Retele de calculatoare – Informatica anul 3 (2019-2020)

Note de Laborator	**Specializare: Informatica anul 3**
Retele de calculatoare	**Contact:**
	retelecdsd@gmail.com
	http://www.cdsd.ro

Comunicatii de Date si Sisteme Distribuite
http://www.cdsd.ro

Laborator 9

1. Obiective:

- **Protocoalele TCP** si **UDP** – Aplicatie Riverbed Modeler – mediu de simulare a retelelor de calculatoare (Varianta "programare" C++: OMNeT++ Network Simulation Framework http://www.omnetpp.org/)
- **Socket-uri TCP**: Aplicatii Java; Aplicatii Python
- **Socket-uri UDP**: Aplicatii Java; Aplicatii Python

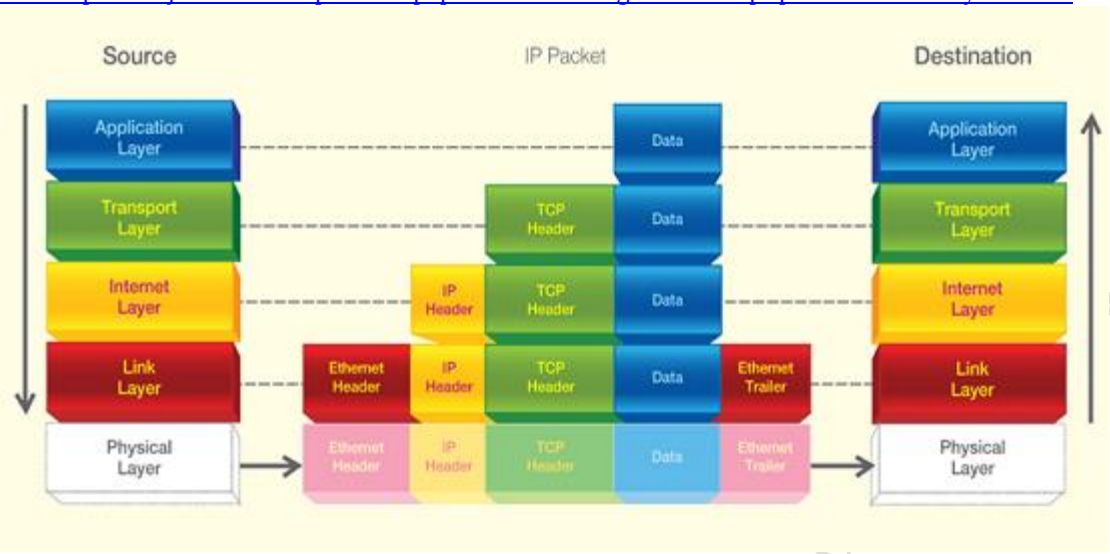
2. Consideratii teoretice (Partea practica- pag.14; Tema pag. 23)

2.0 OSI - Open System Interconnection

Layer	Application/Example		Central Device/ Protocols			DOD4 Model
Application (7) Serves as the window for users and application processes to access the network services.	**End User layer** Program that opens what was sent or creates what is to be sent Resource sharing • Remote file access • Remote printer access • Directory services • Network management		**User Applications** SMTP		**GATEWAY** Can be used on all layers	Process
Presentation (6) Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network.	**Syntax layer** encrypt & decrypt (if needed) Character code translation • Data conversion • Data compression • Data encryption • **Character Set Translation**		JPEG/ASCII EBDIC/TIFF/GIF PICT			
Session (5) Allows session establishment between processes running on different stations.	**Synch & send to ports** (logical ports) Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc.		**Logical Ports** RPC/SQL/NFS NetBIOS names			
Transport (4) Ensures that messages are delivered error-free, in sequence, and with no losses or duplications.	**TCP** Host to Host, Flow Control Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing	PACKET FILTERING	TCP/SPX/UDP			Host to Host
Network (3) Controls the operations of the subnet, deciding which physical path the data takes.	**Packets** ("letter", contains IP address) Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting		**Routers** IP/IPX/ICMP			Internet
Data Link (2) Provides error-free transfer of data frames from one node to another over the Physical layer.	**Frames** ("envelopes", contains MAC address) [NIC card — Switch — NIC card] (end to end) Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgment • Frame delimiting • Frame error checking • Media access control		**Switch** **Bridge** **WAP** PPP/SLIP	Land Based Layers		Network
Physical (1) Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium.	**Physical structure** Cables, hubs, etc. Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts		**Hub**			

The Department of Defense *Four Layer Model* is used to discuss the architecture of TCP/IP. The four layers (from top to bottom) are the *Application Layer*, the *Transport Layer*, the *Internet Layer* and the *Link Layer*. The DoD Four Layer model was used during the creation of TCP/IP, but was not formalized until well afterwards (in RFC 1122, "Requirements for Internet Hosts -- Communications Layers", October 1989). This model is not suitable for discussing all protocol suites (for example, OSI doesn't really fit into it), but is ideal for discussing

TCP/IP, which is the dominant protocol suite in use today. Source:
http://www.sixscape.com/joomla/sixscape/index.php/technical-backgrounders/tcp-ip/the-dod-four-layer-model



Obs: This list shows some protocols that are commonly placed in the transport layers of TCP/IP, OSI, NetWare's IPX/SPX, AppleTalk, and Fibre Channel.

- ATP, AppleTalk Transaction Protocol
- CUDP, Cyclic UDP
- DCCP, Datagram Congestion Control Protocol
- FCP, Fibre Channel Protocol
- IL, IL Protocol
- MPTCP, Multipath TCP
- RDP, Reliable Datagram Protocol
- RUDP, Reliable User Datagram Protocol
- SCTP, Stream Control Transmission Protocol
- SPX, Sequenced Packet Exchange
- SST, Structured Stream Transport
- TCP, Transmission Control Protocol
- UDP, User Datagram Protocol
- UDP Lite
- µTP, Micro Transport Protocol

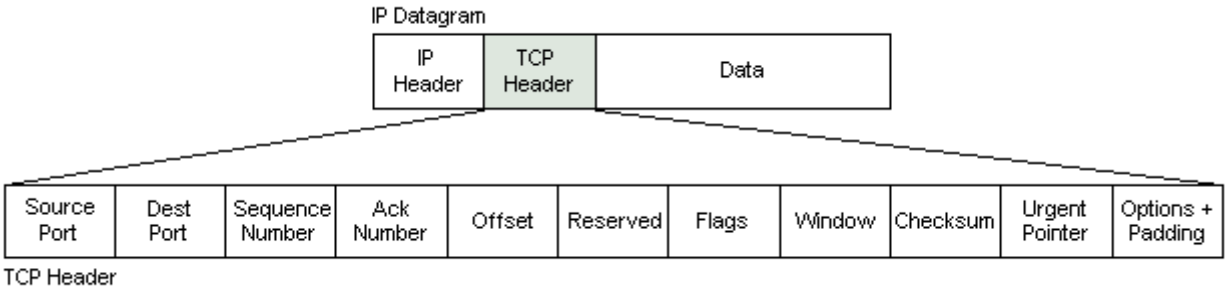
2.1. Suita TCP/IP: protocoale de transport

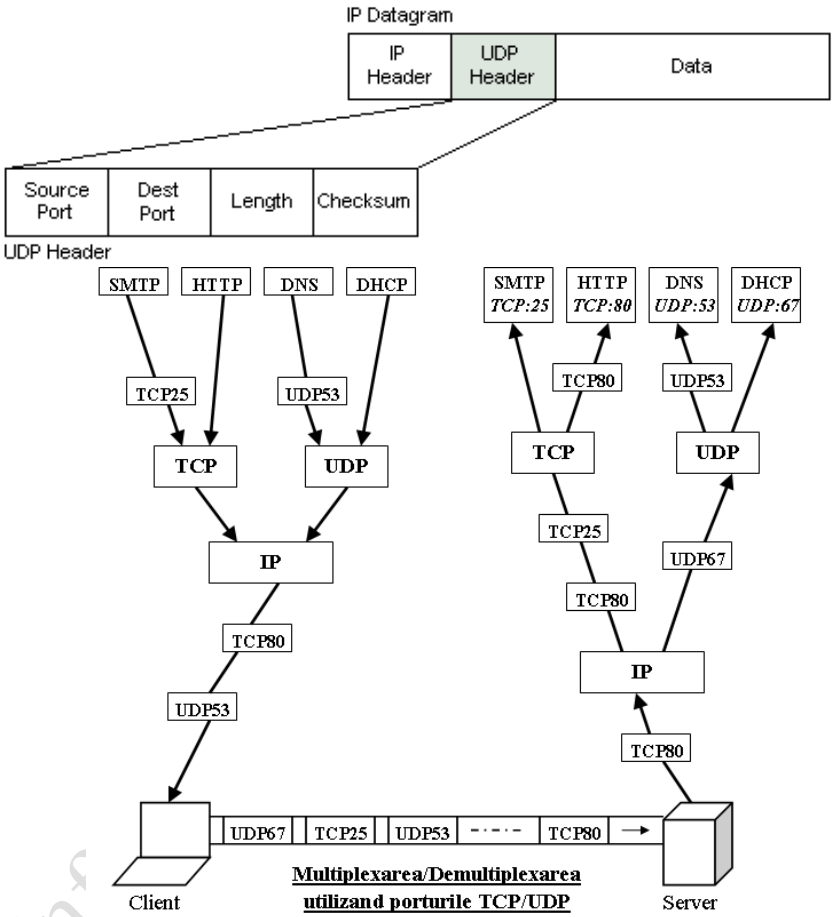
TCP

- Creaza un circuit virtual intre aplicatiile end-user
  - Orientat conexiune
  - Sigur (reliable)
  - Imparte mesajele in segmente
  - Reasambleaza mesajele la destinatie
  - Retransmite tot ce nu fost primit
  - Reasambleaza segmentele primite din afara

