MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
CATEDRA CALCULATOARE

# Tema 3
# -Rețele de calculatoare-

Student:
Pop Ruxandra Maria
Anul 3, grupa 30236

# Cuprins

# 1.Enunțul problemei

Să se realizeze într-un limbaj de programare scenariu de simulare pentru topologia de mai jos.Testarea funcționalității se va face cu ajutorul tool-ului WireShark.


Fig 1.1 Scenariu de simulare

## Specificații:

- Sunt 4 noduri prezente , sender-ul și 3 posibile destinații (D1, D2, D3).
- Nodul Sender transmite 100 de pachete care conține un număr random care va fi transmis la unul din cele 4 destinații. În cazul de față este un numar întreg care va fi incrementat la fiecare recepție, astfel va fi mai ușor de urmărit dacă cele 100 de pachete au fost transmise.
- Fiecare nod poate trimite date doar la o singură destinație/hop la care este conectat.

# 2.Descrierea soluției

Această topologie am ales să o implementez în Java. După cum se poate observa obiectul RelayNode, are 2 instanțe Client și Server. Obiectul Client va transmite mesajul spre următorul nod, spre server-ul corespunzător lui, în timp ce obiectul Server va receptă mesajele primite de la RelayNode-ul dinînainte.
Am folosit librăria java.net, pentru a creea Socket-ul din Client, și ServerSocket-ul, repesctiv Socket-ul din Server.

# 3.Analiză clase implementate

După cum se poate observa în figura 3.1, pentru realizarea topologiei am implementat 5 clase cu nume sugestive.

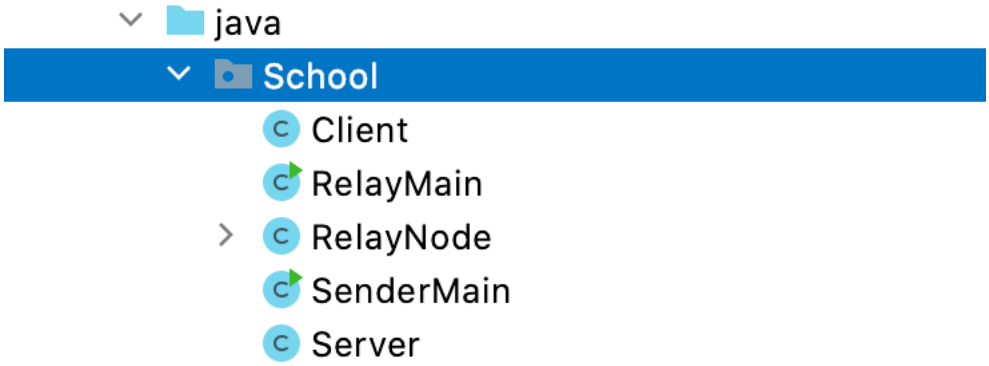


Fig 3.1 Clasele din Java

Conexiuniile și relațiile dintre ele pot fi observate în imaginea de mai jos care specifică diagrama de clase .

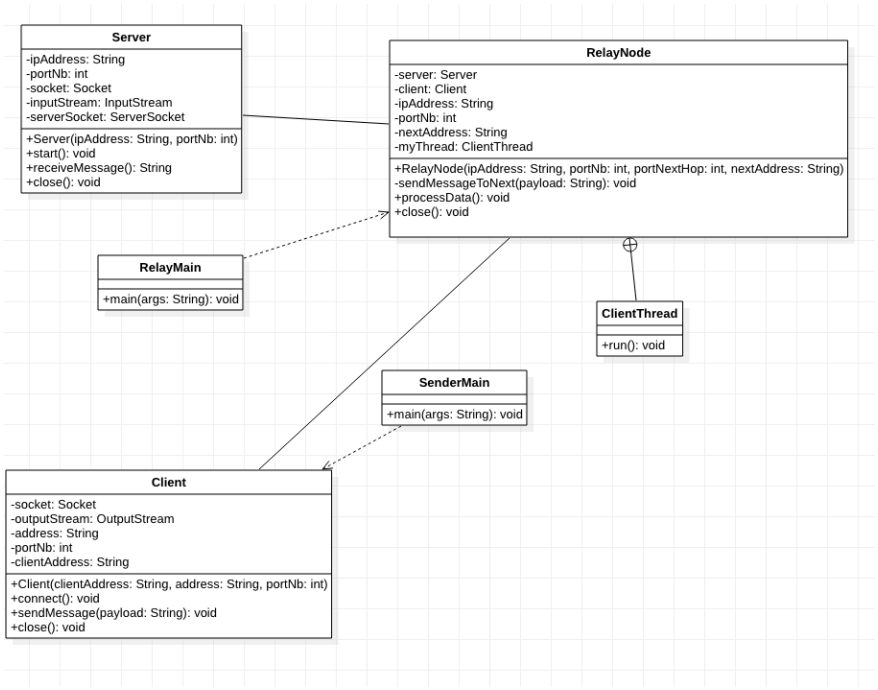


Fig 3.2 Diagrama de clase

Clasa RelayNode:

