UNIVERZITET U BANJOJ LUCI
ELEKTROTEHNIČKI FAKULTET

Ognjen Joldžić

ADAPTIVNI SISTEM ZA DETEKCIJU DDOS NAPADA U RAČUNARSKIM MREŽAMA

Doktorska disertacija

Banja Luka, 2017.

UNIVERSITY OF BANJA LUKA
FACULTY OF ELECTRICAL ENGINEERING

Ognjen Joldžić

ADAPTIVE SYSTEM FOR DDOS ATTACK DETECTION IN COMPUTER NETWORKS

Doctoral dissertation

Banja Luka, 2017.

Informacije o mentoru i disertaciji

Mentor: Zoran Đurić, vanredni profesor, Elektrotehnički fakultet, Univerzitet u Banjoj Luci

Naslov doktorske disertacije: Adaptivni sistem za detekciju DDoS napada u računarskim mrežama

Rezime: Disertacija predstavlja rezultat istraživanja mogućnosti detekcije Denial of Service (DoS) napada u konvergentnim računarskim mrežama, kao i rada na razvoju rješenja za detekciju posebne klase napada koji izazivaju prekid servisa kroz slanje ogromne količine podataka prema malom broju krajnjih hostova, koje pritom nije moguće na efikasan način detektovati metodama detekcije baziranim na specifikaciji potpisa napada. Dodatno, u disertaciji je predložen i namjenski algoritam za raspoređivanje opterećenja koji razvijenom rješenju omogućava skalabilnost u uslovima velikih brzina prenosa podataka. U prvim poglavljima je dat teorijski pregled najznačajnijih napada na računarsku infrastrukturu sa posebnim akcentom na distribuirane DoS napade, kao i presjek postojećeg stanja u oblasti detekcije napada. Rješenje prikazano u ovoj disertaciji omogućava detekciju napada baziranu na anomalijama u rasporedu saobraćaja po procesnim elementima i entropiji odradišnih adresa u odnosu na ukupan broj paketa u jedinici vremena. Algoritam za raspoređivanje opterećenja omogućava dinamičko proširivanje kapaciteta, zadržavajući funkcionalnost kompletnog sistema bez obzira na trenutne uslove okruženja. U završnim poglavljima su dati rezultati eksperimenata koji kroz različite scenarije potvrđuju ispravnost hipoteze i pokazuju da je moguće razviti skalabilno i adaptivno rješenje za detekciju (D)DoS napada bazirano na anomalijama, bez obzira na karakteristike saobraćaja i veličinu zaštićene mreže.

Ključne riječi: detekcija intruzije, prevencija intruzije, distribuirana obrada, raspoređivanje opterećenja, sigurnost

Naučna oblast: Tehnološke nauke

Naučno polje: Sistemski inženjering, računarska tehnologija

Klasifikaciona oznaka: T 120

UDK broj: 004.7.056.5:004.85]:004.735(043.43)

Information about mentor and dissertation

Mentor: Zoran Đurić, associate professor, Faculty of electrical engineering, University of Banja Luka

Title of doctoral dissertation: Adaptive System for DDoS Attack Detection in Computer Networks

Abstract: The dissertation represents the result of research in Denial of Service (DoS) attack detection in converged computer networks, as well as work on the development of solution that is able to detect a special class of network attacks that cause service disruptions by sending vast amounts of traffic aimed at a small number of network hosts, which, at same time, cannot be efficiently detected by signature-based detection methods. Additionally, the dissertation proposes a load balancing algorithm that enables the solution to achieve scalability in high-bandwidth conditions. The initial chapters give a theoretical outline of the most important network attacks, with a special analysis of distributed denial of service (DDoS) attacks, as well as an overview of the current state of the field of attack detection. The solution presented in this dissertation enables the detection of (D)DoS attacks based on the anomalies on traffic patterns of different processing elements and the entropies of the destination addresses in the total number of destination packets over a short period of time. The load balancing algorithm enables the capacity of the system to be dynamically extended, while retaining its efficiency regardless of the current network conditions. The final chapters provide the experimental results that prove the initial hypothesis and show that it is possible to develop a scalable and adaptive solution to detect (D)DoS attacks based on traffic anomalies, which does not depend on the traffic characteristics and the size of the protected network.

Keywords: intrusion detection, intrusion prevention, distributed processing, load balancing, security

Scientific area: Technological sciences

Scientific field: Systems engineering, computer technology

Classification code: T 120

UDC number: 004.7.056.5:004.85]:004.735(043.43)

Sadržaj

Slike

Zahvalnice

Posebno se zahvaljujem mentoru, prof. dr Zoranu Đuriću za motivaciju i podršku tokom trajanja studija, kao i za sugestije i pomoć tokom izrade rada.

Zahvaljujem se porodici na stalnoj podršci mom radu.

Zahvaljujem se prof. dr Slavku Mariću, prof. dr Zoranu Jovanoviću, a naročito doc. dr Pavlu Vuletiću na sugestijama i pomoći tokom izrade rada.

Banja Luka, Bosna i Hercegovina Ognjen Joldžić, 2017.

1. Uvod

Povećanje dostupnosti računarskih mreža i brzine komunikacije na Internetu je imalo veliki uticaj na razvoj informacionih tehnologija. Takav razvoj je uzrokovao povećanje broja korisnika mreža, kao i pojavu čitavog niza novih aplikacija i servisa. Računarske mreže se koriste u svakodnevnim poslovima i imaju značajan uticaj i na oblasti ljudskog djelovanja koje nisu primarno vezane uz informacione tehnologije.

Da bi se omogućio ovakav rast primjene računarskih mreža, potrebno je voditi računa o različitim pratećim aspektima funkcionisanja mreža i aplikativnih servisa, među kojima je svakako jedan od najvažnijih sigurnost. Visoka dostupnost mrežnih tehnologija i brzina komunikacije imaju negativan efekat u smislu znatno manjih resursa koji su potrebni za generisanje različitih vrsta napada usmjerenih protiv samih mreža ili pratećih aplikativnih i mrežnih servisa. Pružaoci usluga komunikacije (*provajderi*) moraju omogućiti sigurnu komunikaciju krajnjim korisnicima bez narušavanja performansi mreže ili unošenja kašnjenja u komunikaciji, bez obzira na tipove napada i osobine algoritama kojima se vrši njihova detekcija. Tipičan napadač na raspolaganju ima dovoljnu količinu resursa da čak i sam kreira efektan napad na neki informacioni sistem. Opasnost se višestruko uvećava ako se u obzir uzme mogućnost distribuiranja napada putem većeg broja pošiljalaca čime se potencijalno uvećava njegova snaga, a smanjuje mogućnost uspješne detekcije.

1.1 Formulacija problema

Među tehnologijama koje su uticale na stanje u kojem se nalaze današnje računarske mreže jedno od najznačajnijih mjesta zauzima mrežna konvergencija. Pojava konvergiranih mreža predstavlja, prije svega, promjenu u načinu projektovanja mrežne infrastrukture koja je za sobom povukla čitav niz promjena u fizičkoj i logičkoj organizaciji mrežnog hardvera i pratećih servisa.

Nekonvergentne računarske mreže su projektovane tako da je saobraćaj fizički grupisan prema tipu sadržaja koji se prenosi preko medijuma, odnosno tako da se preko istog medijuma prenosi samo jedna vrsta komunikacije, bilo da se radi o prenosu paketa, audio ili video sadržaja. Ovakav pristup je omogućavao jednostavniju implementaciju infrastrukture, jer je bilo znatno jednostavnije izvršiti procjenu potrebnih mrežnih kapaciteta. Implementacija sigurnosnih mehanizama je uključivala detekciju samo onog skupa napada koji su bili primjenjivi na dati tip saobraćaja i medijum, pri čemu najčešće nije bilo potrebe za prioritetizacijom

mrežnih paketa, pa je kašnjenje koje bi nastalo kao posljedica rada algoritma za detekciju (ako je postojalo) bilo konstantno.

S druge strane, nedostatak ovakvog pristupa projektovanju ogleda se u visokoj cijeni implementacije u slučaju kad je istom korisniku potrebno omogućiti prenos različitih tipova saobraćaja. Za svaki tip saobraćaja (npr. internet, kablovska televizija) je bilo potrebno planirati i projektovati odvojene fizičke kapacitete. Sa druge strane, razvoj korisničkih uređaja različitih karakteristika i povećanje dostupnosti različitih sadržaja na mreži konstantno povećavaju potrebu korisnika za istovremenim pristupom različitim resursima mreže, često u isto vrijeme. Pritom je potrebno održati zadovoljavajući kvalitet servisa bez obzira na tip korisničkog uređaja ili trenutnu lokaciju korisnika. Samim tim, implementacija ovakve infrastrukture korištenjem nekonvergiranih mreža bi značajno povećala cijenu implementacije kompletnog sistema. Ovakav tip mreža se i danas zadržao u zatvorenim sistemima kakav je sistem kablovskih mreža za prenos televizijskog programa, ali je i u toj oblasti prisutan sve veći broj rješenja koje ne ograničavaju komunikaciju na samo jedan tip saobraćaja.

Konvergentne mreže koriste zajednički medijum za prenos svih vrsta saobraćaja, pri čemu je zadatak hardverskih uređaja i odgovarajućih softverskih komponenti da obezbijede jednak kvalitet servisa kao u slučaju nekonvergiranih mreža. Implementacija ovakve mreže mora uključivati i mehanizme za određivanje prioriteta pojedinih paketa, jer za različite tipove komunikacije postoje različiti zahtjevi po pitanju dozvoljenog kašnjenja i pouzdanog prenosa (na primjer, kod glasovne komunikacije je dozvoljeno kašnjenje pojedinih paketa znatno manje nego kod HTTP saobraćaja; određenim tipovima video komunikacije nije potreban pouzdan prenos i sl.). Logično, implementacija konvergentne mreže je značajno komplikovanija po pitanju konfiguracije i administracije, ali je, zbog postojanja zajedničkog medijuma za prenos podataka, jeftinija od nekonvergentne mreže.

Opisana promjena u načinu projektovanja konvergiranih mreža ima veliki uticaj i na sigurnost mreža [1]. Konvergencijom velikog broj različitih protokola i tipova saobraćaja prema zajedničkoj infrastrukturi, uvećava se i broj napada na mrežne i aplikativne servise koji se odvija u okviru iste mreže. Ono što je ranije bila fizički odvojena komunikacija i što je uključivalo korištenje zasebnih, namjenskih sigurnosnih sistema, u ovom slučaju predstavlja posao samo jednog sistema. Detekcija napada se mora omogućiti bez značajnijeg narušavanja performansi sistema - obrada paketa ne smije uticati na konfigurisane prioritete pojedinih tipova saobraćaja ili unositi drastično kašnjenje u komunikaciji. Iz svega navedenog proizilazi čitav niz problema koje mora riješiti svaki sistem, odnosno algoritam za detekciju napada na mrežnom nivou OSI modela [1].

Zbog činjenice da je u današnje vrijeme u upotrebi jako veliki broj protokola i različitih tipova servisa, proces detekcije napada ne može biti sveobuhvatan. Naime, različite klase napada se detektuju i onemogućavaju na različitim slojevima OSI modela, pa bi razvoj sistema koji bi bio u stanju da detektuje sve napade bio nepraktičan i kompleksan. Primjera radi, napadi kakvi su *SQL injection* (*SQLi*)

[1]OSI (Open Systems Interconnection) - referentni model komunikacije za računarske mreže koji predviđa grupisanje protokola i servisa prema funkcionalnim slojevima

i *Cross-site scripting* (*XSS*) su usmjereni prema aplikativnim servisima i jednostavno se mogu onemogućiti u samom programskom kodu aplikacije. Sa druge strane, napadi usmjereni prema mrežnom sloju OSI modela ne mogu biti detektovani od strane aplikacije i uključuju drugačiju analizu saobraćaja od one koja bi bila potrebna u slučaju prethodne grupe napada. U ovom radu su od posebnog interesa napadi koji nastaju kao posljedica generisanja ogromne količine saobraćaja sa ciljem da se zauzmu resursi odredišnog sistema i da se na taj način onemogući njegovo korištenje od strane legitimnih korisnika. Ovakav tip napada naziva se Denial of service (DoS) napad. Ako je napad generisan od strane velikog broja korisnika, onda se radi o distribuiranom DoS (DDoS) napadu. Ovi napadi su detaljno analizirani u sekciji 2.1.

Način detekcije bilo kojeg napada predstavlja jedan od najbitnijih aspekata razvoja i implementacije rješenja za zaštitu računarskih mreža. Sistem za detekciju se može u manjoj ili većoj mjeri oslanjati na znanje administrativnog osoblja o funkcionisanju mreže i potencijalnim napadima. Idealan sistem bi trebao biti autonoman, odnosno biti u stanju prilagoditi proces detekcije trenutnim uslovima funkcionisanja mreže i detektovati i napade čiji način izvršavanja nije poznat u trenutku inicijalnog konfigurisanja sistema. Takav sistem bi bio u stanju donijeti odluku o početku napada na osnovu pojave neke anomalije u funkcionisanju mreže u odnosu na uobičajeni način rada. Ovakav sistem je znatno kompleksniji za razvoj i implementaciju, ali načelno mnogo robusniji od sistema koji se u svom radu oslanjaju na unaprijed definisane konfiguracije. Tipovi sistema za detekciju i prevenciju upada (engl. *intrusion*, intruzije) su detaljno analizirani u sekciji 2.2.

Konačno, zbog ubrzanog razvoja tehnologije koji dovodi do konstantnog povećanja brzine komunikacije, potrebno je obratiti posebnu pažnju na skalabilnost sistema za detekciju napada. U protivnom, ako sistem nije u stanju da obradi pristigli saobraćaj, postoji rizik od stvaranja uskog grla za protok paketa na mjestu na kojem je on postavljen. Time sistem za detekciju postaje slaba tačka kompletne mreže i otvara mogućnost izvršavanja napada koji bi bili usmjereni direktno protiv sistema za detekciju, a ne protiv nekog konkretnog aplikativnog ili mrežnog servisa.

1.2 Cilj i doprinos istraživanja

Osnovni cilj istraživanja je verifikacija hipoteze da je moguće izvršiti uspješnu detekciju DDoS napada koji nastaju kao posljedica generisanja ogromne količine saobraćaja prema zaštićenom dijelu računarske mreže, bez obzira na arhitekturu i način funkcionisanja mreže, kao i na servise koje ona sadrži. Za takvu detekciju potrebno je razviti adaptivno i skalabilno rješenje koje će biti u stanju da detektuje anomalije u normalnom funkcionisanju mreže, a da pritom ne naruši performanse mreže, odnosno njenih servisa.

Osnovni doprinos istraživanja je predstavljen kroz sljedeće faze:

- analiza postojećeg stanja u oblasti detekcije i prevencije (D)DoS napada na mrežnu infrastrukturu - detaljan pregled postojećih rješenja u oblasti detekcije napada, kao i analiza njihove primjenjivosti na problem detekcije (D)DoS napada,

- dizajniranje algoritma za detekciju napada - algoritam za detekciju (D)DoS napada treba biti u stanju da kroz praćenje normalnog rada mreže prepozna bilo kakve anomalije u njenom funkcionisanju bez posebnog uticaja administratora, i da na osnovu detektovanih anomalija utvrdi postojanje malicioznih aktivnosti na mreži,
- dizajniranje algoritma za raspodjelu opterećenja među procesnim elementima kod detekcije napada - radi omogućavanja skalabilnosti sistema, potrebno je dizajnirati algoritam koji će omogućiti jednostavno proširivanje kapaciteta sistema uz ravnomjeran raspored opterećenja po svim aktivnim procesnim elementima. Algoritam uključuje i mehanizam praćenja aktivnosti elemenata kako bi se preraspodjela opterećenja, ako je to potrebno, izvršila dinamički, bez prekida rada sistema,
- razvoj adaptivnog sistema za detekciju DDoS napada - sistem treba obavljati svoju primarnu funkciju bez narušavanja performansi mreže, odnosno unošenja kašnjenja u komunikaciji. Osim toga, treba omogućiti jednostavno proširivanje kapaciteta sistema kroz dodavanje procesnih elemenata koji će odmah po uključenju biti opterećeni jednakom količinom saobraćaja kao i već aktivni elementi. Sistem treba biti dizajniran tako da bude otporan na napade koji su usmjereni protiv njega samog, tako da nije moguće ostvariti prekid funkcionisanja servisa kroz onesposobljavanje sistema za detekciju.

1.3 Pregled i sadržaj rada

Nakon uvodnih razmatranja, u poglavlju 2 dat je pregled osnovnih tipova napada na računarske mreže, sa posebnim osvrtom na klasu napada koja je od interesa za ovaj rad. Ovo poglavlje sadrži pregled osnovnih koncepata detekcije i prevencije napada na računarske mreže, kao i osnovne osobine sistema za detekciju i prevenciju intruzije.

Poglavlje 3 sadrži pregled postojećih rješenja i naučnih radova iz oblasti obuhvaćenih ovim radom, u prvom redu iz oblasti detekcije napada i softverski definisanih mreža (SDN).

U poglavlju 4 je data detaljna analiza rješenja za detekciju (D)DoS napada razvijenog u okviru ovog rada, uključujući i kratak opis najznačajnijih korištenih tehnologija. Poglavlje sadrži pregled arhitekture rješenja, softverskih i hardverskih komponenti, kao i komunikacionog protokola koji se koristi za razmjenu poruka između elemenata rješenja. U istom poglavlju je data i detaljna analiza razvijenih algoritama za detekciju napada i raspoređivanje opterećenja, kao i pregled osnovnih konfiguracionih parametara koji utiču na rad sistema.

U poglavlju 5 su predstavljeni rezultati opis eksperimentalnog dijela istraživanja kojim je potvrđena postavljena hipoteza. Radi preglednosti, grafički rezultati provedenih testova su dati u okviru priloga u poglavlju 7.

U poglavlju 6 je dato poređenje razvijenog rješenja sa postojećim rješenjima iz iste oblasti, kao i kritički osvrt na rad i potencijalni pravci daljeg istraživanja. Na kraju rada je dat spisak korištene literature.

2. Osnovni koncepti detekcije i prevencije intruzije

2.1 Vrste napada na računarske mreže

Napad ili intruzija na računarsku mrežu/sistem se definiše kao bilo kakva akcija izvršena sa ciljem prekida rada mreže/sistema ili pribavljanja neovlaštenog pristupa nekom njegovom zaštićenom dijelu. Napadi mogu biti usmjereni prema različitim dijelovima sistema, počevši od nižih slojeva OSI modela, pa sve do napada koji iskorištavaju neki sigurnosni propust u samim aplikacijama ili servisima.

Proces detekcije napada je uslovljen tipom napada. Napadi koji su usmjereni prema aplikativnom sloju najčešće koriste propuste u programskom kodu, načinu pristupa skladištu podataka ili načinu formiranja prezentacionog sloja web aplikacije. Efekti takvih napada su najčešće ograničeni na napadnutu aplikaciju i servise koje aplikacija direktno koristi (aplikativni server, sistem za upravljanje bazama podataka i sl.) i ne utiču na druge aplikacije koje se izvršavaju na istom sistemu. U ovu grupu spadaju različiti napadi koji koriste neadekvatne mehanizme validacije ulaznih parametara: SQLi, XSS, buffer overflow i sl [2],[3]. Iako postoje mehanizmi kojima bi se detekcija ovakvih napada i sigurnosnih propusta koji bi do njih mogli dovesti mogla izvršiti izvan same aplikacije [4],[5], najjednostavniji način za sprečavanje napada ove vrste predstavljaju izmjene u samom programskom kodu aplikacije.

Sa druge strane, napadi koji funkcionišu ispod aplikativnog sloja OSI modela ne mogu biti detektovani od strane aplikacija i servisa, već za cilj imaju onesposobljavanje neke komponente nižeg nivoa na sistemu. Zajedničko za sve takve napade je da su njihovi efekti vidljivi na većem broju servisa, ili čak dovode u pitanje funkcionisanje kompletnog sistema.

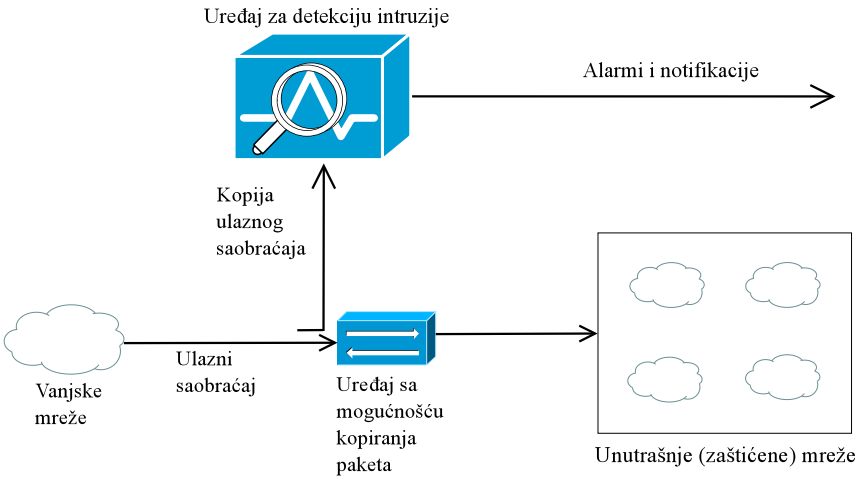
Napadi na nižim slojevima OSI modela, analogno aplikativnim napadima, mogu koristiti sigurnosne propuste u dizajnu ili implementaciji mrežnih protokola. Najčešće se izvode slanjem posebno formiranih paketa [6] ili modifikovanjem postupka komunikacije u okviru protokola na način na koji to nije predviđeno originalnim dizajnom [7].

Posebnu podgrupu ovih napada predstavljaju napadi koji se zasnivaju na zauzimanju resursa odredišnog sistema, čime bi legitimnim korisnicima bio onemogućen ili značajno otežan pristup. Kod ovakvih napada servis troši svo dostupno procesorsko vrijeme na obradu nevalidnih zahtjeva. Zbog ovakvog efekta

koji imaju na krajnji sistem, često se nazivaju i Denial-of-Service (DoS) napadi, pri čemu njihovo generisanje najčešće uključuje slanje velikog broja paketa (engl. *flooding*). Jedan od najpoznatijih primjera *flooding* napada je SynFlood, koji se sastoji iz iniciranja ogromnog broja *SYN* zahtjeva koje predstavljaju početni korak kod uspostavljanja komunikacije kod TCP protokola (*3-way handshake*) [8]. Nakon odgovora servera, napadač ne kompletira proceduru uspostavljanja konekcije (konekcija ostaje poluotvorena), već prelazi na otvaranje nove. Inicirane konekcije ostaju otvorene neko vrijeme, čime se smanjuje dostupna količina resursa (između ostalog, broj slobodnih portova za nove korisnike i količina dostupne memorije).

S druge strane, DoS napadi ne moraju koristiti posebno formiran sadržaj paketa. Mogu biti izvedeni slanjem paketa dobijenih praćenjem standardne komunikacije na nekoj proizvoljnoj računarskoj mreži. Iako server koji je meta ovakvog napada nije adresiran kao primalac generisanih paketa, da bi donio odluku o njihovom odbacivanju mora izvršiti dekapsulaciju zaglavlja i analizirati adresu primaoca i pošiljaoca. Sama ova obrada može predstavljati dovoljno opterećenje po primaoca tako da je svo procesorsko vrijeme utrošeno na obradu saobraćaja koji će sigurno biti odbačen. Posebna varijanta istog napada može uključivati pakete čiji je sadržaj izmijenjen tako da u polju predviđenom za adresu primaoca sadrži adresu napadnutog mrežnog čvora, čime bi se spriječilo jednostavno odbacivanje paketa na osnovu nepostojeće adrese primaoca na samom ulazu u mrežu. Pritom treba imati u vidu da adresirani pošiljalac (tj. onaj čija je adresa upisana u odgovarajućem polju paketa) ne mora biti i stvarni pošiljalac paketa.

U zavisnosti od broja napadača, DoS napadi se dijele na standardne (nedistribuirane) i distribuirane. Standardni napadi potiču od samo jednog napadača (ili malog broja napadača) i u određenoj mjeri su jednostavniji za detektovanje jer je moguće pratiti količinu saobraćaja koja dolazi od pojedinačnih pošiljalaca. S druge strane, kod distribuiranih DoS napada (DDoS) saobraćaj dolazi sa velikog broja različitih lokacija. Pošiljaoci su najčešće korisnički računari (tzv. *botovi*) koje stvarni napadač kontroliše sa udaljene lokacije instaliranjem odgovarajućeg malicioznog softvera. Napad u tom slučaju počinje tako što se slanje paketa istovremeno inicira sa svih kompromitovanih hostova, dok stvarni napadač uopšte ne mora biti uključen u aktivno izvođenje napada. Nakon agregiranja poslatog saobraćaja na ulazu u mrežu u kojoj se nalazi cilj napada, stvara se kritična količina saobraćaja potrebna za prekid rada servisa. Izolovanje saobraćaja koji predstavlja dio ovakvog napada je gotovo nemoguće, jer je broj paketa koji upućuje svaki pojedinačni pošiljalac dovoljno mali da se ničim ne izdvaja od saobraćaja koji šalju legitimni korisnici. Dodatno, ovaj saobraćaj može biti i sastavljen od prikupljenih legitimnih zahtjeva za nekim od postojećih servisa na napadnutom hostu, čime se proces odbacivanja takvih paketa dodatno otežava.



