RPC – Remote Procedure Call

1. Ce este RPC ?

RPC este o tehnica puternica pentru construirea aplicatiilor distribuite bazate pe modelul client-server. Modelul extinde notiunea de apel local de procedura, diferenta fiind ca procedura apelata nu se afla in acelasi spatiu de adresare cu procedura apelanta. Cele doua procese implicate pot sa fie pe acelasi calculator sau pot sa fie pe doua calculatoare in retea. Utilizand PRC, programatorii de aplicatii distribuite ocolesc dezvoltarea interfatarii aplicatiei cu reteaua.

Protocolul RPC se afla la nivelul ***Prezentare*** din ***stiva OSI.***

RPC foloseste protocolul XDR (eXternal Data Representation) – RFC 1832- pentru codarea datelor. Acest protocol se afla tot la nivelul ***Prezentare*** din ***stiva OSI.***

Versiunea originala de RPC a fost definita in RFC 1050. Versiunea 2 de RPC e definita in RFC 1057, iar versiunea ONC-RPC (Open Network Computing) versiunea 2 e definita in RFC 1831.

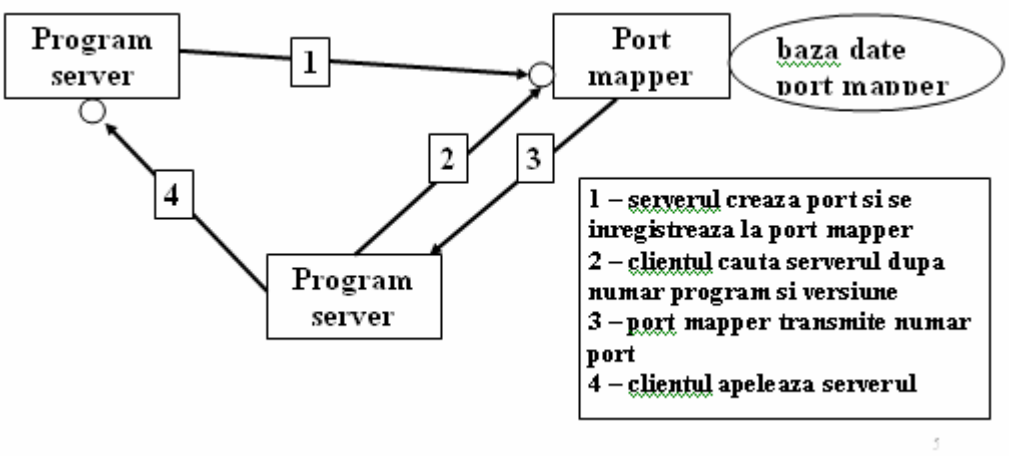
Aplicatii ca **NFS** (Network File System) si **NIS** (Network Information System) sunt bazate pe RPC.

