DIZAJN I IMPLEMENTACIJA APLIKACIJE ZA MERENJE PERFORMANSI MREŽE U NS-3 SIMULATORU

Nenad Jevtić, Marija Malnar, Pavle Bugarčić
Univerzitet u Beogradu - Saobraćajni fakultet,
n.jevtic@sf.bg.ac.rs, m.malnar@sf.bg.ac.rs, p.bugarcic@sf.bg.ac.rs

Sadržaj: *Simulacije predstavljaju jedan od veoma pouzdanih menahizama za analizu performansi mreže. Kako bi se što bolje procenile ove performanse potrebno je analizirati veliki broj mrežnih parametara kao što su protok, procenat izgubljenih paketa, prosečno kašnjenje sa kraja na kraj, medijana kašnjenja sa kraja na kraj, džiter i slično. U trenutnim izdanjima NS-3 simulatora ne postoji odgovarajući alat kojim mogu da se dobiju svi ovi parametri, te je iz tog razloga, potrebno da se razvije nov alat za analizu mrežnih performansi. U ovom radu dat je detaljan opis i implementacija aplikacije za merenje preformansi mreže u NS-3 simulatoru. U cilju evaluacije rezultata izvršene su serije simulacija MANET (mobile ad hoc network) mreže koja koristi DSR (dynamic source routing) protokol rutiranja i prikazani su rezultati simulacija.*

Ključne reči: *mrežni simulator, NS-3, MANET, performanse mreže*

1. Uvod

Savremene komunikacije ne mogu se zamisiti bez svakodnevnog korišćenja bežičnih mreža. Osim mobilnih bežičnih mreža, svakodnevno se koriste i bežične lokalne mreže - WLAN (*Wireless Local Area Networks*). Sa porastom broja pametnih uređaja i paradigmom IoT (*Internet of Things*) raste i broj uređaja koji se povezuju u tzv. bežične personalne mreže - WPAN (*Wireless Personal Area Networks*). Svakako veoma značajna grupa bežičnih mreža su i bežične *ad hoc* mreže - WANET (*Wireless Ad hoc NETworks*). Među WANET mrežama postoje različite podkategorije kao što su bežične meš mreže - WMN (*Wireless Mesh Network*) kod kojih su čvorovi statični, ali i *ad hoc* mreže sa pokretnim čvorovima, kao što su mobilne *ad hoc* mreže MANET (*Mobile Ad hoc NETworks*) ili *ad hoc* mreže koje čine vozila - VANET (*Vehicular Ad hoc NETworks*)

Istraživači telekomunikacionih mreža u poslednje vreme predlažu veliki broj različitih protokola koji služe za poboljšanje karakteristika mreže. Bez obzira na prirodu bežične, ali i žične mreže, često je veoma komplikovano, vremenski zahtevno i finansijski neisplativo izvršiti eksperimentalnu evaluaciju novih protokola. Iz tog razloga, veliki broj istraživanja koristi neki od mrežnih simulatora kao alternativu kojom se vrši apstrakcija detalja implementacije, s jedne strane, dok se, sa druge strane, zadržava realna slika

mreže u velikoj meri. Mrežni simulatori omogućuju korisnicima da testiraju scenarije koji su teški ili skupi za realizaciju u realnom svetu.

Većina mrežnih simulatora koristi paradigmu diskretne simulacije zasnovane na događajima (*discrete event simulation*). Neki od najčešće korišćenih diskretnih mrežnih simulatora su Omnet++ [1], Glomosim [2], Network Simulator 2 (NS-2) [3], Network Simulator 3 (NS-3) [4], itd. Mrežni simulatori Omnet++, GloMoSim i NS-2, realizovani su kao *dual language* simulatori. Kod *dual language* simulatora jedan programski jezik, najčešće C++, koristi se za modele, dok se drugi programski jezik koristi za konfiguraciju mreže. Ukoliko bi se mrežni simulator koristio samo za pokretanje simulacija sa već dostupnim modelima, korišćenje dva programska jezika ne bi predstavljalo problem, posebno imajući u vidu da je jezik za konfiguraciju mreže često prilično jednostavan. Međutim, ukoliko korisnici simulatora žele da implementiraju svoje protokole i modele, korišćenje dva jezika unosi nepotrebnu kompleksnost. Imajući u vidu da je NS-3 *single language* simulator, napisan u C++, nije iznenađujuće što se sve veći broj korisnika odlučuje upravo za njega u svojim istraživanjima. Upravo iz tog razloga je u ovom radu korišćen NS-3 simulator.

