Univerzitet u Beogradu
Elektrotehnički fakultet

Aleksandar Popović, 2020/0059

Simulacija zagušenja pomoću ns-3

projekat iz predmeta Primena modernih telekomunikacija

mentor:
prof. dr Milan Bjelica

Beograd, jul 2023.

Sažetak

OVDE NAPISATI SAŽETAK

Ključne reči: ns-3, simulacija, zagušenje

Sadržaj

Spisak slika

Spisak tabela

Glava 1

Uvod

U ovom radu će se obraditi simulacija zagušenja mreže pomoću simulatora NS-3(Network Simulator 3). Simulator je slobodan softver; njegov izvorni kod je javno dostupan za proučavanje i izmenu.
U simulaciji će se pokazati kako preopterećenje veze dovodi do gubitaka paketa i kako se performanse poboljšavaju posle povećanja kapaciteta.
U nastavku će biti opisan postupak pokretanja simulacije, opis simulirane topologije i pregled rezultata.

Glava 2

Pokretanje simulacije

2.1 Preduslovi

Simulator je potrebno instalirati na *Linux* operativnom sistemu pridržavajući se zvaničnog uputstva. [1] Posle instalacije je potrebno napraviti promenljivu okruženja NS3ROOT koja će imati vrednost putanje do korena ns-3 simulatora pokretanjem sledeće komande u terminalu:

```
$ NS3ROOT=putanja_do_direktorijuma
```

Pošto je doseg ove promenljive vezan za sam proces terminala, preporučuje se da se na kraj fajla `~/.bashrc` doda linija:

```
export NS3ROOT=putanja_do_direktorijuma
```

Tako će i prilikom ponovnog pokretanja sistema biti definisana ova promenljiva.

2.2 Pokretanje skripte

U direktorijumu projekta se nalazi *Bash* skripta `run.sh`. Potrebno ju je korišćenjem terminala učiniti izvršivom i pokrenuti je.

```
$ chmod u+x run.sh
$ ./run.sh
```

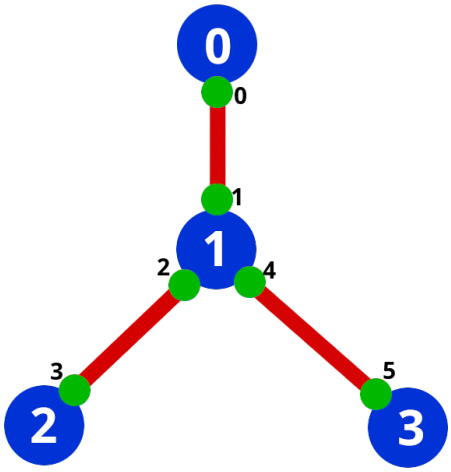
U `output/` direktorijumu će se pojaviti dva izlazna XML fajla u kojima će se naći broj odbačenih paketa na svakom mrežnom uređaju.

Glava 3

Opis simulacije

3.1 Opis mreže

Topologija simulirane mreže je predstavljena na slici 3.1. Mreža se sastoji od četiri računara koja su povezana PPP (*Point to Point Protocol*) vezama. Na računarima označenim brojevima 2 i 3 su instalirane aplikacije koje šalju UDP (*User Datagram Protocol*) pakete ka računaru označenim brojem 0.



Slika 3.1: *Topologija*

Prvobitno su kapaciteti svih veza isti iznose $1Mbps$. Aplikacije konstantno šalju pakete veličine $200b$ brzinom od 5000 paketa u sekundi. Tako se

prouzrokuje zagušenje saobraćaja u mrežnom uređaju u računaru 0 koji se manifestuje gubitkom paketa. Potom se kapacitet linka poveća na $10Mbps$ i posmatra se kako su se promenile performanse mreže.
IP adrese dodeljene mrežnim uređajima su navedene u tabeli 3.1

Tabela 3.1: Adrese mrežnih uređaja

Mrežni uređaj	Adresa
0	10.0.0.2
1	10.0.0.1
2	10.0.1.2
3	10.0.1.1
4	10.0.2.2
5	10.0.2.1

3.2 Opis skripte

Glavni deo skripte je u funkciji `void buildTopology(bool optimize)` koja u zavisnosti od parametra pravi mrežu u kojoj je veza između računara 0 i 1 kapaciteta $1Mbps$ ili $10Mbps$.

