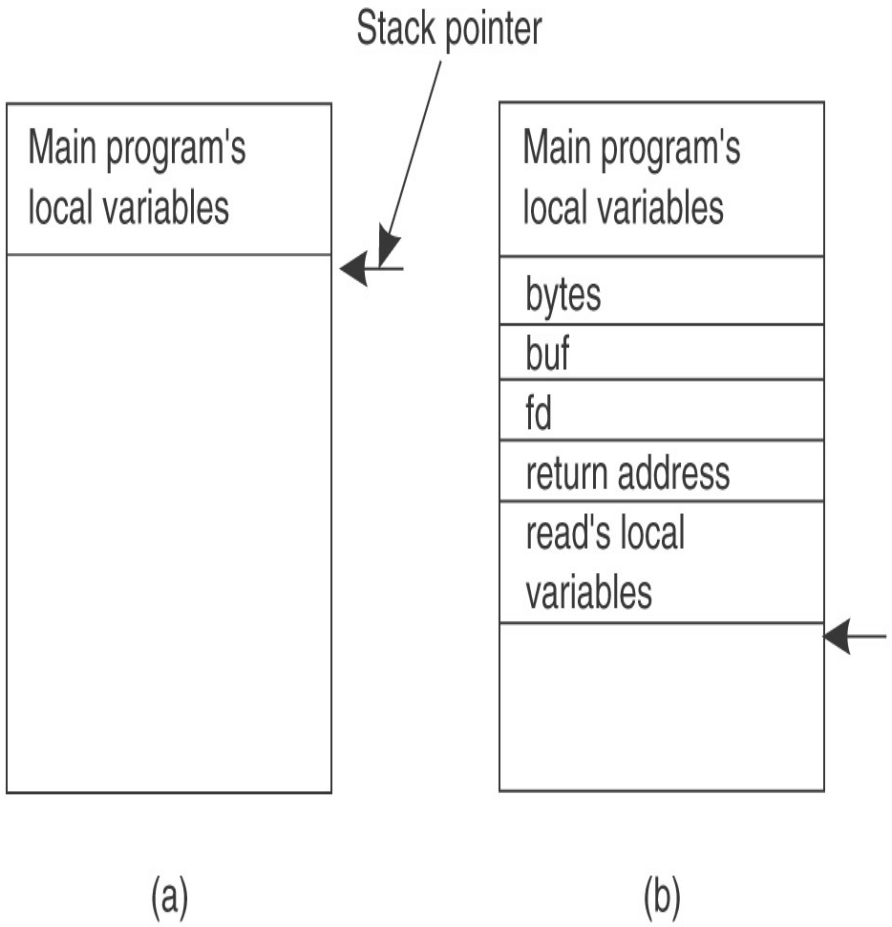
RPC – Remote Procedure Call

De ce apel la distanta ?

- Procedurile (subrutinele) locale au avantaje incontestabile ???
  - structureaza programele
  - permit reutilizarea codului (biblioteci)
  - reduc costul dezvoltarii si intretinerii programelor mari
  - foarte folosite in programarea secventiala
- Motivatia apelurilor la distanta
  - sa beneficieze de aceleasi avantaje si daca procedurile ruleaza in alte sisteme (la distanta)
- Problema
  - un sistem de calcul are incorporate mecansimele necesare apelului local, dar nu si pentru apeluri la distanta

