



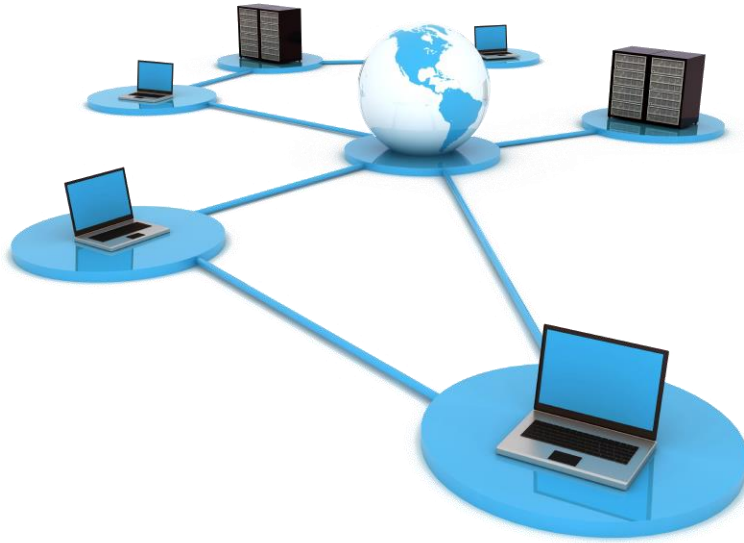
PROGRAMARE JAVA IN RETEA

POSA BOGDDAN-PETRE
27.04.2017 - LINK Academy

CUPRINS

- Cum comunica calculatoarele in retea
 - Protocole de comunicatie
 - Protocolul Internet
 - Socket
- Cum comunicam in retea folosind Java
- Thread-uri si sisteme distribuite

RETEA



2 sau mai multe calculatoare a.i. un calculator poate accesa date, programele și facilitățile sau resursele unui alt calculator conectat la aceeași rețea

Protocoale de retea



-> un set strict de reguli ce guverneaza schimbul de date intre calculatoare.



Protocoale de retea

Hypertext Transfer Protocol (HTTP)

File Transfer Protocol (FTP)

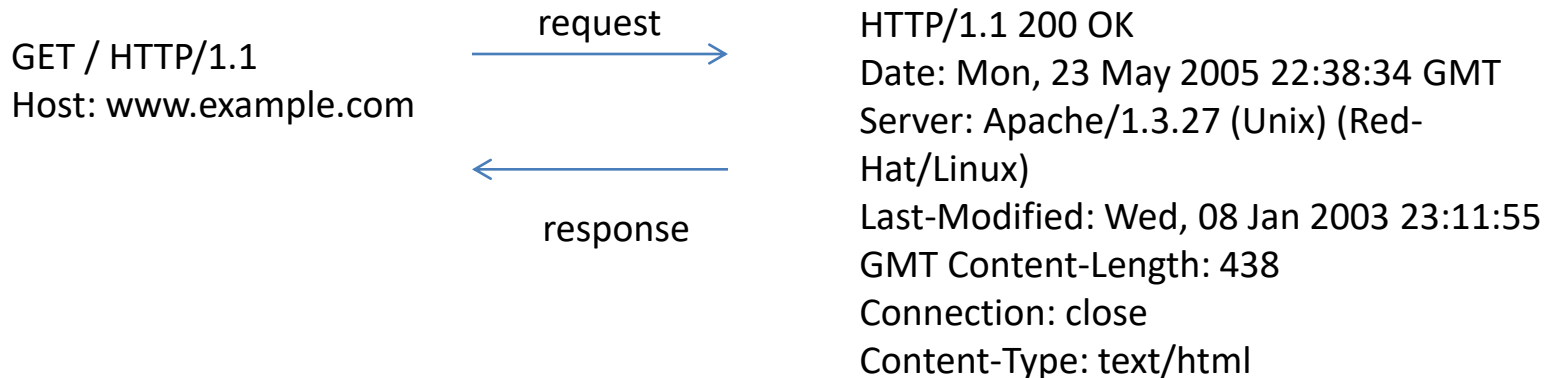
Internet Message Access Protocol (IMAP)

Transmission Control Protocol (TCP)