- Descrie comportamentul unui relay node, și anume un obiect de tip relay node va transmite mesaje spre o singură destinație sau hop și va recepționa mesaje doar de la o sursă, de la relay node-ul dinînainte.
- Datorită faptului ca are un rol dublu (de transmisie și recepție), prezintă 2 atribute de tip Client (pentru transmisie) și Server (pentru recepție).
- Totodată mai prezintă un atribut de tipul ClientThread care are ca scop să astepte să se facă conexiunea unui client (precedent) la server-ul său.Un thread din clasa ClientThread va executa o bucla infinită de recepție/transmisie de pachete, până când toate cele 100 pachete vor ajunge la destinațiile corecte.
- Metoda processData, are rolul, după cum ii sugerează și numele, de a analiza payload-ul primit de RelayNode, pentru a verifica dacă mesajul primit este pentru el, iar dacă este pentru el îl procesează, altfel îl v-a transmite la urmatorul hop, și acest lucru se repetă până când pachetul ajunge la destinația corespunzătoare.
- Metoda sendMessageToNext, specifică ca mesajul va fi transmis la urmatorul hop. Prin intermediul clientului v-a reusi să-l transmită.
- Payload-ul respectă forma impusă și anume IpAdress / Value.

Clasa Client:

- Are ca scop transiterea informaților de un nod la altul.
- Prezintă un atribut de tipul Socket care se conectează la portul server-ului de la adresa address și care ascultă pe portul portNb.
- Metoda sendMessage va transmite mesajul la urmatorul hop.

Clasa Server:

- Are ca scop recepția mesajelor primite de la nodul anterior.
- Metoda receiveMessage îndeplineste rolul acestei clase, și anume rolul de receptor. Odată ce mesajul a fost recepționat de destinația corespunzătoare, numaărul pachetului se va incrementa cu 1.

! În clasa RelayMain am creat cele 3 destinații specificate în capitolul 1.

```
new RelayNode( ipAddress: "127.0.0.1",  portNb: 5001,  portNextHop: 5002, nextAddress: "127.0.0.2");
new RelayNode( ipAddress: "127.0.0.2",  portNb: 5002,  portNextHop: 5003, nextAddress: "127.0.0.3");
new RelayNode( ipAddress: "127.0.0.3",  portNb: 5003,  portNextHop: 5004, nextAddress: null);
```

Fig 3.3 Clasa RelayMain

Fiecare astfel de nod, are adresa corespunzptoare lui, numărul portului, numarul următorului hop (și anume numele server-ului) și mai prezintă și adresa următorului nod (adresa clientului).

Clasa SenderMain, după cum ii spune și numele, se ocupă de transmiterea mesajului de la Sender-ul programului, care are ipAddress egal cu "127.0.0.15" la Server-ul cu adresa "127.0.0.1", de aici începând transmiterea celor 100 de pachete.

# 5.Verificare rezultate obținute

În acest capitol, voi compară rezultate afișate în consola aplicației în care s-a scris codul pentru proiect și ceea ce se capturează cu ajutorul tool-ului Wireshark.

!Mesajele în consolă au fost afisate cu System.out.println();

- Se poate observa ca Sende-ul a fost creat cu succes

```
Sender with address 127.0.0.15 was connectec
```

Fig 5.2 Mesaj consola creare Sender

- Observare ca cele 100 pachete au fost transmise.

```
127.0.0.2/0     127.0.0.2/89
127.0.0.3/1     127.0.0.2/90
127.0.0.2/2     127.0.0.3/91
127.0.0.3/3     127.0.0.3/92
127.0.0.2/4     127.0.0.3/93
127.0.0.2/5     127.0.0.2/94
127.0.0.2/6     127.0.0.3/95
127.0.0.1/7     127.0.0.2/96
127.0.0.3/8     127.0.0.3/97
127.0.0.3/9     127.0.0.1/98
127.0.0.3/10    127.0.0.1/99
```

