RPC

- Exemplo de servidores acessados via RPC:
  - servidores NFS (Network File System): permitem acesso transparente à sistemas de arquivos remotos;
  - portmapper: diretório de servidores RPC;
  - servidor de NIS (Network Information System):
  - serviço de nomes e passwords;
  - servidores de estatísticas.
- RPC permite ao projetista de uma aplicação distribuída focalizar na aplicação e não na comunicação entre os vários componentes que compõem a aplicação.

... 1 - Introdução ao RPC

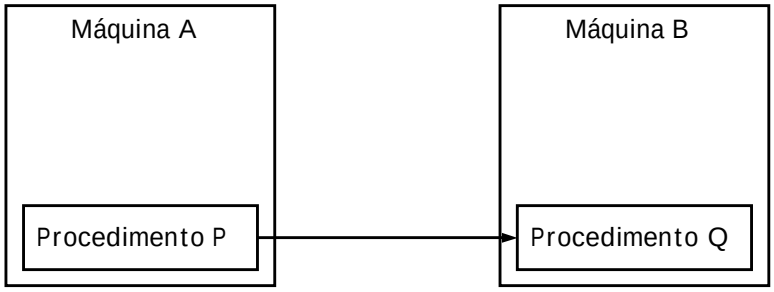
- RPC utiliza os conceitos sedimentados de programação modular e os conceitos mais recentes de orientação a objetos. Assim a aplicação é dividida em módulos:

módulo = { proc. relacionados + dados associados }

  - procedimentos de um módulo podem chamar procedimentos de outro módulo de forma arbitrariamente aninhada
  - módulos podem ser localizados em máquinas distintas para aumentar o desempenho da aplicação

… 1 - Introdução ao RPC

- **chamada de procedimento remoto** - um procedimento P de um módulo localizado numa máquina A chama um procedimento Q de outro módulo localizado numa máquina distinta B.



Pag. 5

2 - Objetivos do RPC

- **Objetivos** de uma implementação de RPC:
  - a chamada de um procedimento deve ser transparente no que diz respeito à localização (local ou remota) do procedimento chamado;
  - a semântica de uma chamada remota deve ser idêntica à de uma chamada local: passa parâmetros e retorna resultados;
  - no caso de erro de implementação, uma chamada local pode não retornar, caso o procedimento chamado entre num laço infinito, porém falhas de comunicação local entre o procedimento chamador e o chamado equivalem a uma falha total do sistema.

Pag. 6

... 2 - Objetivos do RPC

- Esses objetivos foram ***parcialmente conseguidos*** nas implementações correntes:
  - a sintaxe de uma chamada local é idêntica à de uma chamada remota, porém o programador deve codificar uma rotina de inter-face com o sistema **run-time** do RPC chamada de rotina **stub;**
  - módulos remotos devem ter seus procedimentos "registrados" junto ao ambiente de execução do RPC;
  - a semântica de execução não é transparente a falhas de comunicação, ou seja, uma chamada remota pode retornar um erro do tipo "timeout".

Pag. 7

3 - Padronização do RPC

- como já mencionado, a Sun desenvolveu o RPC na década de 80 com o objetivo de permitir o desenvolvimento de aplicações cliente/servidor sem a necessidade de programar diretamente sobre a camada de transporte.
- um sistema RPC completo foi especificado e implementado pela Sun (inspirado em implementações pioneiras na Xerox) e posteriormente padronizado pela **ONC - Open Network Computing**.

Pag. 8

... 3 - Padronização do RPC

- Esta padronização da ONC inclui:
  - um protocolo de comunicação incluindo formatos de mensagens para chamada remota e retorno de resultados;
  - um protocolo para especificação de representação de dados independente de arquitetura ("big endian *versus* little endian")';
  - regras para identificação de módulos remotos e de seus procedimentos;
  - escolha pela aplicação de dois protocolos: TCP ou UDP;
  - especificação de semântica de execução compatível com a do protocolo de transporte escolhido pela aplicação.

Pag. 9

4 - Semântica de Execução

- Idealmente uma chamada remota deveria garantir a execução do procedimento chamado exatamente uma vez, mas isto é impossível de garantir na presença de erros de comunicação, mesmo se utilizarmos o protocolo TCP .
- Se ocorre *timeout* no chamador após uma chamada ser encaminhada pelo protocolo de comunicação, várias situações podem ocorrer:
  - a comunicação foi interrompida e a chamada se perdeu ou o servidor caiu antes de executá-la;
  - a chamada foi executada mas a comunicação foi interrompida e a resposta se perdeu, ou o servidor da chamada caiu após executar a chamada.