Jedan od najvažnijih ciljeva svake simulacije je generisanje izlaznih statistika o posmatranim parametrima mreže. Iako u okviru samog simulatora postoji nekoliko načina za analizu mreže, ne postoji jedan unificiran način na koji se mogu dobiti svi osnovni parametri kao što su protok, procenat izgubljenih paketa, srednje kašnjenje sa kraja na kraj (*end to end delay* - E2E), medijana E2E kašnjenja i džiter. Umesto toga, korisnici NS-3 simulatora mogu se osloniti na specifične alate kao što je *Flow monitor* ili mogu da koriste ASCII i PCAP sisteme za praćenje (*tracing*). Međutim, *Flow monitor* daje rezultate na mrežnom, a ne na aplikacionom sloju i nije realizovan tako da radi sa svim raspoloživim protokolima rutiranja. Na primer, *Flow monitor* se može koristiti kao alat za analizu mreže sa AODV (*Ad hoc On demand Distance Vector*) [5] protokolom rutiranja, međutim ukoliko mreža koristi DSR (*Dynamic Source Routing*) [6] protokol, *Flow monitor* nije moguće koristiti. S druge strane, ASCII i PCAP sistemi za praćenje generišu izlazne datoteke koje zahtevaju dodatnu analizu rezultata simulacija u eksternim alatima kao što su *Matlab* ili *Wireshark*. Iz tog razloga, veoma je korisno da se razvije nov alat za analizu svih navedenih mrežnih performansi, koji radi nezavisno od protokola rutiranja i ne zasniva se na dodatnom korišćenju pomoćnih eksternih alata. U ovom radu dat je detaljan opis implementacije aplikacije za merenje performansi mreže u NS-3 simulatoru koja predstavlja osnovni element realizovanog alata. U cilju evaluacije rezultata izvršene su simulacije MANET mreže koja koristi DSR [6] protokol rutiranja i prikazani su rezultati simulacija. Izvorni kôd opisane aplikacije javno je dostupan [7].

Rad je organizovan na sledeći način. U drugom poglavlju dat je detaljan opis aplikacije za merenje performansi mreže u NS-3 simulatoru. U okviru trećeg poglavlja opisan je scenario za simulaciju MANET mreže i dati su rezultati simulacija. U okviru poslednjeg, četvrtog poglavlja data su zaključna razmatranja i planovi za dalja istraživanja.

2. Aplikacija za merenje performansi mreže u NS-3 simulatoru

Razvijeni alat za analizu performansi mreže zasniva se na modifikaciji NS-3 simulatora na aplikacionom sloju modela mreže. U tu svrhu realizovane su posebne test

aplikacije za generisanje i prijem korisničkih paketa, kao i poseban deo softvera za statističku obradu dobijenih rezultata i upis podataka u odgovarajuću datoteku.

Aplikacija za generisanje paketa, nazvana `StatsPacketSource`, bazira se na klasi `OnOffApplication` koja je sastavni deo NS-3 simulatora i definisana je u modulu `src/applications`. Ova aplikacija omogućava generisanje periodičnog CBR (*Constant Bit Rate*) saobraćaja pri čemu se veličina paketa zadaje atributom `PacketSize`, a brzina generisanja paketa određena je atributom `DataRate`. Nova aplikacija smeštena je u isti modul, a modifikacije koje su izvršene odnose se na generisanje specifičnog zaglavlja aplikacionog sloja u cilju njegove jedinstvene identifikacije, određivanja procenta gubitka paketa i određivanja kašnjenja paketa na prijemu.