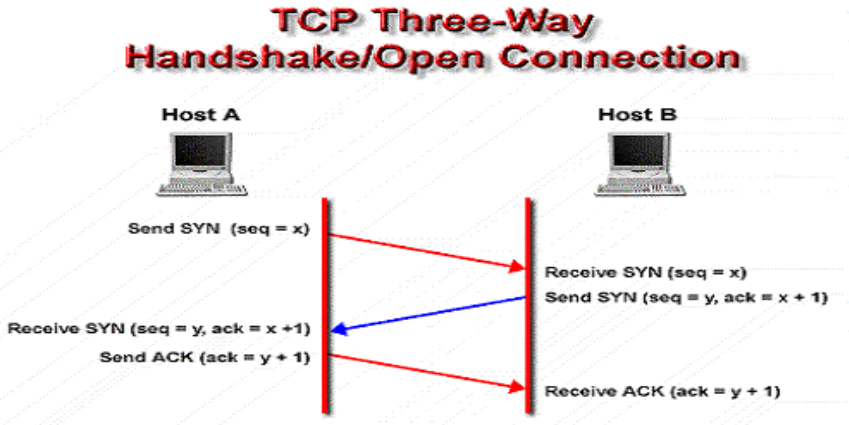
UDP

- Caracteristici:
  - Fara conexiune
  - Nesigur (unreliable)
  - Transmite mesaje numite datagrame
  - Nu verifica transmiterea mesajelor
  - Nu segmenteaza/reasambleaza mesajele
  - Nu foloseste confirmari
  - Nu face retransmiteri





TCP – Stabilirea conexiunii



Exemplu (capturi Wireshark):

```
⊞ Frame 1 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: Xerox_00:00:00 (00:00:01:00:00:00), Dst: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)
⊞ Internet Protocol, Src: 145.254.160.237 (145.254.160.237), Dst: 65.208.228.223 (65.208.228.223)
⊟ Transmission Control Protocol, Src Port: tip2 (3372), Dst Port: http (80), Seq: 0, Len: 0
    Source port: tip2 (3372)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 0    (relative sequence number)
    Header length: 28 bytes
  ⊟ Flags: 0x02 (SYN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgement: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
    ⊞ .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 8760
  ⊞ Checksum: 0xc30c [validation disabled]
  ⊞ Options: (8 bytes)
```

```
⊞ Frame 2 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: fe:ff:20:00:01:00 (fe:ff:20:00:01:00), Dst: Xerox_00:00:00 (00:00:01:00:00:00)
⊞ Internet Protocol, Src: 65.208.228.223 (65.208.228.223), Dst: 145.254.160.237 (145.254.160.237)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: tip2 (3372), Seq: 0, Ack: 1, Len: 0
    Source port: http (80)
    Destination port: tip2 (3372)
    [Stream index: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 28 bytes
  ⊟ Flags: 0x12 (SYN, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgement: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
    ⊞ .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 5840
  ⊞ Checksum: 0x5bdc [validation disabled]
  ⊞ Options: (8 bytes)
  ⊞ [SEQ/ACK analysis]
```

```
⊞ Frame 3 (54 bytes on wire, 54 bytes captured)
⊞ Ethernet II, Src: Xerox_00:00:00 (00:00:01:00:00:00), Dst: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)
⊞ Internet Protocol, Src: 145.254.160.237 (145.254.160.237), Dst: 65.208.228.223 (65.208.228.223)
⊟ Transmission Control Protocol, Src Port: tip2 (3372), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
    Source port: tip2 (3372)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x10 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgement: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 9660
  ⊞ Checksum: 0x7964 [validation disabled]
  ⊞ [SEQ/ACK analysis]
```

2.2. Socket-uri

- Un ***socket*** (soclu, canal de comunicatie) este un ***punct terminal al unei comunicatii punct–la–punct***, avand un **nume** si o **adresa.** Din perspectiva programatorului, un *socket* ***ascunde detaliile retelei*.**
- O **adresa socket** pe o retea TCP/IP consta din doua parti: **o adresa IP** *si* **o adresa (numar) de port**.

Un socket furnizeaza facilitati pentru crearea de fluxuri de intrare/iesire, care permit schimburile de date intre client si server. Atunci cand se stabileste o conexiune, atat clientul cat si serverul vor avea cate un socket, comunicarea efectiva realizandu-se intre socketuri.
Alocarea numărului de porturi este gestionată de IANA pentru a asigura compatibilitate universală pe întregul Internet. Există **trei domenii de numere de porturi:**

- *Well-known (Privileged) Port Numbers*

0-1023 → system port numbers → utilizate pentru cele mai universale aplicaţii TCP/IP (standardizate - RFC)

- *Registered (user) Port Numbers*

1024-49151 → user port numbers → utilizate pentru aplicaţii neprecizate prin RFC-uri. Pentru a asigura că nu există conflicte, IANA alocă numărul de porturi celor care au creat aplicaţii server viabile, de regulă accesibile oricărui utilizator.

- *Private/Dynamic Port numbers*

49152-65535 → nu sunt rezervate şi gestionate de IANA. Pot fi utilizate de oricine, fără înregistrare → protocoale private pentru organizaţii private.

Exista doua forme (nivel transport) ale comunicarii prin sockets:

- orientata spre conexiune
- prin datagrame (neorientat spre conexiune)

Obs: Exista si socket-uri brute (raw sockets – nivel retea http://en.wikipedia.org/wiki/Raw_socket ; http://www.linuxchix.org/content/courses/security/raw_sockets; http://mixter.void.ru/rawip.html; http://www.security-freak.net/raw-sockets/raw-sockets.html, http://msdn.microsoft.com/en-us/library/ms740548(v=vs.85).aspx, http://www.savarese.com/software/rocksaw/)

Modelul TCP/IP suporta ambele metode prin implementarea protocolului TCP (Transmission Control Protocol) si a protocolului UDP (User Datagram Protocol).

2.2.1. TCP (Transmission Control Protocol)

Protocolul TCP este orientat spre conexiune aflat pe nivelul transport -asigura servicii de comunicare sigure cu detectarea si corectarea erorilor intre doua gazde. Stabilirea unei conexiuni se bazeaza pe adresa IP a maşinii destinaţie si pe numărul portului pe care aceasta asteaptă cereri de conectare.

2.2.1.1. Clase si metode pentru programarea in Java cu sockets (TCP)

Pentru a putea realiza un program care utilizeaza socket-uri trebuie sa importam pachetele `java.io` http://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html si
`java.net` https://docs.oracle.com/javase/8/docs/api/java/net/package-summary.html

- **`java.net.Socket`** - Socket client TCP

- `public Socket(String host, int port) throws UnknownHostException,IOException` - constructor care deschide o conexiune TCP catre host-ul si portul specificati ca parametri.
- `public Socket(InetAddress host, int port) throws UnknownHostException,IOException` - constructor care deschide o conexiune TCP catre host-ul si portul specificati ca parametri.
- `public OutputStream getOutputStream()` – intoarce fluxul de iesire pentru socket.
- `public InputStream getInputStream()` – intoarce fluxul de intrare pentru socket.
- `public void close()` – inchide socket-ul.