Pag. 10

... 4 - Semântica de Execução

- Mesmo se o "run-time" do RPC for implementado sobre TCP, no 1º caso a chamada remota não foi executada e no 2º caso foi executada uma vez. Não obstante, o chamador não fica sabendo qual das duas possibilidades ocorreu, ou seja, a semântica de execução é: "no máximo uma execução" ("at most once").

- No caso de retorno sem erro, o protocolo TCP garante que a execução ocorreu exatamente uma vez , o que caracteriza situação de normalidade.

... 4 - Semântica de Execução

- Se o protocolo utilizado for UDP, a situação é bem pior, pois nenhuma garantia pode ser dada:
  - o pacote com o pedido de chamada pode se perder ou ser duplicado, e a resposta pode se perder, de forma que a chamada pode ser não ser executada, ou ser executada uma ou mais vezes;
  - a semântica de execução é - se há retorno: "**execução uma ou mais vezes**" ("at least once semantics"), se não há retorno: "**execução zero ou mais vezes**".

5 - Idéia Básica do Proc. RPC - *Stub*

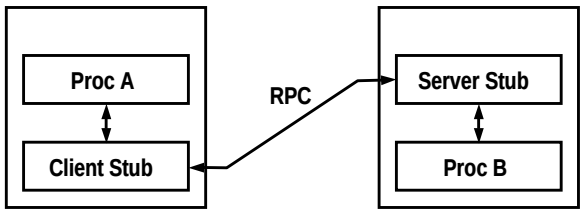
- ***stub*** **-** ***"a short piece left over"*** **ou**
  "any short or blunt projection"

- Suponha que um procedimento A (no cliente) chama B e queremos que B execute remotamente;
  - no cliente, um procedimento "stub", BB, faz o papel de um "dublê" de B - ele apresenta exatamente a mesma interface para A, recebendo os parâmetros passados por A e retornando os resultados esperados por A;
  - no servidor, o procedimento "stub" AA faz o papel de "dublê" de A, passando para B os parâmetros que seriam passados por A e recebendo os resultados retornados por B.

Pag. 13

... 5 - Idéia Básica do Proc. RPC - *Stub*

- BB chama uma rotina que vai enviar uma mensagem via socket para a máquina onde está B e recebe dessa máquina outra mensagem contendo os resultados da chamada remota ;
- AA recebe (no servidor) a mensagem originada no cliente e chama B, passando os parâmetros recebidos e gerando uma mensagem de volta com os resultados retornados por B;
- AA e BB devem ser escritos pelo programador da aplicação RPC e é um dos poucos pontos onde a transparência é perdida.

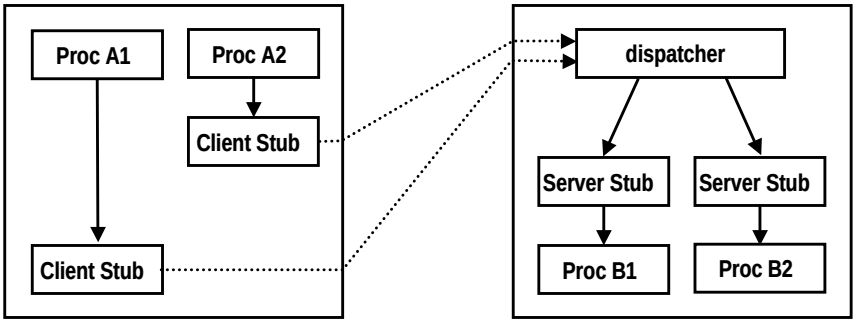


Pag. 14

… 5 - Idéia Básica do Proc. RPC - *Stub*

- O cenário acima é aproximado, mas transmite a idéia básica de uma implementação RPC, onde vários procedimentos são envolvidos para emular a chamada remota do lado do cliente e do lado do servidor:
  - um procedimento em cada lado faz a transformação dos parâme-tros e resultados para/de uma representação independente de arquitetura, e outros procedimentos formatam e encaminham mensagens via sockets.
  - existe uma ferramenta especial, "**rpcgen**", que gera vários desses procedimentos, liberando o programador de conhecer detalhes das rotinas que constituem a biblioteca de execução ou "run-time" do sistema RPC.