Zaglavlje paketa definisano je klasom `StatsHeader` koja je naslednica NS-3 klase `Header`. Klasa `Header` koristi se kao osnovna klasa iz koje se nasleđuju sve klase za realizaciju zaglavlja paketa bez obzira na mrežni sloj u kom se zaglavlje koristi. U ovom slučaju kreira se zaglavlje aplikacionog sloja kao na slici 1.

```
class StatsHeader : public Header
{
public:
  StatsHeader ();
  void SetSeq (uint32_t seq);
  uint32_t GetSeq (void) const;
  Time GetTs (void) const;
  void SetNodeId (uint32_t nodeId);
  uint32_t GetNodeId (void) const;
  void SetApplicationId (uint32_t appId);
  uint32_t GetApplicationId (void) const;
  static TypeId GetTypeId (void);

  virtual TypeId GetInstanceTypeId (void) const;
  virtual void Print (std::ostream &os) const;
  virtual uint32_t GetSerializedSize (void) const;
  virtual void Serialize (Buffer::Iterator start) const;
  virtual uint32_t Deserialize (Buffer::Iterator start);

private:
  uint32_t m_seq; //!< Sequence number
  uint64_t m_ts; //!< Timestamp
  uint32_t m_nodeId; //!< Sender Node Id
  uint32_t m_appId; //!< Sender Application Id
};
```

Slika 1. Kreiranje zaglavlja aplikacionog sloja.

Ovo zaglavlje očigledno sadrži četiri polja: redni broj paketa (`m_seq`), vreme slanja paketa (`m_ts`), identifikator čvora pošiljaoca (`m_nodeId`) i identifikacioni broj aplikacije koja šalje paket (`m_appId`). Ovi podaci dovoljni su da se na prijemnoj strani na jedinstven način identifikuje od kog čvora i od koje aplikacije je paket poslat, dok se redni broj koristi da se proceni koliko je paketa izgubljeno u toku prenosa. Vreme slanja paketa je neophodno da bi se pri prijemu paketa utvrdilo koliko je tačno iznosilo kašnjenje.

Osim standardnih funkcija za postavljanje i očitavanje vrednosti privatnih podataka članova klase, važno je napomenuti da zaglavlje paketa (kao i sam paket) mora

da poseduje dve specifične funkcije: `Serialize` koja obezbeđuje prevođenje zaglavlja u niz bita i `Deserialize` koja radi obrnut proces rekonstruisanja podataka zaglavlja iz niza bita. Ove funkcije neophodne su kako bi se ostvario realističan prenos preko mreže, jer se svi podaci u jednom paketu (i svim zaglavljima) moraju prevesti u niz bita koji se zatim prenose preko fizičkog interfejsa predajnog čvora da bi se na prijemnom čvoru iz niza primljenih bita ponovo rekonstruisao originalni paket i odgovarajuća zaglavlja.

Modifikacija originalne funkcije za slanje paketa aplikacije `OnOffApplication` sastoji se u tome da se realizovano zaglavlje popuni odgovarajućim podacima, a zatim doda paketu i tako modifikovani paket pošalje korišćenjem kôda, kao na slici 2.

```
StatsHeader statsHeader;
statsHeader.SetSeq (m_sent);
statsHeader.SetNodeId (GetNode ()->GetId ());
statsHeader.SetApplicationId (m_applicationListIndex);
Ptr<Packet> packet =
  Create<Packet> (m_pktSize-(statsHeader.GetSerializedSize ()));
packet->AddHeader (statsHeader);
m_socket->Send (packet);
```