Slika 2.1 *Implementacija IDS uređaja u računarskoj mreži*

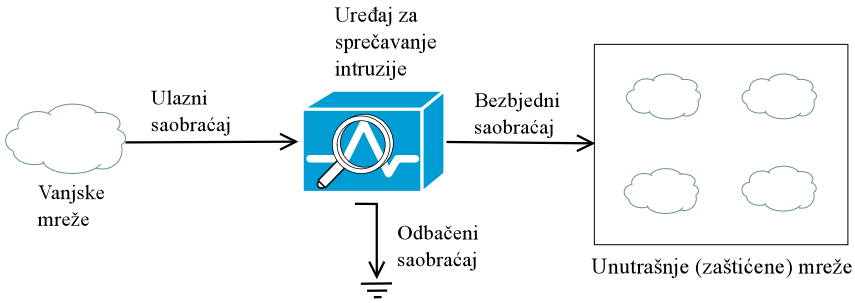
2.2 Osobine sistema za detekciju i prevenciju intruzije

Sistemi za detekciju i prevenciju intruzije u računarskim mrežama se primarno razlikuju u načinu integracije u mrežu, kao i u akcijama koje preduzimaju u situacijama kada je potvrđeno postojanje neke maliciozne aktivnosti.

Zbog velikih razlika u napadima na različitim slojevima OSI modela, u praksi nije izvodljiva implementacija rješenja za detekciju ili prevenciju koje bi pokrilo sve moguće napade [9]. Rješenja koja su ograničena na samo jednu klasu napada unose manje kašnjenje u komunikaciji zbog manjeg broja operacija koje se moraju izvršiti nad svakim pojedinačnim paketom, čime se smanjuje i kompleksnost rješenja.

Sistem za detekciju intruzije (engl. *Intrusion Detection System - IDS*) predstavlja pasivni sigurnosni uređaj čija je osnovna svrha otkrivanje napada. IDS sistemi su dizajnirani tako da nakon uspješne detekcije obavještavaju administratora ili neki drugi mrežni uređaj o postojanju sumnjivih aktivnosti na mreži. Kao posljedica toga, IDS ne mora aktivno učestvovati u prenosu paketa, pa se ne mora nalaziti na putanji kojom se komunikacija odvija sa unutrašnjom (zaštićenom) mrežom (na primarnoj putanji paketa). Najčešća implementacija uključuje postojanje mrežnog uređaja koji šalje kopiju saobraćaja prema IDS-u za analizu, dok se originalni saobraćaj upućuje dalje prema unutrašnjem dijelu mreže. Ovim se eliminiše kašnjenje koje bi IDS unosio i omogućava nastavak komunikacije bez obzira na kompleksnost procedure za detekciju koju IDS koristi. Za to je potrebno posebno konfigurisati mrežni uređaj, ali uglavnom ne predstavlja dodatno opterećenje po pitanju performansi kompletne mreže [10]. Samim tim, ne postoje striktna ograničenja po pitanju brzine analize i obrade paketa, jer ona u svakom slučaju ne utiče direktno na njihovu isporuku na odredište. Na slici 2.1 je prikazana tipična implementacija IDS uređaja u računarskoj mreži.

Nedostatak IDS uređaja leži u visokom stepenu generisanih alarma [11]. Tipičan mrežni saobraćaj sadrži veliki broj neispravnih paketa za koje IDS može



Slika 2.2 *Implementacija IPS uređaja u računarskoj mreži*

samo generisati notifikacije, iako bi bilo dovoljno odbacivati takve pakete bez obavještenja. S druge strane, IDS je izložen manjem riziku da i sam bude meta napada jer se ne nalazi na primarnoj putanji komunikacije - u opštem slučaju ne mora biti vidljiv hostovima koji se nalaze izvan mreže. I u slučaju prekida njegovog rada neće doći do problema u komunikaciji između legitimnih korisnika i servisa koji se nalaze na unutrašnjem dijelu mreže.

Uređaji za prevenciju intruzije (engl. *Intrusion Prevention Device - IPS*) su aktivni mrežni čvorovi čija je osnovna funkcija analiza saobraćaja u realnom vremenu i njegovo filtriranje u slučaju detektovanja malicioznih aktivnosti [12], [13], što uslovljava njihovo pozicioniranje na primarnoj putanji paketa između potencijalnih napadača i zaštićenog dijela mreže. IPS mora funkcionisati tako da ne unosi velika kašnjenja u komunikaciji, kao i da kašnjenje koje mora biti uneseno zbog obrade paketa bude konstantno u toku rada. Na slici 2.2 je prikazana tipična implementacija IPS sistema u računarskoj mreži.

Najveća efikasnost IDS i IPS uređaja se postiže njihovim pozicioniranjem na samom ulazu zaštićene mreže, tako da imaju pristup agregiranom saobraćaju koji ulazi u mrežu (kod IDS uređaja ovo pozicioniranje se odnosi na mrežni uređaj koji vrši replikaciju podataka). Na ovaj način uređaji za detekciju imaju kompletnu sliku komunikacije, povećavajući time šanse za uspješnu detekciju. S druge strane, ovakvim pozicioniranjem se kod IPS uređaja otvara mogućnost da oni postanu "usko grlo" u komunikaciji i da visoka opterećenja dovedu do narušavanja performansi čitave mreže. Stoga je jedan od najvažnijih ciljeva svakog IPS-a da omogući jednostavnu skalabilnost, ali da proširivanje kapaciteta, odnosno distribuiranje procesne logike ne naruši ispravnost algoritma za detekciju. Procesni elementi u sastavu IPS-a moraju zadržati ispravnu analitičku sliku saobraćaja koji se prenosi, bez obzira što će količina saobraćaja koju svaki procesni element obrađuje biti proporcionalno manja. Posebna klasa (D)DoS napada za cilj ima upravo onesposobljavanje IPS čvora u mreži, jer se prekidom njegovog funkcionisanja prekida i veza sa dijelom mreže koji se nalazi iza njega. Ovo je jedan od problema kojem je posvećena posebna pažnja u rješenju koje je razvijeno u okviru ovog rada.

Jedan od najčešćih načina klasifikacije IDS/IPS sistema se odnosi na način obrade pristiglih paketa. Za neke vrste napada potrebno je analizirati zaglavlja i sadržaje prenosnih jedinica (engl. *Protocol Data Unit - PDU*) sa različitih slojeva OSI modela. Ovakav način obrade naziva se duboka analiza paketa (engl. *Deep Packet Inspection - DPI*). Sistemi koji vrše DPI obradu paketa unose značajno veća kašnjenja od sistema koji se baziraju na samo jednom sloju, ali omogućavaju

i detekciju šireg spektra napada [14], [15], [16].

U zavisnosti od načina specifikacije potencijalnih napada IDS/IPS sistemi se dijele na sisteme bazirane na potpisu (engl. *signature-based, pattern-based*) i sisteme bazirane na anomalijama (engl. *anomaly-based*) [17], [18]. Kod sistema baziranih na potpisu, konfiguracija sistema sadrži niz šablona koje će sistem koristiti kako bi identifikovao napad. Šabloni se najčešće definišu kroz vrijednosti polja u zaglavljima različitih PDU-ova. Ovakvi sistemi su jednostavniji za razvoj, ali komplikovani za održavanje, jer se procesna logika sistema svodi na jednostavno upoređivanje sadržaja zaglavlja PDU-ova sa definisanim pravilima. Istovremeno, ovakvi sistemi zahtijevaju odlično poznavanje načina funkcionisanja mreže za koju se implementira da bi pravila u svakom trenutku bila ažurna. Poznavanje potpisa napada mora uključuvati i specifikaciju svih potencijalnih varijacija u njegovom načinu izvođenja, što u praktičnim uslovima nije uvijek moguće. Praćenjem i analiziranjem rada IDS/IPS sistema, napadač može modifikovati napad tako da on ne bude detektovan od strane konfigurisanih pravila, što znači da je kod sistema baziranih na potpisu potrebno redovno ažuriranje definicija potpisa kako bi se održao jednak nivo sigurnosti sistema. U literaturi su, zbog jednostavnijeg razvoja i potvrde funkcionalnosti, popularnija rješenja bazirana na potpisima, o čemu će biti više riječi u sekciji 3.1.

S druge strane, za sisteme bazirane na anomalijama je potrebna vrlo mala inicijalna konfiguracija. Rad ovakvih sistema se zasniva na inicijalnom periodu učenja u toku kojeg sistem mjerenjem količine, frekvencije i vrste saobraćaja sam formira statističke podatke koji pokazuju način funkcionisanja mreže u situacijama u kojima nema aktivnog napada (engl. *baseline*). Nakon inicijalnog perioda učenja, sistem je u stanju da poređenjem trenutnog i naučenog stanja u mreži utvrdi da trenutno stanje u dovoljnoj mjeri odstupa od normalnog ponašanja da bi se sa određenim stepenom sigurnosti moglo klasifikovati kao maliciozno. Inicijalnom konfiguracijom se zadaju samo granični parametri koji se koriste kod poređenja trenutnog stanja i *baseline* vrijednosti i vrlo često se mogu dinamički mijenjati u toku rada. Međutim, procesna logika mora biti znatno kompleksnija nego u slučaju sistema baziranih na potpisu. Sistem mora biti u stanju da reaguje i na anomalije koje nisu poznate u trenutku inicijalnog konfigurisanja i da se prilagodi promjeni uslova u mreži kako bi se izbjeglo klasifikovanje redovnih promjena u tokovima saobraćaja kao napada. Rješenje razvijeno za potrebe ovog rada je bazirano na detekciji anomalija na mrežnom sloju OSI modela.

3. Pregled postojećih rješenja i istraživanja

Postojeća rješenja i istraživanja koja su od značaja za rezultate predstavljene u ovom radu se mogu klasifikovati u tri grupe. Najvažniju oblast predstavljaju istraživanja u domenu detekcije i prevencije napada na računarske mreže. Iz te oblasti su izdvojeni oni radovi koji se bave DoS napadima (u distribuiranom i nedistribuiranom obliku), kao i tehnikama detekcije takvih napada u realnim okruženjima. Drugi značajan segment istraživanja se odnosi na softverski definisane mreže, koje predstavljaju tehnologiju koja je u rješenju razvijenom za potrebe ovog rada iskorištena da bi se omogućila programabilnost i fleksibilnost sistema. Konačno, s obzirom da je za potrebe rada razvijen i algoritam za raspoređivanje opterećenja po procesnim elementima, detaljno su analizirani i radovi iz ove oblasti. Kroz poređenje postojećih rješenja su posebno naglašene prednosti upotrebe SDN tehnologija za implementaciju razvijenog algoritma za raspoređivanje u odnosu na postojeća rješenja koja u svom radu koriste drugačije mehanizme iste namjene.

3.1 Sigurnost računarskih mreža, DoS napadi i rješenja za detekciju i prevenciju

Koncept detekcije i prevencije intruzije u računarskim sistemima nije nov u literaturi. Kao jedan od prvih značajnih dokumenata na tu temu izdvaja se tehnički izvještaj za potrebe Američkog vazduhoplovstva (USAF) [19] u kojem su definisani najznačajniji pojmovi vezani za sigurnosne mehanizme. Iako je sa tehničkog aspekta većina rješenja iznesenih u dokumentu prevaziđena, neki od principa zaštite sigurnosti su validni i u današnjim sistemima. Definisan je koncept malicioznog korisnika i maliciozne upotrebe koji nastaju kao posljedica korištenja otvorenih sistema, te se u dokumentu predlaže razvoj mehanizma koji će omogućiti kontrolu izvršnih programa u realnom vremenu kako bi se izbjegao neovlašten pristup zaštićenim dijelovima sistema (tzv. monitor referenci). Monitor referenci je hardversko/softverska komponenta koja bi bila zadužena za odobravanje ili odbacivanje zahtjeva za pristupom nekoj komponenti sistema. Monitor bi funkcionisao u sprezi sa sigurnosnom politikom, putem koje bi za svakog korisnika bio definisan niz dozvoljenih akcija na sistemu. Svaki zahtjev je svrstan u jednu od tri grupe, u zavisnosti od traženih privilegija: čitanje, modifikacija i izvršavanje.

Sličnosti sa idejama iznesenim u ovom tehničkom izvještaju su vidljive i u

današnjim računarskim sistemima opšte namjene, počevši od koncepta zaštićenog načina rada operativnih sistema, pa do sistema za analizu mrežnog saobraćaja u realnom vremenu.

Koncept monitora referenci i zahtjevi koje on mora ispuniti su definisani i u [20],[21]. Neki od tih zahtjeva se mogu prenijeti bez izmjene i na moderne sisteme:

- kompletnost - kontroli podliježu svi korisnički zahtjevi. U kontekstu monitora referenci, ovdje je riječ o zahtjevima upućenim za resursom na lokalnom sistemu, ali se zahtjev može generalizovati tako da se odnosi na sve elemente sistema, odnosno da sistem nema mogućnost razlikovanja zahtjeva i korisnika, već da svaka upotreba sistema mora biti u skladu sa definisanom sigurnosnom politikom,
- zaštićenost samog sistema - nije moguće izvršiti modifikaciju sigurnosne politike i na taj način zaobići kontrolu. Šire gledano, ovo je jedan od zahtjeva kojem je i u ovom radu posvećena posebna pažnja. Naime, svaki efikasan sigurnosni sistem mora biti projektovan na takav način da, osim napada usmjerenih prema zaštićenim resursima, onemogući napade koji pristup neovlaštenim dijelovima sistema traže kroz onesposobljavanje sigurnosnih mehanizama, tj. napade usmjerene protiv same sigurnosne politike,
- validnost - sistem mora biti projektovan tako da omogući validaciju rezultata i jasno razgraničenje između ispravne i neispravne upotrebe. Ovo se postiže kroz jasno definisanje sigurnosne politike, te validaciju i kompletnost definisanih pravila. Taj zadatak je nešto jednostavniji na neadaptivnim sistemima koji ne mogu prilagođavati trenutna pravila u zavisnosti od stanja okruženja, već se oslanjaju na statičku definiciju kompletne sigurnosne politike u trenutku inicijalne konfiguracije.

Opisani radovi pripadaju grupi rješenja koja funkcionišu u realnom vremenu. Paralelno sa ovakvim rješenjima, u literaturi se pojavljuju i radovi na temu detekcije sigurnosnih propusta koji ne rade u realnom vremenu (engl. *offline*), već se zasnivaju na analizi logova i podataka dobijenih radom sistema (engl. *log auditing*). Među prvim radovima u kojima je definisana ideja analiziranja logova u svrhu detekcije intruzije treba navesti [22], gdje je predložen koncept definisanja posebne notacije kojom bi se specifikovali šabloni (engl. *patterns*) zapisa u logovima koji bi predstavljali dokaz o neovlaštenoj upotrebi. Ovo je jedan od prvih radova u kojima je dat pregled tada poznatih napada na operativne sisteme.

Ideja analiziranja logova operativnog sistema u svrhu detekcije intruzije je iskorištena i u radu [23], koji definiše prijetnju kao mogućnost da neovlašteni korisnik sistema izvrši jednu od sljedećih akcija:

- pristup informacijama,
- manipulacija informacijama,
- dovođenje kompletnog sistema u nepouzdano ili neupotrebljivo stanje.

Osim navedene podjele, u istom radu je definisana podjela napadača u zavisnosti od načina izvođenja napada i željenog efekta koji napad treba da postigne. S obzirom na to da se radi o upotrebi operativnog sistema, koncept korisnika je ograničen na izvršavanje programa ili pristup podacima.

Napadači su na najvišem nivou podijeljeni na vanjske i unutrašnje. Unutrašnji napadači pripadaju grupi korisnika koji imaju ovlaštenja za korištenje jednog dijela sistema, ali različitim tehnikama nastoje dobiti pristup elementima sistema koji nisu pokriveni datim ovlaštenjima. S druge strane, vanjski napadači nemaju ovlaštenje za upotrebu sistema, već prvenstveno imaju za cilj pronalaženje sigurnosnog propusta koji će im omogućiti neometan ulaz u sistem [24]. Ovakva podjela korisnika se može iskoristiti i kod modernih sistema, jer korisnici koji imaju ovlaštenje za upotrebu sistema (ili se nalaze unutar zaštićene mreže) ne podliježu jednakim sigurnosnim mjerama i procedurama detekcije.

Razvojem tehnologije i povećanjem brzine komunikacije u računarskim mrežama, dolazi do specijalizacije istraživanja iz oblasti sigurnosti, posebno prema tipovima napada i načinima njihovog izvođenja [11]. U radu [25] je data detaljna analiza svih aspekata DDoS napada i mehanizama odbrane. Prema tipu, napadi su klasifikovani u zavisnosti od sloja OSI modela na kojem se manifestuju. Kako je već opisano u uvodnom dijelu rada, razlike između napada koji pripadaju različitim grupama su do te mjere izražene da čine jako nepraktičnim bilo kakvo rješenje koje bi omogućilo detekciju svih vrsta napada. Napadi na aplikativnom sloju su najčešće usmjereni prema konkretnim servisima i usko su prilagođeni načinu funkcionisanja ili aplikativnom protokolu koji taj servis koristi. Sa druge strane, napadi koji funkcionišu na nižim slojevima ne razlikuju osobine konkretnih aplikacija, već zajedničku osnovu za komunikaciju koju pružaju protokoli na tim slojevima. Navedene grupe se djelimično preklapaju u slučaju napada koji za cilj imaju zauzimanje resursa sistema, jer se vrlo često manifestuju na svim slojevima sistema (iako su na aplikativnom sloju usmjereni protiv jednog konkretnog servisa).

Od posebnog značaja za ovaj rad je analiza grupa napada koji funkcionišu na mrežnom/transportnom sloju OSI modela i koja je u [25] podijeljena na sljedeće podgrupe:

- napadi koji generišu ogromne količine saobraćaja u cilju potrošnje propusnog opsega mreže (engl. *flooding*) - kod ovog tipa napada napadač ima najveću slobodu formiranja paketa i njihovog slanja prema odredišnoj mreži. Sam sadržaj nije u tolikoj mjeri bitan za izvođenje napada koliko činjenica da je već samim prolaskom saobraćaja kroz mrežu do odredišta cilj napada ispunjen. Napadnuti čvor mora obraditi sve dobijene pakete kako bi utvrdio njihov sadržaj i odredio da li je potrebno odgovoriti na dobijene zahtjeve ili se radi o paketima nevalidnog sadržaja. U toj obradi će biti angažovana velika količina sistemskih resursa, te će sistem ostati nedostupan za druge korisnike,
- napadi koji koriste konkretnu osobinu nekog protokola (engl. *exploit*) - kod ovakvih napada količina saobraćaja ne mora biti presudna za njihovo uspješno izvođenje. Napadi se izvode tako da se formira poseban sadržaj čijom će se obradom izazvati reakcija odredišnog sistema koja će napadaču dati pristup povjerljivim podacima ili će izazvati prekid rada jednog ili više servisa. Taj sadržaj može iskoristiti način funkcionisanja protokola (npr. kod *SynFlood* napada [7]), neku anomaliju u dizajnu protokola ili nedostatak u implementaciji ispravno dizajniranog protokola [1],

[1] jedan od najpoznatijih primjera je nedostatak u implementaciji SSL komunikacije kod

- napadi bazirani na reflektorima - napadi iz ove grupe se razlikuju po načinu izvođenja, dok po osobinama pripadaju *flooding* napadima. Umjesto direktnog slanja paketa prema meti napada, napadač šalje drugim hostovima (reflektorima) posebno formirane zahtjeve kod kojih je adresa stvarne mete napada upisana u paketu kao adresa pošiljaoca. Nakon obrade zahtjeva, reflektor šalje odgovor hostu koji je meta napada, čime je otežano otkrivanje stvarnog izvora,
- amplifikacioni napadi - napadi koji su po izvedbi vrlo slični napadima baziranim na reflektorima (vrlo često uključuju i reflektore), pri čemu kompromitovani hostovi vrše umnožavanje primljenog sadržaja i njegovo slanje prema krajnjem odredištu. Ovo često uključuje upotrebu *broadcast* i *multicast* načina komunikacije kod kojih je umnožavanje paketa ugrađeno u sam mehanizam slanja paketa.

U istom radu [25] je data i analiza mehanizama odbrane, pri čemu je podjela alata za detekciju i prevenciju izvršena na osnovu mjesta implementacije takvog rješenja. Ovo je jedan od dva najčešća načina klasifikacije IDS i IPS rješenja. Prema ovoj podjeli, alati za detekciju i prevenciju intruzije mogu biti:

- sistemi implementirani na odredištu (engl. *destination based*) - ovakvi sistemi su najčešće u upotrebi iz nekoliko razloga. U prvom redu, postavljanjem sistema blizu napadnutog hosta se dobija realna slika napada, jer se saobraćaj koji dolazi od većeg broja geografski udaljenih napadača agregira pri ulasku u mrežu u kojoj se nalazi odredišni host. Osim toga, za implementaciju rješenja ovakvog tipa je odgovorna najčešće samo jedna kompanija, pa su administrativni problemi implementacije manji nego u situaciji kad je za efikasnost implementiranog rješenja potrebna interakcija većeg broja kompanija ili provajdera. U ovom slučaju, administrator sistema može izabrati rješenje koje odgovara specifičnim osobinama date mreže i ne mora biti adekvatno za implementaciju u nekoj drugoj mreži. Rješenje predstavljeno u ovom radu spada u ovu grupu rješenja,
- sistemi implementirani na izvoru podataka (engl. *source-based*) - ovoj grupi pripadaju sistemi koji teže da minimizuju upotrebu sistemskih resursa kod detekcije postavljanjem sistema bliže izvoru podataka. Prednosti ovakvog sistema se ogledaju u tome što je količina saobraćaja koju je potrebno obraditi u slučaju distribuiranog DoS napada mnogo manja od ukupne količine agregiranog saobraćaja koji bi stigao do odredišta. Zaustavljanjem takvog napada na izvoru, rasterećuju se i mrežni uređaji na putanji do krajnjeg primaoca, jer više nije potrebno uložiti resurse za prenos paketa do odredišta. Međutim, za potpunu efikasnost ovakvih rješenja je najčešće potrebno izvršiti njihovu implementaciju u svakoj mreži u kojoj se nalaze potencijalni napadači. Obzirom da provajderi nemaju motiva za dodatna ulaganja u rješenje koje će pružiti zaštitu drugim mrežama na Internetu, zastupljenost ovakvih rješenja je znatno manja od rješenja implementiranih na odredištu napada,
- mrežni sistemi za zaštitu (engl. *network-based*) - kod ovakvih rješenja, sistem za detekciju se nalazi na mrežnim uređajima autonomnog sistema (AS).

OpenSSL biblioteke poznat pod nazivom Heartbleed

Osnovni nedostatak ovakvog pristupa je velika potrošnja resursa u toku procesa detekcije. Osim toga, kompromitovanje ovakvog sistema može imati i posljedice po druge dijelove sistema (koji ne bi bili obuhvaćeni originalnim napadom), jer se prekidom rada mrežnih uređaja prekida prosljeđivanje svih paketa do odgovarajućih dijelova mreže,

- hibridni sistemi - kod sistema iz ove grupe se kombinuju najbolje karakteristike iz prethodno opisanih grupa kako bi se omogućio efikasan i pouzdan sistem. Udaljavanjem od odredišta napada se smanjuje količina saobraćaja koja se mora obraditi, ali se ujedno gubi kompletna slika sistema i napada koji se prema njemu generiše, pa hibridni sistemi teže pronalaženju kompromisa između navedenih osobina.

U literaturi su dostupne i drugačije klasifikacije DDoS napada. U radu [18] su napadi posmatrani kroz alate kojima se izvode, pa je podjela izvršena na osnovu stepena automatizacije, iskorištenim propustima i dinamici napada (da li se slanje paketa vrši konstantnom ili varijabilnom brzinom). Ovo je bitno sa stanovišta detekcije, jer se napad koji se izvodi varijabilnom brzinom u opštem slučaju može prilagoditi tako da zaobiđe mehanizme detekcije smanjivanjem intenziteta napada u odgovarajućem trenutku.

U radu [9] su date dvije klasifikacije IDS i IPS rješenja. Prema prvoj klasifikaciji, sistemi za detekciju i prevenciju se dijele u zavisnosti od toga da li procedura detekcije funkcioniše u realnom vremenu, ili se zasniva na naknadnoj analizi podataka dobijenih u toku rada mreže. U starijoj literaturi preovladava drugi pristup, jer je za rad u realnom vremenu potrebna znatno veća količina dostupnih resursa. Osim toga, rad u realnom vremenu je znatno korisniji u slučaju mreža velikih brzina, jer se napadi znatno brže odvijaju pa je jedino reakcijom u realnom vremenu moguće na vrijeme detektovati ili zaustaviti takav napad. Nedostaci pasivnog analiziranja logova u modernim računarskim mrežama su detaljno izloženi u [26], gdje je kao problem posebno naglašena činjenica da ovakvi sistemi uglavnom reaguju samo na događaje visokog nivoa (jer se na tim nivoima i vrši logovanje), ali da nemaju dodira sa problemima koji nastaju na nižim slojevima sistema.

Druga podjela se odnosi na način funkcionisanja samog algoritma, o čemu je već bilo riječi u prethodnim poglavljima. Rješenja bazirana na specifikaciji napada (engl. *signature-based*) najčešće uključuju namjenski razvijen jezik kojim je potrebno specifikovati sve napade prije početka rada sistema. S druge strane, rješenja bazirana na anomalijama (engl. *anomaly-based*) detektuju napad poređenjem normalnog rada mreže sa trenutnim stanjem i utvrđivanjem stepena odstupanja od normalnih parametara.