Fig 5.2 Mesaj consolă transmitere pachete

- Observare mesajele intermediale despre felul în care au fost transmise pachetele.

```
Hop from 127.0.0.1 to 127.0.0.2 for 127.0.0.2/0
Packet (0) for 127.0.0.2 from 127.0.0.1
Hop from 127.0.0.1 to 127.0.0.2 for 127.0.0.3/1
Hop from 127.0.0.2 to 127.0.0.3 for 127.0.0.3/1
Packet (1) for 127.0.0.3 from 127.0.0.2
Hop from 127.0.0.1 to 127.0.0.2 for 127.0.0.2/2
Packet (2) for 127.0.0.2 from 127.0.0.1
Hop from 127.0.0.1 to 127.0.0.2 for 127.0.0.3/3
Hop from 127.0.0.2 to 127.0.0.3 for 127.0.0.3/3
Packet (3) for 127.0.0.3 from 127.0.0.2
Hop from 127.0.0.1 to 127.0.0.2 for 127.0.0.2/4
```

Fig 5.3 Mesaj consola  intermediale

Acum vom trece în aplicația Wireshark, să vedem dacă rezultatele observate în consolă au fost corecte și dacă s-au transmis corespunzător.

| | |
|---|---|
| 127.0.0.15 | 127.0.0.1 |
| 127.0.0.1 | 127.0.0.15 |
| 127.0.0.1 | 127.0.0.2 |
| 127.0.0.2 | 127.0.0.1 |
| 127.0.0.15 | 127.0.0.1 |

Fig 5.4 Transmisie payload 127.0.0.2/0

După cum se poate observa Senderul 127.0.0.15  va transmite pachetul spre hop-ul 127.0.0.1, pe urmă acesta transmițându-l spre nodul destinație și anume spre nodul cu adresa 127.0.0.2, după cum se poate observa si in figura Fig 5.3

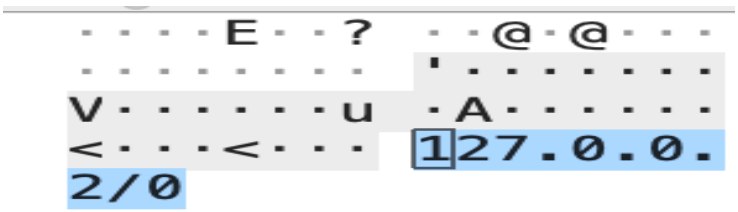


Fig 5.5 Recepție payload 127.0.0.2/0

Nodul cu adresa 127.0.0.2 a receptionat pachetul 0.Vezi figura Fig 5.1


Fig 5.6 Transmisie payload 127.0.0.3/1

După cum se poate observa Senderul 127.0.0.15 va transmite pachetul spre hop-ul 127.0.0.1, iar 127.0.0.1 il va transmite mai departe spre 127.0.0.2,iar clientul corespunzător nodului 127.0.0.2 îl va transmite spre 127.0.0.3,care il va recepționa după cum se poate observa si in figura Fig 5.3

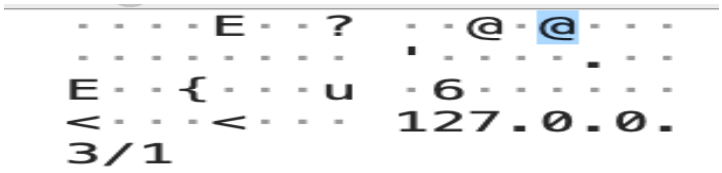


Fig 5.7 Recepție payload 127.0.0.3/1

Nodul cu adresa 127.0.0.3 a receptionat pachetul 1. Vezi figura Fig 5.1

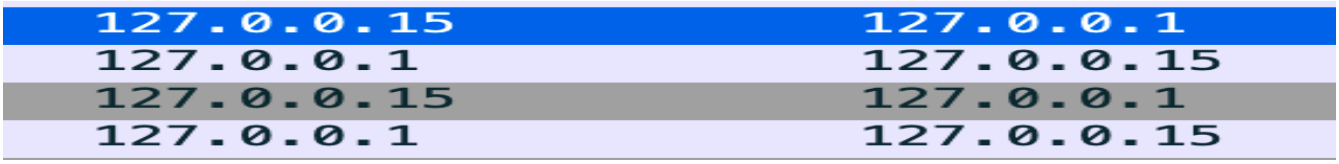


Fig 5.8 Transmisie payload 127.0.0.1/99

După cum se poate observa Senderul 127.0.0.15 va transmite pachetul spre destinația 127.0.0.1, iar 127.0.0.1 il va recepționa

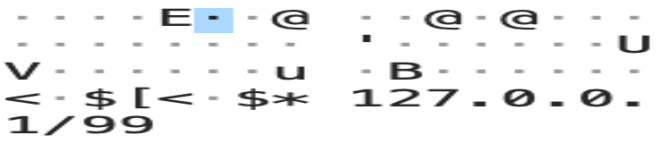


Fig 5.9 Recepție payload 127.0.0.1/99

Nodul cu adresa 127.0.0.1 a receptionat pachetul 99.