Slika 2. Unapređenje `OnOffApplication` aplikacije

Prilikom kreiranja zaglavlja, polje koje se odnosi na vreme slanja paketa automatski se popunjava. Takođe, pri kreiranju paketa, veličinu paketa (koja se kontroliše atributom `PacketSize` odnosno odgovarajućom promenljivom `m_pktSize`) potrebno je umanjiti za veličinu zaglavlja. Ovo se radi kako paketi (u koje je uključeno zaglavlje) ne bi bili većih dimenzija od onih koje je zahtevao korisnik. Ovakva procedura moguća je kod test aplikacija kod kojih sadržaj paketa nema poseban značaj, dok kod realnih aplikacija koje u paketima aplikacionog sloja prenose koristan sadržaj ovo ne bi bilo moguće, već bi se veličina paketa uvećala za veličinu zaglavlja.

Za određivanje kašnjenja i procenta gubitka paketa potrebno je realizovati i odgovarajuću aplikaciju za prijem paketa. Ova aplikacija je nazvana `StatsPacketSink`, a predstavalja izmenjenu verziju aplikacije `PacketSink` koja je sastavni deo NS-3 simulatora. U ovom slučaju izmena je veoma jednostavna i sastoji se u dodavanju izvora za praćenje koji emituje upravo primljeni paket aplikacionog sloja, kao i identifikator čvora i aplikacije koja je primila paket. Emitovani paket primaju funkcije sakupljači koje vrše dalju statističku obradu podataka iz paketa.

U datoteci zaglavlja definisan je izvor podataka za praćenje:

```
TracedCallback<Ptr<const Packet>, uint32_t, uint32_t> m_rxTrace;
```

Takođe, kako bi bio dostupan ostatku kôda simulatora, izvoru se mora dodeliti ime (u ovom slučaju `Tx`) i tip povezane funkcije sakupljača podataka, što se definiše u funkciji `GetTypeId`, slika 3.

Kao što se može primetiti, izmene koje su unete u već postojeće aplikacije su relativno jednostavne i sastoje se samo u dodavanju novog zaglavlja paketu aplikacionog sloja i odgovarajućih izvora za praćenje. Stoga se može zaključiti da se na ovaj način svaki par aplikacija za slanje i prijem paketa lako može izmeniti tako da se može koristiti

za analizu osnovnih parametara mreže. Ovo je od posebnog interesa ako je potrebno testirati mrežu u nekim specifičnim scenarijima u kojima korišćenje test aplikacija nije moguće.

```
TypeId
StatsPacketSink::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::StatsPacketSink")
    .SetParent<Application> ()
    .SetGroupName("Applications")
    .AddConstructor<StatsPacketSink> ()
// ...
    .AddTraceSource ("Rx","A packet has been received",
      MakeTraceSourceAccessor (&StatsPacketSink::m_rxTrace),
      "ns3::StatsPacketSink::StatsTracedCallback");
  return tid;
}
```

Slika 3. Unapređenje funkcije `GetTypeId`.

Za analizu performansi potrebno još definisati i funkcije sakupljače podataka koji na osnovu primljenih paketa treba da izvrše potrebne statističke analize i određivanje parametara mreže. Realizacija funkcija sakupljača podataka je nešto složeniji zadatak i sastoji se u tome da se obezbedi jedna (ili više) funkcija koja kao argumente prima pametni pokazivač na paket i identifikatore čvora i aplikacije koji su primili paket, a kao rezultat ne vraća vrednost (`void`), slika 4.

```
typedef void (* StatsTracedCallback) (Ptr<const Packet> packet,
  uint32_t sinkNodeId, uint32_t sinkAppId);

class StatsFlows
{
public:
  StatsFlows (std::string fileName = "noname");
  void PacketReceived (Ptr<const Packet> packet,
                       uint32_t sinkNodeId, uint32_t sinkAppId);
  void Finalize ();

private:
  std::vector<NetFlowId> m_flowIds;
  std::vector<FlowData> m_flowData;
  std::string m_fileName;
  uint32_t m_allRxPackets;
  bool m_singleFile;
};
```

Slika 4. Realizacija sakupljača podataka.