… 5 - Idéia Básica do Proc. RPC - *Stub*



Multiple Remote Procedures e Dispatching

6 - Identificação de Módulos Remotos e de seus Procedimentos

- Terminologia
  - **programa remoto ≅ módulo remoto ≅ servidor RPC**
  - **chamada remota ≅ cliente**

- como o RPC é um mecanismo genérico, é necessário indicar qual servidor será contactado e qual procedimento dentro deste servidor deverá ser evocado;
- o parâmetro correspondente ao nó remoto é usado pelo RPC para obter o endereço IP do host onde se encontra executando o servidor;

... 6 - Identificação de Módulos Remotos e de seus Procedimentos

- O parâmetro do nome do programa é, na realidade, um número atribuído por um administrador da rede para designar processos servidores individuais.
- O Padrão RPC especifica que para cada programa remoto executando em um computador, deve ser associado um par de números de 32 bits escolhidos pelo programador.

(#prog, #versão)

  - **#prog** - definidos pela SUN
  - **#versão** - define a versão de implementação do Sist. RPC, começando por 1.

... 6 - Identificação de Módulos Remotos e de seus Procedimentos

- Da mesma forma, também é atribuído um inteiro para cada procedimento dentro do programa remoto, identificado univocamente pela tripla:

  (#prog, #versão, #proc)

  - **#prog** identifica o programa remoto e
  - **#proc** identifica um procedimento dentro do programa remoto

- Os procedimentos são numerados sequencialmente a partir de 1. Por convenção, o número 0 é reservado para um procedimento de ***echo*** que pode ser utilizado para testar se o programa remoto pode ser alcançado.

... 6 - Identificação de Módulos Remotos e de seus Procedimentos

- Para evitar conflitos entre programas de diferentes organizações, a Sun divide os números dos programas em grupos, como ilustrado a seguir:

de	até	valores atribuídos por
0x00000000	0x1FFFFFFF	Sun Microsystems, Inc.
0x20000000	0x3FFFFFFF	Gerente Local
0x40000000	0x5FFFFFFF	Transiente (temporário)
0x60000000	0x70000000	Reservado
... ...	... ...	Reservado
0xE000000	0xFFFFFFFF	Reservado

... 6 - Identificação de Módulos Remotos e de seus Procedimentos

Nome	Número Atribuído	Descrição
portmap	100000	portmapper
rstatd	100001	rstat, rup, e perfmeter
rusersd	100002	remote users
nfs	100003	network file system
ypserv	100004	yp (NIS)
mountd	100005	mount, showmount
dbxd	100006	DBXprog (debugger)
ypbind	100007	NIS binder
walld	100008	rwall, shutdown
yppassword	100009	yppassword
etherstatd	100010	ethernet statistics

Pag. 21

... 6 - Identificação de Módulos Remotos e de seus Procedimentos

Nome	Número Atribuído	Descrição
rquotad	100011	rquotaprog, quota, rquota
sprayd	100012	spray
selection_svc	100015	selection service
dbsessionmgr	100016	unify, netdbms, dbms
rexd	100017	rex, remote_exec
office_auto	100018	alice
lockd	100020	kimprog
lockd	100021	nimprog
statd	100024	status monitor
bootparamd	100026	bootstrap
pcnfsd	150001	NFS for PC

Pag. 22

... 6 - Identificação de Módulos Remotos e de seus Procedimentos

- A comunicação do cliente com um programa remoto é feita através de sockets TCP ou UDP.
- É inconveniente prefixar um número de port para o programa remoto ou servidor, pois:
  - não há um mapeamento entre #prog (32 bits) e porta (16 bits);
  - programas remotos são em geral transientes.

