



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Relay Nodes

Temă la disciplina Rețele de calculatoare
- Tema 3 -

Student:

Bîrluțiu Claudiu-Andrei

Calculatoare și tehnologia informației

Grupa 30236

An universitar 2021/2022

Table of Contents

1. Enunțul problemei.....	3
1.1 Cerințe.....	3
2. Soluția propusă.....	3
3. Proiectare și implementare.....	4
4. Rezultate obținute.....	6
5. Analiză WireShark.....	11
6. Concluzii.....	14

1. Enunțul problemei

Să se realizeze un scenariu de simulare pentru topologia **Relay Nodes** care să reflecte figura și cerințele enumerate mai jos.

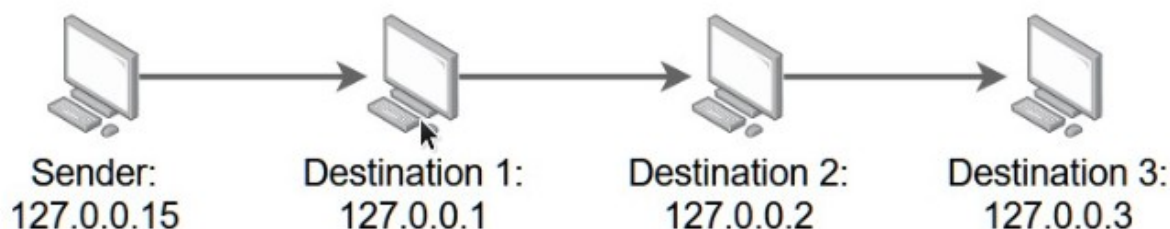


Figure 1: Nodurile din topologie pentru procesul de simulare

Specificații:

- sunt 4 noduri în topologie, **Sender** și 3 posibile destinații (D1, D2 și D3)
- sender-ul va transmite 100 pachete la o destinație random din cele 3 la un moment dat, iar valoarea transmisă e un număr întreg ce se incrementează la fiecare transmisie de la Sender
- restricția de transmisie/recepție: fiecare node poate trimite date doar la o singură destinație/hop la care este conectat

1.1 Cerințe

Se cere implementarea unui program într-un limbaj de programare cunoscut pentru a simula scenariul descris. Testarea programului se va face cu ajutorul tool-ului **Wireshark** și de asemenea se vor analiza comunicarea dintre diferitele adrese IP, se va observa raportul dintre totalul de payload livrat și traficul relevant al aplicației și se vor căuta header-ele pentru protocolul TCP.

2. Soluția propusă

Limbajul de programare folosit pentru implementarea programului de simulare a scenariului descris la capitolul anterior este **JAVA**(jdk 17). S-a ales acest limbaj pentru existența librăriei **java.net** care are 2 clase importante din care se pot instanția obiecte de tipul Client (**Socket**) și obiecte de tipul Server (**ServerSocket**) pentru realizarea comunicării client-server via sockets folosind loopback addresses.

În scenariu propus mai sus avem două categorii de obiecte: **Sender** și **Destination**. Cele două clase pot fi create într-o singură clasă, **RelayNode**, numai pentru diferențiere mai clară s-a recurs la varianta de avea 2 clase, una specifică **Sender**-ului, care va conține o instanță de client ca field principal și o clasă **RelayNode** ce conține componentele și descrie comportamentul unei destinații, cum e descrisă mai sus.

La o primă analiză a schemei din figura 1 și a comportamentului sistemului, se deduce faptul că un **RelayNode** trebuie să conțină o instanță de server ce va recepta mesajele primite de la RelayNode-ul de dinainte (în cazul de față din partea stângă), și o instanță de client ce va transmite mesaje spre relay node-ul următor (cel din dreapta), adică spre serverul corespunzător acestuia din urmă. În figura 2 se observă legătura dintre 3 relay node-uri adaptate celor 3 **destinații**.

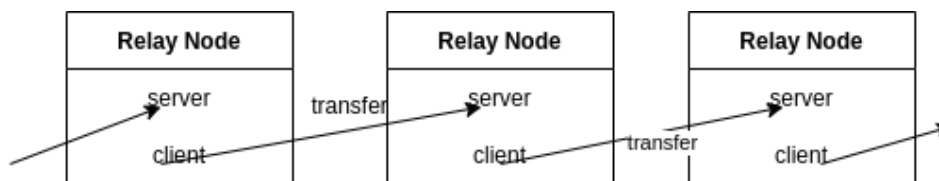


Figure 2: Relay nodes realizare comunicare

Sender-ul va avea o instanță de **client** ce se va conecta la prima destinație din rețea, mai precis se va conecta la serverul specific al acesteia.

3. Proiectare și implementare

În figura 3 se regăsește diagrama de clase pentru implementarea sistemului descris mai sus și simularea scenariului de transmitere a 100 pachete la destinații random.