Jedan od najzastupljenijih alata za detekciju i prevenciju baziran na specifikaciji napada je Snort [27]. Snort pruža mogućnost praćenja i čuvanja paketa (engl. *sniffer*), kao i aktivnog detektovanja napada kroz namjenski definisan jezik za specifikaciju pravila na osnovu kojih se analiziraju paketi. Snort omogućava definisanje vrlo detaljnih pravila koja uključuju vrijednosti većine polja iz zaglavlja svih PDU-ova, kao i odgovarajuću akciju koju treba preduzeti u slučaju da je pronađen paket koji odgovara nekom od definisanih pravila.

Međutim, kao i slučaju ostalih rješenja baziranih na potpisu, postojanje Snorta u mreži ne znači samo po sebi uspješnu detekciju napada, već efikasnost detekcije

zavisi od načina implementacije samog rješenja i definicije pravila. U literaturi je dostupan veliki broj rješenja koja kao polaznu tačku za detekciju uzimaju Snort, te na osnovu njegovih mogućnosti predlažu implementaciju efikasnog sistema za detekciju. U radu [28] je predstavljeno rješenje za detekciju napada korištenjem Snort-a u *cloud* okruženju. Kao jedan od značajnih problema implementacije koji je razmatran je problem skaliranja takvog sistema u *cloud*-u, jer je potrebno omogućiti ispravno funkcionisanje sistema i u uslovima jako velikih brzina prenosa podataka. Osim toga, u radu se govori i o optimalnoj lokaciji za postavljanje bilo kojeg rješenja za detekciju i njenog uticaja na efikasnost, performanse i sigurnost cjelokupnog sistema.

Sličan pristup je iskorišten i u [29], pri čemu su mogućnosti Snort-a proširene korištenjem OpenFlow protokola kako bi se omogućila programabilnost sistema. U [30] je pokazana primjena Snort-a u kombinaciji sa *firewall* servisima kakav je *iptables*. Modularna struktura Snort-a i mogućnost specifikovanja pravila na osnovu bilo kojeg polja u većini zaglavlja, daju mogućnost primjene i za zaštitu mreža preko kojih se komunikacija odvija korištenjem SIP protokola za glasovnu komunikaciju [17].

Detekcija i prevencija intruzije bazirana na anomalijama je u novije vrijeme dobila na popularnosti u literaturi, jer je povećanjem brzine komunikacije na Internetu značajno olakšano izvođenje DoS napada, pa je teže pridvidjeti sve moguće načine odvijanja napada prije pokretanja sistema za detekciju. Ovo je posebno značajno za *cloud* infrastrukture, jer se radi o velikim količinama podataka, velikom broju različitih protokola za komunikaciju i velikom broju hostova koji su potencijalne mete napada.

Od presudnog značaja za efikasnost detekcije bazirane na anomalijama je izbor metrike za klasifikaciju događaja. Događaj predstavlja neku promjenu u manifestaciji saobraćaja na osnovu koje sistem detektuje napad i mora biti odabran tako da napadač ne može prilagoditi osobine napada i izbjeći detekciju (odnosno, da u slučaju svakog takvog prilagođavanja napad postaje neefektivan).

Jedan od čestih pristupa detekciji baziranoj na anomalijama je otkrivanje nepravilnosti u razmjeni poruka koje sugerišu postojanje malicioznih aktivnosti. Za veliku količinu podataka, ovakav pristup traži i veliku količinu sistemskih resursa jer je uglavnom potrebno pratiti svaki pojedinačan tok komunikacije, što otežava (ili čak onemogućava) skaliranje sistema. Konkretno, sistem prati kompletnu komunikaciju kako bi utvrdio broj zahtjeva na koji nije dobijen odgovor, broj nekompletiranih sekvenci razmjene poruka koje moraju biti ispoštovane kako bi se komunikacija nastavila (npr. *three-way handshake* kod TCP protokola), broj prekinutih konekcija itd. [31], [32]. Ovakvi događaji mogu biti detektovani u realnom vremenu ili dobijeni analiziranjem logova na sistemu, nakon čega se statističkom analizom utvrđuje da li je njihova frekvencija dovoljno velika da bi takav događaj bio klasifikovan kao napad.

Među rješenjima koja pripadaju ovoj grupi treba izdvojiti D-WARD [33], [34], [35], koji je značajan iz nekoliko razloga. D-WARD predstavlja kompletno rješenje koje je bazirano na anomalijama i koje koristi nešto drugačiji pristup prevenciji napada od pristupa koji je najčešće korišten u literaturi. Autori sistema polaze od pretpostavke da će u periodima rada mreže u kojima nema aktivnog (D)DoS

napada većina zahtjeva biti propraćena odgovarajućim odgovorima i da će se komunikacija odvijati bez zastoja. Nakon početka napada, većinu saobraćaja čine paketi koji ne predstavljaju validne zahtjeve, pa će broj odgovora biti znatno manji od broja upućenih zahtjeva. Smatra se da će ovakva pretpostavka važiti i u slučaju UDP-baziranog napada, jer će broj odgovora na TCP ili ICMP pakete biti smanjen zbog zauzeća propusnog opsega do kojeg će napad dovesti. Ovim se omogućava uspješna detekcija čak i u slučajevima kada je napad baziran na protokolima koji ne uključuju dvosmjernu komunikaciju. Ovakav pristup ima i svojih slabosti, naročito u slučaju kad su napadači svjesni načina funkcionisanja sistema za detekciju. Dio napada može uključivati i posebno formirane pakete koji će od strane sistema za detekciju biti klasifikovani kao odgovori na zahtjeve, pa sistem neće biti u mogućnosti da detektuje anomalije u komunikaciji i da na osnovu toga preduzme odgovarajuće mjere.

Pristup prevenciji napada je kod D-WARD-a mnogo značajniji od procesa detekcije. D-WARD pripada grupi rješenja baziranih na izvoru napada (engl. *source-based*) koji u prevenciji koriste činjenicu da je količina saobraćaja koju je potrebno blokirati manja što se uređaj za detekciju nalazi bliže napadaču. Za implementaciju rješenja se koriste modifikovani ruteri (tzv. D-WARD ruteri) koji su postavljeni u mreži napadača. Ruter je odgovoran za blokiranje saobraćaja koji dolazi od hosta za koji se utvrdi da učestvuje u napadu na osnovu ranije opisanog algoritma za detekciju.

Problem sa ovakvim pristupom leži u motivu za implementaciju ovakvog sistema od strane provajdera Internet usluga (engl. *Internet Service Provider, ISP*). ISP koji u svojoj mreži instalira ovakav mrežni uređaj ne doprinosi sigurnosti sopstvene mreže, nego štiti druge mreže od napadača koji se nalaze u njegovoj mreži, pa bi za potpunu efikasnost sistema bi bilo potrebno implementirati data rješenja u svim mrežama. U slučaju djelimične pokrivenosti, i dalje postoji opasnost od napada koji dolaze sa onih dijelova mreže koji nisu zaštićeni D-WARD-om. Implementacija ovakvog sistema bi za manje provajdere predstavljala dodatno ulaganje u infrastrukturu koja im ne donosi direktnu korist i povećanu sigurnost, što potencijalno predstavlja ograničavajući faktor za široku prihvaćenost svih rješenja ovog tipa. Iako postoje rješenja koja podrazumijevaju komunikaciju između različitih provajdera i koja su implementirana u velikom broju mreža [36], njihovo proširenje na kompletan Internet je upitno.

Rješenje pod nazivom SIFF (engl. *Stateless Internet Flow Filter*) [37] takođe predstavlja primjer sistema za čiju potpunu funkcionalnost je potrebna izuzetno široka prihvaćenost. SIFF podrazumijeva modifikaciju zaglavlja IP paketa kako bi se na poseban način označili paketi koji predstavljaju dio legitimne komunikacije između klijenta i servera. Oznaka je implementirana tako da komunikacija sa hostovima koji nemaju mogućnost razumijevanja ovih oznaka nije degradirana. Na pakete koji sadrže odgovarajuću oznaku se primjenjuju posebna pravila kvaliteta servisa (engl. *QoS*) koja osiguravaju brži prenos i manje kašnjenje. Slično, rješenje pod imenom SENSS [38], [39], [40] podrazumijeva omogućavanje napadnutom hostu da od provajdera iz kojeg dolazi napad zatraži blokiranje nekog dijela saobraćaja. Skaliranje svih sistema ovakvog tipa je direktno zavisno od broja hostova koji podržavaju predložene izmjene, pa su prisutni isti problemi prihvaćenosti kao kod D-WARD rješenja.

Sistem predložen u [41] uključuje dodatni korak u komunikaciji koji je potrebno izvršiti da bi se ostvarila konekcija. Ovaj korak je označen kao zagonetka (engl. *puzzle*) i predstavlja matematički zadatak koji server šalje klijentima. Nakon izračunavanja i slanja rezultata, server vrši validaciju dobijenih podataka i dozvoljava klijentu dalju komunikaciju i upućivanje stvarnog zahtjeva prema serveru.

Zajednički problem kod svih rješenja koja funkcionišu praćenjem ili akcijom nad svakim paketom u komunikaciji predstavlja skaliranje sistema. Sa porastom brzine raste i broj paketa koje je potrebno obraditi u istom vremenskom intervalu kako ne bi bilo kašnjenja u prenosu koje dolazi kao posljedica rada sistema za detekciju. Jedno od predloženih rješenja koristi činjenicu da je u virtuelnom okruženju (kakva je i većina *cloud* okruženja) relativno jednostavno proširiti procesnu moć sistema uvođenjem novih procesnih čvorova. Takvim pristupom bi se, u situacijama povećanog opterećenja, sistem automatski aktivirao dodatne procesne elemente da bi lakše bila obrađena veća količina saobraćaja [42].

Drugačije rješenje ovog problema su sistemi koji ne donose odluku na osnovu pojedinačnih paketa, već detekciju vrše statističkom analizom velikih količina podataka. U takvoj postavci pojedinačni paketi nemaju poseban značaj, nego ukupna slika saobraćaja u datom trenutku. Rješenja iz ove grupe su pogodna za detekciju napada koji se zasnivaju na slanju ogromne količine podataka koji se ne moraju sadržajem izdvajati od validnog saobraćaja (paketi koji čine napad mogu čak biti dobijeni prikupljanjem validnog saobraćaja sa iste mreže koja je napadnuta). I u ovom slučaju je izbor metrike presudan, jer osim mogućnosti detekcije utiče i na performanse sistema. Na primjer, rješenje predloženo u [43] koristi brzinu (frekvenciju) pristizanja paketa, pri čemu se pretpostavlja da će u slučaju napada ta frekvencija biti znatno veća nego u slučaju normalnog toka saobraćaja. Ovakav način detekcije je teško zaobići prilagođavanjem načina izvođenja napada bez gubitka njegove efektivnosti. Rješenje koristi hardversku implementaciju, što mu povećava brzinu rada, ali zbog cijene umanjuje mogućnosti skaliranja na velike mreže. Mjerenje frekvencije pristizanja uključuje mjerenje proteklog vremena između svaka dva susjedna paketa, što i u slučaju hardverske implementacije može predstavljati veliko opterećenje za sistem.

Najšire mogućnosti detekcije pruža upotreba entropije, pa je ova tehnika dosta zastupljena u literaturi. Entropija daje dobre rezultate jer omogućava detekciju promjena u obimu saobraćaja i njegovom rasporedu prema proizvoljnim vrijednostima iz zaglavlja PDU-a, bez obzira na broj paketa ili brzinu izvođenja napada. Čak i kod promjene obima saobraćaja, ako je raspodjela konstantna, to se neće odraziti na vrijednost entropije, pa je sistem otporan na povećanja i smanjivanja broja paketa koji su redovna pojava u svakoj mreži tokom dana, a ne predstavljaju napad.

Ove osobine čine entropiju pogodnom za detekciju napada u *cloud* okruženjima i mrežama sa velikim propusnim opsegom i velikim brojem aktivnih hostova u svakom trenutku. Iz tog razloga se i najveći dio radova bavi upravo primjenom detekcije bazirane na entropiji na ovakva okruženja. Rješenja predstavljena u [44], [45] koriste upravo razlike u vrijednostima entropije koje nastaju kao posljedica promjene u raspodjeli saobraćaja kako bi izvršila detekciju DDoS napada u mrežama Internet provajdera. Osnovni problem sa kojim se suočava rješenje predloženo u

[44] je distribucija logike za detekciju tako da su pokrivene sve ulazne tačke u mreži, čime bi napadaču bilo onemogućeno da rasporedi napad preko različitih putanja kako bi mu prividno umanjio intenzitet i otežao detekciju. Uređaji za detekciju koji su postavljeni na svim ulaznim tačkama u mrežu računaju pojedinačne vrijednosti entropije, nakon čega se računa zajednička vrijednost na centralnoj lokaciji. Na osnovu zajedničke vrijednosti se donosi odluka o postojanju aktivnog napada.

Oba rješenja prate trenutno stanje entropije i porede ga sa vrijednostima za koje je inicijalnim učenjem utvrđeno da su bezbjedne, tj. da predstavljaju standardnu entropiju u periodima u kojima nema aktivnih napada. Međutim, rješenja ne nude odgovore na probleme skaliranja i raspodjele opterećenja, što u *cloud* okruženjima predstavlja zahtjev od izuzetnog značaja. Iako se pojedinačni paketi ne mogu zasebno iskoristiti za detekciju, i dalje moraju biti obrađeni kako bi se izdvojila vrijednost polja koja će biti korištena za računanje entropije. Povećanjem broja paketa koje je potrebno obraditi u kratkom vremenskom periodu dolazi do potencijalnog uskog grla u komunikaciji, pa se mora pronaći rješenje koje će omogućiti proširenje kapaciteta bez narušavanja preciznosti.

Osnovni nedostatak primjene entropije na detekciju DDoS napada predstavljaju neispravne detekcije koje su posljedica pojave koja je u tehničkoj literaturi poznata pod imenom *flash crowd*. *Flash crowd* je naglo povećanje količine saobraćaja upućene prema jednom hostu u mreži koje nastaje kao posljedica objavljivanja interesantnog sadržaja na tom hostu. Nakon objavljivanja, ogroman broj korisnika upućuje zahtjeve prema istom servisu, zbog čega dolazi do degradacije performansi ili potpunog prekida rada servisa. Ako bi se kategorizovao kao napad, *flash crowd* se može svrstati u kategoriju distribuiranih DoS napada zbog ogromnog broja pošiljalaca (posjetilaca) i malog broja paketa po jednom pošiljaocu. Korisnicima koji upućuju zahtjeve prema serveru nije cilj izazivanje ovakvog prekida, ali je efekat upotrebe isti kao u slučaju uspješno izvršenog DDoS napada. U situaciji u kojoj u mreži postoji i *flash crowd* i DDoS napad, nije moguće pouzdano izvršiti razdvajanje saobraćaja koji je dio napada od legitimnih paketa. Upotreba sistema koji ne razlikuje ove dvije pojave na mreži [46] bi u rezultovala blokiranjem pristupa korisnika servisu, ali bi integritet servisa bio sačuvan. Postavlja se pitanje da li je potrebno razlikovati *flash crowd* ako se on po posljedicama ne razlikuje od tipičnog napada. Ovo je problem koji je u literaturi zastupljen velikim brojem radova [47], iako objektivno još ne postoji sveobuhvatno i potpuno ispravno rješenje.

Rad [47] sumira metrike za detektovanje *flash crowd*-a tretirajući ga kao tipičan DDoS napad. U radu je predloženo rješenje za detekciju DDoS napada i razlikovanje *flash crowd*-a od stvarnog napada, ali autori iznose veliki broj pretpostavki na kojima se u realnim situacijama ne bi smjela zasnivati detekcija. Naime, predstavljeni algoritam razlikuje DDoS napad od *flash crowd*-a prema jačini napada, iako pažljivo izveden DDoS napad može u potpunosti oponašati *flash crowd* kako bi izbjegao upravo takav način detekcije.

Na sličan način se u radovima [48], [49], [50] predlaže razlikovanje *flash crowd*-a od DDoS napada uvođenjem niza pretpostavki koje su tačne samo u kontrolisanim laboratorijskim uslovima ili u nedovoljno pripremljenim napadima. Autori u [48] pretpostavljaju da se *flash crowd* može razlikovati od standardnog DDoS

napada na osnovu raspodjele IP adresa pošiljaoca. Međutim, većina dostupnih alata za simuliranje DDoS napada nudi mogućnost kontrole korištenih vrijednosti polja u koje je upisana adresa pošiljaoca. Samim tim, napadač je u mogućnosti da simulira bilo kakvu raspodjelu adresa i da time prikrije napad.

Istraživanje dato u [51] pokazuje da je *flash crowd* moguće detektovati u slučaju da je vezan za konkretan protokol aplikativnog sloja, ali da ga nije moguće pouzdano i efikasno potvrditi na mrežnom i transportnom sloju. Predloženi mehanizam uključuje korištenje posebnog grafičkog zadatka (engl. *captcha* [52]) koji se zasniva na prepoznavanju sadržaja slike i predstavlja izuzetno komplikovan problem za automatizovanog korisnika (engl. *bot*), ali je relativno jednostavan za čovjeka. Svaki upućeni zahtjev prema serveru mora proći validaciju ovakvog zadatka prije nego što mu se dozvoli dalja komunikacija. Osim što pokazuje nedostatke u performansama pri povećanju brzine, ovakav pristup nije ni moguće implementirati na nižim slojevima OSI modela jer se komunikacija odvija bez uticaja korisnika. Veliki broj paketa se šalje od strane automatizovanih servisa kako bi dobili informacije potrebne za korištenje protokola aplikativnog sloja (npr. IP adresu vezanu za neko DNS ime).

3.2 Softverski definisane mreže

Brojne prednosti koje donose softverski definisane mreže dovele su do velike popularnosti ove tehnologije i u literaturi. Zbog svoje programabilnosti i jednostavnosti implementacije, našle su primjenu u različitim segmentima informacionih sistema, uključujući i detekciju napada na svim slojevima OSI modela.

U radovima [53] i [54] je dat detaljan pregled stanja SDN tehnologija. Detaljno je opisan istorijski razvoj, počevši od ranih implementacija programabilnih mreža i tehnika mrežne virtuelizacije, pa sve do opisa OpenFlow protokola u onom obliku u kojem se on danas koristi. Dat je i detaljan pregled kontrolera koji su već u upotrebi u SDN mrežama, kao i eksperimentalnih projekata koji za cilj imaju dodatno povećanje programabilnosti sistema.

Pozitivni aspekti SDN tehnologije su vidljivi i kroz različite komercijalne i istraživačke implementacije koje povezuju SDN mreže sa postojećim mrežnim tehnologijama. U radovima [55], [56], [57] su izložene mogućnosti implementacije IPv6 mreža korištenjem SDN tehnologija, kao i potencijalni problemi kod tranzicije sa IPv4 na IPv6 mreže koji mogu biti olakšani ili riješeni korištenjem SDN-a. Što se tiče komercijalnih implementacija, posebno je značajna mreža pod nazivom B4 [58], implementirana od strane kompanije Google koja se koristi u svrhu povezivanja centara sa podacima u različitim dijelovima svijeta i koja, prema istraživanjima, prenosi veću količinu podataka od javne mreže u vlasništvu iste kompanije. SDN je iskorišten u ovoj implementaciji i za obezbjeđivanje kvaliteta servisa kroz dinamičko raspoređivanje opterećenja.

U literaturi su detaljno analizirane i prednosti SDN mreža u odnosu na klasične mrežne arhitekture [59], [60]. Programabilnost mreža je u ovim radovima iskorištena kako bi se implementirala zamjena za VLAN tehnologije i tako olakšala logička izolacija mrežnog saobraćaja.

Značajan dio istraživanja [53] je posvećen i problemima sa kojima se suočavaju SDN mreže. U prvom redu se ističu potencijalni problemi sa performansama koji dolaze kao posljedica dizajna SDN tehnologija. Paket koji ne podliježe postojećim pravilima (najčešće prvi paket svakog toka) se uvijek šalje kontroleru, što može dovesti do latencije u obradi paketa.

Potpuna kontrola primljenih i poslatih paketa, kao i mali broj paketa koji se uopšte prosljeđuje prema kontroleru, omogućava transparentnost za druge uređaje o kojoj je bilo riječi u prethodnim poglavljima. Ova osobina rješava sigurnosne probleme koji nastaju kao rezultat napada usmjerenih direktno protiv SDN elemenata mreže [61].

Od velikog značaja za mogućnost skaliranja SDN mreža je njihova decentralizacija. U standardnom načinu rada, svi mrežni uređaji komuniciraju sa jednim centralnim kontrolerom. Ovo dovodi do potencijalnih problema u slučaju prekida rada kontrolera, i uz to usporava rad kompletne mreže u slučaju da postoji veliki broj mrežnih uređaja čiji rad zavisi od istog kontrolera. Veliki dio istraživanja u oblasti SDN-a se odnosi na mogućnost istovremene upotrebe većeg broja kontrolera u istoj mreži i komplikacije u domenu konfiguracije sistema koje to može izazvati [62].

U radu [63] je dat pregled SDN arhitektura sa većim brojem kontrolera, te su unapređenja koja one donose u odnosu na implementacije sa jednim kontrolerom klasifikovana u tri grupe:

- efikasnost - u mrežama sa većim brojem uređaja se povećanjem broja kontrolera skraćuje vrijeme potrebno da mreža konvergira, odnosno da svi mrežni uređaji pređu u stabilno stanje u kojem će imati ažurne konfiguracione podatke,
- skalabilnost - ako rast SDN mreže prati i povećanje broja kontrolera, performanse kompletnog sistema neće biti narušene,
- redundancija - uvođenjem većeg broja kontrolera se izbjegava nekonzistentnost mreže u slučaju kvara na centralnom kontroleru.

U istom radu je dat i pregled načina organizacije mreža sa većim brojem kontrolera. Pokazano je da nema ograničenja u načinu organizacije takve mreže, te da upotreba hijerarhijske ili jednonivovske organizacije kontrolera, odnosno logičke centralizacije ne zavisi u velikoj mjeri od arhitekture mreže, već više od načina implementacije kontrolera.

Unapređenja SDN-a u domenu povećanja efikasnosti mreža sa više kontrolera su tema istraživanja i u radovima [64] i [65]. Autori u [64] iznose probleme sa održavanjem tabela rutiranja kod mreža sa većim brojem kontrolera, kao i načine raspoređivanja svičeva po kontrolerima. U cilju povećanja efikasnosti, predlaže se implementacija logike za raspoređivanje opterećenja kao dio same kontrolne ravni. Slično, u radu [65] su predložena proširenja mehanizma za slanje konfiguracionih podataka u vidu uključivanja verzije konfiguracije, kojim bi se osigurala konzistentnost konfiguracionih poruka u slučaju postojanja više kontrolera u mreži.

U radovima [66] i [67] je analizirana mogućnost distribuiranja kontrolne ravni

i predložena su unapređenja koja bi takvom sistemu donijela redundantnost i sigurnost u slučaju prekida rada jednog ili više kontrolera. Kao osnovna prepreka za implementaciju ovakvog mehanizma i ovdje se ističe problem sinhronizacije i održavanja konzistentnog stanja između različitih kontrolera u mreži. Predložen je razvoj mehanizma za sinhronizaciju koji minimizuje efekte otkaza kontrolera i omogućava nastavak rada sistema, a sve u okviru postojećih funkcionalnosti koje OpenFlow protokol već podržava.

U velikom broju radova je razmatran problem upravljanja politikama konfiguracije i konfiguracionim pravilima, pogotovo u kompleksnijim mrežama. Ove politike se mogu odnositi na pravila kojima se definiše kvalitet servisa (QoS) [68], ili na uopšteno upravljanje mrežom kroz SDN. U radu [69], autori predlažu jezik za specifikaciju politika za upravljanje mrežom koji bi omogućio mreži da reaguje na događaje nižeg nivoa. Jezik za specifikaciju pravila za funkcionisanje mreže je predložen i u [70], pri čemu je njegova osnovna namjena da riješi probleme sa upravljanjem memorijom kod rada sa velikim adresnim prostorima (npr. kod korištenja ugrađene funkcije jezika koja prikazuje histogram za broj primljenih paketa za kompletan IPv4 adresni prostor).

Zbog programabilnosti i jednostavnosti implementacije, SDN predstavlja pogodnu platformu za razvoj rješenja za detekciju napada baziranih na specifikaciji [71]. Od dinamičkih rješenja za detekciju posebno se izdvaja [72], u kojem je predložen mehanizam detekcije zasnovan na promjeni vrijednosti entropije odredišnih adresa. U radu je potvrđena ideja da će usmjeravanje većeg broja paketa prema malom broju hostova izazvati smanjenje vrijednosti entropije koja se jednostavno može detektovati, pri čemu se izračunavanje vrijednosti entropije vrši na SDN kontroleru. U tom smislu, potencijalni problem u implementaciji ovog rješenja mogu biti nedostaci u domenu performansi SDN komponenti koji su dokumentovani u literaturi [73], [74]. Uprkos ovome, postoji čitav niz radova koji se bavi primjenom softverski definisanih mreža (i, konkretnije, OpenFlow protokola) za statističku analizu saobraćaja, bilo da se radi o primjenama vezanim za detekciju kroz agregiranje tokova podataka [75] ili za karakterizaciju saobraćaja za potrebe dimenzionisanja linkova [76], [77] i koji baziraju svoj rad na agregiranju tokova koji se prate preko mehanizama ugrađenih u sam OpenFlow protokol.