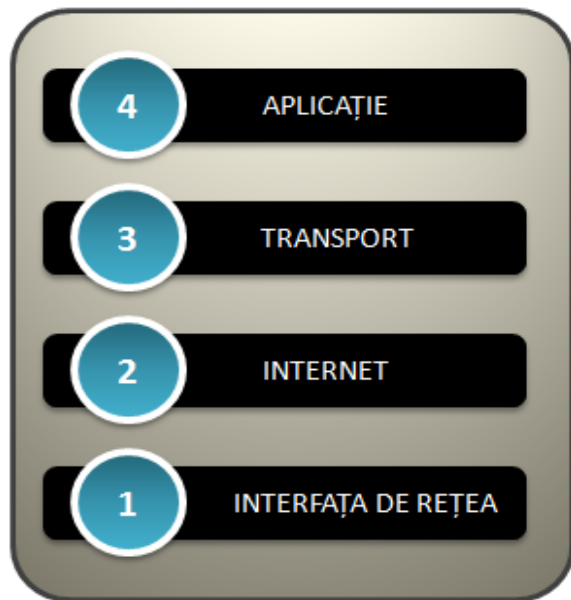
Protocoale de retea

HTTP protocol

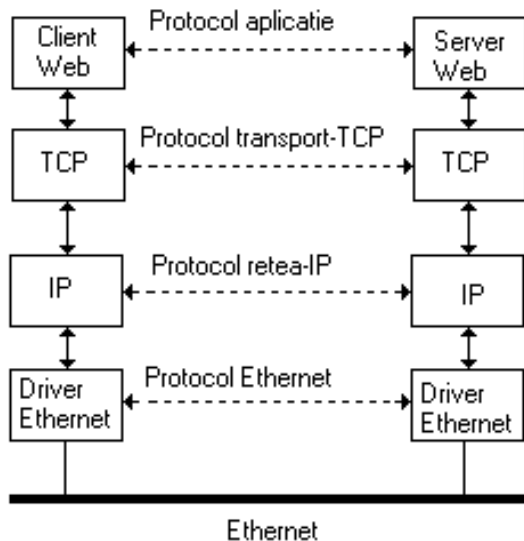




TCP/IP



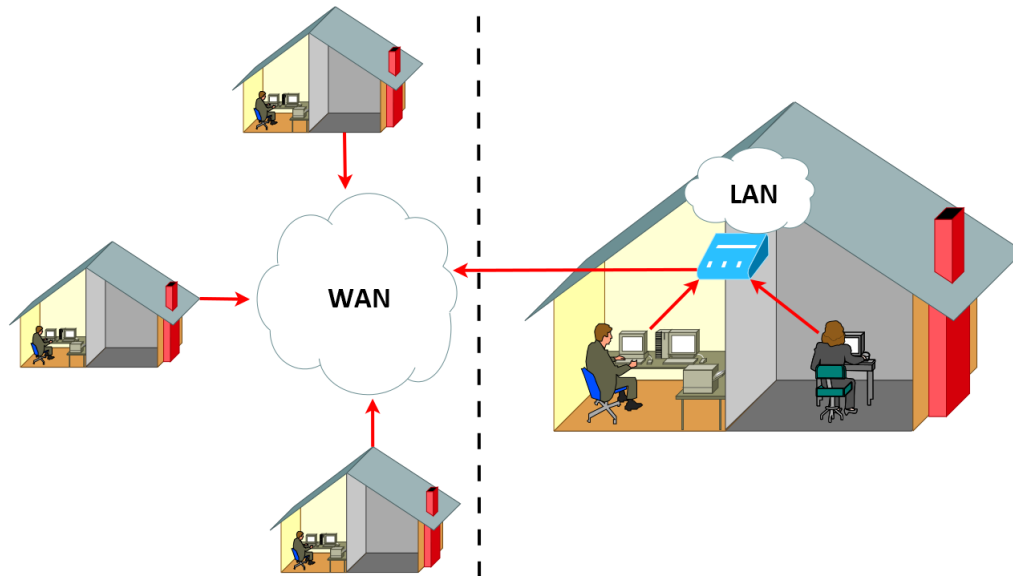
- creat de US Department of Defence
- flexibilitate
- layere de protocoale



Comunicarea client-server intre doua calculatoare aflate pe aceasi retea



Notiuni networking



Notiuni networking

IP: identificarea unui PC in retea; 32 de biți (de ex. 192.168.0.1)

Port: endpoint al unei comunicatii

IP intern: in cadrul retelei

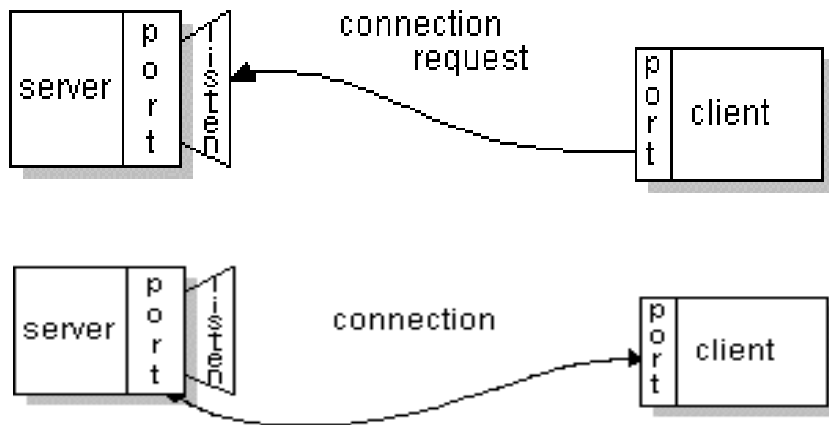
IP extern: in WAN; asignat de ISP



Demo 😊



Client – Server





Client – Server

- comunicare între 2 calculatoare dintre care unul joacă rol de server, celălalt de client.
- clientul utilizează serviciile oferite de către un server
- clientul identifică un server prin: adresă server, port de comunicare deschis de către server, protocolul folosit