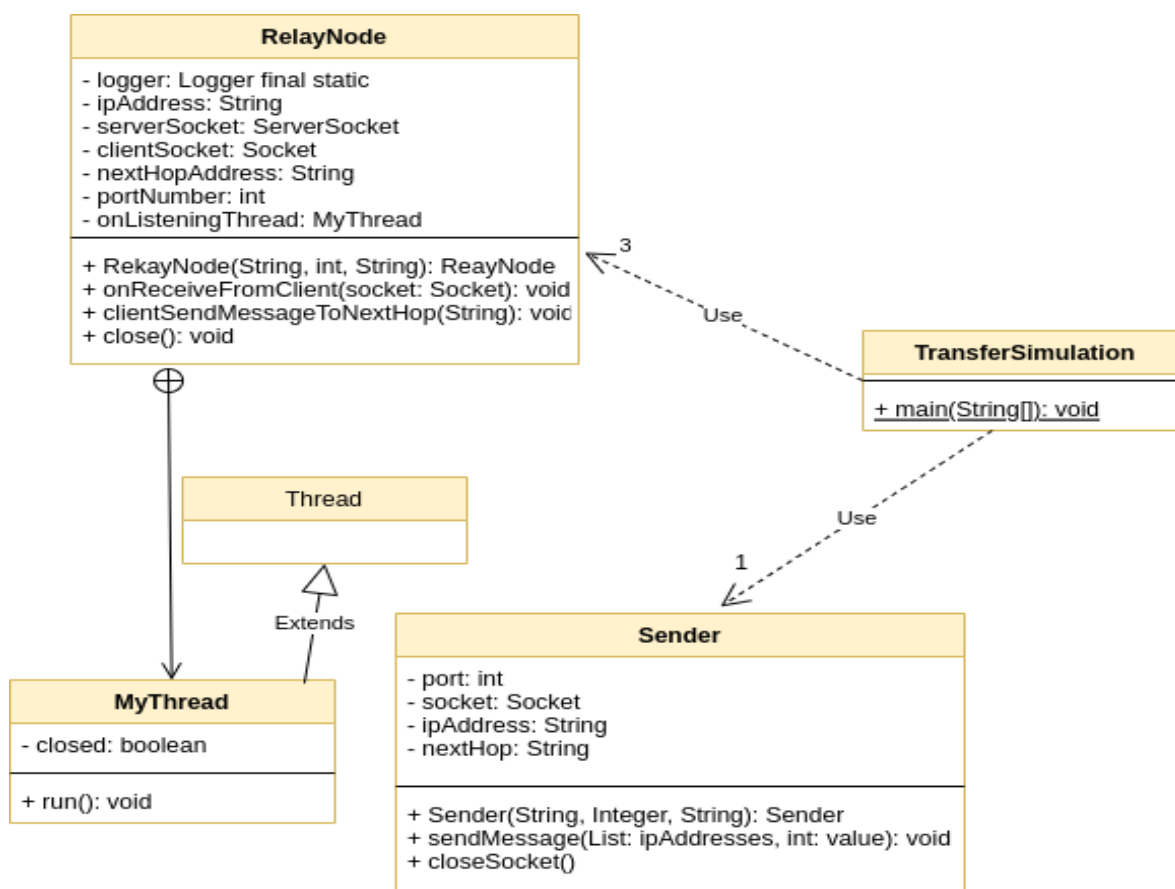


Figure 3: Diragrama de clase

Clasa **Sender**:

- această clasă are ca field principal un obiect de tip client (**Socket**). S-a ales implementarea acestei clase pentru a contura doar comportamentul de trimitere mesaje, nu și recepție așa cum e descris în scenariu
- un obiect de tipul sender va avea și o adresă ip la care să se conecteze pentru a putea trimite mesaje spre următorul nod din rețea (**cazul de față destinație 1**)
- metoda **sendMessage** va alege aleator una dintre destinațiile primite ca parametru și va trimite payload-ul spre aceasta. Transmiterea nu se realizează direct, ci prin hop-uri. Sender-ul poate trimite mereu mesaje doar la o singură destinație, iar aceasta din urmă va procesa mesajul primit, iar dacă mesajul nu e destinat ei, atunci îl va transmite mai departe spre nodul cu care este conectată (next hop-ul).
- Din cele descrise mai sus reiese faptul că payload-ul trebuie să aibă un format anume, în care pe lângă mesajul/valoarea care se dorește a fi transmisă conține și date despre destinație.

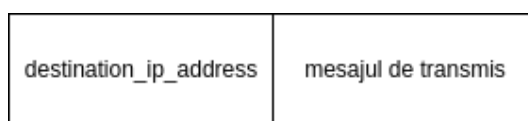


Figure 4: Format-ul payload-ului

Clasa **RelayNode**:

- această clasă va descrie comportamentul unui relay node, componenta ce va recepționa mesaje de la o singură sursă și va transmite mesaje spre o singură destinație sau hop
- având acest rol dublu de transmisie și recepție, o soluție imediată ar fi compunerea acestuia dintr-un **server** și un **client**. Partea de server se va ocupa de recepția mesajelor, iar partea de client va transmite mesaje spre următorul nod din rețea cum se observă în figura 2.
- pentru a putea rula simularea dintr-un singur program, în momentul creării unui relay node se va porni un thread, unde se așteaptă conexiunea unui client (hop anterior) la serverul relay-node-ului.
- în momentul în care un client se conectează la serverul relay-node-ului, se va porni automat un thread nou ce va executa bucla infinită de așteptare/recepție mesaje de la client.
- pentru scenariul problemei descrise mai sus, în momentul în care relay node-ul primește un mesaj de la hop-ul anterior, acesta va despacheta și analiza payload-ul primit și va verifica dacă mesajul primit este pentru el și atunci îl procesează sau în caz contrar îl trimite la următorul hop din rețea la care este conectat.
- Payload-ul este transmis sub formă de UTF, și are următorul format în cod : **destination_ip_address/value**. Astfel despachetarea payload-ului se va face în funcție de „/”. Splituind stringul cu separatorul „/” se vor obține cele două părți relevante ale payload-ului.

Clasa **TransferSimulation**:

- această clasă conține funcția **main** unde se dă flow-ul simulării
- se vor instanția 3 obiecte de tipul *RelayNode* astfel :
 - **destination 1**: ipAddress(127.0.0.1) ; PORT(5000); nextHop(127.0.0.2);
 - **destination 2**: ipAddress(127.0.0.2) ; PORT(5000); nextHop(127.0.0.3);
 - **destination 3**: ipAddress(127.0.0.3) ; PORT(5000); nextHop(NULL);

- la crearea unei destinații, se va porni un thread pentru aceasta, în care server-ul așteaptă conexiuni
- se va instanția un obiect de tipul **Sender**
 - **sender:** ipAddress(127.0.0.15); PORT(5000); nextHop(127.0.0.1)
- se ia o buclă for cu un contor **i** de la 0 la 100 și se va apela metoda **sendMessage** pe obiectul de tip sender cu următorii parametri: lsita de adrese destanție (D1,D2, D3 ip addresses) și valoarea curentă a lui i. Astfel, la fiecare pas al buclei *for*, sender-ul va trimite valoarea curentă a lui **i** la o destinație random dintre cele din lista transmisă, prin intermediul nexHop-ul atașat lui. Mesajul va fi transmis din hop în hop, până când unul dintre receptorii mesajului se identifică cu adresa ip specificată în payload

Observații:

- la terminarea simulării, se vor închide conexiunile și implicit se vor finaliza și toate thread-urile create
- pentru afișarea mesajelor de informare asupra stării simulării s-a folosit **Logger**;

4. Rezultate obținute

În continuare vor fi analizate în paralel mesajele de logging obținute în urma executării programului și ceea ce se capturează în **Wireshark**. S-a folosit maven pentru obținerea executabilului *RelayNodesConnection-1.0-SNAPSHOT.jar* cu versiunea de jdk17, iar din terminal a fost rulat programul de simulare cu **java -jar**.

- Observare creare noduri: cele 3 destinații și senderul

```
(base) birlutiuclaudiu@birlutiuclaudiu:~/Facultate/RC/RelayNodesConnection/target$ ls
classes generated-sources maven-archiver maven-status RelayNodesConnection-1.0-SNAPSHOT.jar
(base) birlutiuclaudiu@birlutiuclaudiu:~/Facultate/RC/RelayNodesConnection/target$ java -jar RelayNodesConnection-1.0-SNAPSHOT.jar
May 07, 2022 11:55:49 AM RelayNode <init>
INFO: Creating server with ipAdrees 127.0.0.1 and port 5000...
May 07, 2022 11:55:49 AM RelayNode <init>
INFO: Created server with ipAddress 127.0.0.1 and port 5000
May 07, 2022 11:55:49 AM RelayNode <init>
INFO: Creating server with ipAdrees 127.0.0.2 and port 5000...
May 07, 2022 11:55:49 AM RelayNode <init>
INFO: Created server with ipAddress 127.0.0.2 and port 5000
May 07, 2022 11:55:49 AM RelayNode <init>
INFO: Creating server with ipAdrees 127.0.0.3 and port 5000...
May 07, 2022 11:55:49 AM RelayNode <init>
INFO: Created server with ipAddress 127.0.0.3 and port 5000
May 07, 2022 11:55:49 AM Sender <init>
INFO: Created sender with ipAddress 127.0.0.15 and port 5000
May 07, 2022 11:55:49 AM RelayNode$MyThread run
```