2. Cum functioneaza RPC ?

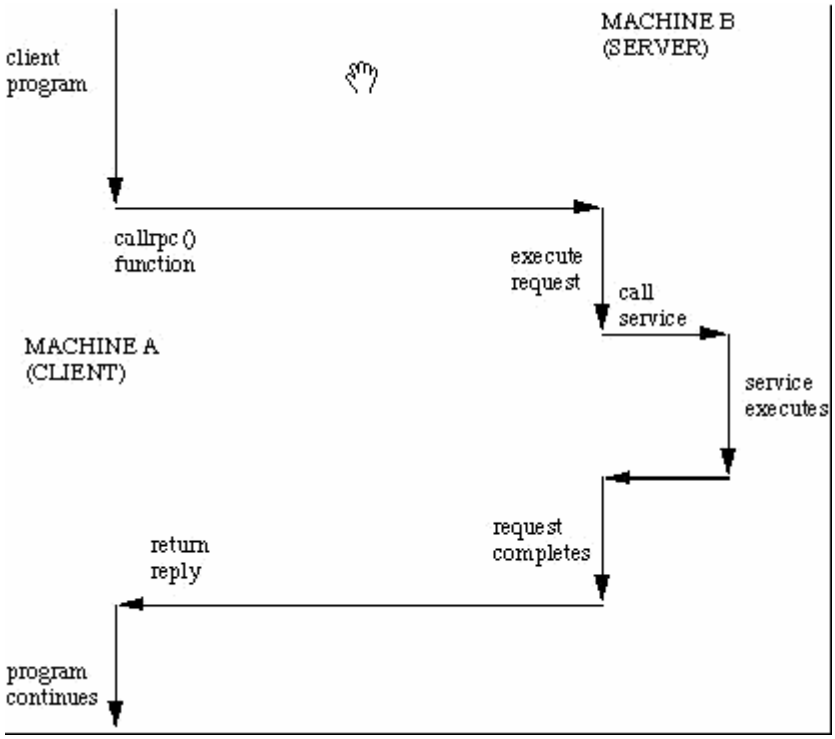
RPC furnizeaza un punct central pentru aplicatiile server sa obtina porturi TCP/UDP pe masina respectiva pe care ruleaza. Aplicatiile nu trebuie « legate » pe anumite porturi specificate pentru ca **serviciul de port mapping** al RPC-ului furnizeaza porturi libere mai mari de 1024 serverelor RPC. Serviciul de lookup folosit pentru asa ceva include **PORTMAPPER-UL (PMAP)** si **RPCBIND** care sunt descrise in RFC 1833. Portmapper are un port fix pe care sta deschis (111) fie TCP, fie UDP. Acest serviciu de lookup trebuie deschis inaintea serverului/clientului si trebuie sa ramana functional pe toata durata executiei aplicatiei RPC.

O problema este introdusa de acest mecanism. Serverele RPC nu utilizeaza porturi rezervate (asa cum sunt cele specificate pentru servicii in fisierul */etc/services*); atunci cand pornesc, utilizeaza portul disponibil dat de serviciul **portmapper.** Atunci cand un program client doreste sa apeleze un anumit serviciu RPC, el nu stie pe ce port ruleaza acesta pentru a face apelul. Trebuie sa existe o metoda de aflare a acestui port.

Deamonul portmapper rezolva aceasta problema. Atunci cand un client face un apel RPC, mai intai apeleaza serviciul de portmapper pentru aflarea portului serverului.



Un apel RPC este analog cu un apel de functie. Ca la un apel de functie, atunci cand un apel RPC este facut, argumentele de apelare sunt trimise procedurii apelante si procesul apelant asteapta intoarcerea raspunsului de la procedura de la distanta. Figura urmatoare arata functionarea apelului RPC intre doua procese pe doua sisteme din retea.



O procedura la distanta este identificata in mod unic prin (numar program, numar versiune, numar procedura). **Numarul de program** identifica un numar de proceduri inrudite, ficare din aceste avand un **numar unic de procedura**. Fisierul */etc/rpc* mapeaza numele de serviciul cu numarul de program dat :

```
# RPC program number data base
# $Revision: 1.5 $       (from 88/08/01 4.0 RPCSRC)
#
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rup perfmeter
rusersd         100002  rusers
nfs             100003  nfsprog
```

```
ypserv          100004  ypprog
mountd          100005  mount showmount
ypbind          100007
walld           100008  rwall shutdown
yppasswdd       100009  yppasswd
etherstatd      100010  etherstat
rquotad         100011  rquotaprog quota rquota
```

Un program poate avea mai multe versiuni. Fiecare versiune are un numar de proceduri care pot fi apelate la distanta. **Numarul de versiune** permite posibilitatea existentei simultane a mai multor versiuni ale unui protocol RPC.

3. Dezvoltarea de aplicatii RPC

Sa consideram urmatorul exemplu simplu :

O aplicatie client/server care cauta intr-o baza de date de pe o masina din retea. Presupunem ca baza de date nu poate fi accesata de pe statia locala folosind NFS. Exista urmatoarele alternative:

- *Utilizam un shell remote* si executam comanda in acest fel. Exista urmatoarele probleme cu aceasta metoda:
  - comanda se executa greu (timp mare) ;
  - trebuie ca utilizatorul sa aibe un cont pe masina respectiva.