Socket

- legatura de comunicare bi-directionala intre 2 calculatoare (important) aflate in aceeași rețea clientul utilizează serviciile oferite de către un server



Socket Java

Class Socket

Constructors:

[Socket\(InetAddress address, int port\)](#)

Creates a stream socket and connects it to the specified port number at the specified IP address.

[Socket\(InetAddress address, int port, InetAddress localAddr, int localPort\)](#)

Creates a socket and connects it to the specified remote address on the specified remote port.

Methods:

[getInputStream\(\)](#) Returns an input stream for this socket.



Socket Java

Class ServerSocket

Constructors:

[ServerSocket](#)(int port) Creates a server socket, bound to the specified port.

Methods:

[accept](#)() Listens for a connection to be made to this socket and accepts it.



Socket Java

Class ServerSocket

```
ServerSocket ss = new ServerSocket(5678);  
Socket socket = ss.accept();  
InputStream in = socket.getInputStream();
```



Socket Java





2016 What happens in an INTERNET MINUTE?



Socket Java

Probleme:

- Serverul accepta un singur client
- Serverul se opreste dependent de client



Socket Java

Cerinte server:

- Clienti multipli / contexte diferite
- Management de clienti
- Erorile clientului sa nu afecteze server-ul



Threads

- > Fir de executie
- > Cea mai mica unitate ce poate fi programata spre executie de catre SO
- > Faciliteaza existenta activitatilor multiple in cadrul unui proces

Server:

- Task-uri in background
- Handle pentru clienti multipli

Video game:

- Thread pentru grafica
- Thread pentru interfata UI
- Thread retea



Threads in JAVA

Extindere clasa **java.lang.Thread**

- Method `run()` trebuie suprascrisa – similara metodei `main` al unui program secvential
- `run()` este chemata cand incepe executia thread-ului
- Thread-ul se termina odata cu incheierea executie metodei `run()`



Threads

Cerinte server:

- Clienti multipli / contexte diferite
- Management de clienti
- Erorile clientului sa nu afecteze server-ul



Threads





Client – Server

with a smart server

Server:

1. Serverul alocă un port de comunicare care va fi cunoscut de către clienți
2. Cât timp (mai sunt clienți la port)
 - {
 - stabilire conexiune cu client (creare socket)
 - deservire client (flux de intrare, flux de iesire)
 - eliberare conexiune (eliberare socket)
 - }

Client:

1. Alocă un port de comunicare
2. Se conectează la server la portul cunoscut de dinainte
3. Se stabilește o conexiune prin care se trimite și se citește date (soclu + fluxuri)



Client – Server

with a smart server





Client – Server

with a smart server

Server:

1. Serverul alocă un port de comunicare care va fi cunoscut de către clienți
2. Cât timp (mai sunt clienți la port)
 - {
 - stabilire conexiune cu client (creare socket)
 - deservire client (flux de intrare, flux de iesire)
 - eliberare conexiune (eliberare socket)
 - }

Client:

1. Alocă un port de comunicare
2. Se conectează la server la portul cunoscut de dinainte
3. Se stabilește o conexiune prin care se trimite și se citește date (soclu + fluxuri)



Client – Server

with a smart server

Server:

1. Serverul alocă un port de comunicare care va fi cunoscut de către clienți
2. Cât timp (mai sunt clienți la port)
 - {
 - stabilire conexiune cu client (creare socket)
 - deservire client (flux de intrare, flux de iesire)
 - eliberare conexiune (eliberare socket)
 - }

Client:

1. Alocă un port de comunicare
2. Se conectează la server la portul cunoscut de dinainte
3. Se stabilește o conexiune prin care se trimite și se citește date (soclu + fluxuri)

Sisteme distribuite

Aplicatie/software care ruleaza pe mai multe calculatoare in retea, in acelasi timp, toata lucrând la acelasi task/job.

Exemple (https://en.wikipedia.org/wiki/List_of_distributed_computing_projects):

PrimeGrid – probleme numere prime

DNA@home – descompunere/pattern-uri in ADN



Sisteme distribuite

SA VEDEM IN PRATICA !!!

(nu am mai gasit alte poze amuzante 😊)

Sisteme distribuite

Testam daca 15 este prim

Folosim 3 clienti:

1. Clientul 1 va cauta divizori in intervalul 0,5
2. Clientul 2 va cauta divizori in intervalul 5,10
3. Clientul 3 va cauta divizori in intervalul 10,14