Pag. 23

... 6 - Identificação de Módulos Remotos e de seus Procedimentos

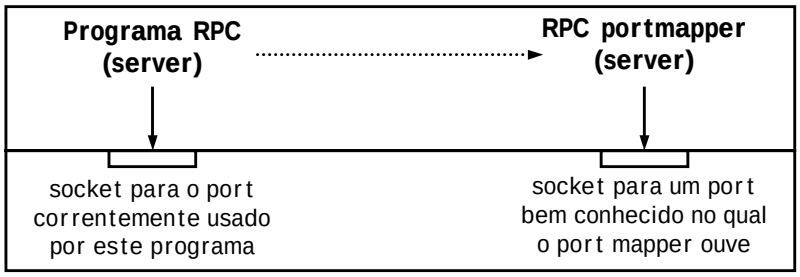
- Solução: mapeamento dinâmico entre **#prog** e **#port**
  - cada máquina que suporta RPC possui um servidor RPC chamado **portmapper** associado à um **port** conhecido 111;
  - quanto um programa remoto é iniciado, ele obtém um *socket* local com um #port transiente e o repassa ao servidor **portmapper** de sua máquina através da mensagem RPC contendo (#prog, #versão, #port transiente);
  - ao receber a mensagem, o servidor **portmapper** registra numa tabela local em memória;
  - quando a aplicação do cliente é iniciada ela envia uma mensagem RPC contendo (#prog, #versão) para o #port 111 do **portmapper**;
  - o **portmapper** consulta sua tabela e devolve ao cliente o número do **port** associado ao programa remoto, ou seja, do servidor;

Pag. 24

... 6 - Identificação de Módulos Remotos e de seus Procedimentos

- a partir desse ponto os procedimentos de execução localizados no cliente utilizarão este número de **port** recebido, para comunicar com o servidor RPC (programa remoto).

- Todos os passos acima são transparentes ao programador da aplicação RPC.



... 6 - Identificação de Módulos Remotos e de seus Procedimentos

- **Política de Retransmissão** - temporização de duração fixa e um número fixo de retransmissões.

- **Problema da Exclusão Mútua**
  - em princípio, vários clientes poderiam estar chamando simultaneamente procedimentos do mesmo módulo remoto e a execução concorrente desses procedimentos poderia gerar inconsistência nas estruturas de dados globais aos procedimentos.
  - em vez de permitir execução concorrente com mecanismos de exclusão mútua, a especificação de RPC resolveu o problema da forma mais simples possível --- apenas um procedimento de cada vez de um módulo remoto pode estar em execução, assim chamadas pendentes são enfileiradas.

7 - Cliente RPC

- O lado do cliente do protocolo RPC é bastante simples. O cliente ativa **callrpc**, passando-lhe oito parâmetros:
  - nome do nó remoto;
  - nome do servidor;
  - número da versão do servidor;
  - procedimento a ser ativado;
  - tipo do parâmetro de entrada enviado ao procedimento remoto;
  - parâmetro de entrada ou uma estrutura no caso de mais de um parâmetro de entrada;
  - tipo do parâmetro de saída retornado pelo procedimento remoto;
  - parâmetro de saída ou estrutura de dado no caso de mais de um parâmetro de saída.

Pag. 27

... 7 - Cliente RPC

- nó remoto - usado para obter o endereço IP do host onde se encontra executando o servidor;
- nome do servidor - qual servidor será contactado;
  - parâmetro do nome do programa é, na realidade, um número atribuído por um administrador da rede para designar processos servidores individuais.
- versão do servidor - RPC facilita a expansão do servidor ao permitir a atribuição de um número de versão para cada nova versão do servidor, assim versões novas e antigas de um servidor poderão existir;
- procedimento - procedimento dentro deste servidor que deverá ser evocado;

Pag. 28

... 7 - Cliente RPC

- parâmetro de entrada - parâmetros individuais precisam ser combinados em estruturas para que possam ser enviados ao servidor ou programa remoto;
- parâmetro de saída - no caso de mais de um parâmetro de saída, os resultados serão retornados em uma estrutura de dados, e só após serem recebidos serão individualmente extraídos da estrutura.

Pag. 29

8 - XDR - Formatação no RPC

- Como máquinas diferentes possuem formatos diferentes, o RPC utiliza um protocolo chamado ***eXternal Data Representation*** (XDR) para traduzir dados de/para um formato de intercâmbio pré-definido:
  - Várias rotinas de serviço são fornecidas com a RPC para conversão de tipos básicos de dado para XDR.
  - Estas incluem **xdr_int**, **xdr_double**, **xdr_float**, **xdr_long**, **xdr_string**, etc. O parâmetro tipo na evocação do procedimento remoto corresponde ao nome da rotina de serviço XDR que o RPC deve chamar para converter o parâmetro correspondente a ser enviado/recebido para/do procedimento.