3.2.1 Kreiranje uređaja i veza između njih

```
NodeContainer nodes;
nodes.Create(4);

InternetStackHelper internet;
internet.Install(nodes);

PointToPointHelper p2p;
p2p.SetQueue("ns3::DropTailQueue", "MaxSize", StringValue(QueueSize));
p2p.SetDeviceAttribute("DataRate", StringValue(BaseBandwidth));
p2p.SetChannelAttribute("Delay", StringValue(LinkDelay));
NetDeviceContainer n2n1 = p2p.Install(nodes.Get(2), nodes.Get(1));
NetDeviceContainer n3n1 = p2p.Install(nodes.Get(3), nodes.Get(1));
if(optimize){
p2p.SetDeviceAttribute("DataRate", StringValue(IncreasedBandwidth));
}
NetDeviceContainer n1n0 = p2p.Install(nodes.Get(1), nodes.Get(0));
```

Možemo primetiti da će veza između računara 0 i 1 biti povećanog kapaciteta ako parametar `optimize` ima vrednost `true`.

3.2.2 Dodela adresa

U ovom delu skripte će se mrežnim uređajima dodeliti IP adrese i popuniće se tabele rutiranja.

```
Ipv4AddressHelper ip;

ip.SetBase("10.0.0.0", "255.255.255.0");
Ipv4InterfaceContainer i1i0 = ip.Assign(n1n0);

ip.SetBase("10.0.1.0", "255.255.255.0");
Ipv4InterfaceContainer i2i1 = ip.Assign(n2n1);

ip.SetBase("10.0.2.0", "255.255.255.0");
Ipv4InterfaceContainer i3i1 = ip.Assign(n3n1);

Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

3.2.3 Izvor saobraćaja

Na računarima 2 i 3 su instalirane aplikacije koje šalju UDP pakete ka računaru 0. Aplikacije će biti pokrenute u trajanju od 20 sekundi.

```
uint16_t port = 9;
OnOffHelper sender("ns3::UdpSocketFactory",
    InetSocketAddress(i1i0.GetAddress(1),port));
sender.SetAttribute("OnTime",
    StringValue("ns3::ConstantRandomVariable[Constant=1]"));
sender.SetAttribute("OffTime",
    StringValue("ns3::ConstantRandomVariable[Constant=0]"));
sender.SetAttribute("DataRate", StringValue(BaseBandwidth));
sender.SetAttribute("PacketSize", UintegerValue(200));
ApplicationContainer senderApps =
    sender.Install(NodeContainer(nodes.Get(2), nodes.Get(3)));
senderApps.Start(Seconds(0.0));
senderApps.Stop(Seconds(20.0));
```

3.2.4 Prijem saobraćaja

Aplikacija za prijem saobraćaja na računaru 0 će biti aktivna 10 sekundi duže zbog eventualnih paketa u redovima čekanja na slanje.

```
PacketSinkHelper sink("ns3::UdpSocketFactory",
    InetSocketAddress(Ipv4Address::GetAny(),port));
ApplicationContainer sinkApps = sink.Install(nodes.Get(0));
sinkApps.Start(Seconds(0.0));
sinkApps.Stop(Seconds(30.0));
```

3.2.5 Pokretanje simulacije i snimanje izveštaja

Za analizu simulacije je upotrebljena `ns3::FlowMonitor` klasa čiji opis je dostupan u dokumentaciji.[2]

```
Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowMonitorHelper;
flowMonitor = flowMonitorHelper.InstallAll();

Simulator::Stop(Seconds(30.0));
Simulator::Run();

flowMonitor->SerializeToXmlFile(optimize?"izlazsa.xml":"izlazbez.xml",
    false,true);
Simulator::Destroy();
```

Glava 4

Rezultati

U tabeli 4.1 su prikazani rezultati simulacije pre optimizacije, a u tabeli 4.2 posle optimizacije Rezultati su pokazali da je zbog uskog grla drastič-

Tabela 4.1: Pre optimizacije

Poslato paketa	24998
Primljeno paketa	13144
Izgubljeno paketa	11854

Tabela 4.2: Posle optimizacije

Poslato paketa	24998
Primljeno paketa	23808
Izgubljeno paketa	1190

no smanjen kvalitet saobraćaja. Posle optimizacije je procenat izgubljenih paketa smanjen sa 47.42% na 4.76%.

Glava 5

Zaključak

U ovom projektu je prikazana simulacija zagušenja korišćenjem NS-3 simulatora. Opisana je mreža u kojoj je zagušenje pokazano i objašnjena je skripta kojom je simulacija kreirana. Pokazano je kako neoptimalna topologija negativno utiče na kvalitet saobraćaja. Topologija je relativno jednostavna jer cilj rada nije bila kompleksnost, nego pokazivanje mogućnosti analize mreže uz pomoć simulatora.

Literatura

[1] *ns-3 Tutuorial*, izdanje za ns-3.38, 2023; dostupno online: https://www.nsnam.org/docs/release/3.38/tutorial/ns-3-tutorial.pdf

[2] ns-3 dokumentacija: https://www.nsnam.org/docs/doxygen/index.html

Prilozi

Struktura direktorijuma projekta