Mecanisme apel de procedura conventional

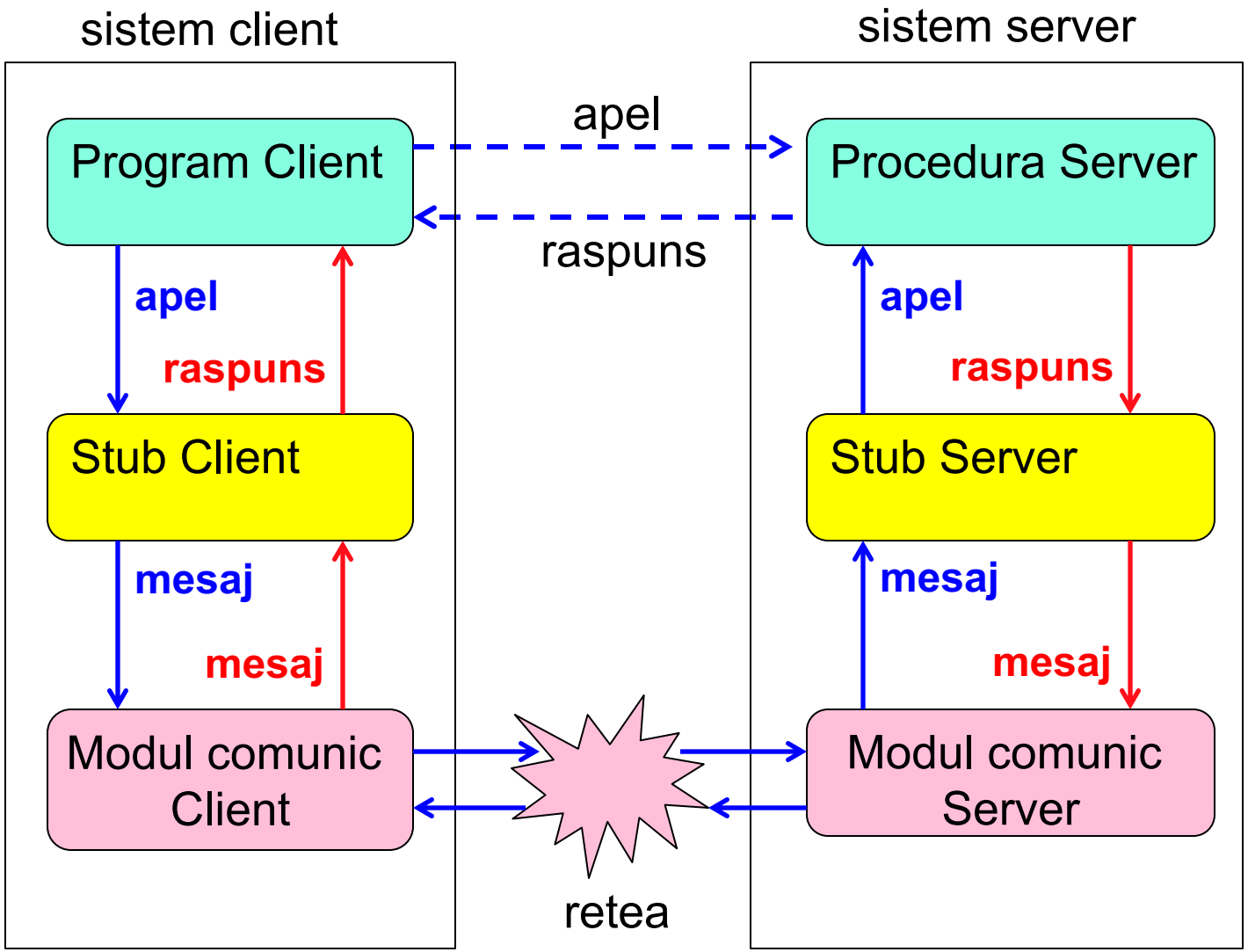
- count = read (fd, buf, bytes)
- la apel, programul principal pune pe stiva, in ordine, parametrii si adresa de intoarcere (fig. b)
- **read** transfera date in zona **buf**, intoarce rezultatul count, elimina adresa de intoarcere din stiva si da controlul programului
- programul elimina din stiva parametrii si continua executia
- scheme de pasare parametri
  - apel prin valoare (fd, bytes)
  - apel prin referinta (buf)
- alta schema posibila: "copy / restore"