- **`java.net. ServerSocket`** – Socket server TCP
  - `public ServerSocket(int port) throws IOException,BindException` - constructor care inregistreaza serverul la portul specificat ca parametru.
  - `public Socket accept()` – asculta cererile de conexiuni, intoarce un obiect Socket care va fi utilizat pentru comunicarea cu clientul.
- **`java.io.DataOutputStream`** – flux de intrare pentru un socket
  - `public DataOutputStream(OutputStream out)` - constructor
  - `public final void writeBytes(String s)` – scrie un sir catre socket
- **`java.io.InputStreamReader`** – citeste octeti de la un socket si îi transformă în caractere. care sunt apoi trimise catre un `BufferedReader`
- **`java.io.BufferedReader`** – cititor de secvente de caractere cu zona tampon
  - `public String readLine()` – citeste urmatoarea linie de text dintr-un flux

2.2.1.2. Comunicare TCP la server

1. Se deschide un socket TCP pentru ascultarea cererilor de conexiuni
   a. Se creeaza o instanta a clasei ServerSocket

   Exemplu:

```
ServerSocket servSocket = new ServerSocket(15876);

Serverul asculta cererile de conexiuni pe portul 15876. Un client se
poate conecta la server folosind acest port.
```

2. Asteapta ca un client sa se conecteze si creaza un nou socket pentru a realiza comunicarea
   a. Se foloseste metoda accept a clasei ServerSocket

   Exemplu:

```
Socket conexSocket = servSocket.accept();
```

   Metoda `accept` va bloca procesul apelant pana cand serverul este contactat de catre un client folosind portul indicat. Dupa stabilirea conexiunii (TCP handshake) se creeaza un socket nou pentru comunicarea cu clientul.

   b. Se deschide un flux de iesire pentru a scrie in socket, prin instantierea unui obiect `DataOutputStream`

   Exemplu:

```
DataOutputStream out = new
      DataOutputStream(conexSocket.getOutputStream());
```

   Putem utiliza metoda `writeBytes` pentru a trimite date catre client

   c. Se deschide un flux de intrare pentru a citi de la socket prin instantierea unui obiect `BufferedReader`

Exemplu:
```
BufferedReader in = new BufferedReader (
   new InputStreamReader(conexSocket.getInputStream()));
```

Putem utiliza metoda `readLine` pentru a citi o linie de text de la client

3. Se citeste raspunsul clientului
   Folosim metoda `readLine` a clasei `BufferedReader` pentru a cite mesaje de la client
   Exemplu:
   ```
   String raspunsClient = in.readLine();
   ```

4. Se proceseaza raspunsul clientului
5. Se trimite un mesaj catre client
   Folosim metoda `writeBytes` a clasei `DataOutputStream` pentru a trimite mesaje catre client
   Exemplu:
   ```
   out.writeBytes(raspunsServer);
   ```

6. Se inchide socket-ul
   `Folosim metoda close a clasei Socket pentru a termina conexiunea TCP`
   Exemplu:
   ```
   conexSocket.close();
   ```

7. Se asteapta alt client

Serverul trebuie implementat intr-o bucla care sa permita mai multor clienti sa se conecteze succesiv la server. Cum exemplul de mai sus nu este multi-thread, un singur client poate fi conectat la un moment dat la server.

2.2.1.3. Comunicare TCP la client

1. Se deschide un socket TCP catre server
   a. Se creeaza o instanta a clasei Socket
      Exemplu:
      ```
      Socket clientSocket = new Socket("P19-x", 15876);

      Comunicarea cu serverul este acum stabilita. Serverul poarta numele de
      P19-x si asculta cererile pe portul 15876
      ```

   b. `Se deschide un flux de iesire pentru a scrie la socket prin instantierea unui obiect DataOutputStream`
      Exemplu:
      ```
      DataOutputStream out = new
            DataOutputStream(clientSocket.getOutputStream());
      ```

      Putem utiliza metoda `writeBytes` pentru a trimite date catre server

   c. `Se deschide un flux de intrare pentru a citi de la socket prin instantierea unui obiect BufferedReader`
      Exemplu:
      ```
      BufferedReader in = new BufferedReader (
                new InputStreamReader (clientSocket.getInputStream()));
      ```

      Putem utiliza metoda `readLine` pentru a citi o linie de text de la server.

2. Mesajul catre server este formatat

```
String mesaj=”Un mesaj catre server”+ ”\n”;
```

3. Se trimite un mesaj la server
   Folosim metoda `writeBytes` a clasei `DataOutputStream` pentru a trimite mesaje catre server
   Exemplu:
   `out.writeBytes(mesaj);`

4. Se citeste raspunsul serverului
   Folosim metoda `readLine` a clasei `BufferedReader` pentru a cite mesaje de la server
   Exemplu:
   `String raspuns = in.readLine();`

5. `Se inchide socket-ul`
   `Folosim metoda close a clasei Socket pentru a termina conexiunea TCP`
   Exemplu:
   `clientSocket.close();`

2.2.2. UDP (User Datagram Protocol)
UDP este un protocol neorientat spre conexiune care transmite datele cu ajutorul protocolului IP. UDP oferă aplicaţiilor acces direct la serviciul de transmitere a datelor dar nu oferă mecanisme de corectare a erorilor. Spre deosebire de TCP, UDP nu realizeaza o conexiune logica intre cele doua gazde, ci incapsuleaza informatia in pachete independente (datagrame), impreuna cu adresa destinatie si numarul portului, si apoi le transmite prin retea.

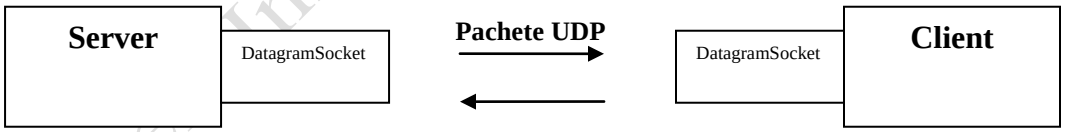
2.2.2.1. Clase si metode pentru programarea cu sockets (UDP)
Pentru a putea realiza un program care utilizeaza socket-uri trebuie sa importam pachetele `java.io` http://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html si
`java.net` `https://docs.oracle.com/javase/8/docs/api/java/net/package-summary.html`

- **`java.net.DatagramPacket`** - clasa pentru realizarea pachetelor pentru transmiterea datelor
  - `DatagramPacket(byte inbuf[], int buflength)` - construieste un `DatagramPacket` pentru receptionarea datagramelor. Parametrul `inbuf` este un tablou de bytes pentru salvarea datelor primite si `buflength` indica numarul de octeti care vor fi cititi.
  - `DatagramPacket(byte inbuf[], int buflength, InetAddress iaddr, int port)` – construieste un pachet pentru transmisia datelor. Fata de datagramele pentru receptie acest construcutor specifica si adresa IP a masinii destinatie si numarul portului.
- **`java.net.DatagramSocket`** – clasa pentru contruirea socketurilor pentru receptia si transmisia datagramelor.
  - `DatagramSocket() throws SocketException` - constructor care creaza un socket utilizand primul port disponibil
  - `DatagramSocket(int port) throws SocketException` – constructor care creaza un socket utilizand portul specificat ca parametru.
  - `void send(DatagramPacket p) throws IOException` – trimite o datagrama
  - `syncronized void receive(DatagramPacket p) throws IOException` – primeste o datagrama
  - `syncronized void close()` – inchide socketul
  - `int getLocalPort()` – intoarce portul pe care asculta socketul pentru datagrame

- **`java.io.DataInputStream`** - flux pentru citirea datelor de tip primitiv într-un format independent de masina pe care se lucreaza
  - `public DataInputStream(InputStream in)` – constructor
  - `readBoolean()`
  - `readByte()`
  - `readChar()`
  - `readDouble()`
  - `readFloat()`
  - `readInt()`
  - `readLong()`
  - `readShort()`
  - `readUnsignedByte()`
  - `readUnsignedShort()`
  - `String readUTF()`
- **`java.io.DataOutputStream`** - flux pentru scrierea datelor de tip primitiv într-un format independent de masina pe care se lucreaza
  - `public DataOutputStream(OutputStream out)` – constructor
  - `writeBoolean(boolean v)`
  - `writeByte(int v)`
  - `writeChar(int v)`
  - `writeDouble(double v)`
  - `writeFloat(float v)`
  - `writeInt(int v)`
  - `writeLong(long v)`
  - `writeShort(int v)`
  - `writeBytes(String s)`
  - `writeChars(String s)`
  - `writeUTF(String str)`
- **`java.io.ByteArrayInputStream`** – flux pentru citirea informatiilor care este creat pe un array de bytes existent.
- **`java.io.ByteArrayOutputStream`** – flux pentru scrierea informatiilor care este creat pe un array de bytes existent.
  - `toByteArray()` – `creaza un array de bytes. Dimensiunea array-ului este data de dimensiunea fluxului de iesire si el va contine octetii din buffer.`

O aplicatie simpla client/server neorientata spre conexiune

Atat clientul cat si serverul folosesc obiecte de tipul `DatagramSocket`.



2.2.2.2. Server UDP:

1. Se creaza un obiect `DatagramSocket` asociat cu un numar de port specificat
   Exemplu:

```
DatagramSocket socket = new DatagramSocket(8400);
```

2. Sa creeze un obiect `DatagramPacket` utilizand constructorul pentru datagrame pentru receptionarea datelor
   Exemplu:

```
byte[] buf = new byte[256];
```

```
DatagramPacket mesaj= new DatagramPacket(buf, buf.length);
```

3. Cu ajutorul metodei `receive()` a clasei `DatagramSocket` se salveaza o datagrama in obiectul de tip `DatagramPacket`
   Exemplu:

```
socket.receive(mesaj);
```

4. Se proceseaza cererea
5. Se creaza un obiect `DatagramPacket` utilizand constructorul pentru datagrame pentru transmisia datelor.
   a) Se identifica adresa si portul de la care vine cererea
   Exemplu:

```
InetAddress adresa = mesaj.getAddress();
```

```
int port = mesaj.getPort();
```

   b) Se construieste raspunsul

   Exemplu:

```
byte[] buf2 = new byte[256];
```

```
buf2 = ("...").getBytes();
```

   c) se completeaza datagrama care va fi trimisa

   Exemplu:

```
DatagramPacket raspuns =
     new DatagramPacket(buf2, buf2.length, adresa, port);
```

6. Cu ajutorul metodei `send()` a clasei `DatagramSocket` se trimite datagrama completata anterior.
   Exemplu:

```
socket.send(raspuns);
```

2.2.2.3. Client UDP: se implementeaza aceleasi etape, cu mentiunea ca mai intai se trimite o datagrama, dupa care se asteapta raspunsul serverului.