Programabilnost SDN mreža je moguće iskoristiti i za dinamičku promjenu strukture mreže, najčešće u cilju prilagođavanja kapaciteta nekog dijela sistema uslovima mreže. Rješenje predloženo u [78] omogućava dinamičko instanciranje procesnih elemenata za inspekciju paketa (engl. *Deep Packet Inspection, DPI*) u zavisnosti od trenutnog opterećenja i konfiguracije mreže. Rješenje teži da pronađe balans između postavljanja minimalnog broja procesora i maksimalnog opterećenja kojem svaki procesor može biti izložen bez degradiranja performansi.

3.3 Algoritmi za raspoređivanje opterećenja

Algoritmi za raspoređivanje opterećenja su veoma čest predmet istraživanja. Tehnike raspoređivanja opterećenja se primjenjuju na sve oblasti informacionih

tehnologija kako bi se postiglo efikasnije korištenje računarskih resursa. Kod mrežnih komunikacija i obrade paketa u realnom vremenu, ovi algoritmi dobijaju na značaju sa porastom brzine komunikacije, jer veći propusni opseg znači i veći broj paketa koje je potrebno obraditi u istom vremenskom intervalu kako se ne bi unosilo kašnjenje.

Postoje različiti pristupi raspoređivanju opterećenja kod računarskih mreža, koji u zavisnosti od mogućnosti sistema, karakteristika saobraćaja i konfiguracije mreže daju različite rezultate, ali i različite nivoe kompleksnosti implementacije i održavanja [79]. Najčešća podjela sistema i algoritama ove vrste je na proaktivne i reaktivne. Proaktivni sistemi su posebno značajni za SDN mreže, jer je, u opštem slučaju, kontroler odgovoran za uspostavljanje tokova. Većina tokova u tipičnoj mreži su kratkotrajni (engl. *mice flows*) [80], [81], ali se obrađuju na isti način kao i dugotrajni tokovi, jer se odluka o njihovom trajanju ne može donijeti samo na osnovu prvog paketa. U mrežama velikih brzina, komunikacija u periodu inicijalnog popunjavanja tabele prosljeđivanja je iz tog razloga sporija nego u periodu u kojem je sadržaj tabele stabilan. Kod proaktivnog raspoređivanja, kontroler definiše skup pravila za prosljeđivanje koja pokrivaju sve pakete i prije početka same komunikacije. Sa jedne strane, nema potrebe za uspostavom pojedinačnih tokova, pa se od starta koriste prednosti hardverskog prosljeđivanja. S druge strane, obzirom da u trenutku definisanja pravila nije poznato kako će izgledati raspored paketa za različite primaoce, u slučaju proaktivnog definisanja pravila je potrebno u pravilnim intervalima vršiti korekcije kako bi se održalo jednako opterećenje.

Drugačija podjela algoritama je zasnovana na načinu tretiranja pojedinačnih tokova kod definisanja pravila [79]. Naime, pravila mogu biti definisana na nivou pojedinačnog toka, ili uz upotrebu agregacije. Logika za izbor načina rada je slična kao u slučaju izbora vrijednosti DP parametra (sekcija 4.5) za TIDS, jer se raspoređivanjem individualnih tokova dobija precizniji raspored, ali se značajno uvećava broj definisanih pravila. Kroz analizu rezultata datu u sekciji 5.4.1 je pokazano da je agregacija bolji pristup, ali da je potrebno naći kompromis između veličine tabele i što veće dužine prefiksa.

Kao jedna od najznačajnijih prednosti upotrebe algoritama za raspoređivanje se često ističe i manja potrošnja energije. U radu [82] je dat detaljan pregled stanja istraživanja u oblasti algoritama i sistema za raspoređivanje opterećenja, pri čemu su kao osnovni ciljevi uz efikasniju potrošnju energije istaknuti i poboljšanje performansi i stabilnost sistema. Od konkretne primjene sistema zavisi i izbor metrike na osnovu koje će sistem funkcionisati, pa je i efikasnost sistema za raspoređivanje usko vezana za metriku kojoj je prilagođen. U radu je dat čitav niz metrika na osnovu kojih se raspoređivanje opterećenja može vršiti:

- efikasno korištenje propusnog opsega,
- smanjenje količine resursa koja se troši na preraspodjelu poslova,
- povećanje otpornosti na greške,
- trajanje migracije poslova sa jednog elementa na drugi,
- minimizacija vremena čekanja na odgovor,
- potrošnja resursa i energije,

- skalabilnost.

Za ispravno raspoređivanje na osnovu pojedinih metrika je od presudnog značaja da sistem bude "svjestan" sadržaja koji se balansira. To znači da neće na isti način tretirati sve pakete (u opštem slučaju, zadatke), već da će prema sadržaju različitim paketima biti dodjeljivani različiti prioriteti [83], [84], [85], odnosno da veći broj paketa na nekom procesnom elementu ne znači automatski i veće opterećenje sistema. U literaturi postoji veliki broj radova koji predlažu sisteme namjenski prilagođene za konkretne tipove saobraćaja. Na primjer, u [86] je dat prijedlog rješenja za upravljanje opterećenjem sistema za prenos video sadržaja (engl. *video streaming*). Kod takvog sistema ne važi pretpostavka da većinu tokova čine kratkotrajni tokovi, već glavninu saobraćaja čine duži tokovi za koje je u interesu da se ne prekidaju u toku prenosa i da se minimizuje promjena redoslijeda paketa. Za upravljanje su iskorištene SDN komponente za koje su autori iznijeli čitav niz prednosti u odnosu na tradicionalne mehanizme za kontrolu kvaliteta servisa (QoS). Sličan, namjenski definisan pristup se koristi u bilo kojem sistemu u kojem metrika zavisi od konkretnog sadržaja ili specifične infrastrukture mreže [87], [88].

Sistemi opšte namjene, prilagođeni funkcionisanju u *cloud* infrastrukturama, moraju omogućiti jednostavno skaliranje, odnosno prilagođavanje pravila za balansiranje trenutnom stanju mreže koje je podložno naglim promjenama [89]. Iz tog razloga postoji veliki broj radova u kojima su predstavljena rješenja bazirana na SDN-u, zbog svoje visoke programabilnosti i fleksibilnosti [90], [74]. Programabilni sistem za raspoređivanje na bazi SDN-a pruža mogućnost implementacije adaptivnih algoritama koji su u stanju da se prilagode uslovima mreže. Kod takvih sistema najveće opterećenje je na kontroleru, pa se skalabilnost najčešće postiže implementacijom većeg broja kontrolera. Veliki broj radova se bavi problemom raspoređivanja opterećenja između više kontrolera u SDN mrežama, kao i problemom sinhronizacije stanja SDN svičeva prilikom prelaska sa jednog kontrolera na drugi [91]. I ovdje je prisutan problem o kojem je već bilo riječi u prethodnoj sekciji, a koji se tiče problema u sinhronizaciji stanja između kontrolera i njihove međusobne sinhronizacije [92].

Čest pristup rješenju problema sa opterećenjem *cloud* infrastruktura je dinamička alokacija dodatnih resursa, bilo da se radi o aktiviranju novih kontrolera [93], [94] ili dodatnih servera koji će obrađivati krajnje zahtjeve [95], [96]. Podrazumijeva se da ovakva implementacija ima smisla samo u *cloud* okruženju, gdje postoji veća količina dostupnih resursa, dok bi se u ostalim okruženjima postavilo pitanje opravdanosti takvog proširenja sa stanovišta troškova.

Na efikasnost sistema za raspoređivanje opterećenja baziranog na SDN-u utiče i način formiranja pravila za prosljeđivanje. U radu [97] je dat prijedlog tehnike za minimizaciju broja pravila u tabeli prosljeđivanja grupisanjem pravila prema mrežnom prefiksu. Sistem pronalazi šira pravila koja mogu zamijeniti veći broj specifičnih pravila i tako smanjiti ukupan broj unosa u tabeli. Ovakav pristup je efikasan kod mreža kod kojih je saobraćaj raspoređen uniformno po IP adresama prema kojima se kreiraju pravila (bilo da se radi o izvorišnim ili odredišnim adresama), pa je primjenjiv na *cloud* infrastrukture sa jako velikim brojem hostova.

4. Pregled razvijenog rješenja

4.1 Cilj razvijenog rješenja

Osnovni cilj sistema za detekciju intruzije koji je razvijen za potrebe ovog rada (TIDS - *Transparent Intrusion Detection System*) je da pruži odgovor na probleme navedene u sekciji 1.1 i da omogući jednostavnu i pouzdanu detekciju (D)DoS napada u mrežama velikih brzina.

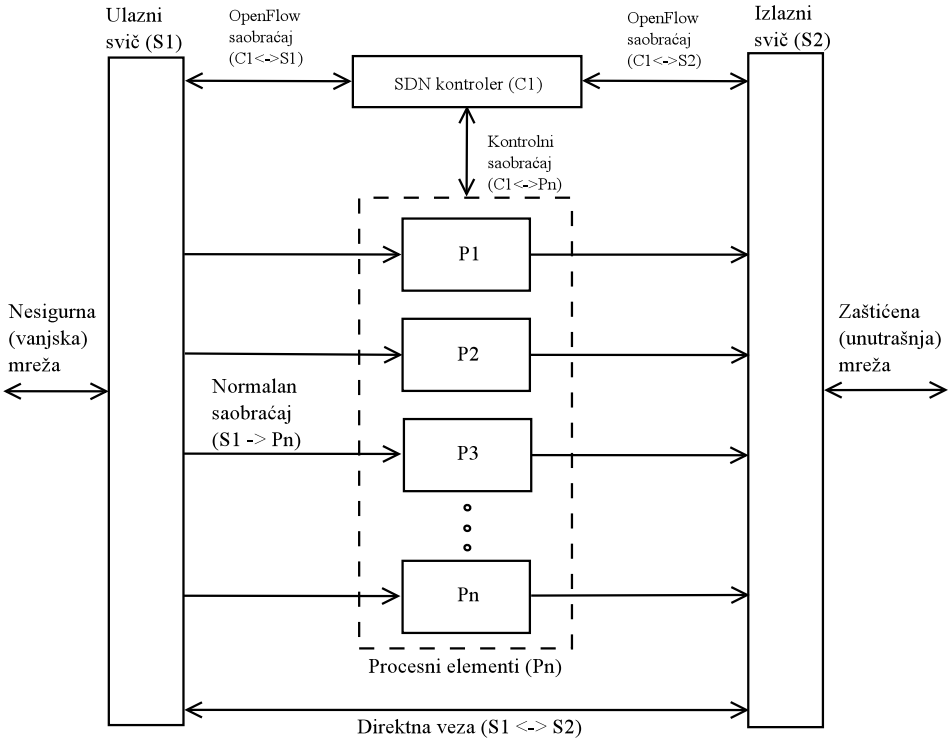
Jedan od primarnih ciljeva sistema je skalabilnost, koja je ostvarena distribuiranjem procesne logike na proizvoljan broj procesnih čvorova, pri čemu je sistem dizajniran na način da sa porastom broja elemenata ne bude narušena mogućnost detekcije napada niti da se unose dodatna kašnjenja u komunikaciji. Sistem je dizajniran tako da može biti implementiran korištenjem nenamjenskog hardvera (engl. *commodity hardware*), čime se značajno smanjuje cijena implementacije i pojednostavljuje procedura proširivanja kapaciteta.

Sigurnosni ciljevi su ostvareni kroz transparentnost sistema, odnosno činjenicu da razvijeni IDS nije vidljiv drugim hostovima na mreži, čime su onemogućeni napadi na sam sistem za detekciju. Razvijeno rješenje funkcioniše primarno kao IDS, ali je u stanju da izvrši prevenciju napada u slučajevima gdje je to moguće (detaljan pregled algoritma za detekciju je dat u poglavlju 4.3.

4.2 Arhitektura rješenja

Sistem je sastavljen od proizvoljnog broja procesnih elemenata koji su u mrežu uključeni korištenjem dva SDN mrežna čvora (sviča) koje kontroliše SDN kontroler. Mrežni saobraćaj koji dolazi iz spoljašnje prema unutrašnjoj (zaštićenoj) mreži se analizira i prosljeđuje (ili blokira) u zavisnosti od definisanih pravila. U analizi saobraćaja se, zbog pojednostavljenja procesne logike, polazi od pretpostavke da maliciozni saobraćaj ne može doći iz unutrašnje mreže, te da izlazni saobraćaj nije potrebno analizirati. U slučaju da postoji potreba i za takvom funkcionalnošću, taj problem bi bio riješen postojanjem još jedne implementacije istog sistema koja bi, u tom slučaju, bila okrenuta u suprotnom smjeru i tretirala bi unutrašnju mrežu kao potencijalno nesigurnu. Izgled arhitekture sistema je dat na slici 4.1.

Svič S_1 predstavlja ulaznu tačku za sav saobraćaj koji je upućen prema zaštićenoj mreži. Njegova funkcija je da prati broj aktivnih procesnih elemenata (P_1 - P_n) i da raspoređuje dolazni saobraćaj tako da opterećenje svakog procesnog



Slika 4.1 *Arhitektura sistema*

elementa bude ujednačeno. Svič S_1 sadrži vezu prema svakom od procesnih elemenata, vezu prema SDN kontroleru (C_1), kao i direktnu vezu prema sviču S_2. Vezama prema procesnim elementima se upućuje isključivo dolazni saobraćaj koji je potrebno analizirati, dok se u suprotnom smjeru šalju samo kontrolni paketi koje svič prosljeđuje kontroleru C_1 i preko kojih se inicira promjena pravila za prosljeđivanje na sviču. Promjena pravila je potrebna prilikom promjena u topologiji procesnih elemenata, u situacijama u kojima se detektuje nejednako opterećenje na aktivnim elementima, ili prilikom detekcije napada.

Direktna veza prema sviču S_2 se koristi za sav saobraćaj koji dolazi iz unutrašnje mreže koji se bez analiziranja prosljeđuje prema spoljašnjoj mreži.

Svič S_2 ima funkciju agregiranja saobraćaja koji dolazi sa procesnih elemenata prema unutrašnjoj mreži, kao i upućivanja saobraćaja koji se kreće u suprotnom smjeru direktno prema izlazu. Kontroler konfiguriše pravila za prosljeđivanje paketa na ovom sviču samo prilikom prvog pokretanja, jer broj procesnih elemenata ne utiče značajno na njegov rad.

SDN kontroler upravlja svičevima i vrši dinamičko konfigurisanje pravila za prosljeđivanje i blokiranje saobraćaja. Kontroler vodi računa o prosljeđivanju svih paketa na mreži, uključujući i pakete generisane od strane susjednih hostova na mreži. Ovi pakete uključuju sve vrste ICMP [98] i ARP [99] paketa, kao i specijalizovane frejmove koji služe za otkrivanje susjednih hostova na mreži. Bez obzira na način njihovog formiranja, elementi TIDS-a ne odgovaraju niti na jedan od ovih zahtjeva, već ih samo (nakon analize) prosljeđuje prema unutrašnjem dijelu mreže. Za pakete mrežnog sloja se ne vrši ni umanjivanje vrijednosti TTL polja, čime se efektivno postiže transparentnost kompletnog sistema, čak i za hostove sa kojima

je on fizički povezan. Dodatno, ovakva implementacija znači da elementi TIDS-a ne moraju biti adresirani ni na jednom sloju OSI modela da bi funkcionisali. Izuzetak ovog pravila može biti implementacija interfejsa za udaljeno upravljanje sistemom koji bi morao biti vidljiv ostalim hostovima na mreži, ali njegovo postojanje ne umanjuje sigurnost sistema jer je pristup tom interfejsu moguće ograničiti na kontrolisani dio mreže unutar kojeg nema rizika od (D)DoS napada.

4.2.1 Softverske komponente

TIDS je sastavljen od nekoliko bitnih softverskih elemenata. Odabrane softverske komponente obezbjeđuju transparentnost i mogućnost obrade paketa bez kašnjenja pri velikim brzinama prenosa, kao i automatsku promjenu konfiguracije sistema u toku njegovog funkcionisanja, a bez uticaja administratora.

4.2.1.1 Procesni elementi

Procesni elementi (procesori) su u kontekstu TIDS-a tretirani kao softverske komponentu sistema jer nema gotovo nikakvih ograničenja po pitanju njihovih hardverskih mogućnosti, a aplikacija koja se na svakom elementu izvršava predstavlja srž kompletnog sistema. Procesni elementi vrše prijem dijela kompletnog saobraćaja od sviča S_1 i prosljeđuju ga nakon analize sviču S_2.

Preuzimanje frejmova se u standardnim mrežnim aplikacijama tipično vrši korištenjem prekida koje generiše mrežna kartica (NIC) [100]. Prekid se generiše prilikom prijema frejma, nakon čega se u prekidnoj rutini vrši njegova rekonstrukcija i prenos prema kernelu operativnog sistema koji je odgovoran za dalju obradu.

Za manje brzine komunikacije ili mreže koje imaju malu iskorištenost linkova, ovakav način obrade frejmova nudi dovoljno dobre performanse, jer vrijeme provedeno u izvršavanju prekidne rutine ne unosi kašnjenje - interval između dva frejma je duži od vremena potrebnog za izvršavanje prekidne rutine. Uvođenje mehanizma za prijem frejmova koji nije baziran na prekidnim rutinama bi u slučaju takvih mreža imalo čak negativan efekat po ukupne performanse sistema [101]. S druge strane, kod mreža velikih brzina (1 Gbps i više) kod kojih je iskorištenost linkova velika, vremenski interval između susjednih frejmova na mreži je znatno manji, pa vrijeme koje sistem utroši na prelazak u prekidnu rutinu i povratak u aktivni program unosi bitno kašnjenje u rad. Pokazuje se da u takvim situacijama upotreba drugačije tehnike za pristup medijumu donosi značajno poboljšanje performansi.

Najčešće korištena tehnika za pristup medijumu kod mreža visokih brzina naziva se prozivanje uređaja (engl. *device polling*). Zasniva se na konstantnom provjeravanju stanja mrežnog interfejsa, čime se elimiše vrijeme provedeno u izvršavanju prekidne rutine. Uz pretpostavku da će se u svakom procesorskom ciklusu na mrežnom interfejsu nalaziti veliki broj novih frejmova, ovakvom tehnikom pristupa se povećava efikasnost rada sistema [101].

Prozivanje uređaja je u okviru TIDS-a implementirano kroz Intelov skup biblioteka pod zajedničkim nazivom DPDK (*Data Plane Development Kit*) [102]. DPDK predstavlja skup biblioteka koje se izvršavaju u korisničkom prostoru i

koje funkcionišu iznad apstrakcionog sloja pod nazivom EAL (*Environment Abstraction Layer*). EAL nudi zajedničku platformu za razvoj aplikacija ovog tipa i pruža funkcionalnosti raspoređivanja zadataka po procesorskim jezgrima, apstrakciju adresnog prostora, upravljanja memorijom i sl. Biblioteka sadrži i optimizovane implementacije različitih struktura podataka kako bi se omogućilo što brže izvršavanje razvijenih aplikacija. Istraživanja pokazuju da je DPDK u stanju da obradi saobraćaj čija brzina prelazi i 40 Gbps [103].

DPDK-bazirane aplikacije funkcionišu tako da se oslanjaju na namjenski razvijen drajver mrežnog interfejsa koji je potrebno aktivirati umjesto podrazumijevanog drajvera koji pruža operativni sistem. Deaktiviranje standardnog drajvera kao posljedicu ima i deaktivaciju standardnih mrežnih servisa koji se oslanjaju na rad tog drajvera, pa aplikacija mora sama implementirati potrebne funkcionalnosti čitanja PDU-ova različitih slojeva. Aplikacija koja je razvijena za potrebe ovog rada implementira funkcionalnosti generisanja i slanja frejmova, jer za rad sistema nisu potrebne funkcionalnosti viših slojeva OSI modela. PDU-ovi viših slojeva se prosljeđuju bez promjene, pa nisu potrebni posebni mehanizmi za obradu.

Softverski zahtjevi za izvršavanje DPDK-baziranih aplikacija su minimalni. Aplikacija za analizu saobraćaja se izvršava na bilo kojem standardnom 64-bitnom operativnom sistemu baziranom na Linux-u. Što se tiče hardverskih zahtjeva, jedini uslov je korištenje neke od podržanih mrežnih kartica zbog kompatibilnosti sa namjenskim mrežnim drajverima. Ovo je jedini specifičan hardverski zahtjev za kompletan sistem, jer aplikacija funkcioniše bez problema na nenamjenskom hardveru (engl. *commodity hardware*), što olakšava proširivanje kapaciteta. Ne postoji obaveza za korištenjem jednakih hardverskih komponenti na pojedinačnim procesnim elementima, već je kompletan sistem moguće implementirati u virtuelnom okruženju. Kod implementacije sa virtuelnim mrežnim interfejsima treba obratiti pažnju na činjenicu da virtuelne mašine dijele ukupnu propusnost (engl. *throughput*) fizičkog interfejsa, pa su mogućnosti sistema ograničene mogućnostima fizičkog interfejsa.

4.2.1.2 SDN elementi

Softverski definisane mreže (engl. *SDN, Software Defined Networking*) predstavljaju koncept razdvajanja funkcionalnosti mrežnih uređaja na kontrolnu ravan (engl. *control plane*) i logiku za prosljeđivanje saobraćaja, odnosno ravan podataka (engl. *forwarding, data plane*) [104]. Razvoj SDN mreža je definisan kroz sljedeće principe [53]:

- razdvajanje kontrolne ravni i ravni podataka,
- donošenje odluka bazirano na tokovima (umjesto na odredištu paketa),
- mogućnost izdvajanja kontrolne logike u poseban fizički entitet (ili više distribuiranih entiteta),
- programabilnost.

Svoju popularnost SDN mreže duguju čitavom nizu unapređenja u domenu projektovanja i implementacije računarskih mreža, u prvom redu modularnosti i

mogućnosti virtuelizacije, odnosno apstrakciji mrežne topologije [105]. Kroz programiranje visokog nivoa, SDN dozvoljava implementaciju logičkog pogleda na mrežu koji ne odgovara fizičkoj implementaciji, ali koji olakšava njeno održavanje i ispravno funkcionisanje. SDN tehnologije time donose znatno veću slobodu u projektovanju i održavanju računarskih mreža.

Bitno je naglasiti da SDN mreže u obliku u kojem se danas koriste ne predstavljaju prvu pojavu programabilnih mreža [54]. Sredinom 90-tih godina XX vijeka su se pojavile prve aktivne programabilne mreže, a ubrzo zatim i prve implementacije mrežnih sistema koji su uključivali razdvajanje kontrolne ravni i ravni podataka.

Funkcija ravni podataka je prosljeđivanje saobraćaja na osnovu tabele prosljeđivanja (engl. *Forwarding Information Base, FIB*). Ovaj dio uređaja predstavlja tzv. brzu putanju, jer je pretraživanje tabele prosljeđivanja realizovano hardverskim komponentama uz korištenje brze memorije, čime se minimizuje kašnjenje u komunikaciji. Efikasnost upravljanja memorijom kod SDN mrežnih uređaja izlazi van opsega ovog rada, ali predstavlja temu koja je vrlo popularna u literaturi i koja je detaljno obrađena u velikom broju naučnih i stručnih radova [106].

U slučaju da za neki paket ne postoji odgovarajuće pravilo u tabeli, on se prosljeđuje prema kontrolnoj ravni gdje se donosi odluka o odgovarajućoj akciji i po potrebi ažurira sadržaj FIB tabele kako bi se paketi istog tipa ubuduće prosljeđivali direktno brzom putanjom. Akcije za koje je odgovorna ravan podataka uključuju prosljeđivanje, odbacivanje, markiranje i ažuriranje statističkih podataka o tokovima.

Kontrolna ravan je segment SDN mreže koji je odgovoran za održavanje i popunjavanje tabele rutiranja (engl. *Routing Information Base, RIB*), kao i za konfiguraciju ravni podataka kroz pravila za prosljeđivanje. Dio sistema koji upravlja kontrolnom ravni i vrši popunjavanje tabele rutiranja i prosljeđivanje pravila prema svičevima naziva se kontroler. Kontroler je osnovni izvor programabilnosti SDN mreža i predstavlja softversku komponentu koja analizira pristigle podatke i formira sadržaj RIB tabele. Činjenica da se radi o softverskoj obradi znači da je kontrolna ravan znatno sporija od ravni podataka, ali s obzirom na to da se ne izvršava na specijalizovanom mrežnom hardveru zauzvrat može ponuditi znatno veću procesorsku snagu i količinu dostupne memorije (uz mogućnost jednostavnog kasnijeg proširenja). Pravila se u RIB tabeli formiraju na nekoliko načina:

- najčešće korišten način formiranja RIB tabele predstavlja statička konfiguracija pravila od strane administratora. Statičkom konfiguracijom se postiže brže prosljeđivanje (pa time i manje kašnjenje), jer nema potrebe za prosljeđivanje paketa prema kontroleru na softversku obradu ako su prilikom prvog dolaska paketa na svič odgovarajuća pravila već dostupna,
- nakon što je paket za koji ne postoji pravilo u FIB tabeli proslijeđen prema kontrolnoj ravni, kontroler donosi odluku o načinu daljeg upravljanja paketom, nakon čega može odgovarajućim porukama izvršiti izmjenu FIB tabela na svičevima kako bi se takvi paketi ubuduće prosljeđivali (ili odbacivali) direktno. Operacija prosljeđivanja paketa prema kontroleru je relativno skupa u smislu performansi sistema, pa se u konfiguraciji softverski definisanih mreža teži da se broj ovakvih akcija svede na minimum već u fazi inicijalne

konfiguracije sistema,

- izmjena pravila u RIB/FIB se u distrubuiranim mrežama može izvršiti različitim tehnikama sinhronizacije između različih kontrolera. U literaturi postoje različiti pristupi rješenju sinhronizacije [63], [66], [93], jer od uspjeha rješenja može zavisiti ispravnost i upotrebljivost kompletnog sistema,
- kao specijalizacija prethodno navedenih slučajeva, izmjena pravila se može vršiti pod uticajem nekog eksternog servisa koji nije dio SDN okruženja, čime bi se dio procesiranja prebacio izvan kontrolera. Ovakvi servisi mogu u analizi koristiti eksterne baze podataka sa pravilima ili statističke podatke dobijene na proizvoljan način (čak i izvan sistema). Sistem koji je razvijen za potrebe ovog rada dobija zahtjeve za izmjenama određenih pravila od procesnih elemenata koji nisu dio SDN mreže, pa se mogu posmatrati kao eksterni servisi.