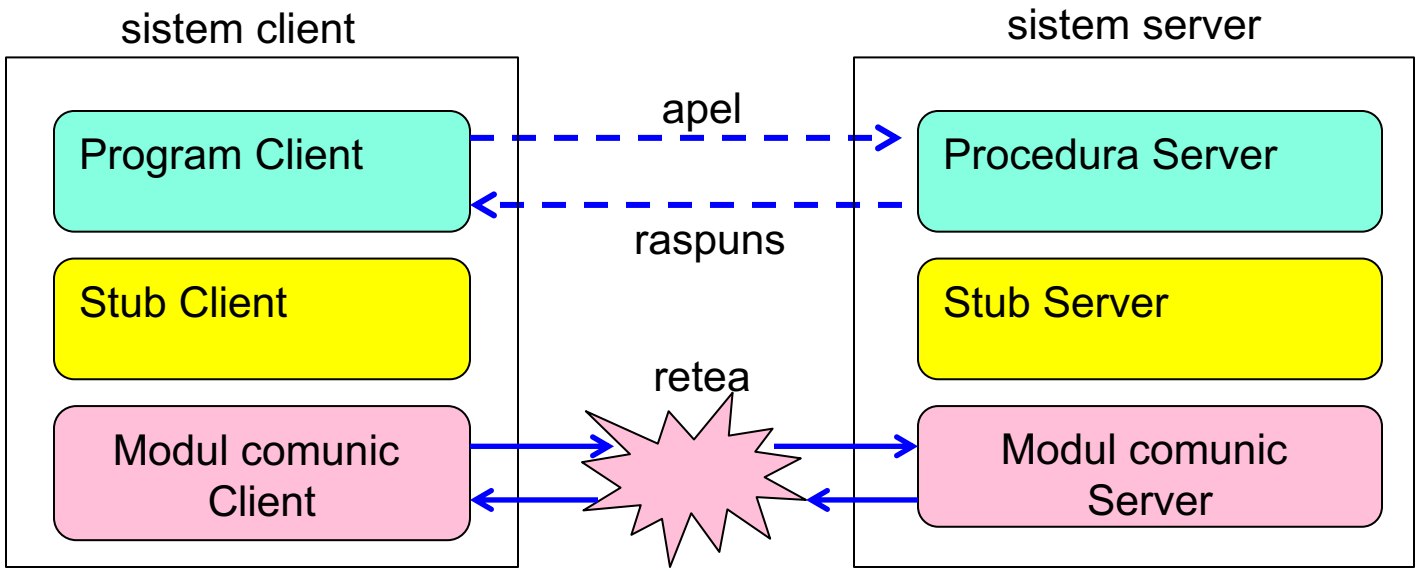
Apelul de proceduri la distanta (RPC)

- Apelantul si procedura ruleaza in sisteme diferite
- Comunicarea se face prin retea
- Vizeaza transparenta
  - aceeasi forma de apel pentru proceduri locale si la distanta
- RPC ofera o solutie completa pentru
  - localizarea procedurii
  - trimiterea apelului la distanta, executia procedurii si intoarcerea raspunsului
  - conversia reprezentarii datelor (daca reprezentarile difera de la un sistem la altul)