1. Se creaza un obiect `DatagramSocket`
   a) Se specifica adresa IP si portul pe care ruleaza serverul
   Exemplu:

```
InetAddress adresa = InetAddress.getLocalHost().getHostName();
int port=8400;
```

   b) Se instantiaza clasa `DatagramSocket`
   Exemplu:

```
DatagramSocket socket = new DatagramSocket();
```

2. Se creaza un obiect `DatagramPacket` utilizand constructorul pentru datagrame pentru transmisia datelor.

   a) Se construieste raspunsul

   Exemplu:

```
byte[] buf = new byte[256];
```

```
buf = ("...").getBytes();
```

b) se completeaza datagrama care va fi trimisa

Exemplu:

```
DatagramPacket mesaj =
     new DatagramPacket(buf, buf.length, adresa, port);
```

3. Cu ajutorul metodei `send()` a clasei `DatagramSocket` se trimite datagrama completata anterior.
   Exemplu:

```
socket.send(mesaj);
```

4. Se asteapta raspunsul serverului
5. Sa creeze un obiect `DatagramPacket` utilizand constructorul pentru datagrame pentru receptionarea datelor
   Exemplu:

```
byte[] buf = new byte[256];
DatagramPacket raspuns= new DatagramPacket(buf, buf.length);
```

6. Cu ajutorul metodei `receive()` a clasei `DatagramSocket` se salveaza o datagrama in obiectul de tip `DatagramPacket`
   Exemplu:

```
socket.receive(raspuns);
```

2.3. Riverbed Modeler Academic Edition

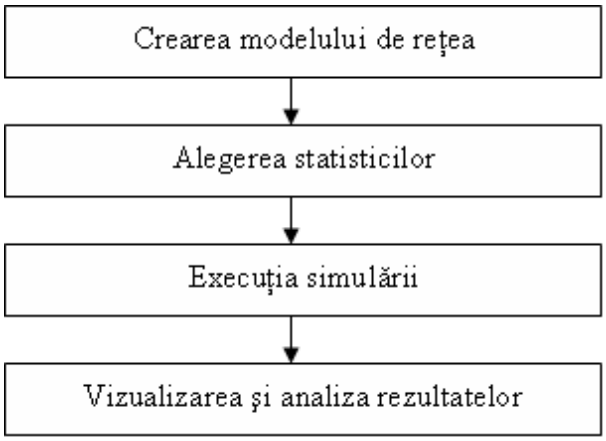
2.5.1. Introducere (vezi Lab 1)

Riverbed Modeler Academic Edition – mediu de simulare a retelelor de calculatoare - furnizează software de management pentru aplicaţii şi reţele, care oferă soluţii pentru:

- Planificarea capacităţii reţelelor,
- Modelare şi simulare pentru rețele și aplicaţii
- Managementul configurării reţelelor
- Managementul performanţelor aplicaţiilor

Riverbed oferă **Modeler Academic Edition**) - include modele standard pentru protocoale şi echipamentele disponibile în tehnologia IT (disponibile, dupa instalare, în subdirectoarele C:\Riverbed EDU\17.5.A\models\std.

Etapele de lucru avute în vedere sunt definite în *Modeler Riverbed workflow*:

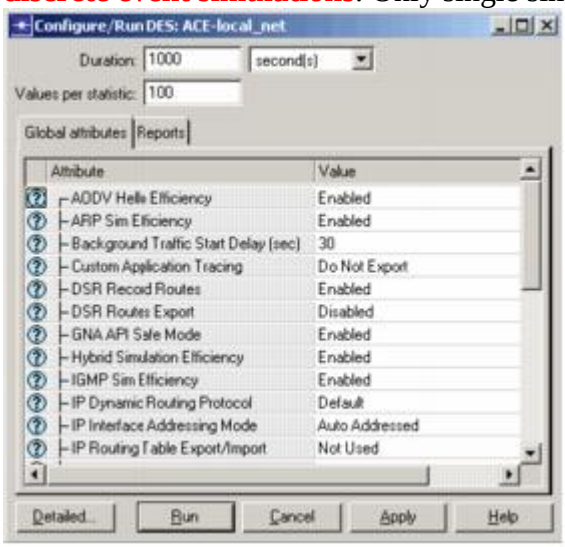


Etapele de lucru Modeler pentru simularea şi analiza unei reţele

Obs: O statistica este o caracteristica numerica a unui esantion (Anexa 3, pag.79, Lab_02)

- **Statistica** este stiinta colectarii, clasificarii, prezentarii, interpretarii datelor numerice si a folosirii acestora pentru a formula concluzii si a lua decizii.
- **Statistica descriptiva (**Descriptive Statistics) se ocupa cu colectarea, clasificarea si prezentarea datelor numerice.
- **Statistica inferentiala** (Inferential Statistics) se ocupa cu interpretarea datelor oferite de statistica descriptiva si cu folosirea acestora pentru a formula concluzii si lua decizii.

Configure/Run DES Dialog Box (Simple) The Configure/Run DES dialog box lets you configure and run a discrete event simulation for the current scenario. The simple version of the dialog box, (shown in the following figure), which appears when the DES configuration mode is set to “simple”, presents a reduced set of controls to simplify configuration and execution of **discrete event simulations**. Only single simulation runs are supported.



The simple Configure/Run DES dialog box has two pages of controls. These controls are organized by type and can be selected by clicking the corresponding tab. The following table lists the controls in this dialog box.

Element	Description
Basic controls	Duration field—Sets the duration of the simulation. Specify units with the pull-down menu following this field. This value sets the “duration” simulation preference. Values per statistic field—Sets the maximum number of values collected for each statistic. This value sets the “num_collect_values” simulation preference.
Global Attributes page	Use this page to define the values of global simulation attributes. This page is similar to the Global Attributes page—Used to define the values of global simulation attributes for the simulation. seen in Detailed mode, except that you cannot set multiple values for an attribute or automatically reset the default value.
Reports page	Use this page to select Statistic reports and Service Level Agreement (SLA) reports for the simulation. Reports are predefined sets of statistic probes. This page is identical to the Configure/Run DES Dialog Box (Detailed)—Report Controls seen in Detailed mode.
Dialog box controls	Detailed… button—Switches temporarily to detailed mode and the detailed Configure/Run DES dialog box, as described in Configure/Run DES Dialog Box (Detailed). (This button does not change the des.configuration_mode preference.)

Run button—Saves the current settings, closes the dialog box, and runs the simulation. Running a simulation from here opens the Simulation Execution Dialog Box.

Cancel button—Closes the dialog box without saving any changed settings.

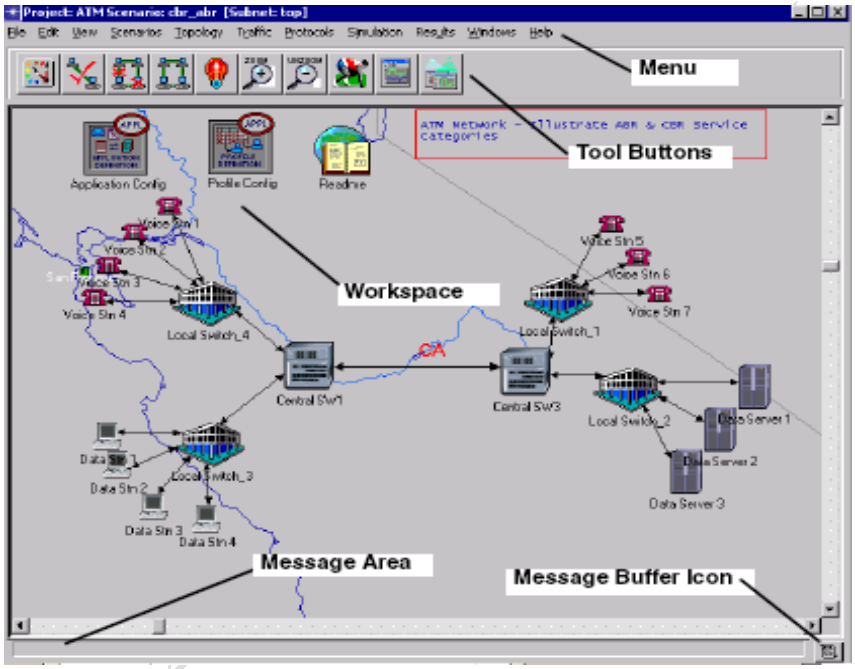
Apply button—Saves the current settings and keeps the dialog box open.