- *Alternativa RPC* este :
  - dezvoltarea unui server care sa ruleze pe masina pe care este baza de date si sa raspunda la apelurile remote ;
  - primirea informatiilor la potentialii clienti printr-o metoda mai rapida ca cea descrisa anterior .

Pentru dezvoltarea unei aplicatii RPC trebuie pasii :

1. Construirea unui protocol pentru comunicatia client/server.
2. Dezvoltarea unui program server.
3. Dezvoltarea unui program client.

Aceste programe vor fi compilate separat. Comunicatia se realizeaza prin generarea de stub si skeleton pentru partea de client, respectiv server.

4. Dezvoltarea de aplicatii RPC :

Cea mai simpla metoda de definire si generare a protocolului este utilizarea unui compilator de protocol, cum ar fi ***rpcgen***.

Pentru protocol trebuie specificate numele procedurilor, tipurile argumentelor procedurii si tipul rezultatului. ***Rpcgen*** citeste aceasta definitie si genereaza automat stub-urile pentru client si pentru server.

Rpcgen foloseste propriul limbaj (RPCL). ***Rpcgen*** citeste fisiere speciale cu prefixul .x.

Pentru a compila un fisier RPCL trebuie executata comanda :

rpcgen rpcprog.x

Acest apel va genera urmatoarele fisiere :
- rpcprog_clnt.c – the client stub ;
- rpcprog_svc.c – the server stub;
- rpcprog_xdr.c – XDR (eXternal Data Representation) filters
- rpcprog.h – the header file needed for XDR filters.

XDR (eXternal Data Representation) este un protocol de reprezentare abstractizata a datelor necesar pentru comunicatia independenta intre doua masini diferite.

4. *Compilarea si rularea aplicatiei:*

Presupunem ca aplicatia client se numeste *rpcprog.c* iar aplicatia server se numeste *rpcsvc.c* si protocolul a fost definit in rpcprog.x, iar *rpcgen* a fost utilizat pentru a produce fisierele stub si filtru: *rpcprog_clnt.c, rpcprog_svc.c, rpcprog_xdr.c, rpcprog.h.*

Programele client si server trebuie sa includa *rpcprog.h (*#include rpcprog.h).

Pasii pentru compilarea clientului :
- compilarea codului client :
  - *cc -c rpcprog.c*
- compilarea stub-ului client:
  - *cc -c rpcprog_clnt.c*
- compilarea filtrului XDR:
  - *cc -c rpcprog_xdr.c*
- construirea executabilului pentru client:
  - *cc -o rpcprog rpcprog.o rpcprog_clnt.o rpcprog_xdr.c*

Pasii pentru compilarea serverului:
- compilarea codului server :
  - *cc -c rpcsvc.c*
- compilarea stub-ului serverului :
  - *cc -c rpcprog_svc.c*
- construirea executabilului serverului :
  - *cc -o rpcsvc rpcsvc.o rpcprog_svc.o rpcprog_xdr.c*

Acum se pot rula programele client, respectiv server pe masinile dorite din retea. Procedurile server trebuie mai intai inregistrate inainte de a putea fi apelate de catre clienti.

5. *Limitari RPC. Solutii.*

RPC nu este suficient pentru crearea de aplicatii distribuite orientate obiect, pentru care comunicatia intre procesele aflate in spatii de adresare diferite se face prin obiecte.

O tehnica pentru rezolvarea acestei probleme este Java RMI (Remote Method Invocation). Dar Java RMI a fost proiectat numai pentru mediul Java. Astfel, toate componentele sistemului distribuit dorit trebuie scrise in Java pentru a functiona corect. (http://www.ecst.csuchico.edu/~amk/foo/advjava/notes/rminotes.html).

JERI (Jini Extensible Remote Invocation) este o extindere a RMI-ului, introdusa in Jini 2.0. Fata de RMI are adaugat mecanismul de securitate puternic si usor configurabil. (http://www.javaworld.com/javaworld/jw-12-2003/jw-1219-jiniology_p.html).