Arhitectura apelurilor la distanta



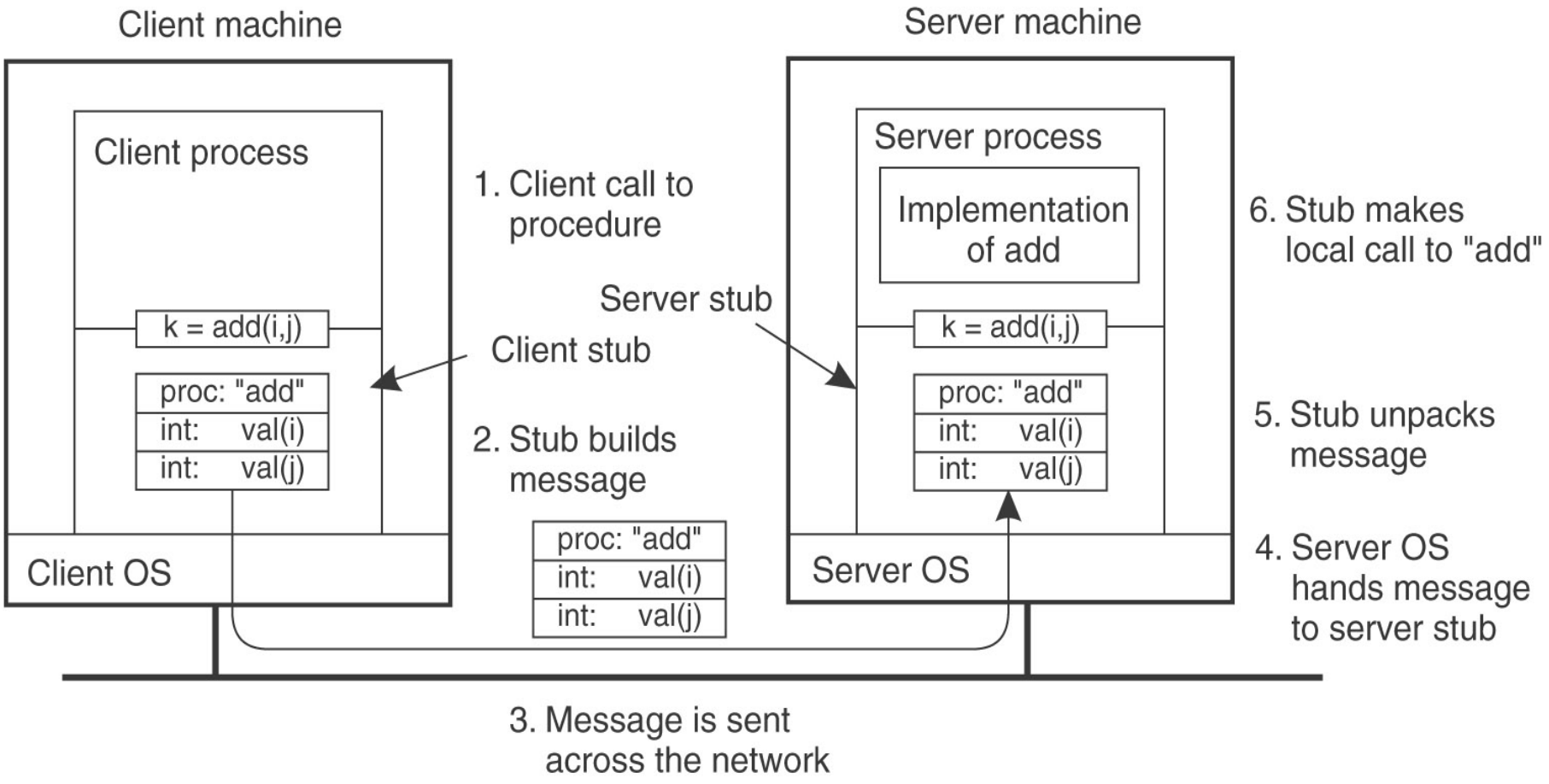
Arhitectura apelurilor la distanta



- Stub client este un proxy al procedurii server
  - accepta aceleasi apeluri dar le transforma in mesaje
- Stub server este un proxy al programului client
  - accepta aceleasi rezultate dar le transforma in mesaje
  - pentru ex. “count = read (fd, buf, bytes)” serverul umple un buf intern stub-ului