Help button—Opens a help file for the dialog box.

Workspace este spaţiul de lucru din partea centrală a ferestrei editorului, care este folosit pentru crearea modelului reţelei, selectarea şi deplasarea obiectelor reţelei, alegerea operaţiilor specifice conextului.

Message Area, plasată în partea de jos a ferestrei, furnizează informaţii despre starea *tool-ului.*

Message Buffer Window, plasata în partea de jos în stânga, permite accesul la o listă de mesaje, notificări, atenţionări.



Project Editor Window

Butoane folosite în *Project Editor*

Semnificaţia butoanelor din *Project Editor*

1. Open object palette	*6. Zoom*
2. Check link consistency	*7. Restore*
3. Fail Selected objects	*8. Configure discrete event simulation*
4. Recover selected objects	*9. View simulation results*
5. Return to parent subnet	*10. Hide or show all graphs*

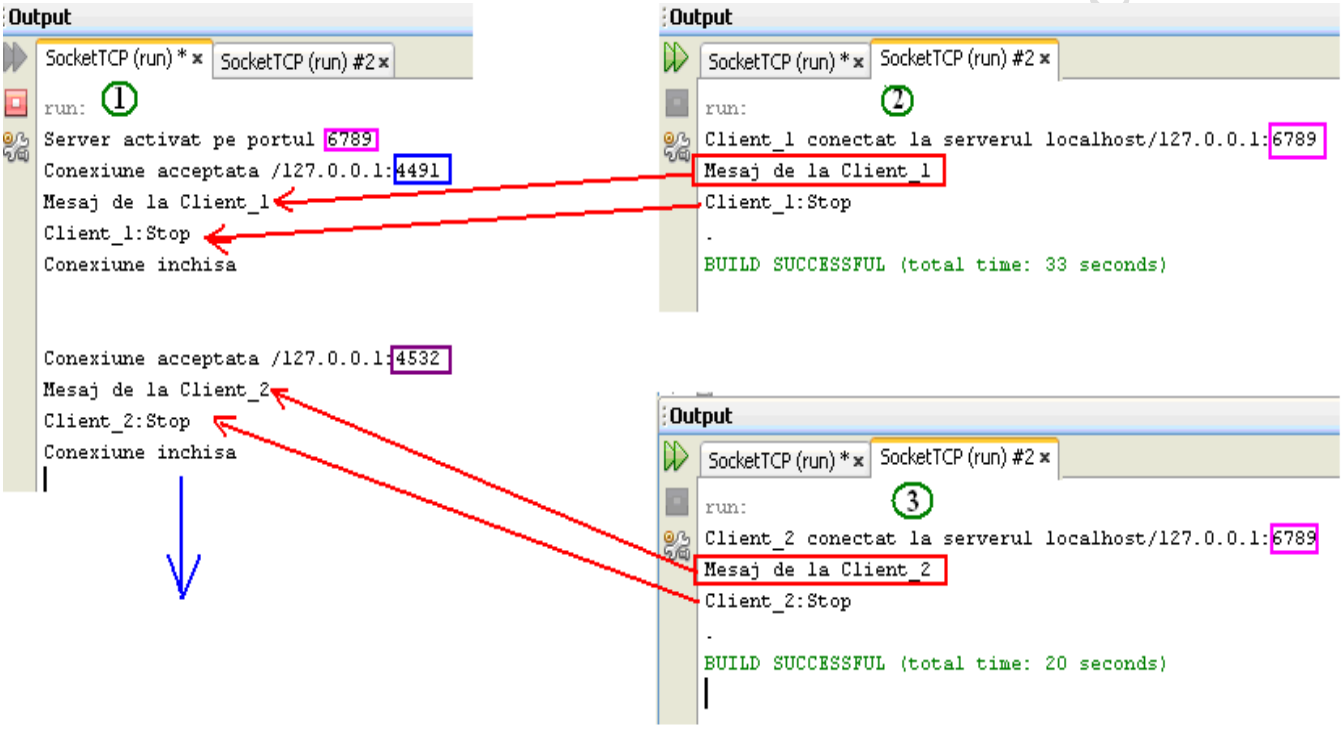
3. Partea practica (Tema pagina 23)

3.1. Aplicatia A1: Predicting the Impact of TCP Window Size on Application Performance (Se vor folosi TCP_WindowSize.pdf + TCP_Window_Size.prj din folderul 3_TCP_WindowSize)

1. Se copiaza folderul **TCP_WindowSizePrj** in C:\Users\.....\op_models;
2. Se ruleaza proiectul conform cerintelor formulate in **3_TCP_WindowSize.pdf**.

3.3. Aplicatia A3: Comunicatie client-server TCP

Exemplu comentariu ...model pentru redactare tema!



3.3.1. TCP Server

Indicatii :

```
import java.net.*;
import java.io.*;

public class TcpServer {

    public static void main(String args[]) {

      int port;
      ServerSocket server_socket;
      BufferedReader input;

      try {
          port = Integer.parseInt(args[0]);
      }catch (Exception e) {
```

```
            System.out.println("port = 15876 (default)");
            port = 15876;
        }

        try {

            server_socket = new ServerSocket(port);
            System.out.println("Serverul este activ la portul " +
                               server_socket.getLocalPort());

            // bucla
            while(true) {

              Socket socket = server_socket.accept();
              System.out.println("Conexiune acceptata " +
                             socket.getInetAddress() +
                             ":" + socket.getPort());
                 input = new BufferedReader(
                   new InputStreamReader(socket.getInputStream()));
              // afiseaza informatiile primite
              try {
                  while(true) {
                    String message = input.readLine();
                    if (message==null) break;
                    System.out.println(message);
                  }
              }catch (IOException e) {
                  System.out.println(e);
              }

              // inchide conexiunea cu clientul
              try {
                  socket.close();
                  System.out.println("Conexiune inchisa de client ");
              }catch (IOException e) {
                  System.out.println(e);}
            }
        }catch (IOException e) {
            System.out.println(e);}
    }
}
```

3.3.2. TCP Client

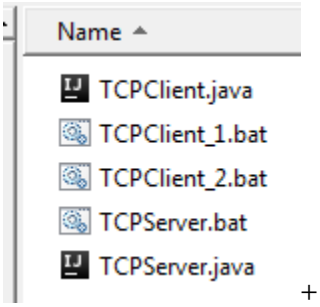
Indicatii :

```
import java.net.*;
import java.io.*;

public class TcpClient {

    public static void main(String[] args) {
```

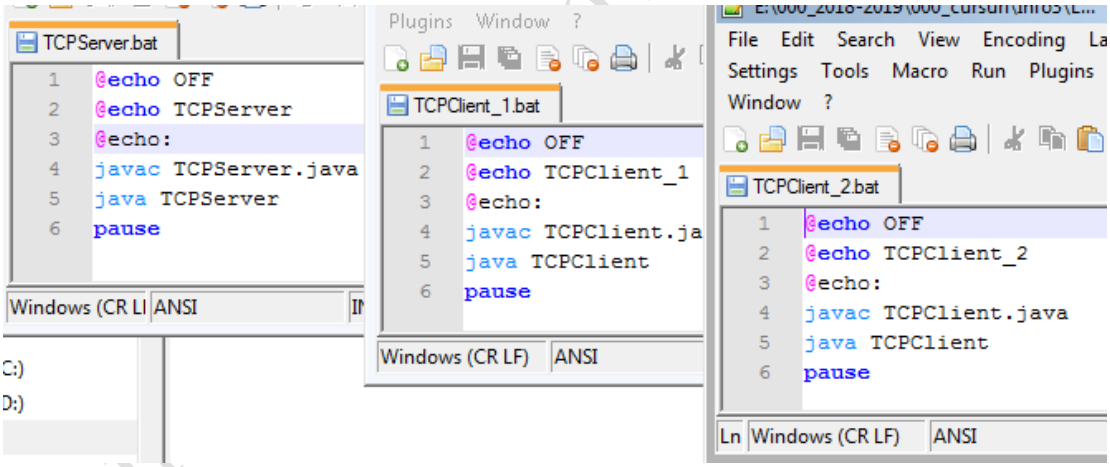
```
int port = 15876;
String server = "localhost";    //clientul va rula pe aceasi masina
Socket socket = null;
String lineToBeSent;
BufferedReader input;
PrintWriter output;
int ERROR = 1;

// citeste argumentele
if(args.length == 2) {
    server = args[0];
    try {
         port = Integer.parseInt(args[1]);
       }
       catch (Exception e) {
         System.out.println("server port = 15876 (default)");
         port = 15876;
       }
   }

   // conectare la server
   try {
       socket = new Socket(server, port);
       System.out.println("Conectat la serverul " +
                          socket.getInetAddress() +
                          ":" + socket.getPort());

   }catch (UnknownHostException e) {
       System.out.println(e);
       System.exit(ERROR);
   }
   catch (IOException e) {
       System.out.println(e);
       System.exit(ERROR);
   }

try {
    input = new BufferedReader(
            new InputStreamReader(System.in));
    output = new PrintWriter(socket.getOutputStream(),true);

    // preia informatiile si le transmite la server
    while(true) {
      lineToBeSent = input.readLine();
      // programul se opreste daca intalneste "."
      if(lineToBeSent.equals(".")) break;
      output.println(lineToBeSent);
    }
}
```

```
        catch (IOException e) {
            System.out.println(e);
        }

        try {
            socket.close();
        }
        catch (IOException e) {
            System.out.println(e);
        }
    }
}
```

Prezentare folder A3_Nume_Prenume



+ **TCP.png**



3.4. Aplicatia A4: TCP_Joc_Nume_Prenume (TEMA !!!!!)

- Realizati o aplicatie client-server care foloseste protocolul TCP: un "joc" al carui scop este ghicirea unui numar. Atunci cand serverul este pornit el va salva un numar aleator intre 0 si 500.