Figure 5: Crearea destinațiilor și crearea sender-ului

- Crearea primei conexiuni între *sender* și *destination1*

```
INFO: Creating server with ipAdrees 127.0.0.3 and port 5000...
May 07, 2022 11:55:49 AM RelayNode <init>
INFO: Created server with ipAddress 127.0.0.3 and port 5000
May 07, 2022 11:55:49 AM Sender <init>
INFO: Created sender with ipAddress 127.0.0.15 and port 5000
May 07, 2022 11:55:49 AM RelayNode$MyThread run
INFO: 127.0.0.1 connected with client 127.0.0.15
May 07, 2022 11:55:49 AM Sender sendMessage
INFO: SEND THE PAYLOAD 127.0.0.1/0 to 127.0.0.1 FROM 127.0.0.15
```

Figure 6: Logging parte de conexiune sender-destination1

Pe captura din Wireshark se observă pașii mecanismului TCP pentru realizarea conexiunii celor două componente: clientul **sender**-ului cu serverul din **destination1**.

Pentru realizarea conexiunii este nevoie de acele pachete de acknowledgement (ACK). Primele trei schimburi de pachete reprezintă **3-way handshake** (figura 7) ce este necesar pentru realizarea conexiunii de tip TCP.

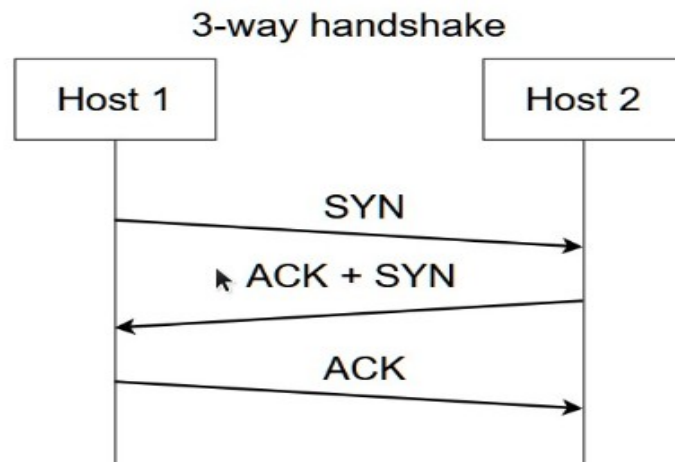


Figure 7: 3-way handshake pentru conexiuni TCP

simulationRealyNode.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.15	127.0.0.1	TCP	74	5000 → 5000 [SYN] Seq=0 Win=65483 Len=0 MSS=65495 SA=127.0.0.15 SD=127.0.0.1
2	0.000008552	127.0.0.1	127.0.0.15	TCP	74	5000 → 5000 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SA=127.0.0.1 SD=127.0.0.15
3	0.000015246	127.0.0.15	127.0.0.1	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=127.0.0.15 TSecr=127.0.0.1
4	0.005445323	127.0.0.15	127.0.0.1	TCP	79	5000 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13 MSS=65495 SA=127.0.0.15 SD=127.0.0.1
5	0.005453864	127.0.0.1	127.0.0.15	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=127.0.0.1 TSecr=127.0.0.15

Figure 8: Conexiunea clientului sender-ului cu serverul destinației 1

- transmiterea primei valori (0) la o destinație random din cele 3 disponibile. În cadrul simulării aduse în discuție, s-a ales aleator destinația 127.0.0.1

```

INFO: Created sender with ipAddress 127.0.0.15 and port 5000
May 07, 2022 11:55:49 AM RelayNode$MyThread run
INFO: 127.0.0.1 connected with client 127.0.0.15
May 07, 2022 11:55:49 AM Sender sendMessage
INFO: ----->SEND THE PAYLOAD 127.0.0.1/0 to 127.0.0.1 FROM 127.0.0.15
May 07, 2022 11:55:49 AM RelayNode onReceiveFromClient
INFO: ----->MESSAGE FOR ME (127.0.0.1) from 127.0.0.15 : 0
May 07, 2022 11:55:49 AM Sender sendMessage
INFO: ----->SEND THE PAYLOAD 127.0.0.3/1 to 127.0.0.1 FROM 127.0.0.15
May 07, 2022 11:55:49 AM RelayNode clientSendMessageToNextHop
INFO: ----->HOP FROM ME (127.0.0.1) to 127.0.0.2 FOR PAYLOAD: 127.0.0.3/1
May 07, 2022 11:55:49 AM RelayNode$MyThread run
INFO: 127.0.0.2 connected with client 127.0.0.1
May 07, 2022 11:55:49 AM RelayNode clientSendMessageToNextHop
INFO: ----->HOP FROM ME (127.0.0.2) to 127.0.0.3 FOR PAYLOAD: 127.0.0.3/1
May 07, 2022 11:55:49 AM RelayNode$MyThread run
INFO: 127.0.0.3 connected with client 127.0.0.2
May 07, 2022 11:55:49 AM RelayNode onReceiveFromClient
INFO: ----->MESSAGE FOR ME (127.0.0.3) from 127.0.0.2 : 1
May 07, 2022 11:55:50 AM Sender sendMessage
INFO: ----->SEND THE PAYLOAD 127.0.0.3/2 to 127.0.0.1 FROM 127.0.0.15
May 07, 2022 11:55:50 AM RelayNode clientSendMessageToNextHop
INFO: ----->HOP FROM ME (127.0.0.1) to 127.0.0.2 FOR PAYLOAD: 127.0.0.3/2
May 07, 2022 11:55:50 AM RelayNode clientSendMessageToNextHop
INFO: ----->HOP FROM ME (127.0.0.2) to 127.0.0.3 FOR PAYLOAD: 127.0.0.3/2
May 07, 2022 11:55:50 AM RelayNode onReceiveFromClient
INFO: ----->MESSAGE FOR ME (127.0.0.3) from 127.0.0.2 : 2
May 07, 2022 11:55:50 AM Sender sendMessage