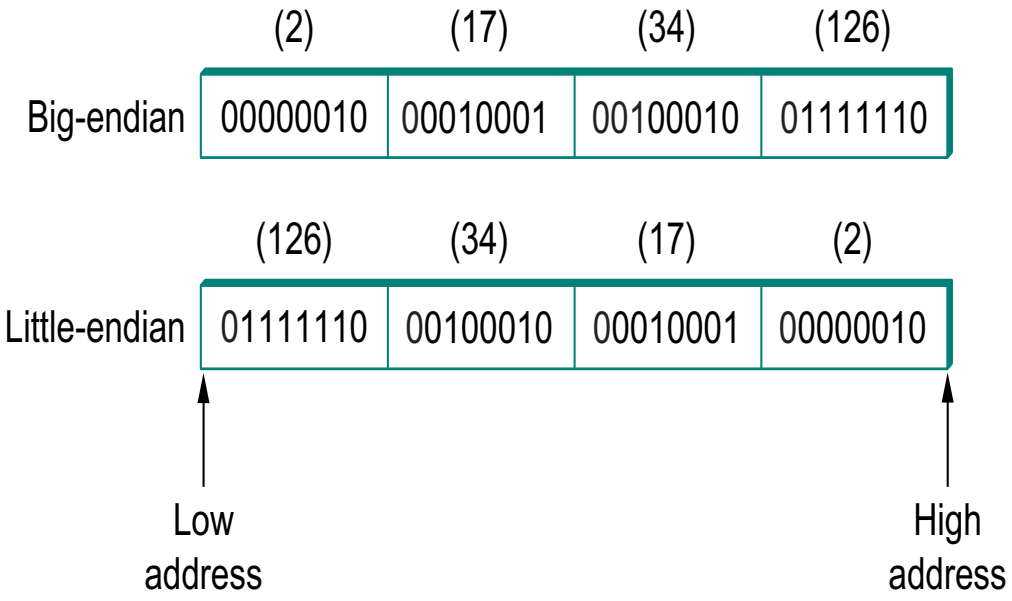
Formatul datelor pentru sisteme omogene

- reprezentarile parametrilor si ordinea sunt identice la client si server
- stub client transmite si numele procedurii (identifica una din procedurile rulate de server)



Probleme: Reprezentarea datelor

- big-endian (cel mai semnificativ bit este in octetul cu adresa mai mica) – adoptat in procesoarele MIPS® and PowerPC™
- little-endian (cel mai semnificativ bit este in octetul cu adresa mai mare) – adoptat in procesoarele Intel® x86
- o valoare intreaga necesita o conversie cand trece de la un tip de masina la altul



In figura: reprezentarea pe octeti a intregului **34 677**

374_{10} = $0211227E_{16}$ in formele big-endian si little-endian

Transferul parametrilor

Marshalling – aranjarea parametrilor pentru transfer

Mecanisme diferite pentru pasarea parametrilor

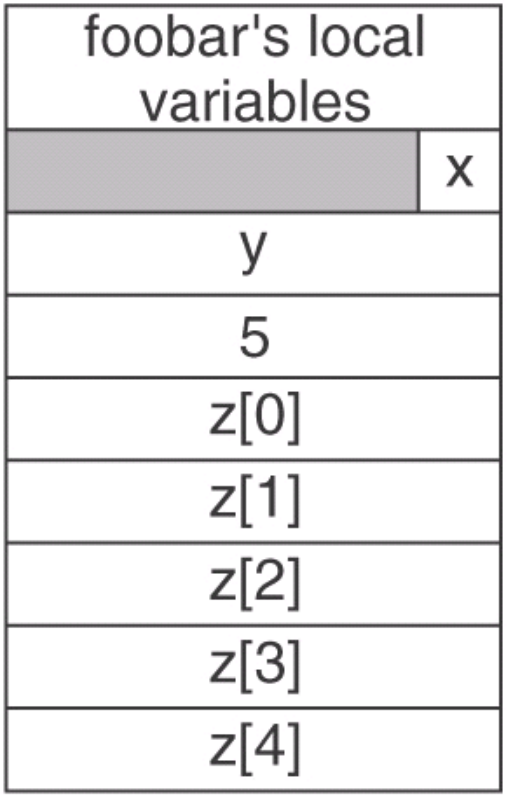
- Parametri valoare
  - adoptarea unor reprezentari standard pentru protocoalele RPC
    - ex. la Sun RPC reprezentarea XDR (eXternal Data Representation)
  - conversia datelor intre reprezentari interne din client / server si reprezentarea standard