Kao osnovni cilj postavljeno je da se svaki podatak o primljenom paketu, u vidu vremenskog niza, upisuje u datoteku, pri čemu korisnik zadaje ime datoteke po želji. Ako korisnik ne zada ime datoteke tada će se ime generisati automatski. Takođe, svi generisani podaci o paketima i statistikama se mogu upisivati u jednu zajedničku datoteku ili u više datoteka, što je definisano promenljivom `m_singleFile`. Podaci se u datoteci snimaju koristeći zarez (`,`) kao separator te se na taj način dobijaju datoteke CSV (*Comma*

Separated Values) tipa koje se lako učitavaju i dalje obrađuju u programskom paketu *Excel* ili nekom dugom sličnom programu.

Kako scenario simulacije zahteva da istovremeno više čvorova generiše saobraćaj, potrebno je da se svaki paket razvrsta prema određenom identifikatoru toka podataka (`NetFlowId`). Identifikator toka podataka čine četiri identifikatora: predajnog čvora, predajne aplikacije, prijemnog čvora i prijemne aplikacije. Svi identifikatori tokova čuvaju se u vektorskoj promenljivoj `m_flowIds`, a odgovarajuće statistike u vektoru `m_flowData`. Na primer, ukoliko tri čvora emituju i primaju saobraćaj u mreži, biće formirana tri toka podataka i tri odgovarajuća identifikatora toka. Vektor `m_flowData` čuva podatke o svakom toku u vidu objekata klase `FlowData`. `FlowData` objekti čuvaju statistike o jednom toku: broj primljenih paketa, ukupnu količinu podataka koja je primljena, trenutak prijema prvog paketa, kao i trenutak i redni broj poslednjeg primljenog paketa. Ovi podaci se ažuriraju po prijemu svakog paketa aplikacionog sloja u funkciji `PacketReceived`. Ova funkcija vrši razvrstavanje pristiglog paketa prema identifikatoru toka i obavlja dve osnovne funkcije:

- ažurira statistike u odgovarajućem objeku tipa `FlowData` koji se čuva u vektoru `m_flowData` i
- upisuje u odgovarajući fajl trenutak prispeća paketa, identifikator toka kome paket pripada, redni broj paketa i kašnjenje koje je pretrpeo paket na putu kroz mrežu od predajnog do prijemnog čvora.

Pozivanje funkcije `PacketReceived` vrši se automatski jer je ta funkcija definisana tako da predstavlja sakupljač podataka. Funkcija `PacketReceived` se povezuje na izvor podataka `Tx` prijemne aplikacije klase `StatsPacketSink` u `main` funkciji simulacionog skripta korišćenjem `Config` sistema:

```
StatsFlows sf;
Config::ConnectWithoutContext(
  "/NodeList/*/ApplicationList/*/$ns3::StatsPacketSink/Rx",
  MakeCallback (&StatsFlows::PacketReceived, &sf));
```

Kada se simulacija završi (funkcija `Run`), svi podaci koji su sakupljeni u vektoru podataka `m_flowData` se dodatno obrađuju i upisuju u datoteku pozivom funkcije `Finalize` u simulacionom skriptu:

```
Simulator::Run ();
sf.Finalize ();
```

U funkciji `Finalize` podaci iz vektora `m_flowData` se najpre obrađuju po svakom toku podataka i izračunava se srednje kašnjenje, medijana kašnjenja, džiter i procenat izgubljenih paketa. Ovi podaci se dalje upisuju u datoteku za svaki tok podataka posebno. Nakon toga, podaci se dodatno statistički obrađuju usrednjavanjem podataka za sve tokove i ponovo računa srednje vreme kašnjenja, medijana, džiter i procenat izgubljenih paketa i to se upisuje u jednu zajedničku datoteku.