Fizički odnos kontrolera i SDN sviča nije strogo definisan specifikacijama [54]. Kontroler može biti implementiran u okviru iste šasije sa svičem ili potpuno fizički odvojen, te može biti centralizovan ili distribuiran na više lokacija, u zavisnosti od potreba i veličine same mreže. Svaki od ovih načina implementacije se odražava različito na brzinu mreže i jednostavnost upravljanja. Centralizovana implementacija kontrolera u opštem slučaju iziskuje veću količinu resursa, ali pojednostavljuje sinhronizaciju informacija na nivou kontrolne ravni unutar mreže. S druge strane, kod distrubuiranih kontrolera je potrebno uložiti dodatni napor da bi se postiglo da svi dijelovi imaju istu sliku mrežne topologije, kako ne bi došlo do pojave petlji u mreži i sličnih problema nekonzistentnosti RIB tabela.

Pravila za prosljeđivanje se od kontrolera prema svičevima prenose korištenjem nekog od namjenski razvijenih protokola. Jedan od najčešće korištenih protokola ove vrste (koji je korišten i za potrebe ovog rada) je OpenFlow protokol [107]. OpenFlow poruke se koriste u komunikaciji između kontrolera i sviča kako bi se uticalo na sadržaj tabela prosljeđivanja i izvršio prenos statističkih podataka od sviča prema kontroleru. Pravila za prosljeđivanje se definišu specifikovanjem vrijednosti polja zaglavlja PDU elemenata drugog, trećeg i četvrtog sloja OSI modela (engl. *match*), kao i akcijom (engl. *action*) koju je potrebno izvršiti za svaki frejm, paket ili segment/datagram kod kojih su specifikovane vrijednosti u pravilima jednake vrijednostima stvarnih polja u zaglavljima. Akcija može uključivati prosljeđivanje prema nekom konkretnom izlaznom portu, prosljeđivanje prema svim portovima sviča (engl. *flooding*), odbacivanje paketa, kao i prosljeđivanje kontroleru.

OpenFlow protokol definiše preko 30 vrsta poruka u trenutno aktuelnoj verziji specifikacije [108], među kojima su najznačajnije:

- *FlowMod* - poruke ovog tipa se koriste za modifikovanje stanja SDN sviča. Za pravila koja se definišu porukama ovog tipa je moguće definisati prioritet i vrijeme važenja kako bi se dodatno uticalo na način njihove primjene,
- *PacketIn* - poruka kojom svič šalje paket kontroleru, bilo da je to eksplicitno traženo nekim definisanim pravilom, ili u situaciji u kojoj ne postoji pravilo koje odgovara poljima u zaglavljima paketa,
- *PacketOut* - poruka kojom kontroler šalje obrađeni paket prema ravni podataka na sviču. Najčešće se koristi kako bi paket koji je na kontroler pristigao

PacketIn porukom bio proslijeđen prema ravni podataka nakon obrade,

- *PortStatus* - asinhrone poruke koje svič šalje kontroleru kako bi signalizirao promjenu stanja nekog od portova.

Zbog velike popularnosti SDN tehnologije, na tržištu postoji veliki broj dostupnih implementacija SDN kontrolera, kako komercijalnih, tako i onih otvorenog koda [109]. Za potrebe razvoja sistema za detekciju napada predstavljenog u ovom radu je korišten kontroler pod nazivom Ryu [110], koji ima nešto manji skup funkcionalnosti u odnosu na druge, ali nudi nešto jednostavniji programski interfejs (engl. *API, Application Programming Interface*).

4.2.2 Komunikacioni protokol

Čvorovi koji se nalaze unutar TIDS-a komuniciraju na dva načina. Prvi način komunikacije predstavlja razmjena OpenFlow poruka, o kojima je više riječi bilo u sekciji 4.2.1.2. Drugi način komunikacije je realizovan kroz namjenski razvijen komunikacioni protokol i odvija se između procesnih elemenata i SDN kontrolera. Osnovna funkcija ovih poruka je iniciranje promjena u tabelama rutiranja i prosljeđivanja, kako bi se izvršila redistribucija i kontrola saobraćaja koji prolazi kroz sistem.

Svaka poruka je enkapsulirana u standardno Ethernet zaglavlje, pri čemu je vrijednost *EtherType* polja u zaglavlju frejma postavljena na (trenutno neupotrebljenu) vrijednost 0x9009 [111], kako bi se pojednostavilo izdvajanje ovih kontrolnih poruka od saobraćaja koji se prenosi putem istih veza. Slanje ovih poruka je ograničeno na vezu između kontrolera i procesnih elemenata i odbacuje se odmah nakon obrade.

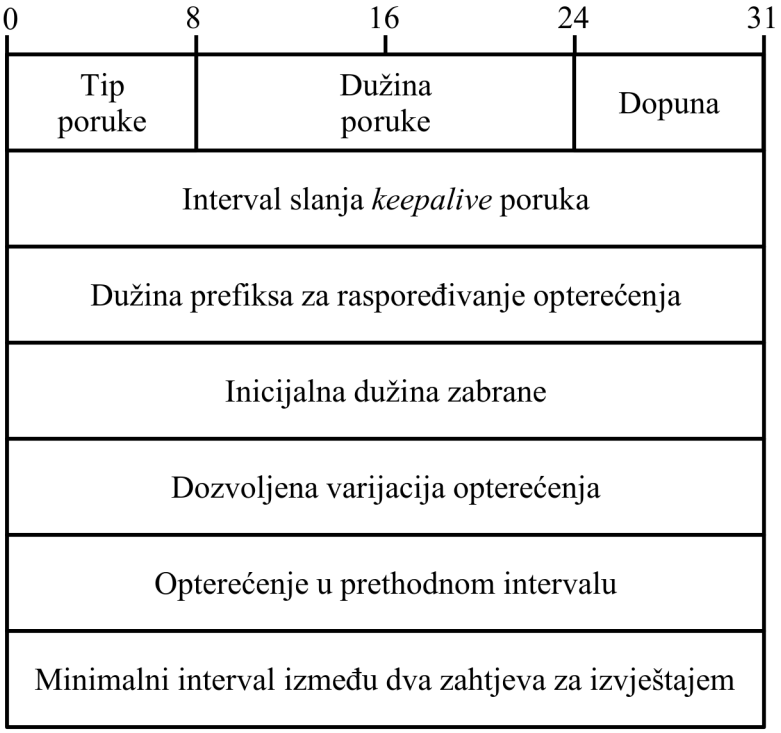
Zaglavlje svih poruka koje se koriste u okviru ovog protokola počinje jednobajtnim poljem koje pokazuje tip poruke, nakon čega slijedi 16-bitno polje koje sadrži ukupnu dužinu poruke u bajtovima. Sljedeći bajt nakon dužine služi za dopunu (engl. *padding*) i uvijek je inicijalizovan na 0. Nakon zajedničkog zaglavlja slijedi varijabilni dio poruke čiji sadržaj zavisi od tipa. Poruke ne sadrže ni adresne informacije ni podatke koji bi se koristili pri provjeri integriteta poruka, jer je primalac uvijek direktno povezan sa pošiljaocem, pa se ovim smanjuje kompleksnost komunikacije. Protokol definiše sljedeće tipove poruka:

- *keepalive* poruka (tip 1) - poruke ovog tipa šalje svaki procesni element prema kontroleru od trenutka pokretanja. Osnovna funkcija *keepalive* poruka je da omoguće kontroleru da vodi računa o broju aktivnih procesora, te da u trenutku prestanka rada nekog elementa (ili uključivanja novog elementa u sistem) izvrši preraspodjelu saobraćaja na trenutno aktivne elemente. *Keepalive* poruke se šalju u pravilnim intervalima (o izabranim vrijednostima dužina intervala će biti više riječi u sekcijama 4.5 i 5.4.1). U okviru *keepalive* poruke se prenosi i niz konfiguracionih parametara koje kontroler koristi da se konfiguriše prilikom prijema prve poruke. Podrazumijeva se da su svi procesni elementi konfigurisani od strane administratora na isti način, pa nije bitno koji procesni element prvi pošalje *keepalive* poruku koja će biti iskorištena za konfiguraciju. Kod svake sljedeće *keepalive* poruke, dio sadržaja

koji se odnosi na konfiguraciju se ignoriše od strane kontrolera, dok se koriste samo podaci o opterećenju procesnog elementa koji je poruku poslao. Na slici 4.2 je dat izgled *keepalive* poruke. Osim zajedničkog zaglavlja, u kojem su sadržani podaci o tipu poruke i njenoj ukupnoj dužini, svaka poruka sadrži sljedeće informacije:

- interval slanja poruka (*keepalive* interval) - 32-bitna vrijednost u sekundama koja pokazuje koliko često će svaki procesni element obavještavati kontroler o svom prisustvu i opterećenju. U slučaju da neki od elemenata ne pošalje *keepalive* poruku u toku 3 *keepalive* intervala, automatski se od strane kontrolera proglašava za neaktivan i saobraćaj se preraspoređuje na ostale aktivne procesne elemente. Broj intervala koji je potreban da bi se element proglasio za neaktivan je odabran po uzoru na druge mrežne protokole [112],
- dužina prefiksa za raspoređivanje opterećenja - 32-bitna vrijednost koja pokazuje koja je veličina najmanjeg mrežnog opsega koji se koristi u raspoređivanju. Određena veličina prefiksa znači da saobraćaj neće biti segmentiran na manje blokove od bloka koji je definisan ovim konfiguracionim parametrom,
- inicijalna dužina zabrane - 32-bitna vrijednost koja pokazuje inicijalnu dužinu intervala na koji će biti izvršeno blokiranje dolaznog saobraćaja za koji je utvrđeno da predstavlja dio aktivnog (D)DoS napada. Dužina trajanja zabrane za ponovljene napade može biti i duža, ali se trajanje uvijek određuje kao umnožak nekog faktora i inicijalne dužine. Način određivanja trajanja zabrana je detaljno obrađen u sekciji 4.3,
- dozvoljena varijacija opterećenja - 32-bitna vrijednost koja pokazuje koliko opterećenje nekog procesnog elementa može odstupati od srednje vrijednosti prije nego što se aktivira procedura redistribucije saobraćaja. Iako je dozvoljena varijacija data kao dio srednje vrijednosti (u opsegu [0,1]), u okviru *keepalive* poruke se, zbog jednostavnijeg čitanja, šalje kao cjelobrojna vrijednost dobijena množenjem stvarne vrijednosti sa 100,
- opterećenje u prethodnom intervalu - 32-bitna vrijednost koja označava broj obrađenih paketa od strane nekog procesora od prethodne *keepalive* poruke. *keepalive* poruka ne sadrži detaljne informacije o tipu obrađenog saobraćaja (grupisane po dužini prefiksa), već samo ukupan broj paketa. Po prijemu *keepalive* poruka od svih procesnih elemenata, kontroler upoređuje dobijene vrijednosti kako bi utvrdio da li opterećenje nekog procesnog elementa odstupa od srednje vrijednosti više od dozvoljene varijacije, pa po potrebi aktivira algoritam za raspoređivanje,
- minimalni interval između dva zahtjeva za izvještajem (engl. *holddown*) - ovaj 32-bitni parametar ima funkciju povećanja stabilnosti algoritma za raspoređivanje i kompletnog sistema. Nakon uspješno završene redistribucije saobraćaja, sistem ulazi u interval u kojem nije dozvoljena nova redistribucija. Trajanje ovog intervala u sekundama je definisano vrijednošću *holddown* polja *keepalive* poruke. Na ovaj način se postiže da sistem ne reaguje na česte uzastopne promjene u opterećenju koje bi

mogle inicirati ponovno pokretanje algoritma za raspoređivanje dok je prethodni ciklus izvršavanja još uvijek aktivan. Dužina ovog intervala zavisi od trajanja algoritma za raspoređivanje kao i obima saobraćaja, ali ne bi trebalo da bude duži od *keepalive* intervala.

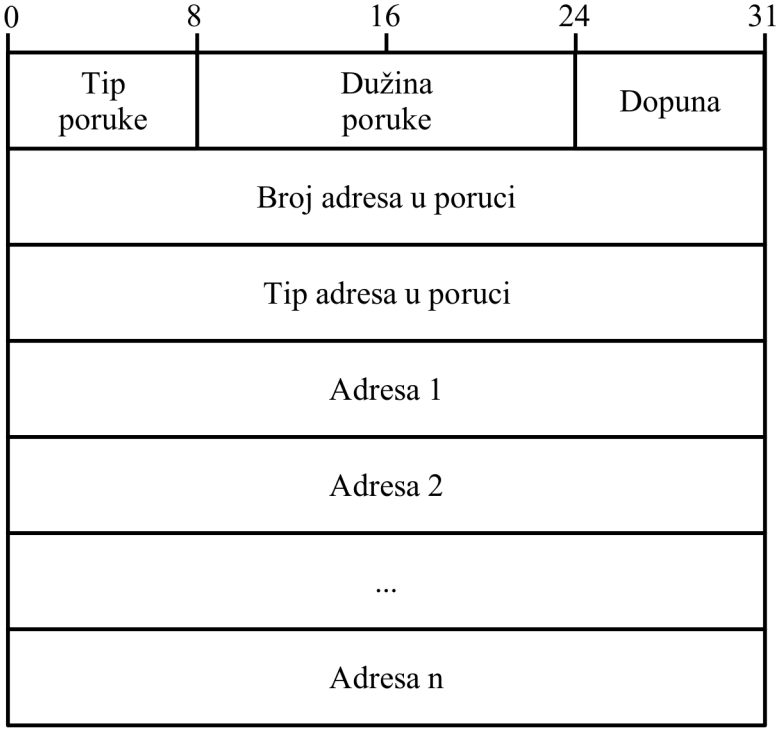


Slika 4.2 *Sadržaj keepalive poruke*

- *shutdown* poruka (tip 2) - porukom ovog tipa procesni element šalje kontroleru obavještenje da prestaje sa radom i da više neće biti aktivan na tom portu. Ako je gašenje procesnog elementa planirano, onda se slanjem ove poruke obezbjeđuje znatno brža adaptacija sistema na izmjene u topologiji. Kontroler ne mora sačekati isticanje 3 *keepalive* intervala da bi proglasio procesor neaktivnim, već može odmah po prijemu poruke inicirati redistribuciju saobraćaja. Ovaj tip poruke ne nosi dodatne informacije u zaglavlju, jer su sve potrebne informacije sadržane u tipu i portu sa kojeg data poruka dolazi,
- zahtjev za promjenom sadržaja tabele rutiranja (engl. *Flow Modification request*) (tip 3) - poruka ovog tipa služi da inicira promjenu sadržaja tabela prosljeđivanja i rutiranja, odnosno da inicira slanje *FlowMod* poruke od strane kontrolera prema sviču S_1. Ovu poruku šalje procesni element koji detektuje napad i ona sadrži jednu ili više mrežnih adresa koje su učestvovale u napadu. Izgled poruke je dat na slici 4.3. U tijelu ove poruke se nalaze sljedeća polja:
  - broj adresa u poruci - 32-bitni podatak koji pokazuje koliko adresa sadrži kompletna poruka, kako bi se kontroleru olakšala verifikacija ispravnosti primljenog sadržaja,
  - tip adresa u poruci - 32-bitni podatak koji pokazuje u odnosu na koje

polje zaglavlja IPv4 paketa je potrebno vršiti poređenje prilikom formiranja novog *FlowMod* pravila. Ako je vrijednost postavljena na 0, onda je potrebno blokirati sve pakete koji su upućeni prema adresama iz nastavka poruke. Analogno tome, ako je vrijednost polja 1, onda se zabrana odnosi na adresu pošiljaoca paketa. Poruka ne nosi informacije o trajanju zabrane, zato što ta vrijednost zavisi od vremena koje je proteklo od prethodne zabrane za svaku od navedenih adresa. Procesni elementi ne čuvaju informacije o prethodnim zabranama, već se evidencija o tome vodi isključivo na kontroleru,

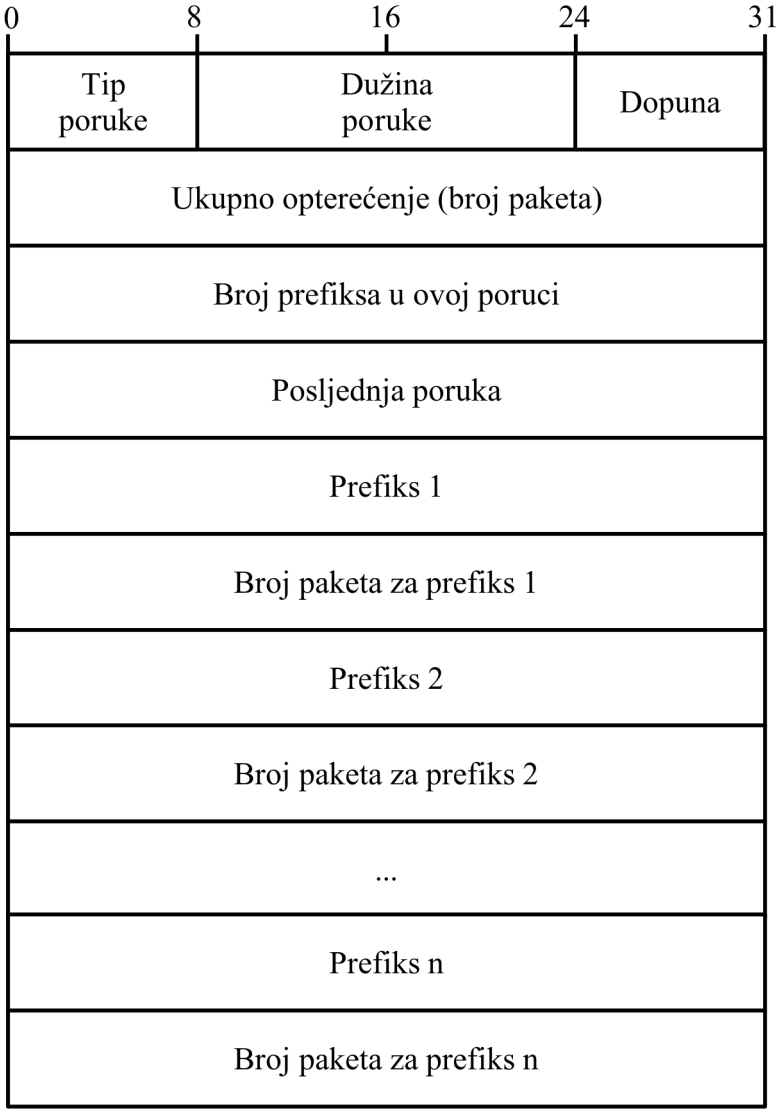
– adrese - nakon tipa adresa, poruka sadrži listu adresa za koje je potrebno formirati nova pravila u tabelama rutiranja i prosljeđivanja i njihov ukupan broj mora odgovarati vrijednosti odgovarajućeg polja iz zaglavlja kako bi se ispravno izvršila validacija primljene poruke. Za svaku adresu je rezervisan opseg od 32 bita.



Slika 4.3 *Sadržaj Flow Modification request poruke*

- zahtjev za detaljnom listom prefiksa (tip 5) - poruke ovog tipa koristi kontroler da bi od procesnih elemenata tražio detaljnu listu mrežnih prefiksa i broj paketa za svaki prefiks koji je obrađen od slanja prethodne *keepalive* poruke. Kontroler šalje ovaj zahtjev nakon prijema *keepalive* poruka od svih procesnih elemenata i nakon utvrđivanja da opterećenje na njima nije jednako. Slično kao kod poruke tipa 2 (*shutdown* poruke), sve potrebne informacije su sadržane u tipu, pa je poruka ovog tipa dugačka samo 4 bajta i sadrži samo zajednički dio zaglavlja,
- detaljna lista prefiksa (tip 4) - poruka ovog tipa ima za svrhu prenos detaljnog spiska mrežnih prefiksa koji je od slanja prethodne *keepalive* poruke obrađen na svakom procesnom elementu. Šalje se kao odgovor na primljenu poruku

tipa 5 (zahtjev za detaljnom listom prefiksa) i zbog potencijalno velikog broja prefiksa obrađenih na jednom procesnom elementu može biti rastavljena na veći broj manjih poruka. Izgled poruke tipa 4 je dat na slici 4.4. Svaka poruka sadrži ukupan broj obrađenih paketa za taj procesni element, nakon čega slijedi polje u kojem je sadržan broj prefiksa koji je prenesen u trenutnoj poruci i polje koje pokazuje da li je data poruka posljednja u nizu. Ostatak poruke sadrži parove prefiksa i broja obrađenih paketa za svaki prefiks. Za svaki prefiks je rezervisano polje od 32 bita, ali je dužina samih prefiksa (tj. veličina podmreže) određena odgovarajućim parametrom *keepalive* poruke.



Slika 4.4 *Sadržaj poruke sa detaljnom listom prefiksa*

4.3 Algoritam za detekciju

4.3.1 Entropija

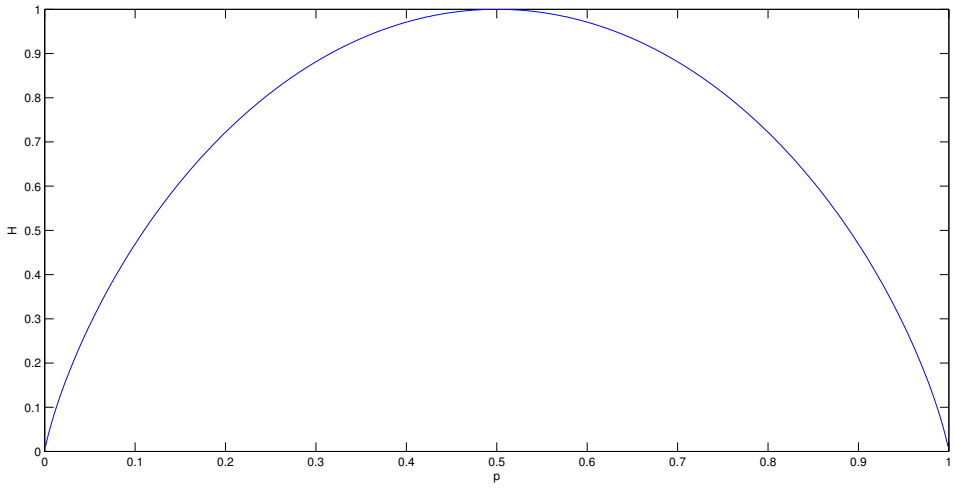
Entropija u teoriji informacija predstavlja osnovnu mjeru neodređenosti [113]. Ona sadrži očekivanu vrijednost informacionog sadržaja koja je sadržana u određenoj poruci. U obliku u kojem se koristi u teoriji informacija je formulisana od strane Kloda Šenona 1948. godine, pa se često u literaturi naziva i Šenonova entropija.

U opštem slučaju, za skup od n promjenljivih čije su vjerovatnoće p_1, p_2,...,p_n, Šenonova entropija se definiše formulom 4.1.

$$H = -\sum p * \log_2 p \tag{4.1}$$

Iako se primarno koristi u teoriji informacija, entropija ima čitav niz numeričkih osobina koje imaju primjenu i izvan oblasti koje su usko vezane za prenos poruka. Ovo se posebno odnosi na svojstva koja se direktno tiču informacionog sadržaja i uticaja vrijednosti vjerovatnoća pojedinačnih promjenljivih na ukupnu vrijednost entropije. Za potrebe ovog rada su posebno značajne sljedeće dvije osobine entropije:

- vrijednost entropije je 0 ako i samo ako svi ishodi osim jedog imaju vrijednost 0, a taj ishod ima vrijednost 1. Drugim riječima, u slučaju u kojem je ishod nekog događaja siguran, nema neodređenosti. Analogno tome, ako svi ishodi imaju jednaku vjerovatnoću, vrijednost entropije je maksimalna i iznosi 1/n. U toj situaciji je nesigurnost ishoda najveća, pa je najveći i informacioni sadržaj koji takav sistem ima. Ova osobina entropije se dodatno može ilustrovati i grafički, na primjeru sa slike 4.5, gdje je prikazana zavisnost entropije u situaciji sa dva ishoda čije su vjerovatnoće p i 1-p.