Web Services – XML-RPC

Web services sunt un set de utilitare care permit construirea de aplicatii distribuite deasupra infrastructurii web existente. Aceste aplicatii utilizeaza web-ul ca un fel de „nivel transport”.

XML-RPC este printre cele mai usoare metode de construire a serviciilor de web si permite apelul usor al procedurilor la distanta. XML (eXtensible Markup Language) furnizeaza un „vocabular” pentru descrierea procedurilor la distanta (RPC) care sunt apoi apelate folosindu-se HTTP (HyperText Transfer Protocol).

1. Protocolul XML-RPC (http://www.xmlrpc.com/spec):

Un apel XML-RPC este condus intre doua parti: un client (procesul apelant) si un server (procesul apelat). Serverul este facut disponibil la un anumit URL (de exemplu http://se.rogrid.pub.ro/rpcserv). Pentru a utiliza procedurile de la acest server sunt necesari urmatorii pasi:

1. Programul client face un apel la distanta utilizand un client XML-RPC in care specifica *numele metodei* apelate, *parametrii* si *serverul tinta.*
2. Clientul XML-RPC ia numele metodei si parametrii acesteia si ii impacheteaza ca mesaj XML. Apoi clientul face un apel HTTP POST care contine informatiile specificate XML cate serverul tinta.
3. Serverul HTTP de pe masina tinta primeste cererea POST si paseaza continutul XML la serverul XML-RPC care asculta.
4. Serverul XML-RPC parseaza mesajul XML primit pentru a gasi numele metodei si parametrii acesteia iar apoi apeleaza metoda specificata.
5. Metoda apelata returneaza un raspuns la procesul XML-RPC care incepe sa-l impacheteze ca un raspuns XML.
6. Serverul de web returneaza acest raspuns XML la cererea HTTP POST.
7. Clientul XML-RPC parseaza raspunsul XML pentru a gasi rezultatul apelului, pe care-l intoarce programului client.
8. Programul client isi continua executia.

Un proces poate fi atat server (pentru unii clienti) cat si client (pentru alte servere).

Utilizarea protocolului HTTP face ca apelurile XML_RPC sa fie ***sincrone*** si ***stateless.***

Un apel XML-RPC este urmat de exact un raspuns, acesta fiind sincron cu cererea. Acest lucru se intampla pentru ca raspunsul trebuie sa se produca pe aceeasi conexiune HTTP ca si cererea.

HTTP este un protocol fara stare (stateless). Aceasta inseamna ca nici un context nu este retinut de la o cerere la urmatoarea. XML-RPC mosteneste aceasta caracteristica. Deci, daca un program client invoca de mai multe ori aceeasi metoda de pe server, XML-RPC trateaza aceste cereri total separat. Acest lucru ocoleste overhead-ul implicat in mentinerea starii pe mai multe sisteme.

2. *Cum arata o cerere si un raspuns XML-RPC:*

Cererea in XML:

```
POST /rpchandler HTTP/1.0
User-Agent: AcmeXMLRPC/1.0
Host: xmlrpc.example.com
Content-Type: text/xml
Content-Length: 165

<?xml version="1.0"?>
<methodCall>
        <methodName>getCapitalCity</methodName>
        <params>
                <param>
                <value><string>England</string></value>
                </param>
        </params>
</methodCall>
```

Raspunsul in XML:

```
<?xml version="1.0"?>
<methodResponse>
        <params>
          <param>
            <value><string>London</string></value>
          </param>
```

```
        </params>
    </methodResponse>
```

Pentru structura detaliata XML-RPC vedeti documentatia de pe site.

3. Alternative XML-RPC:

Protocoalele alternative XML-RPC pentru creare de Web Services sunt :

- SOAP (Simple Object Access Protocol) - http://webservices.xml.com/pub/a/ws/2000/11/01/protocols/quickref.html#soap) ;
- UDDI (Universal Description, Discovery, and Integration) - http://webservices.xml.com/pub/a/ws/2000/11/01/protocols/quickref.html#uddi
- WSDL (Web Services Description Language) ;