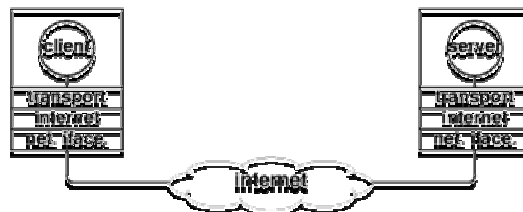


Nivelul Aplicație

Modelul Client-Server

Modelul Client - Server

- Aplicațiile de rețea folosesc paradigma client-server
- Clienții și server-ele sunt **programe**
 - serverul așteaptă *pasiv* să fie contactat
 - clientul *activ* inițiază comunicația
- Un server poate fi client pentru alt server
- Clienții și server-ele folosesc serviciile transport pentru a stabili comunicația și pentru a schimba date.





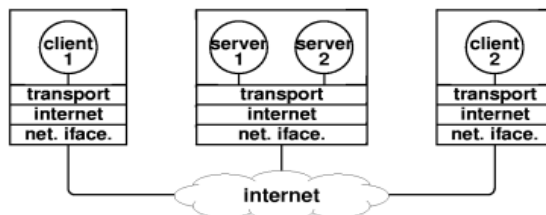
Caracteristici generale

- **Clientul**
 - Un program obișnuit, invocat direct de către utilizator
 - Rulează local (pe PC-ul utilizatorului)
 - Inițiază activ contactarea serverului
 - Poate accesa mai multe servicii dacă este necesar
 - Nu necesită hardware sau software special
- **Serverul**
 - Un program privilegiat, cu destinație specială
 - Rulează pe un calculator partajat
 - Așteaptă pasiv să fie contactat de clienți
 - Necesită hardware și SO performant
 - Trebuie să conțină cod pentru *autentificare, autorizare, securitatea, confidențialitatea și protecția datelor*
 - Servicii standard: remote login, e-mail, ftp, http
 - Mai multe servere pot rula pe același calculator; serviciile oferite sunt identificate prin porturile folosite (23 – telnet, 80 – web etc.)



Acces multiplu

- Două servere pe același calculator, accesate de clienți aflați pe alte două calculatoare.
- Exemplu: server-ul poate rula servicii de FTP, WWW
- Pe sistemele Linux: `/etc/services` conține lista serviciilor și portul folosit.





Exemplu

- Programatorii de aplicații client/server trebuie să aleagă între servicii orientate sau neorientate pe conexiune
 - Varianta fără conexiune poate fi utilizată la orice moment; nu garantează recepția datelor
 - Varianta cu conexiune necesită stabilirea explicită a unei conexiuni; garantează recepția datelor
- Serverul poate fi *iterativ* sau *concurrent*
- Exemplul folosește varianta cu conexiune, server iterativ
 - Serverul apelează entitățile software locale de protocol pentru a accepta cereri de conectare
 - Clientul stabilește o conexiune cu serverul folosind entitățile software locale de protocol
- Clientul și serverul schimbă date după ce s-a stabilit conexiunea

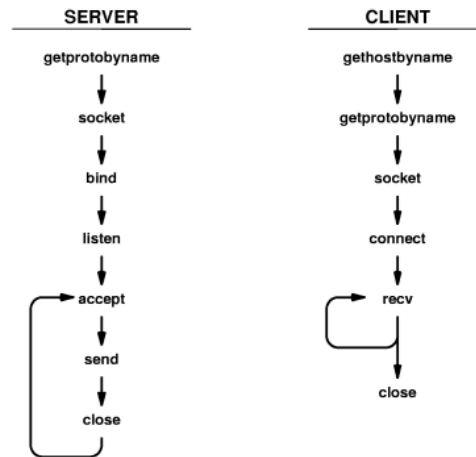


Un exemplu de serviciu

- Serverul
 - Memorează de câte ori a fost contactat
 - Returnează acest număr ca sir ASCII
- Clientul
 - Stabilește o conexiune
 - Așteaptă un mesaj de la server (sir ASCII)
 - Afișează mesajul
- Protocol la nivel aplicație
 - Sintaxa:
 - Fără header-e
 - Clientul trimite un mesaj cu corp vid
 - Serverul răspunde cu un mesaj sir ASCII
 - Semantica - Clientul stabilește conexiunea, serverul returnează un sir ASCII



Operațiunile efectuate de Server și Client



Server-ul

- Inițializare:
 - `getprotobyname` – află care e numărul de protocol pentru TCP
 - `socket` – creează un socket
 - `listen` – asociază socket-ul cu cereri
- Bucla:
 - `accept` – accepta cereri de conexiune
 - `send` – trimite un mesaj către client
 - `close` – închide socket-ul



Clientul

- Initializare:
 - `gethostbyname` – căuta serverul
 - `getprotobyname` – afla care e numărul de protocol pentru TCP
 - `socket` – creează un socket
 - `connect` – se conectează la portul serverului
- Bucla:
 - `recv` – recepționează un mesaj de la server
- Terminare:
 - `close` - închide socket-ul



Clientul apeleaza “recv”

- Apelul este în bucla în client
- Serverul trimite un singur mesaj!!!
 - De ce se folosesc apeluri multiple?
 - Entitățile de protocol *nu* garantează livrarea datelor la receptor în aceleași blocuri care au fost generate de server.
 - Data pot calatori în segmente diferite
 - Apelul `recv` poate returna mai puține date decât au fost cerute
 - Clientul trebuie sa apeleze repetat `recv` pana se returnează valoarea 0



Socket și blocare

- Majoritatea apelurilor socket sunt *blocante*
- Procesul apelant este blocat (scos din coada de procese *ready*) pana când operația pe socket se termină.
- Serverul se blochează:
 - In apelul *accept*, pana se stabilește o nouă conexiune.
 - In apelul *send*, până ce datele se livrează entității de protocol.
- Clientul se blochează:
 - In apelul *gethostbyname*, până ce DNS rezolvă numele serverului.
 - In apelul *recv*, până ce mesajul de la server este livrat.