Slika 4.5 *Vrijednost entropije za dva ishoda sa vjerovatnoćama p i 1-p*

- bilo kakva promjena vrijednosti vjerovatnoća u smjeru ujednačavanja će dovesti do porasta entropije, pri čemu se podrazumijeva da su ishodi međusobno zavisni (tj. da povećanje vjerovatnoće ishoda p_i podrazumijeva smanjenje vjerovatnoće ishoda p_j),
- analogno prethodnoj tački, bilo kakva promjena vrijednosti vjerovatnoća u smjeru povećavanja njihove razlike će dovesti do pada entropije, jer time jedan od ishoda postaje znatno izvjesniji od ostalih.

4.3.2 Primjena entropije na detekciju DDoS napada

U prethodnom razmatranju su navedene samo osobine entropije koje se odnose na međusobno zavisne promjenljive. U slučaju detekcije DDoS napada, od posebnog značaja je raspored izvornih i odredišnih adresa u zaglavlju IP paketa. S obzirom na to da su polja koja sadrže adresne informacije obavezan dio zaglavlja, te da se njihov sadržaj bira iz konačnog skupa elemenata, tada se i broj paketa upućenih prema nekoj adresi (ili upućenih sa neke adrese) može posmatrati kao niz međusobno zavisnih ishoda. Veličina skupa zavisi od bitske dužine adrese (rješenje prikazano u ovom radu radi sa IPv4 protokolom čija dužina adrese iznosi 32 bita), pa se adrese primaoca i pošiljaoca biraju iz skupa od 2^{32} elemenata [1].

Osnovni princip upotrebe entropije za analizu mrežnog saobraćaja se zasniva na ideji da se broj paketa upućenih prema pojedinačnim krajnjim hostovima na mreži u odnosu na ukupan broj svih paketa može posmatrati kao vjerovatnoća da će sljedeći paket biti upućen prema istom hostu, odnosno da svaki host na mreži predstavlja jedan od mogućih ishoda za događaj izbora odredišta paketa. Paket sadrži samo jednu adresu pošiljaoca i primaoca (iako ta adresa može upućivati prema većem broju konkretnih primalaca, o čemu će biti riječi u nastavku), pa se različiti ishodi mogu posmatrati kao međusobno zavisni. Promjena broja paketa (vjerovatnoće) u korist jednog hosta će, za istu propusnost mreže, rezultovati smanjivanjem broja paketa za ostale hostove. Iz osobina navedenih u sekciji 4.3.1 slijedi da će ovakva promjena rezultovati i promjenom vrijednosti entropije za sve posmatrane adrese primalaca. Ako je određeni procesni element u nekom trenutku zadužen za obrađivanje paketa upućenih prema N=10 različitih hostova, te ako je u toku jednog radnog intervala prema svakom od datih hostova upućeno P=20000 paketa, tada se prema izrazu 4.2 dobija vrijednost entropije od 3.3219.

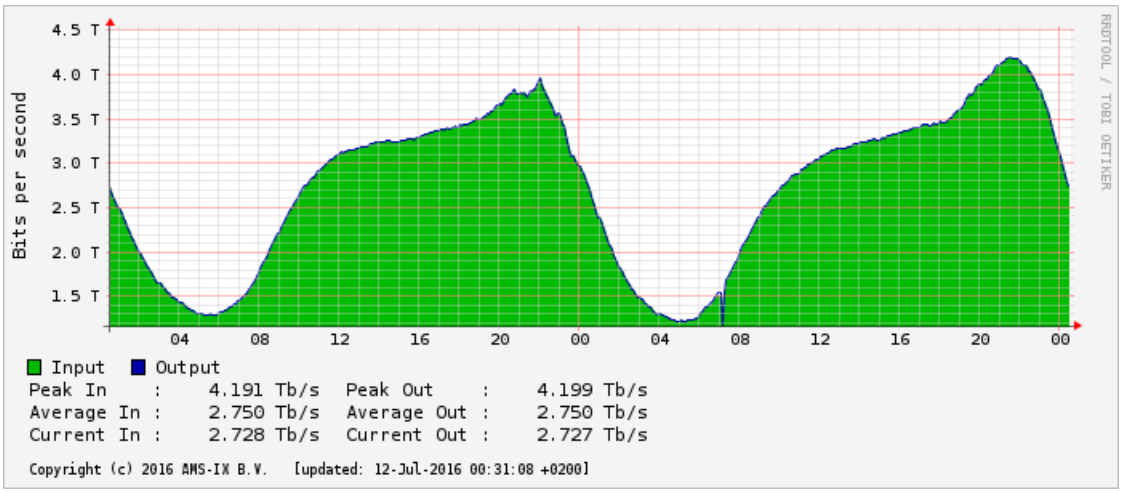
$$H = -\sum \left(\frac{P}{\sum_N P} * \log_2 \frac{P}{\sum_N P} \right) \tag{4.2}$$

Ako se u nekom trenutku broj paketa prema jednom od krajnjih hostova poveća za 10 puta, tada je prema istom izrazu nova vrijednost entropije 2.4995. Daljim povećanjem broja paketa upućenih prema istom krajnjem hostu, dobijaju se još manje vrijednosti entropije.

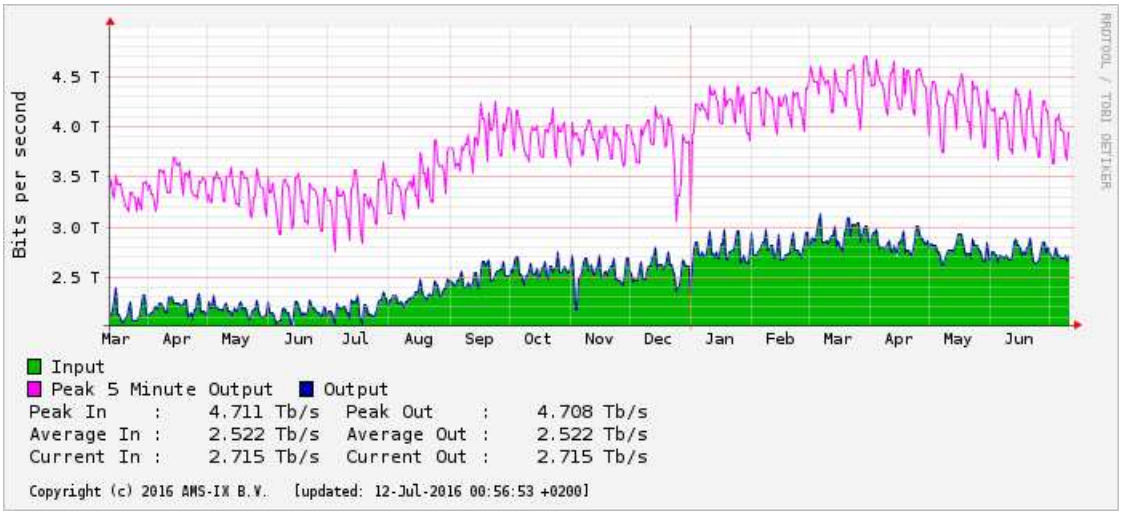
Apsolutna vrijednost broja paketa nema toliki značaj za promjenu vrijednosti

[1] U praksi je broj iskoristivih adresa dužine n bita znatno manji od 2^n, zbog različitih alokacija adresa i ograničenja koja nameće sam protokol [114],[115],[116], ali je uvijek konstantan, pa to ne utiče na opštost zaključka.

entropije koliko relativni odnos između konstantnog broja primalaca. Ova činjenica značajno komplikuje bilo kakvu detekciju napada ako se kao metrika uzima samo broj paketa. U tipičnoj računarskoj mreži se u toku dana dešavaju drastične promjene u količini saobraćaja za određeni vremenski period, ali je relativni odnos primljenog broja paketa između različitih primalaca konstantan ili se dovoljno sporo mijenja da može biti posmatran kao približno konstantan. Slika 4.6 ilustruje tipičnu promjenu količine saobraćaja (engl. *traffic pattern*) na jednoj od tačaka razmjene paketa u Amsterdamu (Amsterdam Internet Exchange Point, AMS-IX [117]), gdje je očigledno postojanje drastične razlike u broju paketa za različite dijelove dana. Međutim, relativni odnos u broju paketa je isti uprkos porastu količine saobraćaja [118],[119],[120], pa takva promjena ne treba biti klasifikovana kao DoS napad. Sličnu pravilnost je moguće uočiti i u kretanju saobraćaja u toku godine (slika 4.7).



Slika 4.6 *Tipična promjena količine saobraćaja u toku dana*



Slika 4.7 *Tipična promjena količine saobraćaja u toku godine*

Uslov koji se odnosi na mapiranje jedne adrese na tačno jednog primaoca, odnosno pošiljaoca paketa je značajan za identifikovanje mete i izvora napada. U današnjim mrežnim protokolima ovaj uslov ne mora biti ispunjen za sve pakete, jer

se u kontekstu broja pošiljalaca i primalaca vezanih za jednu adresu komunikacija može izvoditi na sljedeće načine:

- komunikacija od jednog pošiljaoca prema jednom primaocu (engl. *unicast*) - ovo je najčešći način komunikacije kod svih mrežnih protokola. Adresa pošiljaoca i adresa primaoca ukazuju na tačno jedan čvor na mreži (zanemarujući neispravne pakete u kojima su adresna polja prazna ili imaju neispravnu vrijednost). Svaki primalac upućuje zahtjev za traženim resursima direktno prema pošiljaocu, koji zatim šalje traženi sadržaj kreiranjem direktnog toka podataka prema svakom pojedinačnom primaocu. Broj korisnika koji traže neki resurs je samim tim direktno proporcionalan potrebnom propusnom opsegu. Čak i u slučaju da korisnici koji se nalaze na istoj fizičkoj lokaciji traže isti resurs, *unicast* komunikacija nema mehanizam povećavanja efikasnosti prenosa, već se svaki korisnički zahtjev tretira na isti način i upućuje zasebno. S druge strane, pošiljalac u svakom trenutku ima tačan podatak o broju primalaca, jer svakom zahtjevu odgovara tačno jedan krajnji korisnik,
- komunikacija od jednog pošiljaoca prema većem broju primalaca (engl. *multicast*) - ovaj način komunikacije se koristi kod sistema kod kojih je potrebno omogućiti efikasan prenos podataka od strane jednog pošiljaoca prema većem broju primalaca. Naročitu primjenu je našao kod prenosa multimedijalnog sadržaja (najčešće videa), gdje su kod standardnog (*unicast*) prenosa zahtjevi za potrebnim propusnim opsegom ograničavajući faktor za implementaciju efikasne mrežne infrastrukture. Osnovni princip funkcionisanja *multicast* komunikacije je smanjivanje opterećenja samog servera kroz eliminisanje zavisnosti potrebnog propusnog opsega mreže od broja klijenata koji konzumiraju sadržaj [121]. Ovo se postiže tako što se u komunikaciju aktivno uključuju i mrežni uređaji koji se nalaze na putanji od pošiljaoca prema primaocima. U slučaju *multicast* komunikacije, njihova funkcija je umnožavanje primljenog saobraćaja i slanje kopija paketa kroz one mrežne interfejse na kojima se nalazi jedan ili više primalaca. Pošiljalac u ovom slučaju formira samo jedan tok podataka (engl. *stream*), pa nema podatak o broju primalaca. Podaci se usmjeravaju prema nekoj od adresa iz posebne klase koje su rezervisane za ovu primjenu [122], [123]. Primaoci koji žele da konzumiraju sadržaj koji dolazi od pošiljaoca prema nekoj *multicast* adresi (često poznatoj i kao *multicast* grupa zbog proizvoljnog broja primalaca koji se nalaze iza jedne adrese) se posebnim posebnim porukama prijavljuju za prijem najbližem mrežnom uređaju, nakon čega se formira stablo koje spaja pošiljaoca sa svim primaocima i na osnovu kojeg se vrši umnožavanje saobraćaja. Zbog ograničenja u veličini dostupnog opsega adresa, *multicast* komunikacija je podrazumijevano ograničena na lokalni mrežni segment, i potrebna je upotreba dodatnih protokola rutiranja kako bi se omogućila *multicast* komunikacija preko Interneta [121],
- komunikacija od jednog pošiljaoca prema najbližem primaocu (engl. *anycast*) - ovaj oblik komunikacije se koristi u situacijama u kojima više primalaca koristi istu adresu za komunikaciju. Za razliku od *multicast* komunikacije, zajednička adresa se koristi da bi se klijentima omogućilo više tačaka u mreži preko kojih je moguće korištenje usluga jednog ili više servisa, pri čemu

nije važno od koje krajnje tačke je odgovor dobijen (način funkcionisanja servisa je konzistentan) [124]. Krajnjeg primaoca *anycast* paketa određuju uređaji koji vrše rutiranje, bazirajući odluku na topologiji mreže i lokaciji pošiljaoca zahtjeva. Zbog navedenih osobina, implementacije servisa kakvi su raspoređivanje opterećenja (engl. *load balancing*) je znatno teža, jer je teško uticati na izbor čvora prema kojem će paketi biti upućeni. *Anycast* komunikacija utiče i na efekte (D)DoS napada, jer će napad biti lokalizovan na čvor koji je najbliži ulazu u mrežu, pa se efekti (D)DoS napada upućenog prema *anycast* adresi ne mogu sa sigurnošću predvidjeti.

Upotreba navedenih tipova komunikacije ima različite efekte na vrijednosti entropije odredišnih adresa u mreži. Kako je već opisano, kod *unicast* komunikacije je svaki zahtjev vezan za tačno jednog primaoca (čak i u slučaju da je napadač izvršio promjenu vrijednosti polja i upisao adresu primaoca koji ne postoji u mreži). Samim tim, drastično povećanje broja paketa upućenih prema malom broju primalaca se može jednostavno detektovati kroz promjenu vrijednosti entropije.

S druge strane, u slučaju *multicast* komunikacije, unutar jedne *multicast* grupe se može nalaziti veći broj primalaca koji nisu vidljivi pošiljaocu ili uređaju koji na ulazu mreže analizira saobraćaj. Promjena broja primalaca se uopšte ne odražava na vrijednost entropije, jer se ne mijenja broj *multicast* grupa, već se samo vrši ažuriranje distribucionog stabla. Ovo je procedura koja je ograničena na mrežne uređaje koji su najbliži primaocima saobraćaja, pa nije od značaja za detekciju napada. Međutim, zbog činjenice da se broj poslatih paketa ne mijenja sa povećanjem broja primalaca, kod detekcije se dobri rezultati mogu postići i jednostavnim praćenjem kretanja broja paketa. Detekcija (D)DoS napada usmjerenih prema *multicast* mrežama je zasebno obrađena u literaturi [125], ali nije predmet razmatranja u ovom radu.

Kod razmatranja distribucije adresa pošiljalaca i primalaca je potrebno obratiti pažnju i na situacije u kojima vrijednosti ovih polja ne odgovaraju stvarnim, a da pritom nije riječ o malicioznim aktivnostima. Prevođenje (translacija) mrežnih adresa (engl. *Network Address Translation, NAT*) [126] je tehnologija koja je razvijena sa namjenom efikasnijeg iskorištenja javnih IP adresa. Koristi se da bi se omogućila komunikacija na javnoj mreži hostovima koji nemaju odgovarajuću adresu (koriste samo privatnu adresu u lokalnoj mreži) ili u slučajevima kada je potrebno prikriti adresu pošiljaoca. Polje koje označava adresu pošiljaoca se u tom slučaju prilikom prelaska paketa iz lokalne mreže na internet mijenja adresom iz unaprijed definisanog bloka. Za primaoca, pošiljalac je vidljiv kroz transliranu adresu, pa se odredišne adrese svih paketa upućenih prema toj javnoj adresi na povratku u mrežu prevode nazad u odgovarajuće privatne adrese. Problemi kod detekcije napada nastaju u slučaju kada napad dolazi iz mreže u kojoj se veliki broj privatnih adresa mijenja malim brojem javnih adresa (često samo jednom adresom), jer se time smanjuje mogućnost preciznog određivanja tačnog pošiljaoca svakog paketa. U najboljem slučaju, moguće je izolovati mrežni opseg sa kojeg dolazi napad, ali nije moguće (bez intervencije provajdera koji vrši translaciju) odrediti koji od hostova iz privatne mreže je pravi pošiljalac malicioznog saobraćaja. NAT se rijetko koristi za prevođenje odredišnih adresa, jer bi pozivanje funkcija servisa bilo nepraktično ako bi se njegova lokacija mijenjala (ako bi prevedena adresa bila uvijek ista, onda se translacijom ne postiže nikakav efekat, a unosi

se izvjesno kašnjenje prevođenjem). U slučaju da se NAT iskoristi za prevođenje odredišnih adresa servisa u mreži, identifikacija primaoca i računanje ispravne vrijednosti entropije se može postići korištenjem kombinacije adrese i porta (umjesto samo adrese), pa ovakva implementacija ne utiče direktno na mogućnosti detekcije.

Sličan efekat na proces detekcije ima upotreba servisa čija je osnovna svrha anonimizacija korisničkih adresa na internetu. Ovi servisi (poznati i pod nazivom *proxy* servisi) prihvataju korisničke zahtjeve i prosljeđuju ih prema krajnjem odredištu, mijenjajući pritom vrijednosti adresnih polja. Iako mogu biti upotrebljeni i u svrhu sigurnosne analize i kontrole saobraćaja prilikom ulaska ili izlaska iz mreže, mogu se upotrebljavati kako bi se stvarni korisnički podaci zamijenili drugim vrijednostima. Situacija sa stanovišta izračunavanja entropije je u slučaju *proxy* servisa slična NAT-u, jer je broj različitih adresa pošiljalaca manji ili jednak stvarnom broju hostova koji šalju pakete (u zavisnosti od broja adresa koje se koriste za zamjenu).

Zbog znatno veće zastupljenosti *unicast* saobraćaja u odnosu na ostale vidove komunikacije na Internetu, algoritam koji je opisan u nastavku poglavlja se odnosi isključivo na *unicast* komunikaciju, bez obzira na način konfiguracije mreže pošiljaoca i upotrebu NAT-a i sličnih servisa koji mogu sakriti stvarnu adresu pošiljaoca. Upotreba takvih servisa ne umanjuje efikasnost algoritma za detekciju, o čemu će više riječi biti u sekciji 5.4.2.

Način funkcionisanja algoritma za detekciju (D)DoS napada na jednom procesnom elementu je dat u prikazu 4.1.

Algoritam 4.1 Algoritam za detekciju (D)DoS napada

Require: *entropije* = Kružna lista sa 30 elemenata koja sadrži entropije za prethodnih 30 intervala, *prag* = Konfigurisani prag entropije, *odredista* = mapa odredišta i broja primljenih paketa po svakom odredištu, *izvori* = mapa pošiljalaca i broja poslatih paketa po svakom pošiljaocu, *pragprazan* = vrijeme potrebno da sistem pređe u fazu praznog rada ako nema paketa, *pragucenja* = broj intervala koje sistem treba provesti u intervalu učenja, *radniinterval* = dužina radnog intervala, *pragnapadac* = ukupan dio saobraćaja koji mora biti poslat od strane nekog izvora kako bi taj izvor bio deklarisan kao napadač

1: $stanje \leftarrow$ NEMA_NAPADA ▷ Aktivno stanje sistema
2: $faza \leftarrow$ PRAZAN_HOD ▷ Faza rada sistema
3: $prazanbrojac \leftarrow 0$ ▷ Proteklo vrijeme od prijema prethodnog paketa
4: $intucenja \leftarrow 0$ ▷ Broj proteklih intervala u periodu učenja
5: $radnibrojac \leftarrow 0$ ▷ Vrijeme proteklo u trenutnom radnom intervalu
6: $novaent \leftarrow 0$ ▷ Entropija za upravo završeni interval
7: $srednjaent \leftarrow 0$ ▷ Srednja vrijednost entropije za prethodna 3 bezbjedna intervala
8: $brojnapadaca \leftarrow 0$ ▷ Broj napadača
9: **while** *true* **do**
10: **if** na ulaznom mrežnom interfejsu nema novih paketa **then**
11: $prazanbrojac \leftarrow$ *vrijeme od prethodnog prijema paketa*
12: **if** $prazanbrojac > pragprazan$ && $faza \neq$ PRAZAN_HOD **then**
13: $faza \leftarrow$ PRAZAN_HOD

```
14:     else
15:         if faza == PRAZAN_HOD then
16:             faza ← PERIOD_UCENJA
17:             intucenja ← 0
18:         else
19:             radnibrojac ← vrijeme proteklo od početka trenutnog intervala
20:             Ažuriraj brojač u mapi odredista za odgovarajući prefiks
21:             if stanje == ALARM_FAZA_2 then
22:                 Ažuriraj brojač u mapi izvori za odgovarajuću adresu
23:             if radnibrojac>radniinterval then
24:                 novaent ← entropija za upravo završeni interval
25:                 if stanje == NEMA_NAPADA then
26:                     Ažuriraj listu entropije dobijenom vrijednošću
27:                 if faza == PERIOD_UCENJA then
28:                     intucenja ← 0
29:                     faza ← PERIOD_AKTIVNOG_RADA
30:                 else if faza == PERIOD_AKTIVNOG_RADA then
31:                     srednjaent ← srednja vrijednost entropije za tri prethodna
    (najnovija) elementa liste entropije
32:                     if novaent/srednjaent <prag then
33:                         if stanje == NEMA_NAPADA then
34:                             stanje ← ALARM_FAZA_1
35:                         else if stanje == ALARM_FAZA_1 then
36:                             stanje ← ALARM_FAZA_2
37:                         else if stanje == ALARM_FAZA_2 then
38:                             stanje ← NAPAD_U_TOKU
39:                             brojnapadaca ← broj elemenata u mapi izvori za koje
    ukupan broj paketa prelazi vrijednost pragnapadac
40:                             if brojnapadaca <pragdist then
41:                                 Pošalji kontroleru zahtjev za postavljanje zabrane
    za sve napadače
42:                             else
43:                                 Signaliziraj distribuirani DoS napad
44:                             Isprazni mapu odredista
45:                             Isprazni mapu izvori
46:                             stanje ← NEMA_NAPADA
47:                             brojnapadaca ← 0
48:                     else
49:                         Isprazni mapu odredista
50:                         Isprazni mapu izvori
51:                         stanje ← NEMA_NAPADA
52:                         brojnapadaca ← 0
53:                 radnibrojac ← 0
54:                 novaent ← 0
55:         Proslijedi pročitane pakete prema izlaznom interfejsu
```

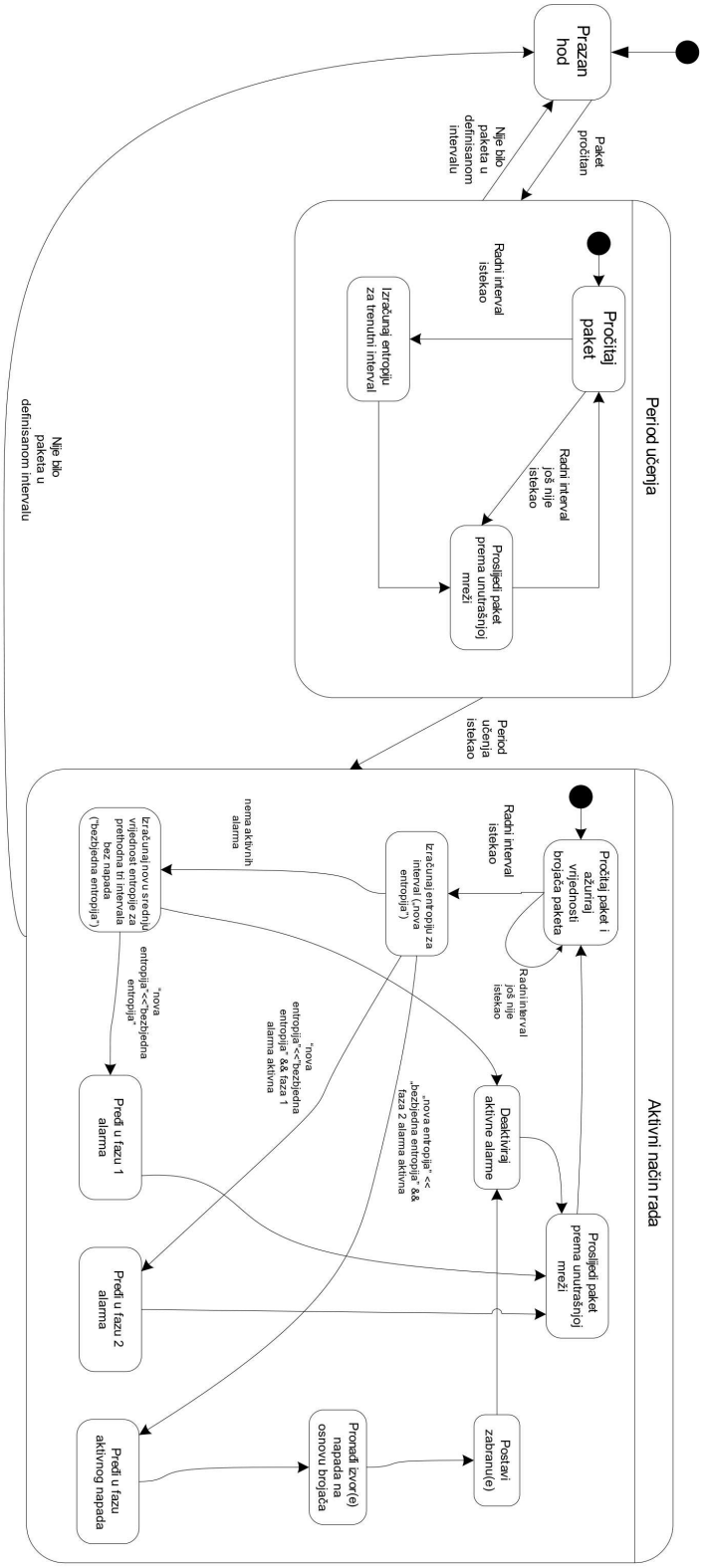
Nakon pokretanja sistema, svaki od procesnih elemenata šalje inicijalnu *keepalive* poruku prema SDN kontroleru. Kontroler koristi prvu primljenu poruku

da bi pročitao konfiguracione parametre koji se odnose na rad algoritama za detekciju i raspoređivanje, dok se sve sljedeće poruke koriste za utvrđivanje broja aktivnih procesnih elemenata. Osim parametara koji se šalju u okviru *keepalive* poruke, postoji čitav niz konfiguracionih parametara koji su lokalni za svaki procesni element i ne šalju se SDN kontroleru. Značenja ovih parametara i njihove predefinisane vrijednosti su date u sekciji 4.5.