Kao rezultat simulacije dobija se CSV datoteka u kojoj su rezultati prikazani u tri dela. U okviru prvog dela dati su podaci o paketima koji se redom šalju u toku simulacije. Za svaki paket obezbeđeni su podaci o indeksu toka, vremenskom trenutku u kom je paket primljen, redni broj i kašnjenje paketa.

U drugom delu, prvo su dati osnovni podaci o svakom generisanom toku podataka: indeks toka, adresa izvorišnog čvora za taj tok podataka, identifikator izvorišne aplikacije, adresa prijemnog čvora za taj tok podataka i identifikator prijemne aplikacije. Zatim, za svaki tok obezbeđen je podatak o srednjem, maksimalnom i medijani kašnjenja sa kraja na kraj i džiteru, kao i podaci o broju izgubljenih paketa i realno ostvarenom protoku na aplikacionom sloju. Osim toga, za svaki tok podataka obezbeđeni su i dodatni podaci za odgovarajući izvorišni i odredišni čvor: trenutak prijema prvog i poslednjeg paketa, vreme tokom kog su slati paketi, broj poslatih/primljenih paketa, količina poslatih/primljenih podataka i protok.

Konačno, u trećem delu izlazne datoteke dati su sumirani podaci za sve tokove podataka zajedno, kao što je prikazano na slici 5.

AVERAGE RESULTS FOR ALL FLOWS	
Average E2E Delay [ms]:	6.734642118
Median E2E Delay [ms]:	4.279466667
Max of E2E Delay [ms]:	464.2007333
Jitter of E2E Delay [ms]:	20.59061677
Number of all Tx packets:	4808
Number of all Rx packets:	4456
Number of all lost packets:	352
Lost packets [%]:	7.321131448
Real troughput [kbps]:	45.98903333

Slika 5. Prikaz sumiranih rezultata simulacije.

Kao što se može videti na slici 5, korisnik opisane aplikacije na raspolaganju ima podatak o srednjem kašnjenju sa kraja na kraj, medijani kašnjenja sa kraja na kraj, maksimalnom kašnjenju sa kraja na kraj, džiteru, broju svih poslatih paketa, broju svih primljenih paketa, broju izgubljenih paketa, procentu izgubljenih paketa i ostvarenom protoku na aplikacionom sloju.

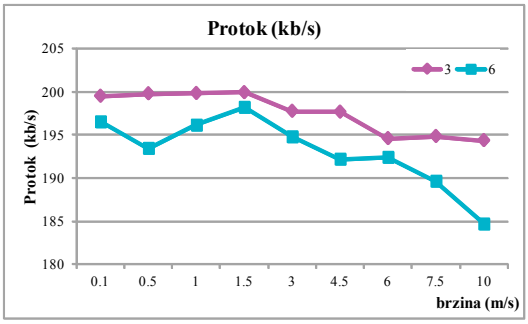
3. Opis simulacionog scenarija i rezultati simulacija

Za potrebe evaluacije realizovanog alata za određivanje performansi mreže, izvršene su simulacije u okviru kojih je analizirana mreža koja se sastoji od 60 čvorova. Početne pozicije čvorova raspoređene su na slučajan način u okviru simulacione oblasti dimenzija 2000m x 2000m. Generisan je CBR saobraćaj pri čemu se šalju paketi fiksne veličine od 512 B. Korišćeni su UDP protokol na transportnom sloju i DSR protokol rutiranja. Za MAC sloj korišćen je 802.11b standard sa protokom od 5,5 Mb/s i `YansWiFiChannel` model kanala. Za model propagacije korišćen je propagacioni model `FriisPropagationLossModel` i model mobilnosti `RandomWaypoint` kod kog je menjana brzina kretanja čvorova na vrednosti 0,1 m/s, 0,5 m/s, 1 m/s, 1,5 m/s, 3 m/s, 4,5 m/s, 6 m/s, 7,5 m/s i 10 m/s.

