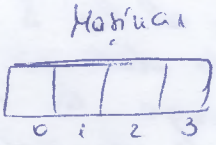
Rețea

Facilitățile p/u lucru în rețea

Norma OSI

- Normele sunt organizate pe nivele ierarhice. Între un nivel superior pot apela orice de la un nivel inferior (reciproca nu este adev.)
  Dacă și reciproca este adev. => cod "spaghetti"

- Norma OSI
  - organizată pe 7 nivele (n. superior → n. inferior):
    (7) Aplicație
    (6) Prezentare → nu este obligatoriu ca din 7 să trecem în 6 dar este recomandat. (dacă endian este diferit)

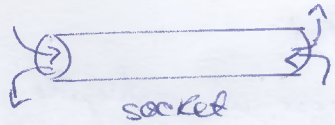


← endian: big / little

→ transformă lb. formală de aplicație într-un lb. universal.
→ protocoalele care transf. lb apl. <-> lb univ (ex. XDR)

(5) Sesiune (site-uri: - am închis browser-ul -> l-am deschis => altă sesiune) - se poate evita

(4) Transport - obligatoriu folosit
- ultimul nivel în care tb să avem acces



- pe lângă pipe dar bidirecțională
- în general, sunt de două feluri:
  pipe ← TCP, UDP ← cozi de mesaje

Norma TCP/IP - alcătuită din 4 + 3 + 1 nivele

③ Rețea - "calculează" ruta pe care îl. să o ia mesajul, următoarea destinație

- protocol: IP (internet protocol)

De la 4 → 1 este invizibil: nu folosim 3, 2, 1 în aplicații
- intervine router-ul

② Controlul de date - încarcă mesajul (obț. după trecerea prin celelalte nivele)
← te încurcă mesajul în check sum înainte să treacă prin celelalte nivele
Prob: $\frac{1}{256}$
→ pt a te ști dacă msg a fost transmis corect

① Fizic (hard) - cu fir sau fără fir
- verifică mesajul (gen check sum)
→ corect: trimite mai departe
→ incorect: îl retrimite la receptor
→ dc este corect: trimite la rețea, pleacă în nivelul fizic al lui B, comunică cu controlul de date → nu e dest. final, ... ajunge la B urcă până la nivel de aplicație și trece prin toate nivelele.

- Tehnologia RPC (Remote Procedure Call)
- o aplicație A vrea să execute o met. pe o altă mașină

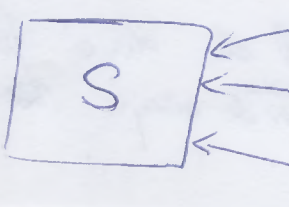
~~Aplicații distribuite~~

Arhitectura client-server

Mașină server - mașină performantă d.p.d.v. hardware
Aplicația server ≠ mașina server

aplicație client

mașina pe care se execută aplicația server



Server-ul așteaptă cereri de la client și îl servește
Prelucrările se fac în server

java.net → pachet

Server:
~~ServerSocket~~ ServerSocket ss = new ServerSocket(port);
Socket s = ss.accept();

Client:
Socket s = new Socket(host, port)
adresa unde lucrează apl. server

- Aplicații client-server:
  - → DNS (Domain Name Server)
  - → SMTP (poștă electronică)
  - → Telnet (SSH)
    - ↳ trimite parola în clar
    - ↳ trimite parola criptat (te folosește la următorul actual)
  - → FTP (SFTP) – file transfer protocol
    - ↳ secure
    - → transfer de fișiere de pe o mașină pe alta
    - două metode de transfer: binar sau ASCII
  - → Baze de date cu arhitectură client-server.
    - Access, FoxPro, dBase - nu sunt client-server.
    - ORACLE, ~~INFORMIX~~ SYBASE (MSSQL), DB2 (IBM), POSTGRES, INGRES, INFORMIX etc.
      Microsoft
      - toate datele sunt pe o mașină-server
      - bazele de date sunt interogate cu ajutorul SQL
  - → www - aplicații de tip web

"\r\n" - end of line