Procesni elementi analiziraju pakete koji dolaze od sviča S_1 u toku vremenskih intervala jednake dužine (tzv. radnih intervala, engl. *time slice*) čija bi dužina trebalo da je veća ili jednaka dužini intervala za slanje *keepalive* poruka. Svaki procesni element čuva statističke podatke za 30 prethodnih intervala u kružnoj listi, tako da se najstarije vrijednosti brišu po isteku 30. intervala. Drugim riječima, za interval dužine 30 sekundi, svaki procesni element čuva statističke podatke za prethodnih 15 minuta, što je dovoljna količina podataka za ispravnu detekciju (što je i eksperimentalno potvrđeno u sekciji 5).

Svaki procesni element funkcioniše u jednom od tri načina rada (što se može prikazati dijagramom stanja na slici 4.8):

- prazan hod (engl. *idle*) - u ovom načinu rada se procesni element nalazi od pokretanja do prijema prvog paketa od sviča S_1. Osim toga, procesni element se vraća nazad u ovaj način rada ako u toku kratkog vremenskog intervala nema primljenih paketa. Dužina ovog intervala utiče na rad algoritma za detekciju, jer se u ovom stanju ne vrši izračunavanje vrijednosti entropije zbog činjenice da bi prvi paket izazvao dovoljno veliku promjenu entropije u odnosu na dotadašnje stanje da bi ona bila detektovana kao napad. Ako je vrijednost intervala prekratka, prelasci u ovaj način rada bi bili prečesti, pogotovo u situacijama manje frekvencije saobraćaja. To bi onemogućilo ispravan rad algoritma za detekciju, jer je prelazak u prazan hod ekvivalentan inicijalnom pokretanju sistema i podrazumijeva da će sistem do ulaska u aktivni način rada morati ponovo proći kroz period učenja i inicijalizacije u toku kojeg nije u stanju da detektuje napade, bez obzira na vrijednost entropije. S druge strane, ako je trajanje intervala predugo, sistemu će biti potrebno dodatno vrijeme da reaguje na prestanak saobraćaja. Kriterijum koji je za potrebe testiranja u ovom radu korišten za izbor dužine intervala je da ona bude jednaka manjem dijelu intervala za slanje *keepalive* poruka (vrijednost sa kojom su vršeni testovi je 5 sekundi),
- period učenja (engl. *learning*) - u ovom načinu rada sistem prikuplja podatke o normalnom ponašanju mreže (engl. *baseline*). Ovaj period mora proteći bez napada, kako bi sistem imao ispravne podatke o standardnoj vrijednosti entropije koje će kasnije moći uporediti sa vrijednostima entropije u aktivnom načinu rada. Da bi se povećala preciznost sistema, potrebno je da saobraćaj u toku perioda učenja bude što sličniji vrijednostima koje se u realnim uslovima pojavljuju na mreži, uključujući broj paketa i njihovu raspodjelu u vremenu, kao i raspodjelu prema odredišnim hostovima. Sistem u toku ovog perioda neće detektovati aktivni napad (bez obzira na izmjerene vrijednosti entropije), već će prihvatiti analizirano stanje kao normalno (bezbjedno) i na osnovu tako prikupljenih podataka kreirati osnovu za dalji rad. Ako je napad u toku za vrijeme trajanja perioda učenja, sistem neće biti u stanju



Slika 4.8 *Dijagram stanja za algoritam za detekciju (D)DoS napada*

da detektuje napade u aktivnom načinu rada, jer se vrijednosti entropije neće dovoljno razlikovati ili će biti mnogo veće od onih koje su učenjem proglašene za normalne. Vrijednosti koje su dobijene u toku ove faze se vremenom prilagođavaju izmjenama koje nastaju kao posljedica promjena u izgledu (engl. *pattern*) saobraćaja. Međutim, ako saobraćaj na osnovu kojeg je proveden period učenja ne odgovara realnom saobraćaju, potrebno je mnogo više vremena da sistem dobije ispravne podatke i da nastavi da funkcioniše punim kapacitetom (da se iz srednje vrijednosti entropije izgubi uticaj pogrešno naučenih vrijednosti).

Dužina perioda učenja je definisana konfiguracionim parametrom koji označava broj radnih intervala koji treba proći prije prelaska u aktivni način rada. Kraći period učenja omogućava sistemu da ranije pređe u aktivni način rada, ali može pružiti negativnu sliku o normalnom toku saobraćaja i vrijednosti entropije. Naime, kratki skokovi u broju paketa upućenih prema jednom ili više hostova koji se mogu pojaviti u toku normalnog funkcionisanja mreže ne predstavljaju pokazatelj napada. Ako je period učenja dovoljno kratak da se poklapa sa pojavom ovakvog skoka, onda povratak na normalan obim saobraćaja nakon prelaska u aktivni način rada može rezultovati dovoljnom promjenom entropije da sistem donese neispravnu odluku. Kod dužeg perioda učenja, efekti kratkih skokova u broju paketa nisu izraženi u tolikoj mjeri jer se vrijednost entropije dobija kao srednja vrijednost izračunatih entropija za sve intervale. Usrednjavanje smanjuje uticaj kratkotrajnih drastičnih promjena u broju paketa (bilo da se radi o porastu ili o padu), čime sistem dobija na robusnosti. Međutim, predugačak period učenja bi učinio sistem neosjetljivim na manje promjene koje mogu predstavljati kratkotrajne napade koji bi sa kraćim periodom učenja bili jednostavno otkriveni.

U toku perioda učenja, svi paketi se prosljeđuju prema unutrašnjoj mreži,

- aktivni način rada (engl. *enforcing*) - u ovom načinu rada sistem analizira saobraćaj i donosi odluke o napadima poređenjem izračunatih vrijednosti entropije sa vrijednostima za koje je učenjem utvrdio da su bezbjedne. U toku svakog intervala sistem broji pakete i grupiše rezultate prema odredišnim adresama. Analiza broja paketa predstavlja operaciju koja zahtijeva najviše procesorskog vremena, pa je implementacija ove strukture podataka od kritičnog značaja za performanse kompletnog sistema. Adrese i brojevi paketa se čuvaju u heš tabeli koja je implementirana u okviru DPDK biblioteke [102]. Brzina pristupa se postiže podjelom memorijskog prostora za čuvanje ključeva u odvojene cjeline (engl. *buckets*), pri čemu se u jedoj cjelini nalaze vrijednosti koje imaju uzastopne vrijednosti ključeva. Ovim se smanjuje broj pretraživanja, jer nije potrebno pretraživati one cjeline koje sigurno ne sadrže traženi ključ. Implementacija koristi konfigurabilnu heš funkciju kojom se ulazni ključ pretvara u 4-bajtni otisak. Pošto se heš tabela koristi za čuvanje brojeva paketa za IPv4 adrese (pri čemu adrese predstavljaju ključeve u tabeli), nema gubitka podataka kod izračunavanja otisaka.

  Broj paketa u toku intervala se za svaku adresu čuva kao 8-bajtni neoznačeni podatak. Zbog ograničenja u dostupnom memorijskom prostoru koji bi bio potreban za čuvanje broja paketa za sve adrese iz IPv4 opsega, maksimalna

veličina tabele je ograničena konfiguracionim parametrom. Parametar utiče na određivanje otiska ključa u heš tabeli, jer se vrijednost određuje po modulu ukupne veličine tabele. Ovo ograničenje ne utiče na efikasnost sistema, jer je tipičan broj odredišnih adresa u okviru jednog *keepalive* intervala mnogo manji od veličine tabele. U konfiguraciji sistema koja je korištena za dobijanje rezultata u ovom radu, veličina tabele je ograničena na 262144 unosa (0x40000).

Po isteku svakog radnog intervala, sistem računa vrijednost entropije za taj interval. Dobijena vrijednost se poredi sa srednjom vrijednošću entropije za prethodna 3 intervala u kojima nije bilo napada (tzv. bezbjedna vrijednost). Ako je odnos vrijednosti entropija za aktivni interval i bezbjedne vrijednosti veći od konfigurisanog praga, to znači da povećanje broja paketa u korist malog broja primalaca nije dovoljno da bi situacija bila klasifikovana kao napad. Ako je odnos manji od konfigurisanog praga, sistem prelazi u prvu fazu alarma.

Različite faze koje prethode konačnoj detekciji napada za svrhu imaju povećanje robusnosti sistema. Paketi se i dalje prosljeđuju prema unutrašnjoj mreži, pa u slučaju kratkotrajnog skoka u broju paketa neće biti trajnih posljedica po funkcionisanje mreže. Nakon isteka novog radnog intervala ponovo se vrši računanje entropije, koja se zatim ponovo poredi sa srednjom vrijednošću za prethodne sigurne intervale (vrijednost na osnovu koje je sistem prešao u prvu fazu alarma se ignoriše). Ako je odnos vrijednosti sigurne entropije i novoizračunate vrijednosti za prethodni interval i dalje ispod zadatog praga, sistem prelazi u drugu fazu alarma.

U drugoj fazi alarma se nastavlja prosljeđivanje paketa prema unutrašnjoj mreži, ali sistem pokušava izolovati izvore potencijalnog napada. Za svaki dolazni paket koji je upućen prema adresama za koje se smatra da su meta napada (uzimaju se adrese koje u ukupnom broju primljenih paketa učestvuju sa odgovarajućim procentom koji je definisan konfigurabilnim parametrom) se u odvojenoj tabeli ažurira broj poslatih paketa za svakog od pošiljalaca.

Ako se u bilo kojoj fazi do proglašenja aktivnog napada vrijednost entropije vrati na bezbjednu vrijednost, tabele sa statističkim podacima se prazne i brišu se vrijednosti entropija za one intervale u kojima je došlo do značajnog odstupanja. Ovim se onemogućava uticaj vrijednosti entropije izračunat u trenutku dok je napad bio aktivan na srednje vrijednosti nakon što je napad završen.

Ako se vrijednost entropije održi ispod praga i po isteku sljedećeg radnog intervala, sistem proglašava stanje aktivnog napada i preduzima odgovarajuće mjere za eliminisanje malicioznog saobraćaja. Ako je broj pošiljalaca relativno mali (manji od vrijednosti konfigurabilnog parametra), tada procesni element koji je detektovao napad kreira odgovarajući paket sa zahtjevom za promjenu tabele rutiranja i šalje ga kontroleru. Kontroler kreira po jedno pravilo za svaku adresu specifikovanu u paketu i definiše zabranu kompletnog saobraćaja koji dolazi sa datih adresa u unaprijed definisanom trajanju. Za specifikovanje trajanja zabrane se koristi funkcionalnost *OpenFlow* protokola koja omogućava da se za pravila definiše vrijeme validnosti (engl. *hard*

timeout), nakon čijeg isteka se pravilo samo briše iz tabele. Inicijalno trajanje zabrane se definiše u formi konfiguracionog parametra i šalje se kao jedan od parametara od strane procesnih elemenata prema kontroleru u okviru inicijalne *keepalive* poruke. Ako se dati pošiljalac nije ranije pojavljivao u listi napadača, tada je dužina zabrane jednaka dužini definisanoj u konfiguracionom parametru. Međutim, ako se dati host pojavi u listi napadača prije isteka intervala čija je dužina jednaka dvostrukoj dužini prethodne zabrane, tada se koristi sistem penalizacije kako bi se dodatno sankcionisalo ponovljeno učešće u napadu. Koncept penalizacije podrazumijeva udvostručavanje dužine trajanja zabrane za svaki ponovljeni prekršaj i koristi se i u drugim mrežnim protokolima kako bi se povećala robusnost sistema (sličan koncept je implementiran kod *route dampening* mehanizma kod BGP protokola [127]).

Problem kod blokiranja saobraćaja nastaje u slučaju da se radi o distribuiranom DoS napadu. Kod uspješno izvedenog distribuiranog napada, ukupan broj paketa po jednom napadaču je približan broju paketa koje je poslao nemaliciozni korisnik. Zbog ogromnog broja napadača, napad dobija na snazi tek u tački agregacije saobraćaja. Ukupna količina saobraćaja raste približavanjem primaocu (meti), pa se detekcija ne razlikuje u odnosu na nedistribuiranu verziju napada, ali bi blokiranje pošiljalaca prema broju poslatih paketa vjerovatno uključivalo i blokiranje nekog od legitimnih korisnika. Jedan od načina zaštite u ovom slučaju bi bio blokiranje paketa usmjerenih prema odredištu napada (u literaturi poznato pod nazivom *blackholing* [128]), čime se može postići eventualna zaštita od fizičkog oštećenja dijelova sistema, ali se istovremeno postiže krajnji efekat koji je jednak efektu koji bi imao uspješan napad (tj. servis je u prekidu).

Procesni elementi nemaju podatak o aktivnim zabranama na sistemu, niti sadrže pravila za blokiranje malicioznih paketa. Nakon detekcije aktivnog napada, vrši se izolovanje napadača i slanje obavještenja kontroleru, nakon čega se nastavlja normalan rad sistema. Iz toga slijedi da će svaki paket koji u bilo kojem trenutku bude proslijeđen do nekog procesnog elementa biti propušten prema unutrašnjoj mreži (procesni elementi ne vrše blokiranje paketa osim u slučaju paketa koji pripadaju komunikacionom protokolu i koji se uvijek odbacuju na prvom koraku), a da se čitav sistem oslanja na usluge SDN segmenta mreže za blokiranje malicioznog saobraćaja.

Dodatni sigurnosni mehanizam je implementiran kako bi napadač bio spriječen da pažljivim praćenjem načina rada sistema prilagodi svoj napad tako da se konačna entropija u svakom intervalu uvijek nađe iznad definisanog praga. U tu svrhu sistem prati i trend kretanja vrijednosti entropije u znatno dužem vremenskom periodu (podrazumijevano postavljen na 10 *keepalive* intervala). Ako je vrijednost entropije u konstantnom padu i po isteku perioda mjerenja ima vrijednost koja se nalazi ispod konfigurisanog praga, sistem i u ovakvoj situaciji prelazi u fazu alarma i nastavlja sa daljim provođenjem algoritma.

4.4 Algoritam za raspoređivanje opterećenja

Osnovna funkcija algoritma za raspoređivanje opterećenja (engl. *load balancing*) je ujednačavanje opterećenja na procesnim elementima i omogućavanje dinamičkog proširenja kapaciteta u toku rada sistema bez narušavanja rada algoritma za detekciju.

Promjena rasporeda saobraćaja po procesnim elementima u toku rada sistema može značajno uticati na vrijednost entropije, pa je algoritam za raspoređivanje dizajniran na način da promjena vrijednosti ne utiče na odluku o napadu. Iz tog razloga, preraspodjela saobraćaja je dozvoljena samo u periodima u kojima nema opasnosti da će vrijednost entropije dovesti do pogrešne odluke, odnosno ako je sistem u bezbjednom stanju.

Pregled algoritma je dat u prikazu 4.2.

Algoritam 4.2 Algoritam za raspoređivanje opterećenja

Require: $aktelem$ = Broj aktivnih procesnih elemenata, određen na osnovu *keep-alive* poruka, $prag$ = Konfigurisani prag za preraspodjelu, $elementi$ = Lista aktivnih procesnih elemenata

1: $brizv \leftarrow 0$ ▷ Broj primljenih izvještaja
2: $srednjibr \leftarrow 0$ ▷ Srednja vrijednost broja paketa
3: **for all** $p \in elementi$ **do** $p.brpak \leftarrow 0$ ▷ Broj paketa za dati procesni element
Inicijalizuj listu $p.prefiksi$ ▷ Lista mrežnih prefiksa i broj paketa po svakom prefiksu
4: **while** $brizv \neq aktelem$ **do**
5: Ažuriraj $p.brpak$ za procesni element p_i
6: $brizv \leftarrow brizv + 1$
7: $srednjibr \leftarrow$ srednja vrijednost broja paketa
8: Inicijalizuj listu nedovoljno opterećenih procesnih elemenata *underutilized*
9: Inicijalizuj listu preopterećenih procesnih elemenata *overutilized*
10: **for all** $p \in elementi$ **do**
11: **if** $p.brpak > srednjibr*(1+prag)$ **then**
12: Dodaj p u *overutilized*
13: **else if** $p.brpak < srednjibr*(1-prag)$ **then**
14: Dodaj p u *underutilized*
15: **if** *underutilized* nije prazna *overutilized* nije prazna **then**
16: $brizv \leftarrow 0$
17: **for all** $p \in elementi$ **do**
18: Pošalji zahtjev za detaljnim spiskom mrežnih prefiksa od procesnog elementa p
19: **while** $brizv \neq aktelem$ **do**
20: ažuriraj listu mrežnih prefiksa $p.prefiksi$ prefiksima i brojevima paketa dobijenim od procesnog elementa p_i
21: $brizv \leftarrow brizv + 1$
22: **for all** $o \in overutilized$ **do**
23: **for all** $u \in underutilized$ **do**
24: **for all** $prefiks \in o.prefiksi$ **do**

25: **if** $o.brpak - prefiks.brpak > srednjibr*(1\text{-}prag)$ && $u.brpak + prefiks.brpak < srednjibr*(1+prag)$ **then**
26: Ukloni $prefiks$ iz liste $o.prefiksi$
27: Dodaj $prefiks$ u listu $u.prefiksi$
28: $o.brpak = o.brpak - prefiks.brpak$
29: $u.brpak = u.brpak + prefiks.brpak$
30: Kreiraj i pošalji FlowMod poruku

Prije početka rada samog algoritma, kontroler generiše niz inicijalnih pravila kako bi se izbjeglo pojedinačno raspoređivanje svakog novog toka paketa (engl. *flow*). SDN svičevi šalju svaki paket za koji ne postoji odgovarajuće pravilo u tabeli prosljeđivanja prema kontroleru, što bi zahtijevalo ogromnu količinu sistemskih resursa, posebno u mrežama velikih brzina [129]. Iz tog razloga se nakon pokretanja proaktivno generišu pravila koja služe za usmjeravanje paketa dok kontroler ne prikupi dovoljno podataka o opterećenju procesnih elemenata. Ova pravila particionišu saobraćaj na bazi najnižih bita odredišne IP adrese, pri čemu broj bita zavisi od broja procesnih elemenata (koristi se najmanji broj bita n za koji je 2^n veće ili jednako broju procesnih elemenata). Ova pravila imaju najniži prioritet i koriste se u slučaju da nema uže definisanih pravila. Sva naknadna pravila se generišu sa višim prioritetom, kako bi kod odlučivanja imala prednost.

Prva primljena *keepalive* poruka sadrži konfiguracione parametre koji se odnose na rad algoritma za raspoređivanje. Ovi parametri uključuju dužinu prefiksa za raspoređivanje i dozvoljenu varijaciju opterećenja. Dužina prefiksa za raspoređivanje određuje najmanji mrežni opseg koji će se koristiti u raspoređivanju i koji se neće dijeliti na manje blokove prilikom prebacivanja sa jednog procesnog elementa na drugi.

Dozvoljena varijacija opterećenja predstavlja prag dozvoljenog odstupanja opterećenja jednog procesnog elementa od srednje vrijednosti izračunate za prethodni *keepalive* interval. Ako opterećenje jednog ili više procesnih elemenata prelazi dati prag, sistem započinje preraspodjelu mrežnih prefiksa na one procesne elemente čije je opterećenje manje. Nakon što je procedura preraspodjele završena, SDN kontroler nastavlja sa upoređivanjem izmjerenih vrijednosti opterećenja za sljedeće intervale i ponavlja kompletnu proceduru ako se ukaže potreba za takvom akcijom.

Prikazana verzija algoritma kao metriku koristi broj paketa, ali je princip rada dovoljno fleksibilan da na isti način može podržati i korištenje neke druge metrike uz trivijalne izmjene u načinu izvođenja algoritma i načinu mjerenja opterećenja procesora.

Svaki *keepalive* paket sadrži 32-bitno polje koje pokazuje opterećenje sistema u toku prethodnog intervala. Nakon prijema *keepalive* paketa od svakog procesnog elementa u istom intervalu, sistem poredi dobijene vrijednosti i računa srednju vrijednost opterećenja. Ako je neki od procesnih elemenata imao opterećenje koje je veće od dozvoljene vrijednosti (koja je izračunata kao proizvod srednje vrijednosti i vrijednosti konfigurisanog praga), sistem šalje poruku tipa 5 svim procesnim elementima kojom se traži detaljna lista prefiksa i broja paketa za svaki prefiks.

Po prijemu svih lista prefiksa, kontroler prolazi kroz listu prefiksa za svaki element čije je opterećenje veće od dozvoljenog. Odluka o migriranju nekog mrežnog

prefiksa [2] sa jednog na drugi procesni element se donosi na osnovu predviđanja efekta koji bi takva akcija imala po opterećenje oba čvora. Neki procesni element ima odgovarajuće opterećenje za prihvatanje dodatnog saobraćaja ako je zbir ukupnog broja paketa u prethodnom intervalu i broja paketa za prefiks koji se trenutno analizira manji od dozvoljene granice opterećenja (proizvod srednjeg opterećenja i konfigurisanog praga). Ako takav element postoji, odgovarajuće liste prefiksa se ažuriraju za oba procesna elementa i šalje se *FlowMod* poruka koja formira odgovarajuće pravilo na sviču S_1. Kontroler vodi računa o rezultatima prethodnih preraspodjela i briše sva pravila koja su u kontradikciji sa novokreiranim.

Procedura preraspodjele se završava kad se projekcije opterećenja svih procesnih elemenata dovedu u dozvoljene granice ili kad više nema procesnih elemenata dovoljno niskog opterećenja koji bi mogli prihvatiti prefikse sa preopterećenih čvorova. Nije moguće garantovati da će se opterećenje na osnovu kojeg su donesene odluke zadržati i u sljedećim intervalima, pa se u slučaju nove nejednakosti opterećenja algoritam pokreće ponovo kako bi se broj paketa ujednačio.

Slanjem *keepalive* poruka koje sadrže samo sumarno opterećenje se štede resursi na procesnim elementima, kao i dostupni propusni opseg na dijelu mreže između procesnih elemenata i SDN kontrolera. Detaljna lista prefiksa se traži samo u slučajevima u kojima sistem potvrdi postojanje razlike opterećenja na štetu jednog ili više procesnih elemenata.

Sistem je implementiran tako da nakon završene iteracije algoritma za raspoređivanje ulazi u period čekanja (engl. *holddown* interval) unutar kojeg ne može biti pokrenuta nova preraspodjela. Ovo ima za svrhu povećanje robusnosti sistema, jer bi česte uzastopne preraspodjele paketa unijele nestabilnost u rad sistema i dale pogrešnu predstavu o realnom opterećenju sistema. Vrijednost ovog polja je još jedan konfiguracioni parametar koji se u prvoj *keepalive* poruci koristi za konfigurisanje kompletnog sistema.

Kao i u slučaju ostalih parametara, različite dužine *holddown* intervala imaju različite efekte na rad sistema. Očekivano je da manje vrijednosti (kraći intervali) donose rizik da će nova procedura preraspodjele biti pokrenuta prije nego što se ažurira tabela prosljeđivanja na sviču (ovaj dio procedure može trajati i duže od samog algoritma za preraspodjelu opterećenja, posebno u sistemima kod kojih je količina dostupnih sistemskih resursa mala). S druge strane, duži interval može značiti da će se razlike u opterećenju značajno povećati između dvije uzastopne preraspodjele, posebno u mrežama sa velikim brojem hostova koji se nalaze u unutrašnjoj mreži. Rezultati testova iz kojih je donesen zaključak o preporučenim vrijednostima parametara koji se odnose na algoritam za raspoređivanje su dati u prilogu rada (sekcija 7.2), a analiza dobijenih rezultata u sekciji 5.4.1.

Kako je prikazano u algoritmu 4.2, raspoređivanje se privremeno suspenduje od trenutka kad sistem pređe u prvu fazu alarma, pa sve do trenutka kad se identifikuje aktivni napad ili do trenutka kada se sistem vrati u normalno stanje prije proglašenja napada. Ova mjera služi da bi se omogućio nesmetan rad algoritma za detekciju u periodu kada je potrebno da izračunata vrijednost entropije

[2] Pod "migracijom prefiksa" sa čvora A na čvor B se smatra promjena sadržaja tabela rutiranja i prosljeđivanja tako da se saobraćaj upućen prema adresama koje pripadaju datom prefiksu usmjerava prema čvoru B, umjesto prema čvoru A

odražava samo promjene u izgledu saobraćaja nastale kao posljedica napada, ali ne i promjene nastale kao posljedica preraspodjele mrežnih prefiksa. SDN kontroler donosi odluku o privremenom suspendovanju nakon što se u jednom ili više *keepalive* poruka dobijenih od procesnih elemenata u polju koje pokazuje opterećenje u prethodnom intervalu nađe vrijednost 0xffffffff, što označava da se sistem nalazi u fazi alarma ili da je napad upravo u toku.