Pag. 30

... 8 - XDR - Formatação no RPC

- Por exemplo, suponha que um procedimento remoto, chamado **LOOKUP**, converta um nome em endereço (trata-se de uma função típica do servidor de nome).
- O comando **callrpc** poderia ser o seguinte:

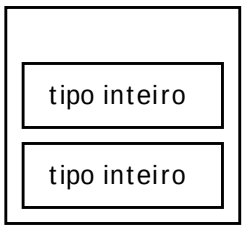
  callrpc(estação, servidor, minha_versão, LOOKUP, xdr_string nome, xdr_long endereço)

- ... no caso onde vários parâmetros devem ser transferidos, estes terão que ser colocados em uma única estrutura de dados.

... 8 - XDR - Formatação no RPC

- Por exemplo, caso dois inteiros tenham que ser enviados a um procedimento remoto, eles podem ser combinados na estrutura mostrada abaixo:

```
struct arg {
   int x;
   int y;
}
```




```
struct arg meus_dados;
meus_dados.x = ...;
meus_dados.y = ...;
```

... 8 - XDR - Formatação no RPC

- Neste caso, o usuário deverá fornecer a sua própria rotina de serviço XDR para o comando **callrpc**.
- Escrever uma rotina deste tipo significa percorrer a estrutura de dados chamando a rotina XDR apropriada para cada elemento na estrutura.
- Seja rotina callrpc usada na seguinte chamada :

  callrpc(ESTAÇÃO, SERVIDOR, VERSÃO, média, xdr_argumentos, meus_dados, xdr_float, resultado)

... 8 - XDR - Formatação no RPC

- Exemplo de rotina para conversão de estrutura de dados para XDR, para o exemplo da rotina apresentada na página anterior:

```
struct arg *pointer;      /* aponta para "meus_dados" */

XDR *xdr_sp;              /* aponta p/ o fluxo de dados XDR */

{
  xdr_int( xdr_sp, &pointer -> x );  /* converte o primeiro elemento */
  xdr_int( xdr_sp, &pointer -> y );  /* converte o segundo elemento */
  return;
}
```

9 - Servidor RPC

- O servidor deve registrar os seus procedimentos que poderão ser acessados remotamente através da rotina **registerrpc** no servidor de **portmapper;**
- Posteriormente, ativa-se a rotina da bilbioteca RPC denominada **svc_run** - esta rotina permite ao servidor esperar as chamadas aos procedimentos registrados.

... 9 - Servidor RPC

- A rotina **registerrpc** da biblioteca RPC utiliza seis (06) parâmetros discriminados a seguir:
  - número do programa a ser registrado;
  - número da versão;
  - número do procedimento que está sendo registrado;
  - nome do procedimento a ser evocado;
  - rotina de serviço XDR a ser evocada para os parâmetros de entrada;
  - rotina de serviço XDR a ser evocada para os parâmetros de saída.

... 9 - Servidor RPC

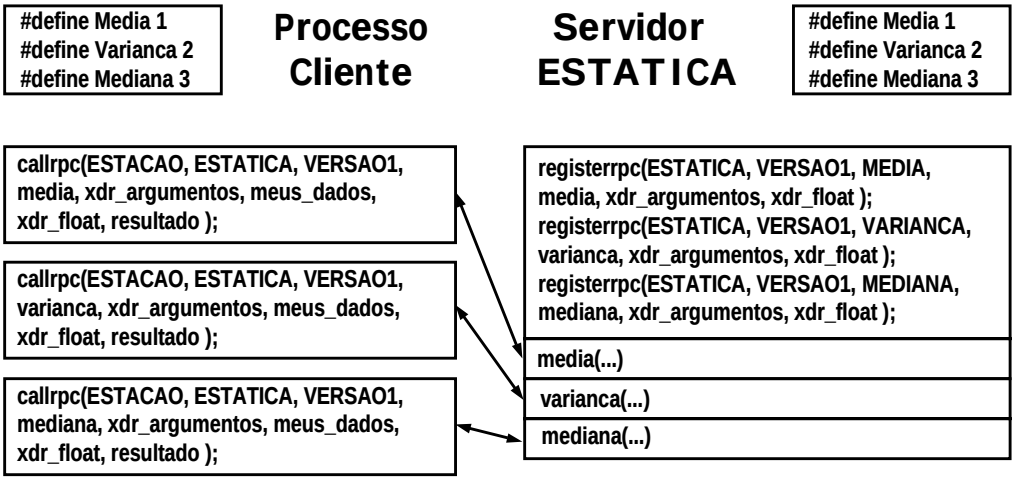
- O diagrama abaixo mostra como registrar o exemplo anterior (conversão de estrutura de dados para XDR):

```
#define UM  1

registerrpc( NÚMERO, SERVIDOR, VERSÃO, UM
             média, xdr_argumentos, xdr_float );

svc_sun;  /* não retorna */
```