Transferul parametrilor (2)

Parametri referinta – solutii diferite

- pentru structuri simple (arrays) – se transfera elementele tabloului
- copy / restore merge in anumite cazuri
- pentru eficienta (ex. transmitere intr-un singur sens) se specifica daca parametrii sunt in, out, in / out

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



Transferul parametrilor (3)

Pointerii din structuri complexe nu pot fi gestionati usor

Incercari

- pentru structuri complexe (grafuri) care folosesc pointeri se face linearizarea structurii, transmiterea si apoi refacerea ei la celalalt capat
- transmiterea pointerilor si cod special pentru acces la datele din client
  - ex. cereri de la server catre client pentru a afla datele referite de pointeri

Solutia apelului la distanta

- specificarea unui **protocol** pentru comunicarea intre programul client si procedura server, care
  - descrie semnatura procedurii la distanta
  - descrie tipurile parametrilor apelurilor si ale rezultatelor invocarilor
    - folosite de un compilator care genereaza stub-urile client si server
  - faciliteaza conectarea intre client si server

Protocolul RPC reglementeaza

- formatul datelor si functiilor
  - reprezentatarea tipurilor si structurilor de date
    - folosesc tipuri specifice (XDR -eXternal Data Representation la Sun RPC)
    - sau standardizate (ASN.1—Abstract Syntax Notation)
  - descrierea (semnaturilor) procedurilor
    - Interface Definition Language, IDL
- regulile schimbului de mesaje (de ex. protocolul de transport folosit)

SUN RPC

Numit si ONC RPC (ONC – Open Network Computing)

Utilizat in scrierea NFS (Network File System)

Poate fi folosit peste TCP sau UDP la alegere

Are un limbaj de descriere a interfetei,

XDR (eXternal Data Representation)

Ofera un compilator asociat, rpcgen

XDR – eXternal Data Representation standard

Specificat in RFC 1014 - permite definirea

Procedurilor utilizabile de la distanta

Tipurilor asociate:

Constante, typedefs, structuri, uniuni, enumerari

Caracteristici

Mecanisme simple de identificare:

Interfata identificata prin **nr program** si **nr versiune**

O definitie de procedura include **semnatura** si **nr procedura**

Semnatura procedurii include

tip_rezultat nume_procedura tip_parametru_intrare

Initial - un singur parametru de intrare si un singur rezultat (restrictie eliminata)

Exemplu - Interfata RPC pentru rezervare loc avion

```
struct reserve_args {
  int flight_number;
  string passenger_name<>;
};

struct reserve_result {
  int ok;
  string seat<>;
};

program RESERVATION_PROG {
  version RESERVATION_VERS {
    reserve_result RESERVATION_RESERVE   (reserve_args) = 1;
  } = 1;
} = 400001;
```

Protocolul - Formatul mesajelor

Call	**Reply**
Xid	Xid
Call / reply	Call / reply
Rpc version	Accepted? (vs bad rpc version or auth failure)
Program #	Auth stuff
Program version	Success? (vs bad prog / proc #)
Procedure #	results
Auth stuff	
Arguments	