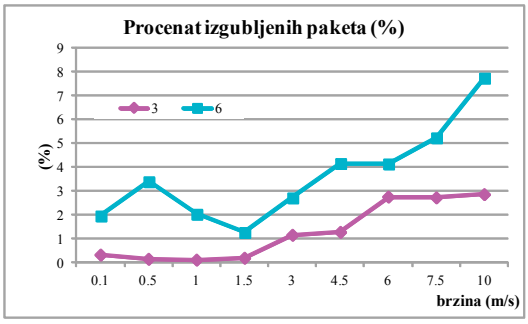
Varijacija saobraćajnog opterećenja vršena je promenom broja aktivnih čvorova koji generišu saobraćaj u mreži. Simulacije su izvršene za slučajeve kada saobraćaj generiše 5% i 10% čvorova od ukupnog broja čvorova u mreži. Pošto mreža ima 60 čvorova, simulacija se izvršava kada saobraćaj generišu istovremeno 3 i 6 čvorova. Svi čvorovi generišu CBR saobraćaj istog intenziteta koji na aplikacionom sloju generiše protok od 200 kb/s. Za određivanje parametara simulacije korišćene su opisane test

aplikacije za generisanje i prijem paketa, na osnovu kojih su dobijeni podaci o srednjem E2E kašnjenju, medijani E2E kašnjenja, džiteru, protoku i procentu gubitka paketa. Trajanje simulacije je 200 s, dok je vreme inicijalizacije mreže 10 s.

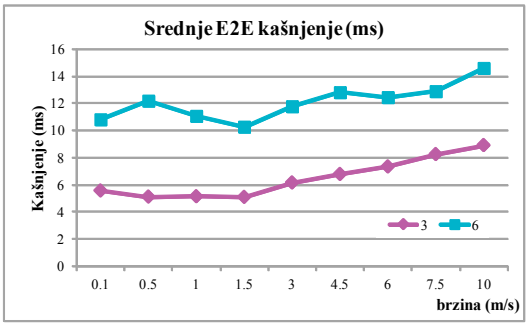
Za svaki skup parametara simulacija je ponavljana četiri puta sa međusobno nekorelisanim vrednostima slučajnih veličina (različit `RngRun`) i kao krajnji rezultat uzete su srednje vrednosti za svaki od parametara mreže. Dobijeni rezultati za protok, procenat izgubljenih paketa, srednje E2E kašnjenje, medijanu E2E kašnjenja i džiter prikazani su na slikama 6-10.



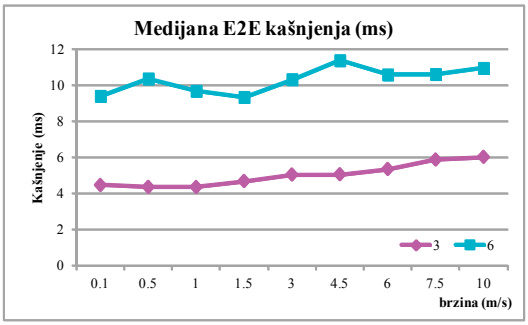
Slika 6. Protok podataka u zavisnosti od brzine kretanja čvorova i broja čvorova koji generišu saobraćaj.



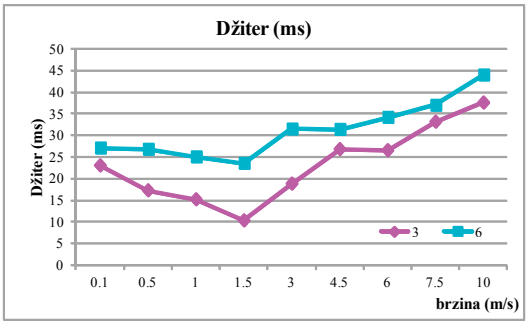
Slika 7. Procenat izgubljenih paketa u zavisnosti od brzine kretanja čvorova i broja čvorova koji generišu saobraćaj.



Slika 8. Srednje E2E kašnjenje u zavisnosti od brzine kretanja čvorova i broja čvorova koji generišu saobraćaj.