```

Figure 9: Transmitere valori 0, 1 și 3

Valoarea 0 a fost transmisă spre *destination1*, valoarea 1 a fost transmisă spre *destination3* și la fel și valoarea 2.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.15	127.0.0.1	TCP	74	5000 → 5000 [SYN] Seq=0 Win=65483 Len=0 MSS=65495 S
2	0.000008552	127.0.0.1	127.0.0.15	TCP	74	5000 → 5000 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0
3	0.000015246	127.0.0.15	127.0.0.1	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=
4	0.005445323	127.0.0.15	127.0.0.1	TCP	79	5000 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
5	0.005453864	127.0.0.1	127.0.0.15	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
6	0.057279723	127.0.0.15	127.0.0.1	TCP	79	5000 → 5000 [PSH, ACK] Seq=14 Ack=1 Win=65536 Len=13
7	0.057296592	127.0.0.1	127.0.0.15	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=27 Win=65536 Len=0 TSval=
8	0.057857031	127.0.0.1	127.0.0.2	TCP	74	5001 → 5000 [SYN] Seq=0 Win=65483 Len=0 MSS=65495 S
9	0.057871218	127.0.0.2	127.0.0.1	TCP	74	5000 → 5001 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 M
10	0.057885092	127.0.0.1	127.0.0.2	TCP	66	5001 → 5000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=
11	0.059245742	127.0.0.1	127.0.0.2	TCP	79	5001 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
12	0.059255465	127.0.0.2	127.0.0.1	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
13	0.060524396	127.0.0.2	127.0.0.3	TCP	74	5001 → 5000 [SYN] Seq=0 Win=65483 Len=0 MSS=65495 S
14	0.060537159	127.0.0.3	127.0.0.2	TCP	74	5000 → 5001 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 M
15	0.060550050	127.0.0.2	127.0.0.3	TCP	66	5001 → 5000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=
16	0.061570230	127.0.0.2	127.0.0.3	TCP	79	5001 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
17	0.061578200	127.0.0.3	127.0.0.2	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
18	0.108989030	127.0.0.15	127.0.0.1	TCP	79	5000 → 5000 [PSH, ACK] Seq=27 Ack=1 Win=65536 Len=13

```
0000 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 .....E
0010 00 41 49 da 40 00 40 06 f2 cc 7f 00 00 0f 7f 00 ..AI@@...
0020 00 01 13 88 13 88 71 ef 79 8c 08 c3 bb 38 80 18 .....q y...8...
0030 02 00 fe 43 00 00 01 01 08 0a bd 43 91 d6 f6 ec .....K...
0040 4a 43 00 0b 31 32 37 2e 30 2e 30 2e 31 2f 30 ..JC..127.0.0.1/0
```

În figura 10, la linia 4 se poate observa trimiterea mesajului cu valoarea 0 spre destinația 1 (127.0.0.1). Mesajul este recepționat, iar la linia 5 este capturat acknowledgment-ul trimis spre sender (127.0.0.15). Transmiterea acestui payload se încheie aici, pentru că destinatarul este 127.0.0.1 .

No.	Time	Source	Destination	Protocol	Length	Info
4	0.005445323	127.0.0.15	127.0.0.1	TCP	79	5000 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
5	0.005453864	127.0.0.1	127.0.0.15	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
6	0.057278723	127.0.0.15	127.0.0.1	TCP	79	5000 → 5000 [PSH, ACK] Seq=14 Ack=1 Win=65536 Len=13
7	0.057296592	127.0.0.1	127.0.0.15	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=27 Win=65536 Len=0 TSval=
8	0.057857031	127.0.0.1	127.0.0.2	TCP	74	5001 → 5000 [SYN] Seq=0 Win=65483 Len=0 MSS=65495 S
9	0.057871218	127.0.0.2	127.0.0.1	TCP	74	5000 → 5001 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 M
10	0.057885092	127.0.0.1	127.0.0.2	TCP	66	5001 → 5000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=
11	0.059245742	127.0.0.1	127.0.0.2	TCP	79	5001 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
12	0.059255465	127.0.0.2	127.0.0.1	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
13	0.060524396	127.0.0.2	127.0.0.3	TCP	74	5001 → 5000 [SYN] Seq=0 Win=65483 Len=0 MSS=65495 S
14	0.060537159	127.0.0.3	127.0.0.2	TCP	74	5000 → 5001 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 M
15	0.060550050	127.0.0.2	127.0.0.3	TCP	66	5001 → 5000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=
16	0.061570230	127.0.0.2	127.0.0.3	TCP	79	5001 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
17	0.061578200	127.0.0.3	127.0.0.2	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
18	0.108989030	127.0.0.15	127.0.0.1	TCP	79	5000 → 5000 [PSH, ACK] Seq=27 Ack=1 Win=65536 Len=13
19	0.109004498	127.0.0.1	127.0.0.15	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=40 Win=65536 Len=0 TSval=
20	0.110211722	127.0.0.1	127.0.0.2	TCP	79	5001 → 5000 [PSH, ACK] Seq=14 Ack=1 Win=65536 Len=13
21	0.110221062	127.0.0.2	127.0.0.1	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=27 Win=65536 Len=0 TSval=

> Frame 6: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface lo, id 0
 > Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
 > Internet Protocol Version 4, Src: 127.0.0.15, Dst: 127.0.0.1
 > Transmission Control Protocol, Src Port: 5000, Dst Port: 5000, Seq: 14, Ack: 1, Len: 13
 > Data (13 bytes)

```

0000  00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  .....E.
0010  00 41 49 db 40 00 40 06 f2 cb 7f 00 00 9f 7f 00  ..A!_@_@_
0020  00 01 13 08 13 88 71 ef 79 09 08 c3 bb 38 00 18  ...-q-y...K.
0030  02 00 fe 43 00 00 01 01 08 0a bd 4b 92 0a f6 ec  ...C.....8...
0040  4a 49 00 0b 31 32 37 2e 30 2e 30 2e 33 2f 31  ..J!..127.0.0.3/1
  
```

Figure 11: Transmite payload 127.0.0.3/1

11	0.059245742	127.0.0.1	127.0.0.2	TCP	79	5001 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
12	0.059255465	127.0.0.2	127.0.0.1	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
13	0.060524396	127.0.0.2	127.0.0.3	TCP	74	5001 → 5000 [SYN] Seq=0 Win=65483 Len=0 MSS=65495 S
14	0.060537159	127.0.0.3	127.0.0.2	TCP	74	5000 → 5001 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 M
15	0.060550050	127.0.0.2	127.0.0.3	TCP	66	5001 → 5000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=
16	0.061570230	127.0.0.2	127.0.0.3	TCP	79	5001 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
17	0.061578200	127.0.0.3	127.0.0.2	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
18	0.108989030	127.0.0.15	127.0.0.1	TCP	79	5000 → 5000 [PSH, ACK] Seq=27 Ack=1 Win=65536 Len=13
19	0.109004498	127.0.0.1	127.0.0.15	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=40 Win=65536 Len=0 TSval=
20	0.110211722	127.0.0.1	127.0.0.2	TCP	79	5001 → 5000 [PSH, ACK] Seq=14 Ack=1 Win=65536 Len=13
21	0.110221062	127.0.0.2	127.0.0.1	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=27 Win=65536 Len=0 TSval=

> Frame 11: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface lo, id 0
 > Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
 > Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.2
 > Transmission Control Protocol, Src Port: 5001, Dst Port: 5000, Seq: 1, Ack: 1, Len: 13
 > Data (13 bytes)

```

0000  00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  .....E.
0010  00 41 2c 03 40 00 40 06 19 b1 7f 00 00 91 7f 00  ..A!_@_@_
0020  00 02 13 09 13 88 09 15 9d 71 a7 28 a3 70 00 18  ...-q-(p...
0030  02 00 fe 36 00 00 01 01 08 0a 16 d9 03 8c 0e ad  ...6.....q...
0040  69 fa 00 0b 31 32 37 2e 30 2e 30 2e 33 2f 31  ..i...127.0.0.3/1
  