Indicatii:

```
int randomNumber=(int)(Math.random()*500);
```

- Clientul va citi numere de la tastatura si va trimite aceste numere la server. Serverul va raspunde prin mesaje (MARE, MIC, CORECT). Clientul trebuie sa poata introduce numere pana cand primeste valoarea CORECT.

3.5. Aplicatia A5: Comunicatie client-server UDP

3.5.1. UDP Server

Indicatii :

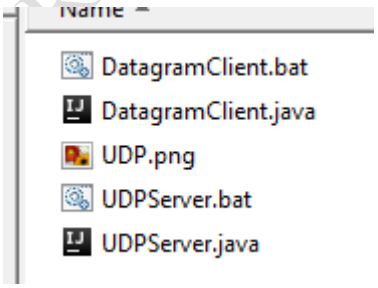
```
import java.net.*;
import java.io.*;

public class DatagramServer {

        public static final int PORT = 8200;
        private DatagramSocket socket = null;
        DatagramPacket cerere, raspuns = null;

        public DatagramServer() throws IOException {

              Socket = new DatagramSocket(PORT);
               try {
                while (true) {

                    //Declara pachetul in care va fi receptionata cererea
                    byte[] buf = new byte[256];
                    cerere = new DatagramPacket(buf, buf.length);

                              //Astepta aparitia unui pachet cu cererea
                              socket.receive(cerere);

              //Afla adresa si portul de la care vine cererea
              InetAddress adresa = cerere.getAddress();
              int port = cerere.getPort();


              //Construieste raspunsul
              buf = ("Server:" +
                        new String(cerere.getData())).getBytes();

              //Trimite un pachet cu raspunsul catre client
              raspuns = new DatagramPacket(buf, buf.length,
                                           adresa, port);

                  socket.send(raspuns);
                           }
                  } finally {
                           socket.close();
                  }
          }

     public static void main(String[] args) throws IOException {
          new DatagramServer();
     }
}
```

3.5.2. UDP Client

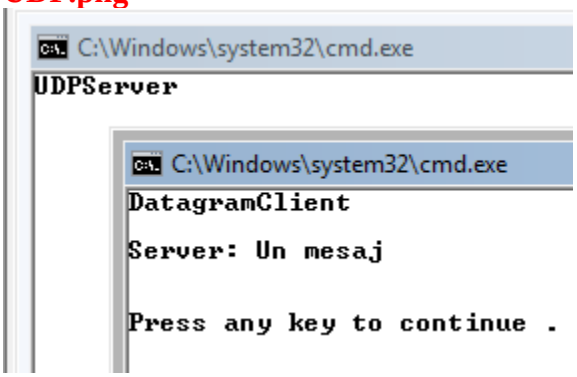
Indicatii :

```
import java.net.*;
import java.io.*;

public class DatagramClient {

	public static void main(String[] args) throws IOException {

		//adresa IP si portul la care ruleaza serverul
		InetAddress address = InetAddress.getByName("127.0.0.1");
		int port=8200;

		DatagramSocket socket = null;
		DatagramPacket packet = null;
		byte buf[];

		try {
		//Construieste un socket pentru comunicare
		socket = new DatagramSocket();

		//Construieste si trimite pachetul cu cerere catre server
		buf = "Un mesaj".getBytes();
		packet = new DatagramPacket(buf, buf.length,
					address, port);
			socket.send(packet);

		 //Asteapta pachetul cu raspunsul de la server
		 buf = new byte[256];
		 packet = new DatagramPacket(buf, buf.length);
		 socket.receive(packet);

		 //Afiseaza raspunsul
		 System.out.println(new String(packet.getData()));
		 } finally {
			socket.close();
		 }
	}
}
```

Prezentare folder A5_Nume_Prenume



UDP.png



3.6.Aplicatia A6:UDP_Tip_Nume_Prenume - Exercitiu (TEMA !!!!!)

- Realizati o aplicatie client-server care foloseste protocolul UDP. Clientul va cere serverului timpul si serverul il va furniza.
  Indicatii:
  http://www.codeproject.com/KB/IP/udptime.aspx;
  http://www.java2s.com/Tutorial/Java/0320__Network/UDPtimeserverbasedonNewIO.htm

Exemple de metode

- Exemplu de metoda pentru construirea unui `DatagramPacket`

```
DatagramPacket buildPacket (String message, String host, int port) throws
IOException {

        // creaza un tablou de bytes dintr-un sir

        ByteArrayOutputStream byteOut = new ByteArrayOutputStream ();
        DataOutputStream dataOut = new DataOutputStream (byteOut);
        dataOut.writeBytes(message);
        byte[] data = byteOut.toByteArray ();

        //intoarce pachetul
        return new DatagramPacket (data, data.length, InetAddress.getByName(host),
                                   port);

}
```

- Exemplu de metoda care primeste un DatagramPacket si afiseaza continutul acestuia.

```
void receivePacket () throws IOException {

       byte buffer[] = new byte[65535];
       DatagramPacket packet = new DatagramPacket (buffer, buffer.length);
       socket.receive (packet);

       // Transforma un tablou de bytes intr-un sir

       ByteArrayInputStream byteIn = new ByteArrayInputStream (
                                  packet.getData (), 0, packet.getLength ());
```

```
DataInputStream dataIn = new DataInputStream (byteIn);

// citeste datele utilizand un format standard
String input = "";
while((input = dataIn.readLine ()) != null)
output.appendText("SERVER-ul raspunde : " + input + "\n");}
```

3.7. Aplicatii de retea in Pyton

3.7.1. Recapitulare (Lab_02, Lab_03)

- **Python_intro**
- **Programare_Python**
- **Byte-of-python**
- **Python socket network programming_1 (Lab_08)**
- **Python socket network programming_2 (Lab_08)**
- **Python Files and os.path (Lab_09)**

Obs: Anexa 2 - The Programming Process (pag.38)

3.7.2. Programarea socket-urilor de retea in Python (7_BasicsOfSockets.pdf)

3.7.2.1 Client-Server TCP Indicatii
3.7.2.2 Client-Server UDP Indicatii

3.7.3. Exercitii (Solutii propuse)

a. Simple_Server
b. Echo_Server
c. Local_File_Transfer

Challenge: Interfate grafice pentru a, b, c

Recomandare: Qt Designer , cu Designer din Anaconda prompt).
http://pythonforengineers.com/your-first-gui-app-with-python-and-pyqt/,
https://www.codementor.io/deepaksingh04/design-simple-dialog-using-pyqt5-designer-tool-ajskrd09n, https://wiki.python.org/moin/PyQt/Tutorials

3.7.4.TCP_Joc_Nume_Prenume (TEMA !!!!!)

Realizati o aplicatie client-server **PYTHON** care foloseste protocolul TCP: un "joc" al carui scop este ghicirea unui numar. Atunci cand serverul este pornit el va salva un numar aleator intre 0 si 500.

Clientul va citi numere de la tastatura si va trimite aceste numere la server. Serverul va raspunde prin mesaje (MARE, MIC, CORECT). Clientul trebuie sa poata introduce numere pana cand primeste valoarea CORECT.

Challenge: Interfata grafica

Recomandare: Qt Designer , cu Designer din Anaconda prompt).
http://pythonforengineers.com/your-first-gui-app-with-python-and-pyqt/,
https://www.codementor.io/deepaksingh04/design-simple-dialog-using-pyqt5-designer-tool-ajskrd09n, https://wiki.python.org/moin/PyQt/Tutorials

==========

Observatii TEMA!!!!!