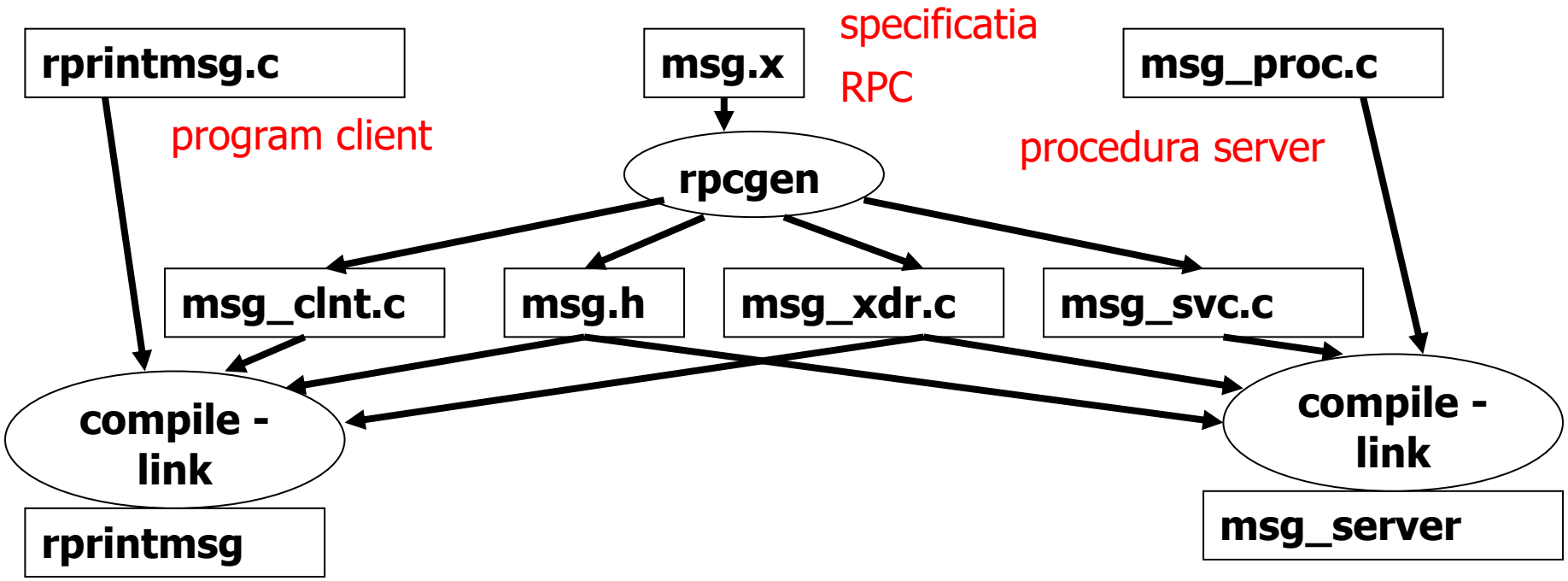
Reprezentarea datelor pentru XDR

int	complement fata de 2, 32 biti, little endian
unsigned int	32 biti little endian
float	32 biti: semn, exponent pe 8 biti, mantisa pe 23 biti
double	64 biti: semn, exponent pe 11 biti, mantisa pe 52 biti
string	lungime sir pe 4 octeti, n octeti sir plus r octeti pentru umplutura la multiplu de 4 octeti
tablou	lungime fixa: n elemente de la 0 la n-1
	lungime variabila: lungime pe 4 octeti urmata de n elemente de la 0 la n-1
struct	componentele in ordinea declararii
union	discriminant pe 4 octeti urmat de valoarea de pe varianta selectata
void	0 octeti

Compilatorul rpcgen

Pe baza specificatiei RPC (fisierul msg.x), rpcgen genereaza :

- procedurile din **stub client** (msg_clnt.c)
- procedurile din server: **main, dispecer, stub** (msg_svc.c)
- procedurile de **marshalling** si **unmarshalling** XDR (msg_xdr.c)
- fisier antet corespunzator definitiilor de proceduri si tipuri din msg.x (msg.h)



Transparenta? Modificari fata de apel local

Programele folosesc direct socketi si **functii** specifice care faciliteaza conectarea intre client si server

Clientul

- initializeaza interfata RPC cu ***clnt_create***
- care creeaza un **RPC handle** la programul si versiunea specificate, pe o gazda data;
  - parametrii includ:
    - nume server
    - nume si versiune program (generate de rpcgen in msg.h)
    - protocolul de transport utilizat
- apelata inaintea oricarei proceduri la distanta