```

Figure 12: Transmite 127.0.0.3/1 din hopul 127.0.0.1 la 127.0.0.2

11	0.059245742	127.0.0.1	127.0.0.2	TCP	79	5001 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
12	0.059255465	127.0.0.2	127.0.0.1	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
13	0.060524396	127.0.0.2	127.0.0.3	TCP	74	5001 → 5000 [SYN] Seq=0 Win=65483 Len=0 MSS=65495 S
14	0.060537159	127.0.0.3	127.0.0.2	TCP	74	5000 → 5001 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 M
15	0.060550050	127.0.0.2	127.0.0.3	TCP	66	5001 → 5000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=
16	0.061570230	127.0.0.2	127.0.0.3	TCP	79	5001 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=13
17	0.061578200	127.0.0.3	127.0.0.2	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=14 Win=65536 Len=0 TSval=
18	0.108989030	127.0.0.15	127.0.0.1	TCP	79	5000 → 5000 [PSH, ACK] Seq=27 Ack=1 Win=65536 Len=13
19	0.109004498	127.0.0.1	127.0.0.15	TCP	66	5000 → 5000 [ACK] Seq=1 Ack=40 Win=65536 Len=0 TSval=
20	0.110211722	127.0.0.1	127.0.0.2	TCP	79	5001 → 5000 [PSH, ACK] Seq=14 Ack=1 Win=65536 Len=13
21	0.110221062	127.0.0.2	127.0.0.1	TCP	66	5000 → 5001 [ACK] Seq=1 Ack=27 Win=65536 Len=0 TSval=

> Frame 16: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface lo, id 0
 > Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
 > Internet Protocol Version 4, Src: 127.0.0.2, Dst: 127.0.0.3
 > Transmission Control Protocol, Src Port: 5001, Dst Port: 5000, Seq: 1, Ack: 1, Len: 13
 > Data (13 bytes)

```

0000  00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  .....E.
0010  00 41 6c ca 40 00 40 06 cf e7 7f 00 00 02 7f 00  ..A!_@_@_
0020  00 03 13 09 13 88 1d 08 57 f6 a1 6b ed 69 00 18  ...-W.k.i...
0030  02 00 fe 38 00 00 01 01 08 0a e3 5b c0 dd a4 81  ...8.....[....
0040  2e e3 00 0b 31 32 37 2e 30 2e 30 2e 33 2f 31  ...127.0.0.3/1
  
```

Figure 13: Transmite 127.0.0.3/1 din hopul 127.0.0.2 la 127.0.0.3

Mesajul a ajuns la destinație și atunci nu mai este trimis mai departe. Se va trimite un nou payload de la 127.0.0.15 la o altă destinație aleasă aleator.

- ```
INFO: ----->SEND THE PAYLOAD 127.0.0.1/98 to 127.0.0.1 FROM 127.0.0.15
May 07, 2022 11:55:54 AM RelayNode onReceiveFromClient
INFO: ----->MESSAGE FOR ME (127.0.0.1) from 127.0.0.15 : 98
May 07, 2022 11:55:55 AM Sender sendMessage
INFO: ----->SEND THE PAYLOAD 127.0.0.3/99 to 127.0.0.1 FROM 127.0.0.15
May 07, 2022 11:55:55 AM RelayNode clientSendMessageToNextHop
INFO: ----->HOP FROM ME (127.0.0.1) to 127.0.0.2 FOR PAYLOAD: 127.0.0.3/99
May 07, 2022 11:55:55 AM RelayNode clientSendMessageToNextHop
INFO: ----->HOP FROM ME (127.0.0.2) to 127.0.0.3 FOR PAYLOAD: 127.0.0.3/99
May 07, 2022 11:55:55 AM RelayNode onReceiveFromClient
INFO: ----->MESSAGE FOR ME (127.0.0.3) from 127.0.0.2 : 99
May 07, 2022 11:55:55 AM RelayNode close
INFO: Closing threads
May 07, 2022 11:55:55 AM RelayNode close
INFO: Closed threads
May 07, 2022 11:55:55 AM RelayNode close
INFO: Closing threads
May 07, 2022 11:55:55 AM RelayNode close
INFO: Closed threads
May 07, 2022 11:55:55 AM RelayNode close
INFO: Closing threads
May 07, 2022 11:55:55 AM RelayNode close
INFO: Closed threads
(base) birlutiucclaudiu@birlutiucclaudiu:~/Facultate/RC/RelayNodesConnection/target$
```

|  |     |             |            |            |     |                                                          |
|--|-----|-------------|------------|------------|-----|----------------------------------------------------------|
|  | 423 | 5.039100848 | 127.0.0.1  | 127.0.0.15 | TCP | 66 5000 → 5000 [ACK] Seq=1 Ack=1377 Win=65536 Len=0      |
|  | 424 | 5.089608347 | 127.0.0.15 | 127.0.0.1  | TCP | 80 5000 → 5000 [PSH, ACK] Seq=1377 Ack=1 Win=65536 Len=0 |
|  | 425 | 5.089775125 | 127.0.0.1  | 127.0.0.15 | TCP | 66 5000 → 5000 [ACK] Seq=1 Ack=1391 Win=65536 Len=0      |
|  | 426 | 5.090145624 | 127.0.0.1  | 127.0.0.2  | TCP | 80 5001 → 5000 [PSH, ACK] Seq=973 Ack=1 Win=65536 Len=0  |
|  | 427 | 5.090171505 | 127.0.0.2  | 127.0.0.1  | TCP | 66 5000 → 5001 [ACK] Seq=1 Ack=987 Win=65536 Len=0       |
|  | 428 | 5.090473990 | 127.0.0.2  | 127.0.0.3  | TCP | 80 5001 → 5000 [PSH, ACK] Seq=486 Ack=1 Win=65536 Len=0  |
|  | 429 | 5.090478944 | 127.0.0.3  | 127.0.0.2  | TCP | 66 5000 → 5001 [ACK] Seq=1 Ack=500 Win=65536 Len=0       |
|  | 430 | 5.140152980 | 127.0.0.15 | 127.0.0.1  | TCP | 66 5000 → 5000 [FIN, ACK] Seq=1391 Ack=1 Win=65536 Len=0 |
|  | 431 | 5.141986909 | 127.0.0.2  | 127.0.0.3  | TCP | 66 5001 → 5000 [FIN, ACK] Seq=500 Ack=1 Win=65536 Len=0  |
|  | 432 | 5.142582379 | 127.0.0.1  | 127.0.0.2  | TCP | 66 5001 → 5000 [FIN, ACK] Seq=987 Ack=1 Win=65536 Len=0  |
|  | 433 | 5.152379630 | 127.0.0.3  | 127.0.0.2  | TCP | 66 5000 → 5001 [FIN, ACK] Seq=1 Ack=501 Win=65536 Len=0  |
|  | 434 | 5.152397684 | 127.0.0.2  | 127.0.0.3  | TCP | 66 5001 → 5000 [ACK] Seq=501 Ack=2 Win=65536 Len=0       |
|  | 435 | 5.152424199 | 127.0.0.2  | 127.0.0.1  | TCP | 66 5000 → 5001 [FIN, ACK] Seq=1 Ack=988 Win=65536 Len=0  |
|  | 436 | 5.152428741 | 127.0.0.1  | 127.0.0.2  | TCP | 66 5001 → 5000 [ACK] Seq=988 Ack=2 Win=65536 Len=0       |

```
> Frame 424: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.15, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 5000, Dst Port: 5000, Seq: 1377, Ack: 1, Len: 14
> Data (14 bytes)
```

```
0000 00 00 00 00 00 00 00 00 00 00 00 08 00 00 45 00 E..
0010 00 42 4a 0e 40 00 00 00 f2 68 7f 00 00 0f 7f 00 .BJ=@...h....
0020 00 01 13 8b 13 8b 71 ef 00 c8 08 c3 b3 38 00 18 q...8...
0030 02 00 fe 44 00 00 01 01 08 0a bd 4b a5 b2 f6 ec ...D.....K...
0040 5d f2 00 0c 31 32 37 2e 30 2e 30 2e 33 2f 39 39 l...127.0.0.3/99
```