Slika 9. Medijana E2E kašnjenja u zavisnosti od brzine kretanja čvorova i broja čvorova koji generišu saobraćaj.



Slika 10. Džiter u zavisnosti od brzine kretanja čvorova i broja čvorova koji generišu saobraćaj.

Kao što se vidi na slikama 6-10, performanse mreže degradiraju sa porastom brzine čvorova, što je i očekivano imajući u vidu da što se čvorovi brže kreću brže i izlaze iz međusobnog dometa te je potrebno više vremena za pronalaženje putanja, samim tim se povećava i broj izgubljenih paketa i smanjuje protok. Takođe, može se primetiti da u slučaju kada veći broj čvorova generiše saobraćaj dolazi do degradiranja mrežnih performansi, što je posledica povećanog saobraćajnog opterećenja. Na osnovu prikazanih rezultata pokazano je da predloženi alat moguće koristiti za jednostavniju i efikasniju analizu bežičnih mreža u NS-3 simulatoru.

4. Zaključak

U ovom radu opisan je poseban softverski dodatak za NS-3 simulator predviđen za merenje performansi mreže koji se bazira na implementaciji test aplikacije za generisanje i prijem paketa kao i odgovarajućoj statističkoj obradi rezultata. Aplikacija je razvijena u cilju obezbeđivanja efikasnog alata za dobijanje osnovnih parametara mreže

kao što su protok, procenat izgubljenih paketa, srednje E2E kašnjenje, medijana E2E kašnjenja i džiter. U trenutnom izdanju NS-3 simulatora nije realizovan jedinstven alat kojim se mogu dobiti svi ovi parametri i koji funkcioniše za sve protokole rutiranja, te predloženi alat predstavlja značajan doprinos efikasnosti korišćenja simulatora. Da bi se potvrdilo da realizovana aplikacija zaista daje odgovarajuće rezultate, izvršen je niz simulacija jedne MANET mreže koja koristi DSR protokol rutiranja i prikazani su dobijeni rezulati.

Planovi za buduća istraživanja obuhvataju dalje unapređenje simulatora, kako novim alatima za analizu mreže, tako i unapređenjem postojećih protokola koji se koriste u *ad hoc* mrežama sa ciljem da se što je moguće više poboljšaju performanse mreže.

Literatura

[1] OMNet++, [Online]. Available at: http://www.omnetpp.org/
[2] GloMoSim, [Online]. Available at: http://pcl.cs.ucla.edu/projects/glomosim/
[3] NS-2. [Online]. Available at: http://www.isi.edu/nsnam2
[4] NS-3. [Online]. Available: http://www.nsnam.org/
[5] C. E. Perkins, E.M. Belding–Royer, S.R. Das, "Ad Hoc On demand Distance Vector (AODV) routing", RFC 3561, IETF, 2003.
[6] D. Johnson, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks for IPv4", RFC 4728, IETF, 2007
[7] Source code [Online]. Available at: https://github.com/neje/ns3-aodv-network-performance-tool

Abstract: *Simulations represent one of the most reliable tools for the analysis of network performance. In order to provide as good as possible performance evaluation it is necessary to analyze large number of network parameters such as throughput, packet loss ratio, average end to end delay, median of end to end delay, jitter, etc. In current release of NS-3 (Network Simulator 3) there is no available tool, which provides all these parameters. Therefore, it was necessary to develop a new tool for the network performance analysis. In this paper, a detailed description and implementation of application for network performance measurements in NS-3 simulator is given. In order to evaluate this implementation, series of simulations of MANET (Mobile Ad hoc NETwork) network within DSR (Dynamic Source Routing) protocol are done, and simulation results are presented.*

Keywords: *Network simulator, NS-3, MANET, network performance*

DESIGN AND IMPLEMENTATION OF THE APPLICATION FOR NETWORK PERFORMANCE MEASUREMENTS IN THE NS-3 SIMULATOR

Nenad Jevtić, Marija Malnar, Pavle Bugarčić