Transparenta? Modificari fata de apel local

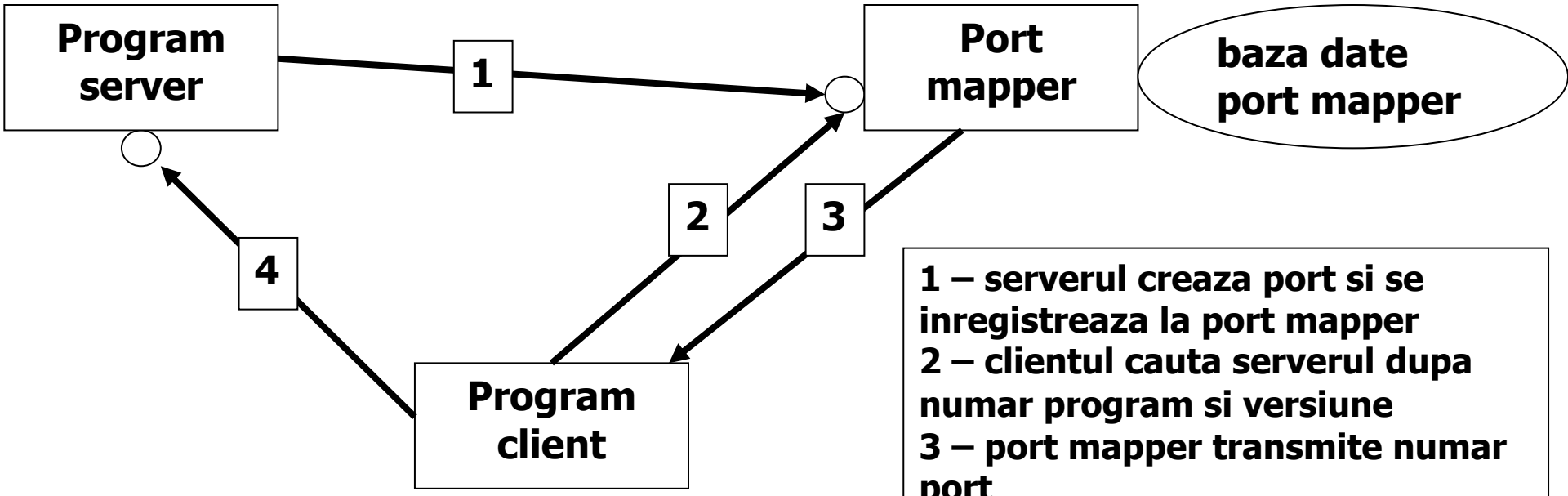
Serverul

- executa stub-ul si pune procesul in background
- creaza un socket si leaga un port local la socket
- apeleaza ***svc_register*** pentru a inregistra portul, numarul si versiunea programului (se contacteaza ***port mapper***)
- executa ***listen*** (asteapta cererile clientilor)

Binding

Foloseste **port mapper** aflat la un **port cunoscut** pe fiecare masina care ruleaza RPC.

La el se inregistreaza toate **serverele** de pe masina locala prin: **nr program, nr versiune, nr port**



DCE RPC

- Distributed Computing Environment / Remote Procedure Call
- Dezvoltat de Open Software Foundation (acum The Open Group)
- Proiectat pentru Unix si extins la alte OS (VMS, Windows)
- Defineste **servicii** client-server (incluse in DCE RPC)
  - Distributed file service
  - Directory service
  - Security service
  - Distributed time service

Obiective DCE RPC

- Transport mesaje intre client si server
- Conversie tip date
- Transparenta locatiei
- Transparenta limbajului (ex. client Java, server C)
- Transparenta SO
- Transparenta hardware
- Transparenta protocoalelor de retea

Legare client - server in DCE

Legarea la server

Sun – programatorul trebuie sa cunoasca masina serverului

DCE – serverul inregistreaza endpoint (port) cu daemon local DCE si corespondenta {nume_server, masina} cu un server de directoare