| No. | Time        | Source     | Destination | Protocol | Length | Info                                                                |
|-----|-------------|------------|-------------|----------|--------|---------------------------------------------------------------------|
| 423 | 5.089108848 | 127.0.0.1  | 127.0.0.15  | TCP      | 66     | 5000 → 5000 [ACK] Seq=1 Ack=1377 Win=65536 Len=0 TSv=665000         |
| 424 | 5.089668347 | 127.0.0.15 | 127.0.0.1   | TCP      | 66     | 80 5000 → 5000 [PSH, ACK] Seq=1377 Ack=1 Win=65536 Len=0 TSv=665000 |
| 425 | 5.089775125 | 127.0.0.1  | 127.0.0.15  | TCP      | 66     | 5000 → 5000 [ACK] Seq=1 Ack=1391 Win=65536 Len=0 TSv=665000         |
| 426 | 5.090145624 | 127.0.0.1  | 127.0.0.2   | TCP      | 66     | 5001 → 5000 [PSH, ACK] Seq=973 Ack=1 Win=65536 Len=0 TSv=665000     |
| 427 | 5.090171505 | 127.0.0.2  | 127.0.0.1   | TCP      | 66     | 5000 → 5001 [ACK] Seq=1 Ack=987 Win=65536 Len=0 TSv=665000          |
| 428 | 5.090473990 | 127.0.0.2  | 127.0.0.3   | TCP      | 66     | 5001 → 5000 [PSH, ACK] Seq=486 Ack=1 Win=65536 Len=0 TSv=665000     |
| 429 | 5.090478944 | 127.0.0.3  | 127.0.0.2   | TCP      | 66     | 5000 → 5001 [ACK] Seq=1 Ack=500 Win=65536 Len=0 TSv=665000          |
| 430 | 5.140152980 | 127.0.0.15 | 127.0.0.1   | TCP      | 66     | 5000 → 5000 [FIN, ACK] Seq=1391 Ack=1 Win=65536 Len=0 TSv=665000    |
| 431 | 5.141986909 | 127.0.0.2  | 127.0.0.3   | TCP      | 66     | 5001 → 5000 [FIN, ACK] Seq=500 Ack=1 Win=65536 Len=0 TSv=665000     |
| 432 | 5.142582379 | 127.0.0.1  | 127.0.0.2   | TCP      | 66     | 5001 → 5000 [FIN, ACK] Seq=987 Ack=1 Win=65536 Len=0 TSv=665000     |
| 433 | 5.152379630 | 127.0.0.3  | 127.0.0.2   | TCP      | 66     | 5000 → 5001 [FIN, ACK] Seq=1 Ack=501 Win=65536 Len=0 TSv=665000     |
| 434 | 5.152397684 | 127.0.0.2  | 127.0.0.3   | TCP      | 66     | 5001 → 5000 [ACK] Seq=501 Ack=2 Win=65536 Len=0 TSv=665000          |
| 435 | 5.152424199 | 127.0.0.2  | 127.0.0.1   | TCP      | 66     | 5000 → 5001 [FIN, ACK] Seq=1 Ack=988 Win=65536 Len=0 TSv=665000     |
| 436 | 5.152428741 | 127.0.0.1  | 127.0.0.2   | TCP      | 66     | 5001 → 5000 [ACK] Seq=988 Ack=2 Win=65536 Len=0 TSv=665000          |

```

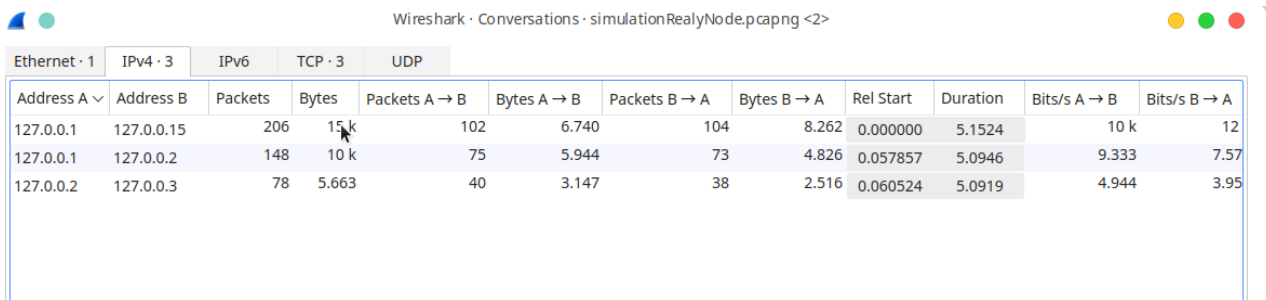
> Frame 428: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.2, Dst: 127.0.0.3
> Transmission Control Protocol, Src Port: 5001, Dst Port: 5000, Seq: 486, Ack: 1, Len: 14
> Data (14 bytes)
0000 00 00 00 00 00 00 00 00 00 00 00 00 45 00 ..BL.@.....E-
0010 00 42 6c ed 40 00 00 00 cf c3 7f 00 00 02 7f 00 ..B.L@.....
0020 00 03 13 39 00 00 00 00 a1 6b ed 69 80 18 00 00 Y.k-
0030 02 00 fe 39 00 00 01 01 08 0a e3 5b 4d 82 a4 81 ..9.....
0040 41 58 00 0c 31 32 37 2e 30 2e 30 2e 33 2f 39 39 AX..127.0.0.3/99