```
pmt-projekat/
├── run.sh
├── h/
│   └── topology.h
├── src/
│   ├── main.cc
│   └── topology.cc
└── output/
```

run.sh

```
#!/bin/bash

mkdir -p "$NS3ROOT/scratch/projekat"

pushd src
for file in $(ls)
do
cp $file "$NS3ROOT/scratch/projekat/$file"
done
popd

pushd h
for file in $(ls)
do
cp $file "$NS3ROOT/scratch/projekat/$file"
done
popd
```

```
mkdir -p output
OUTDIR="$PWD/output"

pushd "$NS3ROOT"
./ns3 build
./ns3 run scratch/projekat/main.cc --cwd=$OUTDIR
popd
```

topology.h

```
#pragma once
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/flow-monitor.h"
#include "ns3/flow-monitor-helper.h"

#define BaseBandwidth "1Mbps"
#define IncreasedBandwidth "10Mbps"
#define QueueSize "1024p"
#define LinkDelaz "5ms"


using namespace ns3;

void buildTopology(bool optimize);
```

topology.cc

```
#include "topology.h"

void buildTopology(bool optimize){
NodeContainer nodes;
nodes.Create(4);

//instalacija steka protokola na cvorovima
InternetStackHelper internet;
internet.Install(nodes);

//pravimo linkove
PointToPointHelper p2p;
p2p.SetQueue("ns3::DropTailQueue", "MaxSize", StringValue(QueueSize));
p2p.SetDeviceAttribute("DataRate", StringValue(BaseBandwidth));
p2p.SetChannelAttribute("Delay", StringValue(LinkDelay));
NetDeviceContainer n2n1 = p2p.Install(nodes.Get(2), nodes.Get(1));
NetDeviceContainer n3n1 = p2p.Install(nodes.Get(3), nodes.Get(1));
if(optimize){
p2p.SetDeviceAttribute("DataRate", StringValue(IncreasedBandwidth));
}
NetDeviceContainer n1n0 = p2p.Install(nodes.Get(1), nodes.Get(0));

//dodela IP adresa
Ipv4AddressHelper ip;

ip.SetBase("10.0.0.0", "255.255.255.0");
Ipv4InterfaceContainer i1i0 = ip.Assign(n1n0);

ip.SetBase("10.0.1.0", "255.255.255.0");
Ipv4InterfaceContainer i2i1 = ip.Assign(n2n1);

ip.SetBase("10.0.2.0", "255.255.255.0");
Ipv4InterfaceContainer i3i1 = ip.Assign(n3n1);

Ipv4GlobalRoutingHelper::PopulateRoutingTables();

//izvor saobracaja
uint16_t port = 9;
OnOffHelper sender("ns3::UdpSocketFactory",
```

```
    InetSocketAddress(i1i0.GetAddress(1),port));
sender.SetAttribute("OnTime",
    StringValue("ns3::ConstantRandomVariable[Constant=1]"));
sender.SetAttribute("OffTime",
    StringValue("ns3::ConstantRandomVariable[Constant=0]"));
sender.SetAttribute("DataRate", StringValue(BaseBandwidth));
sender.SetAttribute("PacketSize", UintegerValue(200));
ApplicationContainer senderApps =
    sender.Install(NodeContainer(nodes.Get(2), nodes.Get(3)));
senderApps.Start(Seconds(0.0));
senderApps.Stop(Seconds(20.0));

//prijem saobracaja
PacketSinkHelper sink("ns3::UdpSocketFactory",
    InetSocketAddress(Ipv4Address::GetAny(),port));
ApplicationContainer sinkApps = sink.Install(nodes.Get(0));
sinkApps.Start(Seconds(0.0));
sinkApps.Stop(Seconds(30.0));


//snimanje izvestaja
AsciiTraceHelper ascii;
Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream(optimize?
    "bezoptimizacije.tr":"saoptimizacijom.tr");
p2p.EnableAsciiAll(stream);
p2p.EnablePcapAll(optimize?"bezoptimizacije":"saoptimizacijom");

Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowMonitorHelper;
flowMonitor = flowMonitorHelper.InstallAll();

Simulator::Stop(Seconds(30.0));
Simulator::Run();

flowMonitor->SerializeToXmlFile(optimize?
    "izlazsa.xml":"izlazbez.xml",false,true);
Simulator::Destroy();
}
```

main.cc

```
#include "topology.h"

int main(int argc, char* argv[]){
Config::SetDefault ("ns3::Ipv4GlobalRouting::RespondToInterfaceEvents",
    BooleanValue(true));

buildTopology(false);
buildTopology(true);

}
```