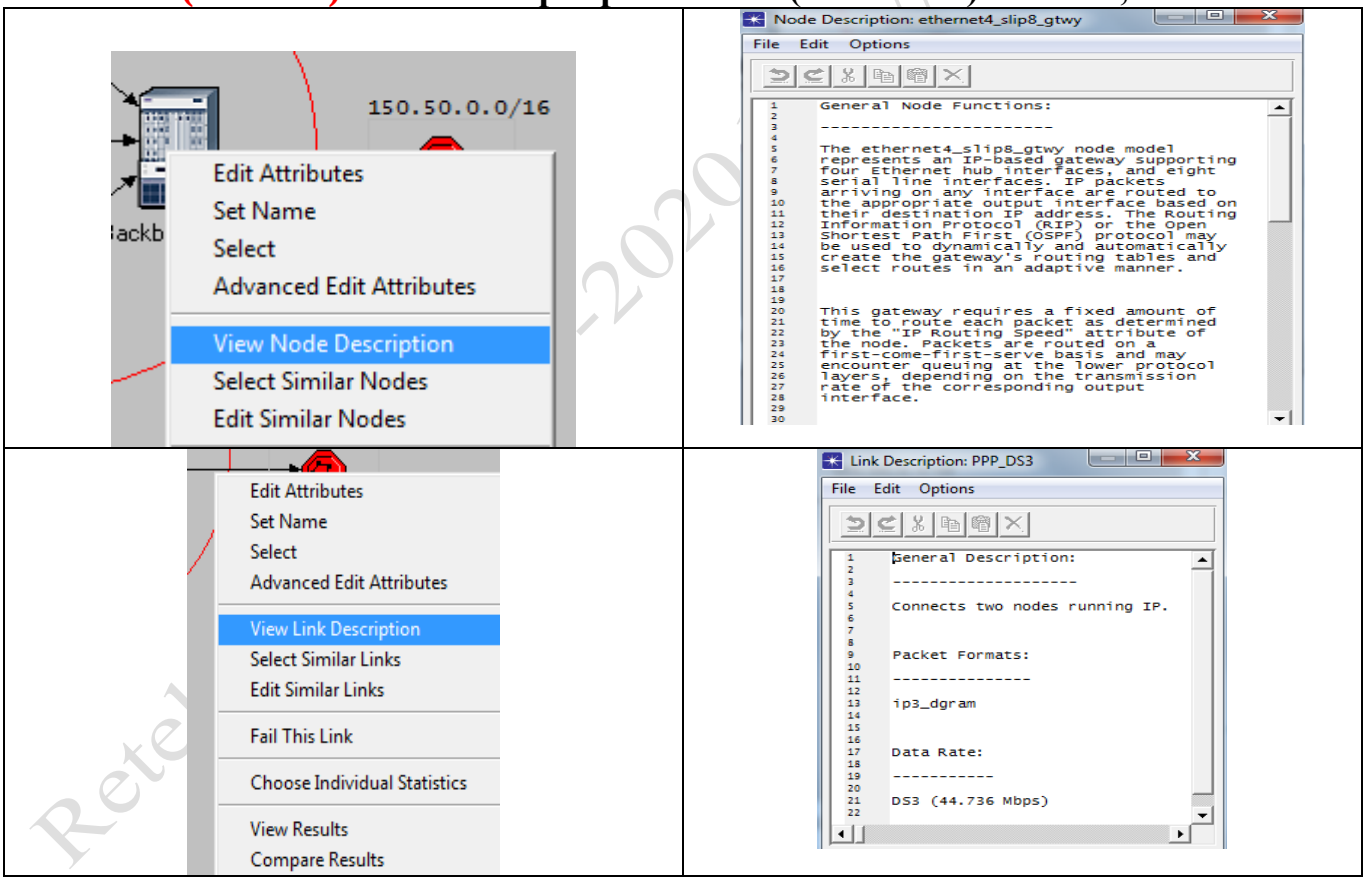
1. **Atentie (Modeler)** – Proiectul creat se salveaza implicit in:
   C:\Users\student(NUME user)\op_model\NUME_PROIECT

NUME_PROIECT ***contine proiectul modeler propriu-zis***

Varianta:

- In directorul **....\Studenti\Info3\Nume_Prenume** se creează directorul (**pentru punctul 3.3**) **\L9_3.3_Modeler_Nume_Prenume** folosind:
  - ***File → New → Folder***
- Se lansează în execuţie Modeler.
- Se selectează directorul în care vor fi plasate fişierele proiectului.
  - ***File → Model Files →Add Model Directory***
  - Se selectează directorul în care se va lucra (în acest director vor fi salvate fişierele proiectului curent)
  - Se arhiveaza **L9_3.3_Modeler_Nume_Prenume**

2. Atentie (Modeler) : Click dreapta pe "obiect" (ex. Router)..."Judec, deci exist!"



…..similar [omnet++](http://www.omnetpp.org)….. (http://www.omnetpp.org)

Retele de calculatoare – Informatica anul 3 (2019-2020)

4. Tema:

- **Toate punctele din sectiunea 3 "partea practica"** se vor relua de catre cursanti, folosind etapele de lucru indicate. Rezultatele experimentale:
  - **L9_nume+prenume_Modeler (folder)** - contine **proiectul Modeler de la pct 3.1** si **L9_nume+prenume_Modeler.doc** (**document .doc):** rezultatele experimentale: **comentarii insotite de capturi (Snipping Tool) corespunzatoare proiectului Modeler (3.1), pasi intermediari importanti/topologia fizica, rezultate/capturi pentru View node description si View link description (obs.2 anterioara), exercitiile rezolvate, raspunsuri la intrebari, rezultate finale, observatii finale)**. ATENTIE: proiectele Modeler vor avea denumiri de tipul 3.1_Nume_Prenume)/ (**Varianta "programare" C++: OMNeT++ Network Simulation Framework** http://www.omnetpp.org/)
  - **L9_nume+prenume_java (folder) : contine subfolderele 3.3., 3.4., 3.5., 3.6., fiecare subfolder cu fisierele pentru fiecare aplicatie (.java, .bat, . png, insotite de un *readme.txt* pentru particularitati de rulare, conform prezentarilor facute). Atentie la modul de prezentare din Anexa 1...asa ar trebui!..si de pe masini diferite: ip sursa != ip destinatie)**
  - **L9_nume+prenume_Python (folder) – cu subfloderele 3.7.2, 3.7.3, 3.7.4 (fiecare din acestea contine scripturile .py si .doc/ .png (snipping tool) pentru aplicatiile Python. Atentie la modul de prezentare din Anexa 1...asa ar trebui!..si de pe masini diferite: ip sursa != ip destinatie. RECOMANDARE:** 3.6/7.1 (Lab2, Lab3, Lab4, Lab5, Lab6, Lab7, Lab8)

se vor arhiva cu numele **L9_nume+prenume_info3.rar si** se va trimite prin e-mail la adresa **retelecdsd@gmail.com** precizandu-se la **subject: L9_nume+prenume_info3**, pana pe data de **6 decembrie 2019 e.n., ora 8.00 a.m. (Atentie, gmail nu "prea vrea" .rar in .rar http://www.makeuseof.com/tag/4-ways-email-attachments-file-extension-blocked).**

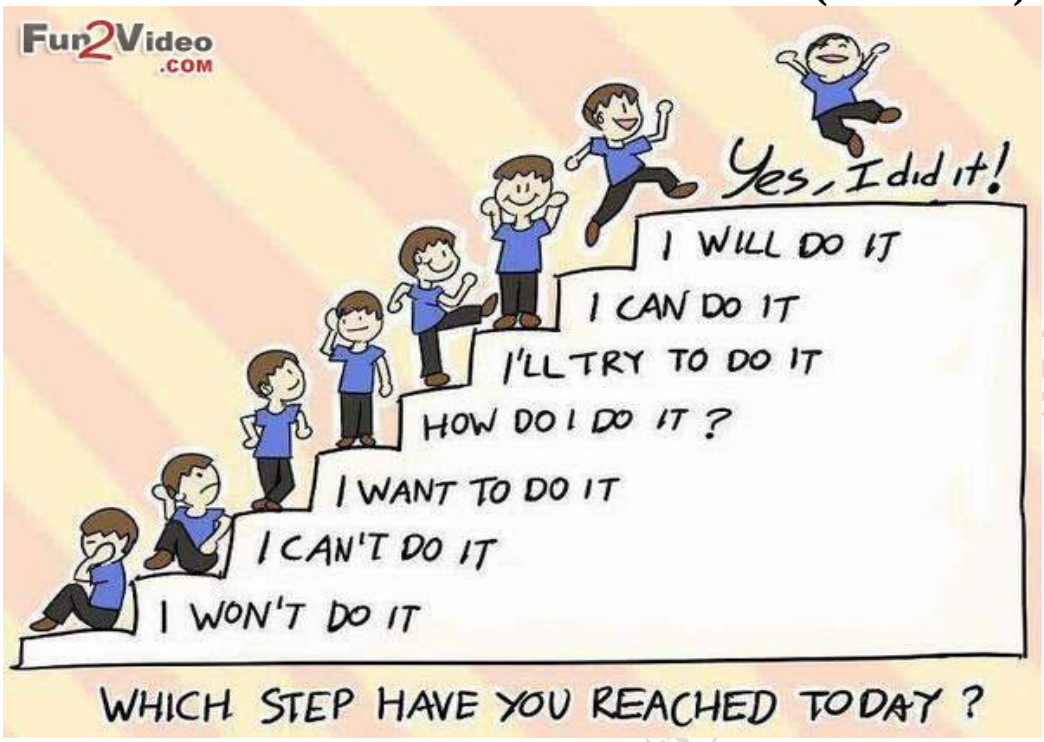
VARIANTE pentru trimiterea arhivei: http://www.gfile.ro; http://www.wetransfer.com
Cursantii sunt incurajati sa analizeze si sa comenteze rezultatele obtinute, studiind si materialele indicate in bibliografie si anexe. **(+ Recapitulare Laboratoarele 1+2+3+4+5+6+7+8)** (Pentru Modeler, varianta "programare" C++: **OMNeT++ Network Simulation Framework** http://www.omnetpp.org/;

Obs:

Punctaj maxim (Data trimiterii temei)			
<= 6.12. 2019	**10.12. 2019**	**14.12.2019**	**18.12.2019**
100 pct	**80 pct**	**60 pct**	**50 pct**

Obs: Participarea (activa!) la Curs si Laborator permite, prin cunostintele acumulate, obtinerea unor rezultate bune si f. bune, asa cum ni le dorim cu totii.