```

10

## 5. Analiză WireShark

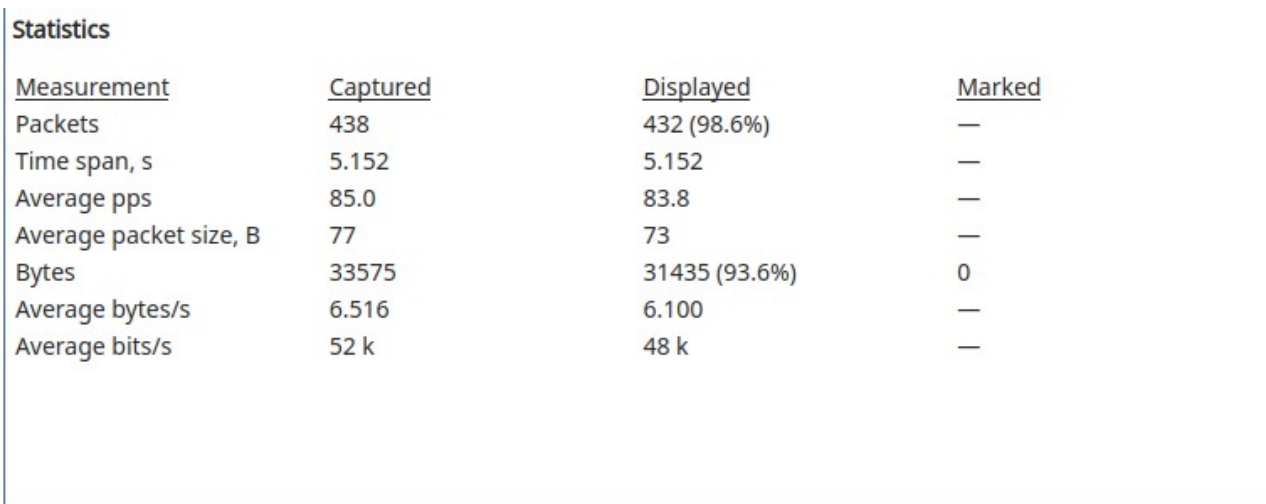
Se vor discuta în continuare rezultatele statistice obținute în urma capturii.



| Address A | Address B  | Packets | Bytes | Packets A → B | Bytes A → B | Packets B → A | Bytes B → A | Rel Start | Duration | Bits/s A → B | Bits/s B → A |
|-----------|------------|---------|-------|---------------|-------------|---------------|-------------|-----------|----------|--------------|--------------|
| 127.0.0.1 | 127.0.0.15 | 206     | 15 k  | 102           | 6.740       | 104           | 8.262       | 0.000000  | 5.1524   | 10 k         | 12           |
| 127.0.0.1 | 127.0.0.2  | 148     | 10 k  | 75            | 5.944       | 73            | 4.826       | 0.057857  | 5.0946   | 9.333        | 7.57         |
| 127.0.0.2 | 127.0.0.3  | 78      | 5.663 | 40            | 3.147       | 38            | 2.516       | 0.060524  | 5.0919   | 4.944        | 3.95         |

Figure 17: Vizualizare statistici comunicare

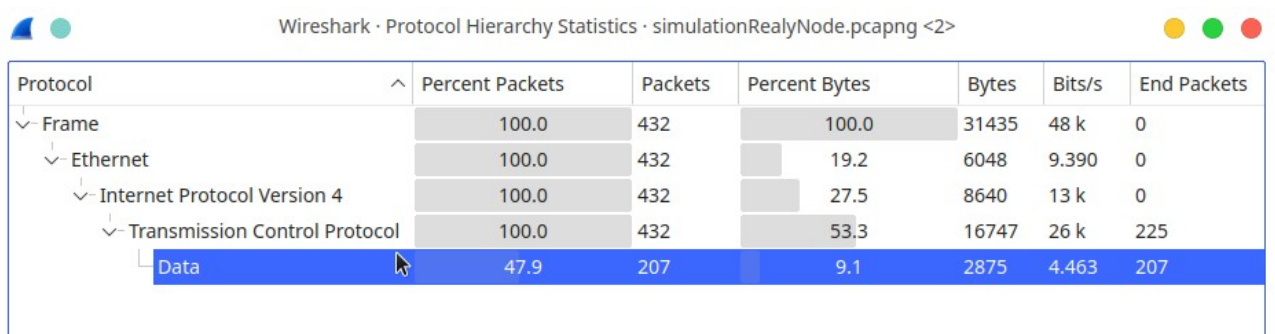
În figura 17 se observă statistica transferurilor de pachete între cele 4 componente ale rețelei. Se observă un număr total de 432 de pachete ( $15360 + 10240 + 5663 = 31263 \text{ b} = 30,53 \text{ kB}$ ).



| Measurement            | Captured | Displayed     | Marked |
|------------------------|----------|---------------|--------|
| Packets                | 438      | 432 (98.6%)   | —      |
| Time span, s           | 5.152    | 5.152         | —      |
| Average pps            | 85.0     | 83.8          | —      |
| Average packet size, B | 77       | 73            | —      |
| Bytes                  | 33575    | 31435 (93.6%) | 0      |
| Average bytes/s        | 6.516    | 6.100         | —      |
| Average bits/s         | 52 k     | 48 k          | —      |

Figure 18: Wireshark -> Statistics -> Capture File Properties

**Observație:** din cele 438 de pachete 6 pachete sunt UDP (127.0.0.1 -> 127.0.0.1)



| Protocol                      | Percent Packets | Packets | Percent Bytes | Bytes | Bits/s | End Packets |
|-------------------------------|-----------------|---------|---------------|-------|--------|-------------|
| Frame                         | 100.0           | 432     | 100.0         | 31435 | 48 k   | 0           |
| Ethernet                      | 100.0           | 432     | 19.2          | 6048  | 9.390  | 0           |
| Internet Protocol Version 4   | 100.0           | 432     | 27.5          | 8640  | 13 k   | 0           |
| Transmission Control Protocol | 100.0           | 432     | 53.3          | 16747 | 26 k   | 225         |
| Data                          | 47.9            | 207     | 9.1           | 2875  | 4.463  | 207         |

Figure 19: Data distributed (payload len)

Raportul dintre totalul de bytes livrați și traficul relevant al aplicației este dat de dimensiunea totală datelor în raport cu totalul dimensiunii frame-urilor transmise.  $2875/31435 = 0.091 \Rightarrow 9\%$