... 9 - Servidor RPC



10 - Servidor *Portmapper*

- Conforme mencionado anteriormente, os números de programa RPC são divididos em grupos de 20000000(hex) como mostrado anteriormente.
- A faixa definida pela Sun é usada para processos servidores de interesse geral para usuários da Internet como, por exemplo, Network File System - NFS, Yellow Pages, e outras aplicações.

de	até	valores atribuídos por
0x00000000	0x1FFFFFFF	Sun Microsystems, Inc.

Pag. 39

10 - Servidor *Portmapper*

- A faixa seguinte pode ser usada para servidores específicos da empresa e os números são alocados pelos administrador da rede. A terceira faixa é alocada dinamicamente por rotinas de serviço RPC.

de	até	valores atribuídos por
0x00000000	0x1FFFFFFF	Sun Microsystems, Inc.
0x20000000	0x3FFFFFFF	Gerente Local
0x40000000	0x5FFFFFFF	Transiente (temporário)

Pag. 40

10 - Servidor *Portmapper*

- **alguns exemplos de números de Programas RPC correntemente atribuídos pela SUN Microsystems.**

Nome	Número Atribuído	Descrição
portmap	100000	port mapper
rusersd	100002	remote users
nfs	100003	network file system
ypserv	100004	yp (NIS)
ypbind	100007	NIS binder
walld	100008	rwall, shutdown
yppassword	100009	yppassword
etherstatd	100010	ethernet statistics

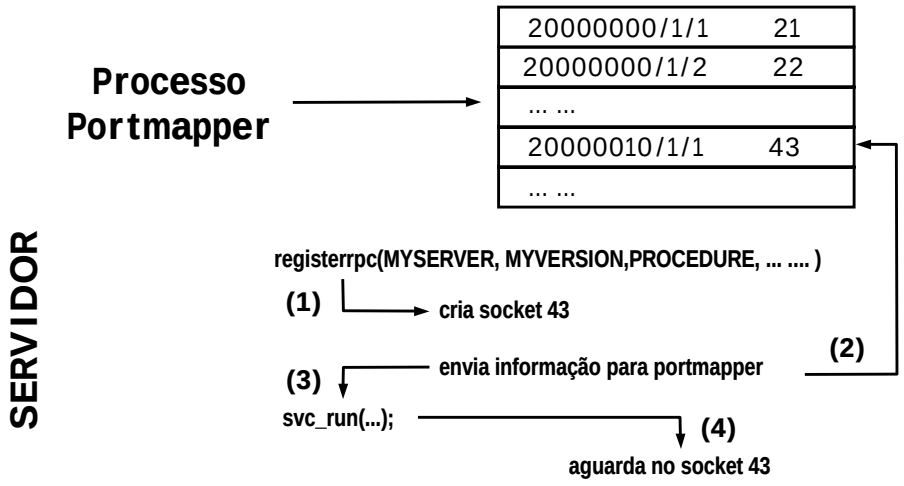
10 - Servidor *Portmapper*

- Os números de programa são ligados dinamicamente a números de porta UDP e TCP por meio do processo RPC denominado **portmapper**. O processo **portmapper** é executado na porta 111.
- Vejamos agora um exemplo que ilustra o uso do **portmapper** - este exemplo foi extraído do livro Introdução aos Sistemas Cliente/Servidor: Guia prático para profissionais de sistemas. Paul E. Renaud.

10 - Servidor *Portmapper*

- Quando um servidor evoca **registerrpc (1),** a rotina da biblioteca cria um *socket* e envia uma mensagem para o **portmapper** informando que o *socket* corresponde a um determinado procedimento remoto **(2).**
- O **portmapper** utiliza esta informação para atualizar uma tabela interna contendo entradas constituídas de **programa /versão/procedimento** associadas a descritores de *socket*.
- A chamada registerrpc retorna **(3)** e o servidor chama **svc_run (4)** passando a aguardar uma evocação do procedimento registrado.