DE ANALIZAT readme-ul readme_mod_work_dir.pdf (si un numai!... de exemplu si **readme_lab_modeler.pdf) de la adresa http://www.cdsd.ro**



Sursa: http://www.funnfun.in/wp-content/uploads/2013/06/steps-of-success-encouraging-quote.jpg

How to send an e-mail

http://lifehacker.com/5803366/how-to-send-an-email-with-an-attachment-for-beginners
https://support.google.com/mail/answer/6584?hl=en "As a security measure to prevent potential viruses, Gmail doesn't allow you to send or receive executable files (such as files ending in `.exe`)."
https://support.google.com/mail/answer/2480713?hl=en
http://fastupload.ro/free.php
http://www.computerica.ro/siteuri-transfer-fisiere-mari-upload/

Bibligrafie:

Lab_01, Lab_02, Lab_03, Lab_04, Lab_05, Lab_06, Lab_07, TL_01, TL_02, TL_03, TL_04
http://www.cdsd.ro/cursuri

http://support.microsoft.com/kb/140859
http://www.windowsreference.com/windows-2000/how-to-add-static-route-in-windows-xp2000vista/
http://www.comptechdoc.org/os/linux/usersguide/linux_ugrouting.html
http://linux-ip.net/html/ch-routing.html
http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf
http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/route.mspx?mfr=true

efg' Mathematics, http://www.efg2.com/Lab/Mathematics/CRC.htm
Java API, https://docs.oracle.com/javase/8/docs/api/
Java Tutorial, Writing Your Own Filtered Streams http://www.rgagnon.com/javadetails/java-0416.html

http://en.wikipedia.org/wiki/Cyclic_redundancy_check

http://www34.brinkster.com/dizzyk/crc32.asp
http://www.createwindow.com/programming/crc32/crcfile.htm
http://webnet77.com/cgi-bin/helpers/crc.pl
http://www.softpedia.com/get/Others/Miscellaneous/CRC32-Calculator.shtml
http://www.wikiera.net/EthernetCRC-readytouseexample.html
http://www.wireshark.org/docs/wsug_html_chunked/ChAdvChecksums.html

Modeler Tutorials
https://rpmapps.riverbed.com/ae/4dcgi/SIGNUP_NewUser
https://supportkb.riverbed.com/support/index?page=content&id=S24443
https://rpmapps.riverbed.com/ae/4dcgi/DOWNLOAD_HOME
https://rpmapps.riverbed.com/ae/4dcgi/REG_TransactionCode

- Install Riverved Modeler 17 5 Windows 10, 8 1, 8 and 7 **(**https://www.youtube.com/watch?v=TpenN2jYbHQ)
- Install Riverbed Modeler (https://www.youtube.com/watch?v=DQ3XhHYuFGA)
- How to activate riverbed modeler 17.5 **(**https://www.youtube.com/watch?v=h-ImeJMqiSA)
- How to solve invalid activation of Opnet Modeler 17.5 **(**https://www.youtube.com/watch?v=13ZBcXkW46s)
- Riverbed Modeler 17.5 Tutorial - Switched Lan (https://www.youtube.com/watch?v=XdebwQLrr0w)
- 6-Virtual LAN (VLAN) configuration in OPNET Riverbed (https://www.youtube.com/watch?v=Ajz7bVO5WJM)
- Riverbed Modeler Configuracion VLAN (https://www.youtube.com/watch?v=rP3jPMcyEFk)
- Ethernet (lab_04)
- Riverbed Opnet 17.5 Tutorial - The Ethernet network **(**https://www.youtube.com/watch?v=fS_J6ApFJtc)
- 6-Virtual LAN (VLAN) configuration in OPNET Riverbed (https://www.youtube.com/watch?v=Ajz7bVO5WJM)
- Riverbed Modeler Tutorial 3 Configuracion VLAN (https://www.youtube.com/watch?v=rP3jPMcyEFk)

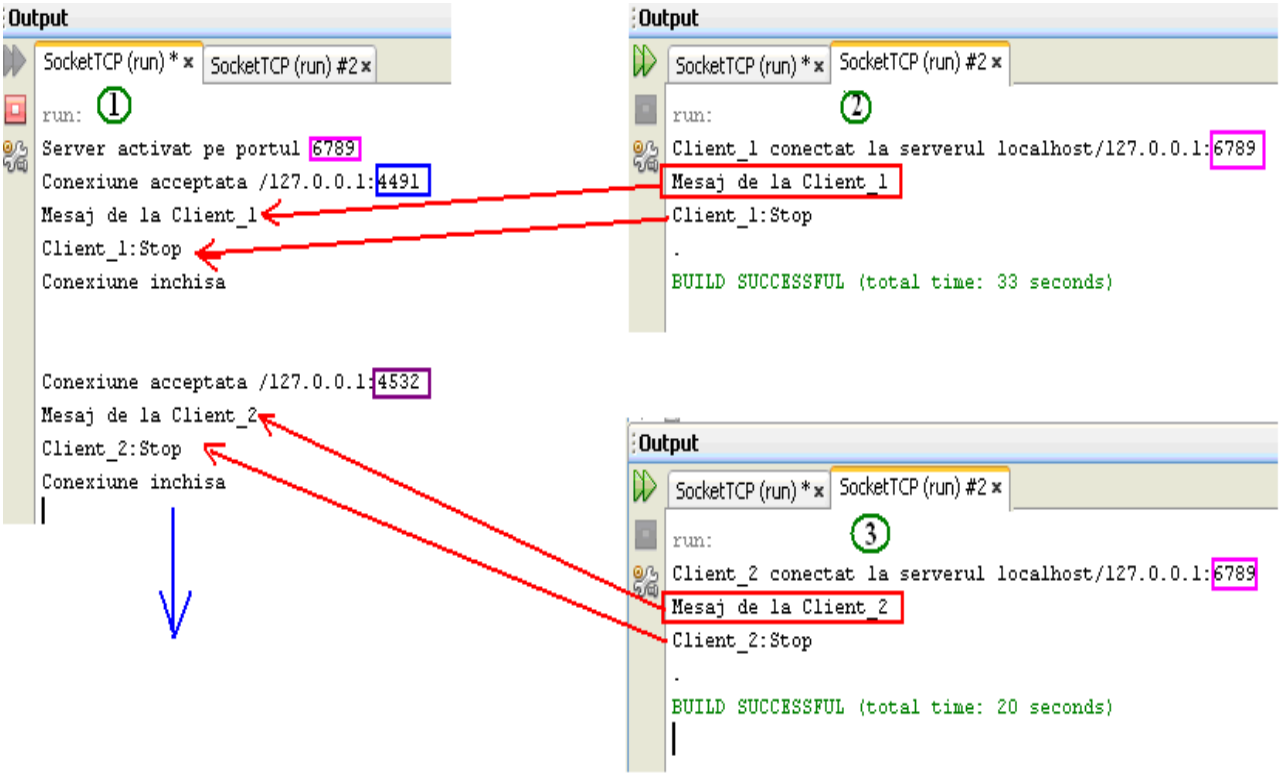
Python (Lab1, Lab2)
Using Python on Windows - https://docs.python.org/3/using/windows.html
The Hitchhiker's Guide to Python - http://docs.python-guide.org/en/latest/intro/learning/
A Byte of Python - https://www.gitbook.com/book/swaroopch/byte-of-python/details
GUI Programming in Python - https://wiki.python.org/moin/GuiProgramming
https://winpython.github.io/ ; https://www.python.org/

https://social.technet.microsoft.com/wiki/contents/articles/910.windows-7-enabling-telnet-client.aspx
http://www.telnet.org/htm/places.htm
rainmaker.wunderground.com : weather via telnet!

https://docs.python.org/3/library/socket.html
18.1. socket — Low-level networking interface

Anexa 1 :

Exemplu comentariu



Anexa 2: The Programming Process

1. Identify the Problem - **What** Are You Trying To Do?
   - Requirements
   - Specification
2. Design a Solution - **How** Is It Going To Be Done?
3. Write the Program - **Teaching** the Computer
   - Code
   - Compile
   - Debug
4. Check the Solution - **Testing** it Understands You