| No. | Time        | Source     | Destination | Protocol | Length | Info                                                |
|-----|-------------|------------|-------------|----------|--------|-----------------------------------------------------|
| 35  | 0.265738358 | 127.0.0.2  | 127.0.0.1   | TCP      | 66     | 5000 → 5001 [ACK] Seq=1 Ack=53 Win=65536 Len=0      |
| 36  | 0.266998387 | 127.0.0.2  | 127.0.0.3   | TCP      | 79     | 5001 → 5000 [PSH, ACK] Seq=40 Ack=1 Win=65536 Len=0 |
| 37  | 0.267008483 | 127.0.0.3  | 127.0.0.2   | TCP      | 66     | 5000 → 5001 [ACK] Seq=1 Ack=53 Win=65536 Len=0      |
| 38  | 0.315967685 | 127.0.0.15 | 127.0.0.1   | TCP      | 79     | 5000 → 5000 [PSH, ACK] Seq=79 Ack=1 Win=65536 Len=0 |
| 39  | 0.315982740 | 127.0.0.1  | 127.0.0.15  | TCP      | 66     | 5000 → 5000 [ACK] Seq=1 Ack=92 Win=65536 Len=0      |
| 40  | 0.317324755 | 127.0.0.1  | 127.0.0.2   | TCP      | 79     | 5001 → 5000 [PSH, ACK] Seq=53 Ack=1 Win=65536 Len=0 |
| 41  | 0.317334105 | 127.0.0.2  | 127.0.0.1   | TCP      | 66     | 5000 → 5001 [ACK] Seq=1 Ack=66 Win=65536 Len=0      |
| 42  | 0.367779986 | 127.0.0.15 | 127.0.0.1   | TCP      | 79     | 5000 → 5000 [PSH, ACK] Seq=92 Ack=1 Win=65536 Len=0 |
| 43  | 0.367794810 | 127.0.0.1  | 127.0.0.15  | TCP      | 66     | 5000 → 5000 [ACK] Seq=1 Ack=105 Win=65536 Len=0     |
| 44  | 0.369038572 | 127.0.0.1  | 127.0.0.2   | TCP      | 79     | 5001 → 5000 [PSH, ACK] Seq=66 Ack=1 Win=65536 Len=0 |
| 45  | 0.369048318 | 127.0.0.2  | 127.0.0.1   | TCP      | 66     | 5000 → 5001 [ACK] Seq=1 Ack=79 Win=65536 Len=0      |

> Frame 38: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface lo, id 0

> Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)

> Internet Protocol Version 4, Src: 127.0.0.15, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 5000, Dst Port: 5000, Seq: 79, Ack: 1, Len: 13

> Data (13 bytes)

```

0000 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 E-
0010 00 41 49 e0 40 00 40 06 f2 c6 7f 00 00 0f 7f 00 -AI_@_@_.....
0020 00 01 13 88 13 88 71 ef 79 da 08 c3 bb 38 80 18 -...q.y...8...
0030 02 00 fe 43 00 00 01 01 08 0a bd 4b 93 0c f6 ec -...C...K...
0040 4b 4c 00 0b 31 32 37 2e 30 2e 30 2e 32 2f 36 -KL-127.0.0.2/6

```

Transmission Control Protocol (tcp), 32 bytes

Packets: 438 · Displayed: 438 (100.0%)

Profile: Default

[illegible]

12



### Cele mai importante elemente ale header-ului:

**Source port: 5000** -> este portul clientului, poru celui care trimite mesajul; în cazzul nostru a sender- ului

**Destination port: 5000** -> reprezintă portul destinației, în cazul nostru e tot 5000 dar la adresa ip 127.0.0.1 ( e serverul destinației D1)

**Sequence number: 79** -> deoarece SYN flag e 0, atunci acesta este numărul de secvență acumulat al primului octet de date al acestui pachet pentru sesiunea curentă.

**Acknowledgment number: 1** -> deoarece ACK este setat (pe 1) valoarea acestui câmp reprezintă următorul număr de secvență pe care îl așteaptă receptorul

**Window size value : 512** -> dimensiunea ferestrei de primire, care specifică numărul de octeți (dincolo de numărul de secvență din câmpul de confirmare) pe care expeditorul acestui segment este dispus în prezent să îi primească

**Checksum: 0xfe43** -> utilizat pentru verificarea erorilor antetului și datelor

**Urgent pointer: 0** -> dacă steag-ul URG este setat, atunci acest câmp de 16 biți este un decalaj față de numărul de secvență care indică ultimul octet de date urgente.

**Options :** lungimea acestui câmp este determinată de câmpul de compensare a datelor

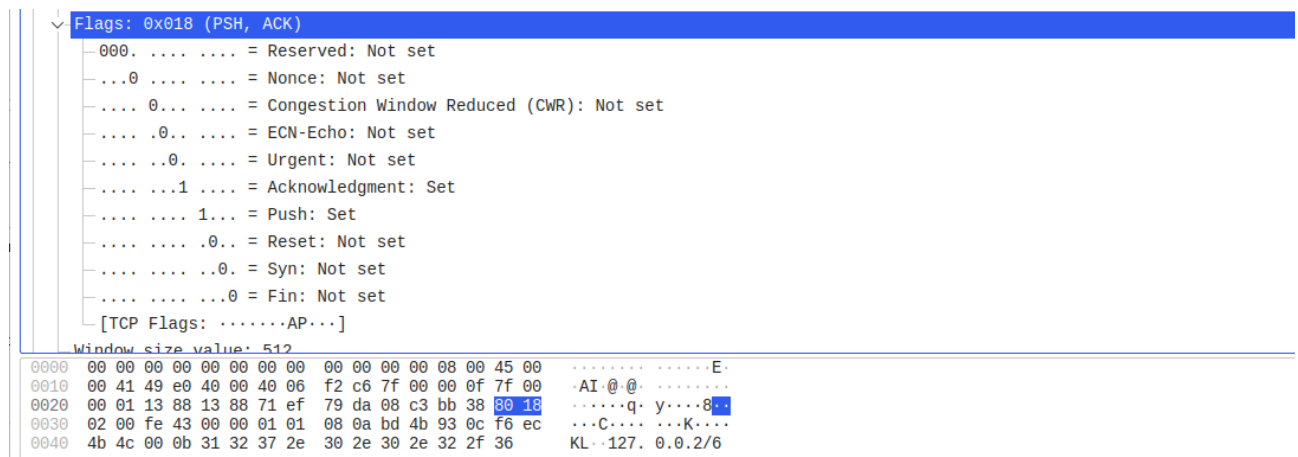


Figure 22: Flag-uri

Dintre flag-uri vom aminti :

**ACK** – indică faptul că field-ul Acknowledgement este semnificativ. Toate pachetele de după pachetul SYN inițial trimis de client ar trebui să aibă acest flag setat.

**PSH** – Funcție Push. Solicită împingerea datelor din buffer către aplicația de primire.

**RST (1 bit)** – Resetați conexiunea

**SYN** – Sincronizează numerele de secvență. Doar primul pachet trimis de la fiecare capăt ar trebui să aibă acest flag setat. Alte flag-uri își schimbă semnificația în funcție de acest flag, iar unele sunt valabile numai atunci când este setat, iar altele când este nesetat(pe 0).

**FIN** – Nu mai sunt date de la expeditor

*Lungimea header-ului este de 32 bytes. Sunt 13 bytes de data.*

## **6. Concluzii**

Implementarea acestei simulări de rețea formată din relay nodes a reprezentat un prilej bun de familiarizarea cu conceptele de Transmission Control Protocol (TCP) și înțelegere a modalității de comunicare între nodurile rețelei bazate relația client-server. Pentru cel care a dezvoltat aplicația, a fost provocare nouă ce a presupus o înțelegere mai în amănunt a arhitecturii client-server implementată în limbajul de programare JAVA.