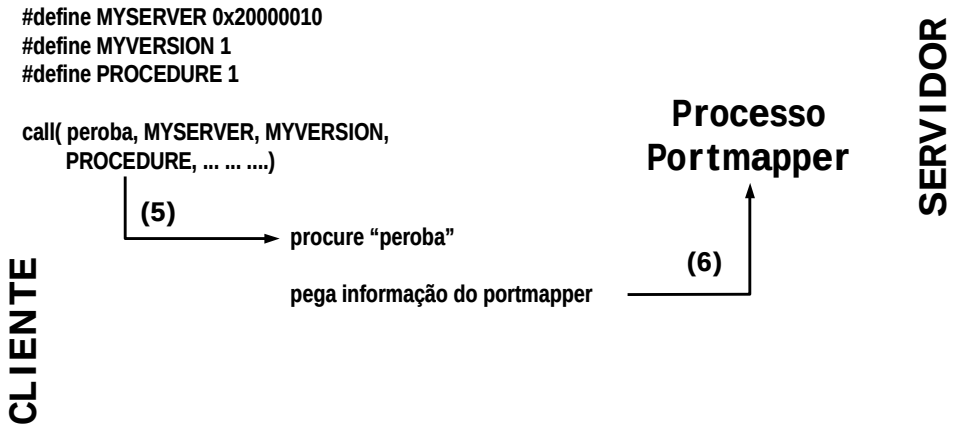
10 - Servidor *Portmapper*



10 - Servidor *Portmapper*

- Quando um cliente ativa **callrpc (5),** a rotina da biblioteca RPC usa o parâmetro correspondente ao nome da estação onde se encontra o servidor para obter o endereço IP da estação.
- Em seguida envia uma mensagem para o **portmapper** rodando nesta máquina **(6).** O **portmapper** pesquisa a tabela usando como entrada o número de **programa/ versão/procedimento** fornecido pelo cliente e determina o descritor de socket que o servidor está usando.
- O **portmapper** responde à rotina da biblioteca RPC que executa do lado do cliente enviando o descritor de socket que o servidor está usando.

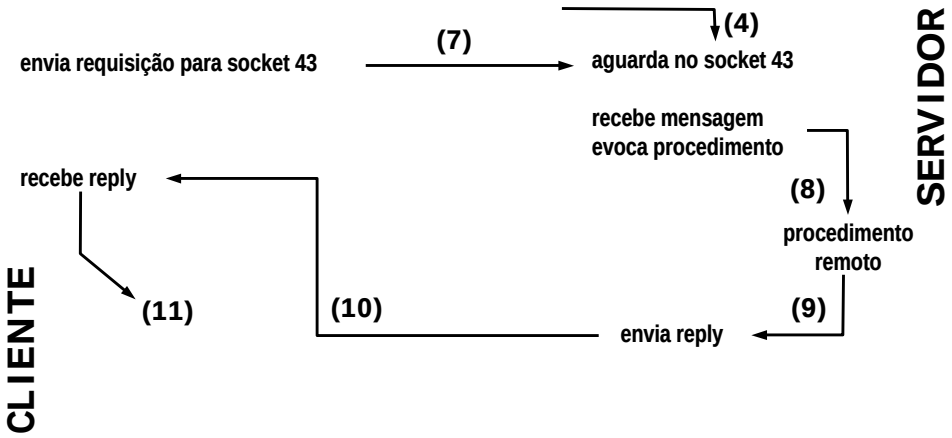
10 - Servidor *Portmapper*



10 - Servidor *Portmapper*

- Na sequência, **callrpc** envia os parâmetros da chamada de procedimento para o processo servidor **(7)** e o procedimento é ativado **(8).**
- Quando o procedimento retorna **(9),** os resultados são enviados de volta a **callrpc (10),** que depois retorna os resultados para o cliente **(11).**

10 - Servidor *Portmapper*



10 - Servidor *Portmapper*

- No caso do exemplo anterior, somente um procedimento remoto foi registrado por parte do servidor.
- Caso mais de um procedimento seja registrado, quando da evocação de **svc_run** o servidor ficará aguardando a evocação de um procedimento remoto em um conjunto de sockets.