Dodatno, algoritam definiše posebnu proceduru brzog oporavka nakon blokiranja napada. Podrazumijeva se da je od prelaska sistema u prvu fazu alarma pa do povratka u normalan režim rada prošlo dovoljno vremena da se informacije o trenutnom rasporedu mrežnih blokova po procesnim elementima mogu smatrati zastarjelim i da ne odgovaraju trenutnom stanju. Stoga, nakon završetka napada, sistem odbacuje sva postojeća pravila koja definišu raspored paketa i ostavlja samo pravila koja se odnose na aktivne zabrane saobraćaja. Nakon toga se algoritam pokreće ispočetka i sistem nastavlja sa analiziranjem opterećenja u okviru svake primljene *keepalive* poruke.

4.5 Konfiguracioni parametri

Svi konfiguracioni parametri za rad sistema su definisani na svakom procesnom elementu. Vrijednosti parametara koji se odnose na rad SDN kontrolera se prenose u okviru *keepalive* poruka koje šalju procesni elementi, pri čemu se za konfiguraciju koristi sadržaj prve primljene poruke. Podrazumijeva se da između procesnih elemenata ne postoji definisana hijerarhija, već da svaki procesni element funkcioniše na isti način, pa redoslijed njihovog pokretanja ne treba biti od značaja za njihovo funkcionisanje. Ne postoji direktan komunikacioni kanal između samih procesnih elemenata, pa je za njihovo ispravno i dosljedno funkcionisanje potrebno da svi konfigurisani parametri imaju iste vrijednosti. Izuzetak su parametri koji se odnose na upotrebu procesorskih jezgara i dostupne memorije, jer vrijednosti tih parametara utiču samo na funkcionisanje elementa na kojem su definisani i nemaju uticaja na ostale elemente.

Osnovni konfiguracioni parametri sistema su:

- broj intervala za učenje (LS) - pokazuje broj *keepalive* intervala koji treba proći od prvog primljenog paketa do prelaska u aktivni način rada, odnosno broj intervala koje sistem provede u fazi učenja. Trajanje perioda učenja ima direktan uticaj na efikasnost sistema, pa je potrebno odabrati vrijednost koja sistemu daje dovoljno vremena da prikupi podatke o normalnom funkcionisanju mreže, ali istovremeno omogućava brzu inicijalizaciju sistema. Efekti različitih vrijednosti ovog konfiguracionog parametra su objašnjeni u sekciji 4.3. Podrazumijevano trajanje perioda učenja je 5 *keepalive* intervala,
- dužina *keepalive* intervala (DKI) - ovaj parametar određuje interval između dvije susjedne *keepalive* poruke. Ovo je najznačajniji parametar za funkcionisanje sistema, jer se vrijednosti velikog broja drugih parametara određuju kao umnožak dužine ovog intervala. Logika izbora ispravne vrijednosti ovog parametra je slična kao i u slučaju ostalih parametara. Male vrijednosti

povećavaju opterećenje kompletnog sistema, jer slanje *keepalive* poruka direktno uslovljava pokretanje algoritma za raspoređivanje opterećenja. S druge strane, duži *keepalive* interval bi otežao otkrivanje eventualnih kvarova na sistemu, jer kontroler donosi odluku o prestanku rada nekog procesnog elementa na osnovu broja propuštenih *keepalive* poruka. U sekciji 5.4.1 su analizirane performanse sistema za različite vrijednosti *keepalive* intervala,

- dužina radnog intervala (engl. *slice*) (SLD) i broj intervala (TS) - ovi parametri određuju trajanje intervala za mjerenje entropije i broj prethodnih intervala za koje sistem čuva podatke. Kako se najstariji podatak o entropiji briše nakon popunjavanja kompletne kružne liste, minimalan broj intervala koji je potrebno čuvati za ispravan rad je određen kao LS+3, gdje je LS broj intervala za učenje. Tri dodatna intervala su u najgorem slučaju potrebna za prolazak kroz sve faze detekcije napada. Nakon završetka napada, vrijednosti entropije za intervale tokom kojih je napad bio aktivan se ignorišu, ali su potrebni za ispravnu detekciju. Bilo koja vrijednost parametra TS koja je veća od minimalne vrijednosti je dovoljna za ispravan rad. Za potrebe testiranja za ovaj rad, vrijednost parametra TS je bila postavljena na 30.

  Iako ne postoji formalno ograničenje po pitanju vrijednosti parametra SLD, nema smisla da njegova vrijednost bude manja od vrijednosti parametra DKI. Najbolji rezultati se postižu kada je vrijednost ovog parametra jednaka vrijednosti DKI parametra (dužini *keepalive* intervala). Procesni elementi putem *keepalive* poruka obavještavaju kontroler o postojanju aktivnog napada kako bi se na vrijeme izvršila suspenzija algoritma za raspoređivanje opterećenja, pa se intuitivno nameće zaključak da *keepalive* interval ne bi smio biti duži od radnog intervala. Za potrebe testiranja u okviru ovog rada, vrijednosti SLD i DKI parametara su uvijek bile jednake,

- prag entropije za detekciju napada (PENT) - ovaj parametar je izražen kao vrijednost u opsegu [0,1] i pokazuje do kojeg praga se smije smanjiti vrijednost entropije izmjerene u aktivnom intervalu prije nego što bude klasifikovana kao (D)DoS napad. Ne predstavlja apsolutnu vrijednost entropije nego procenat bezbjedne vrijednosti entropije. Kako je objašnjeno u sekciji 4.3, povećanje broja paketa prema malom broju hostova će rezultovati smanjivanjem ukupne entropije za odredišne adrese paketa u tom intervalu.

  Izbor vrijednosti PENT parametra zavisi od nekoliko faktora. U prvom redu, broj hostova koji se nalaze u zaštićenoj mreži i tipovi servisa koji se na njima nalaze utiču na količinu saobraćaja koja prolazi kroz svaki procesni element, pa samim tim utiče i na vrijednost entropije u toku napada. Ako procesni element obrađuje saobraćaj za manji broj hostova u zaštićenoj mreži, svako povećanje broja paketa upućenih prema jednom krajnjem hostu u toku jednog intervala će dovesti do izraženije promjene u entropiji na odgovarajućem procesnom elementu nego u slučaju kad je broj krajnjih hostova veći, pa će biti potrebno odabrati manju vrijednost parametra kako bi se izbjeglo klasifikovanje normalnog povećanja obima saobraćaja kao napada. Analogno tome, u situacijama u kojima je jedan procesni element odgovoran za prosljeđivanje paketa prema većem broju hostova u toku svakog intervala, za istu promjenu

entropije kao u prethonom slučaju je potrebno poslati znatno veći broj paketa prema jednom hostu. Drugim riječima, napad istog intenziteta će izazvati manju razliku u entropiji u odnosu na bezbjedni interval što je ukupan broj krajnjih hostova veći.

Uticaj različitih vrijednosti ovog parametra se može posmatrati i kroz zavisnost od broja procesnih elemenata, podrazumijevajući da je broj krajnjih hostova konstantan i da je algoritam za raspoređivanje opterećenja aktivan. Kod manjeg broja procesnih elemenata, svaki element obrađuje veću količinu saobraćaja, pa (D)DoS napad ima slabiji efekat na smanjenje vrijednosti entropije. Povećanje broja procesnih elemenata istovremeno znači da svaki element dobija manju ukupnu količinu saobraćaja (tačnije, analizira pakete za manji broj odredišnih adresnih blokova), pa su efekti isti kao u slučaju variranja broja krajnjih hostova. Da bi se ilustrovala prednost korištenja većeg broja procesora za konstantan broj hostova u mreži (sekcija 5.4.1), za testno okruženje je vrijednost ovog parametra bila postavljena na 0.6,

- prag opterećenja za detekciju napada (POPT) - ovaj parametar predstavlja jedan od sigurnosnih mehanizama protiv neispravne detekcije napada (engl. *false positive detection*). Kao i u slučaju PENT parametra, predstavlja numeričku vrijednost u opsegu [0,1] i pokazuje donju granicu promjene opterećenja između dva susjedna intervala za koju se vrijednost entropije može uzeti kao pouzdana. Na primjer, ako je vrijednost parametra 0.5, izmjerena vrijednost entropije se zanemaruje ako je opterećenje procesnog elementa u aktivnom intervalu manje od 50% opterećenja iz prethodnog intervala. Eksperimenti su pokazali da se u slučaju drastičnog pada broja paketa mogu dobiti vrijednosti entropije koje mogu navesti sistem na pogrešnu odluku ako se prihvate bez dodatne analize,
- minimalni procenat primljenih paketa za napadnuti čvor (MPP) - pokazuje minimalnu količinu saobraćaja koja mora biti upućena prema jednom hostu da bi on bio uvršten u grupu napadnutih hostova. Vrijednost parametra je data u opsegu [0,1] i predstavlja dio ukupnog saobraćaja za aktivni interval. Pošto promjena entropije pokazuje da je ravnomjerna raspodjela saobraćaja narušena u korist malog broja hostova, ovaj parametar je uveden kao kriterijum koji određuje granicu između napadnutih i nenapadnutih odredišnih adresa. Njegova vrijednost najčešće nije od presudnog značaja za funkcionisanje sistema, jer je u slučaju efikasnog (D)DoS napada broj primljenih paketa za napadnuti čvor drastično veća i od zbira primljenih paketa za sve ostale hostove. Za potrebe testiranja u ovom radu, vrijednost ovog parametra je postavljena na 0.1,
- minimalni broj napadača koji je potreban da bi napad bio klasifikovan kao distribuiran (MBN) - vrijednost ovog parametra ne utiče na proces detekcije, jer zaključak koji se donosi na osnovu entropije ne zavisi od broja napadača. Kako je objašnjeno u sekciji 2.1, tip napada utiče isključivo na prevenciju. Osim toga, u distribuiranom DoS napadu može učestvovati ogroman broj napadača, koji se ne razlikuju po broju poslatih paketa od standardnog korisnika. Vrijednost parametra treba biti ograničena dostupnim resursima

sistema, jer se kod prevencije nedistribuiranih napada generiše po jedno pravilo za svaku adresu sa koje je napad upućen,

- inicijalna dužina zabrane (IDZ) - parametar koji pokazuje trajanje zabrane u sekundama za adresu koja se prvi put pojavljuje u svojstvu napadača. Kako je objašnjeno u sekciji 4.3, sistem koristi mehanizam penalizacije kakav se pojavljuje kod BGP protokola. Ako se ista adresa pojavi kao izvor napada u toku intervala čija je dužina jednaka dvostrukoj dužini prethodne zabrane, trajanje zabrane se udvostručuje. Ovaj mehanizam služi da bi se omogućilo jednostavno blokiranje adresa koje konstantno učestvuju u (D)DoS napadima na duže vrijeme. Ako se po isteku zabrane ista adresa ne pojavi u toku sljedećeg intervala čija je dužina jednaka dužini zabrane, podatak o ranijem učestvovanju u napadima se briše. U slučaju ponovne pojave iste adrese u nekom budućem napadu, saobraćaj sa te adrese će biti blokiran u trajanju inicijalne dužine zabrane,
- dozvoljena varijacija entropije između susjednih intervala u toku napada (DVE) - ovaj parametar predstavlja još jedan u nizu sigurnosnih mehanizama kojima se postiže veća robusnost sistema i smanjuje broj potencijalno pogrešnih odluka. Čak i u slučaju da napad ima konstantan intenzitet, zbog promjena u pozadinskom saobraćaju (saobraćaju koji ne učestvuje u napadu) može doći do promjene vrijednosti entropije za broj paketa u toku aktivnog intervala. Vrijednost entropije može biti narušena promjenama u izgledu pozadinskog saobraćaja tako da sistem pretpostavi da je napad završen i da je potrebno deaktivirati sve trenutno aktivne alarme. DVE parametar definiše dozvoljenu varijaciju vrijednosti entropije između dva susjedna intervala unutar koje sistem neće donijeti odluku o prestanku napada. Vrijednost parametra ne bi trebalo da je velika, jer se podrazumijeva da glavninu saobraćaja i dalje čine paketi upućeni od strane napadača, a da pozadinski saobraćaj predstavlja samo manji dio ukupnog broja paketa. Vrijednost DVE parametra korištena u radu je 0.1,
- minimalni broj poslatih paketa da bi adresa pošiljaoca bila proglašena za napadača (MPOP) - na osnovu ovog parametra se određuje broj paketa koji mora biti poslat sa jedne adrese kako bi host na toj adresi bio proglašen za napadača. Kako bi se omogućilo izolovanje napadača za bilo kakav oblik (D)DoS napada, vrijednost ovog parametra se definiše u odnosu na srednju vrijednost broja poslatih paketa za svaku adresu pošiljaoca. Ako je broj poslatih paketa veći od srednje vrijednosti poslatih paketa po jednom pošiljaocu za procenat definisan vrijednošću MPOP parametra, pošiljalac se klasifikuje kao napadač. U slučaju da se radi o distribuiranom napadu, prosječan broj paketa koji pošalje jedan napadač ne odstupa mnogo od srednje vrijednosti za sve pošiljaoce, pa zbog toga i nije moguće uspješno filtrirati napadače u slučaju distribuiranog napada. Broj adresa koje odgovaraju ovom kriterijumu bi, u tom slučaju, prešao vrijednost određenu parametrom MBN i napad bi bio klasifikovan kao (D)DoS,
- dužina prefiksa (DP) - prefiks, u kontekstu raspoređivanja opterećenja po procesnim elementima TIDS-a, predstavlja nedjeljivi mrežni blok koji se može migrirati sa jednog na drugi procesni element. DP predstavlja jedan

od najvažnijih parametara za rad sistema, jer njegova vrijednost značajno utiče na efikasnost i preciznost sistema za detekciju. Algoritam za detekciju napada je testiran sa vrijednostima DP parametra u opsegu od 16 do 28, dok su za testiranje algoritma za raspoređivanje korištene vrijednosti iz opsega od 12 do 28.

- dozvoljena varijacija opterećenja (DVO) - parametar koji pokazuje maksimalno odstupanje opterećenja nekog procesnog elementa od srednjeg opterećenja za sve aktivne procesne elemente. Prilikom izbora vrijednosti parametra, potrebno je imati na umu da će male vrijednosti izazvati često pokretanje algoritma za raspodjelu opterećenja, što može biti problem u slučaju da je količina dostupnih resursa za SDN kontroler nedovoljna za često izvršavanje takve procedure. Sistem je testiran za različite vrijednosti ovog parametra, pri čemu je vrijednost 0.05 dala zadovoljavajuće rezultate. To znači da će algoritam za raspodjelu opterećenja biti pokrenut ako je opterećenje bar jednog elementa za 5% veće od srednje vrijednosti opterećenja za sve elemente koja je izračunata na osnovu sadržaja *keepalive* poruka,
- dužina *holddown* perioda (DHP) - *holddown* period predstavlja interval nakon završetka algotirma za raspoređivanje opterećenja nakon kojeg je dozvoljeno njegovo ponovno pokretanje. Ovaj parametar funkcioniše kao sigurnosni mehanizam u slučajevima premalih vrijednosti DVO parametra ili mreže kod koje su promjene intenziteta saobraćaja česte. Sličan koncept je prisutan kod različitih implementacija protokola rutiranja koji uključuju procedure sa kompleksnim izračunavanjem u situacijama nestabilnosti mreže [130]. U sekciji 7.2 su prikazani efekti različitih vrijednosti ovog parametra na algoritam za raspoređivanje opterećenja,
- maksimalan broj pošiljalaca (MBPO) i primalaca (MBPR) u jednom intervalu - zbog efikasnosti rada kompletnog sistema, memorijski prostor koji je potrebam za čuvanje statističkih podataka u toku aktivnog intervala se alocira prilikom pokretanja. U heš mapi se za svaku IP adresu u 32-bitnom polju čuva podatak o broju paketa, pa bi za kompletan IPv4 adresni prostor bilo potrebno alocirati 32 GB memorije za svaku od mapa. Time bi se onemogućila implementacija na jeftinom, nespecijalizovanom hardveru. Iz tog razloga se definišu parametri koji ograničavaju veličinu mapa na vrijednosti koje su znatno manje od kompletnog IPv4 adresnog prostora. To znači da bi, u slučaju da je mapa popunjena, podaci za neke ključeve morali biti izgubljeni, ali se pretpostavlja da je broj različitih adresa u okviru jednog intervala mnogo manji od broja adresa u kompletnom adresnom prostoru. Za potrebe testnog okruženja, parametar MBPR je bio konfigurisan vrijednošću 0x40000 (262144 različitih adresa, 2 MB potrebne memorije), a parametar MBPO vrijednošću 0x10000 (65536 različitih adresa, 512 kB potrebne memorije), što se na bazi podataka korištenih u ovom radu pokazalo kao dovoljna količina memorije za dobijanje ispravnih rezultata.
- broj intervala za praćenje trenda (BIT) - ovaj parametar određuje broj intervala u toku kojih će biti praćen trend kretanja vrijednosti entropije kako bi se detektovale sporije promjene vrijednosti koje imaju za cilj da ublaže efekte (D)DoS napada kako bi se otežala detekcija. Kako je već opisano u

sekciji 4.3, da bi bili ispunjeni uslovi za detekciju, funkcija vrijednosti entropije u toku ovog perioda mora biti nerastuća, a odnos vrijednosti na početku i na kraju perioda mora biti manji od praga definisanog PENT parametrom. Izbor vrijednosti parametra zavisi od dužine pojedinačnog intervala, ali ne bi trebalo da bude veći od vrijednosti TS parametra (ukupan broj intervala).

Navedeni parametri nemaju jednak značaj ispravno funkcionisanje produkcione implementacije sistema. Od kritičnog značaja su dužina *keepalive* intervala (DKI), dužina radnog intervala (SLD), dužina prefiksa (DP) i prag entropije za detekciju napada (PENT) moraju biti pažljivo odabrani i prilagođeni konkretnom okruženju, dok ostali parametri nude veću robusnost ili fleksibilnost sistema, ali ne mogu dovesti do neispravnog rada ako bi bile zadržane njihove podrazumijevane vrijednosti. Vrijednost ostalih parametara vrlo često zavisi od količine dostupnih resursa sistema (npr. maksimalan broj pošiljalaca (MBPO) i primalaca (MBPR), broj prethodnih intervala za koje se čuvaju podaci i sl.), pa njihove vrijednosti mogu biti prilagođene u bilo kojem trenutku nakon stavljanja sistema u produkciju bez narušavanja ispravnosti rada.

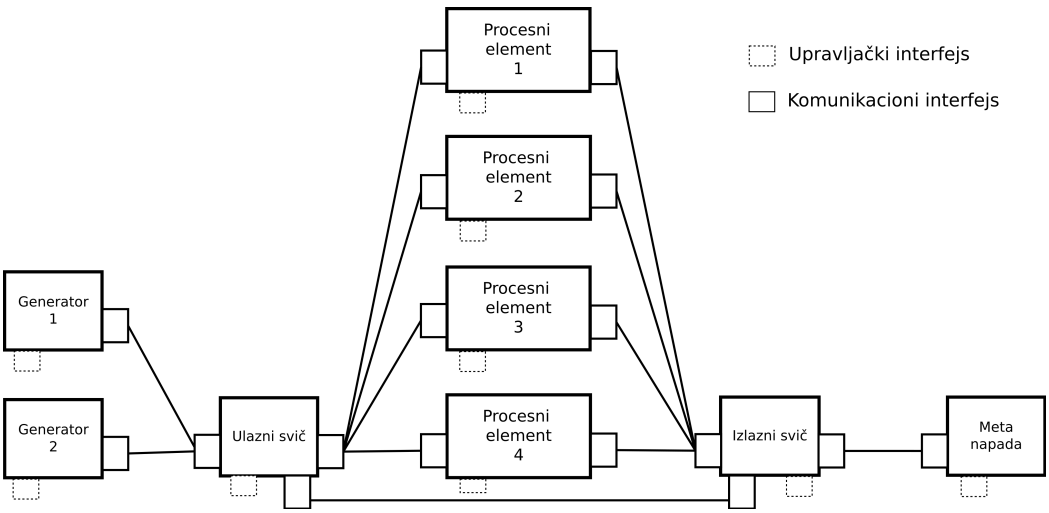
5. Eksperimentalni rezultati

5.1 Opis testnog okruženja

Za potrebe dobijanja rezultata prikazanih u ovoj sekciji implementirano je laboratorijsko okruženje u kojem su simulirani DDoS napadi generisanjem ogromnog broja paketa u kratkom vremenskom periodu. Paketi koji čine napad su pomiješani sa saobraćajem dobijenim prikupljanjem podataka sa stvarnih računarskih mreža. Cilj eksperimenata je bio da se pokaže da je sistem u stanju da detektuje napad bez obzira na izgled legitimnog korisničkog saobraćaja i da se pokaže uticaj izbora različitih parametara na rad sistema.

Sistem koji je korišten za testiranje je implementiran korištenjem virtuelne infrastrukture na standardnom potrošačkom hardveru. Korištenjem virtuelne infrastrukture je pokazana jednostavnost proširenja kapaciteta sistema, jer se dodavanje novih procesnih elemenata svodi na kloniranje postojećih virtuelnih mašina i njihovo uključivanje u mrežu.

Osnovni izgled sistema je dat na slici 5.1. Svi hostovi na sistemu imaju minimalno 2 interfejsa: prvi interfejs se koristi za upravljanje virtuelnom mašinom putem SSH protokola, dok se drugi koristi za interno prosljeđivanje paketa kroz TIDS. Upravljački interfejs ima IP adresu koja je korištena za udeljeni pristup i saobraćaj prema tom interfejsu se ne kontroliše od strane TIDS-a.



Slika 5.1 *Izgled testnog okruženja TIDS-a*

Komunikacioni mrežni interfejsi prosljeđuju saobraćaj koji dolazi iz vanjske

mreže (u slučaju testnog okruženja, sa hostova označenih kao generatori) ili iz unutrašnje mreže (u slučaju testnog okruženja, sa hosta označenog kao meta napada) i za konačni cilj nemaju hostove koji čine TIDS.

Na lijevoj strani dijagrama su prikazani hostovi koji imaju zadatak da simuliraju saobraćaj koji dolazi iz spoljašnje mreže (Generator 1 i Generator 2), od strane potencijalnih napadača ili legitimnih korisnika. Postavka na dijagramu sadrži dva ovakva hosta kako bi se napadački saobraćaj mogao generisati simultano sa korisničkim. Komunikacioni interfejsi na ovim hostovima imaju adresne informacije, ali se adrese sa tih interfejsa najčešće ne koriste u slanju paketa. Tačnije, jedini slučaj u kojem se adresa interfejsa koristi za slanje paketa je ako se generiše saobraćaj koji pripada nedistribuiranom DoS napadu. Kod generisanja distribuiranog DoS napada, koristi se unaprijed definisana lista IP adresa. Bez obzira na to što se adresa ne koristi, interfejs mora imati adresne informacije kako bi se uopšte mogao aktivirati i koristiti od strane aplikacija koje generišu saobraćaj.

Na desnoj strani dijagrama je prikazan host koji predstavlja metu (na slici označen kao "Meta napada"). Za testiranje procedura detekcije, postojanje ovog hosta nije obavezno, jer je dovoljno da paketi dođu do procesnih elemenata. Međutim, odredišni host je korišten zbog lakšeg mjerenja broja proslijeđenih paketa i za testiranje ispravnosti komunikacije kroz transparentni sloj. Da bi komunikacija bila ostvarena kroz transparentni sloj, potrebno je napraviti razmjenu paketa na drugom sloju OSI modela na način na koji bi to bilo urađeno u slučaju da su hostovi spojeni direktno (ARP komunikacija koja je potrebna na lokalnoj mreži za ostvarivanje bilo kakve konekcije na višim slojevima OSI modela). Ako jedan od generatora uspješno ostvari komunikaciju sa metom, to znači da su ARP paketi uspješno razmijenjeni i da su MAC tabele na oba hosta uspješno popunjene MAC i IP adresama druge strane. U slučaju da srednji sloj nije transparentan, MAC tabela bi bila popunjena adresom ulaznog sviča (za generator), odnosno izlaznog sviča (za metu napada).

Komunikacioni intefejs na hostu koji predstavlja metu napada, kao i u slučaju generatora, može biti adresiran bilo kojom adresom. U ovom slučaju, MAC tabela je na generatorima popunjena statički, tako što je ista MAC adresa (stvarna MAC adresa mete) unesena kao odgovarajuća adresa za IP adrese iz dva odvojena adresna bloka. U tom slučaju, generatori ne moraju slati ARP pakete, već imaju sve što je potrebno za slanje paketa. Na ovaj način je omogućeno slanje paketa prema dva odredišna hosta, iako se svi paketi prosljeđuju prema istoj virtuelnoj mašini, čime se pojednostavljuje testiranje.

U srednjem dijelu su hostovi koji čine TIDS. Sa lijeve strane, između generatora i procesnih elemenata, nalazi se virtuelna mašina koja funkcioniše i kao ulazni svič i kao SDN kontroler za taj svič. U pitanju je server koji nema specijalnih hardverskih komponenti i na kojem se izvršava operativni sistem baziran na Linux-u uz softver koji omogućava kreiranje virtuelnog sviča. Server sadrži jednojezgarni Intel Xeon E5-2420 procesor i 8 GB radne memorije. Količina radne memorije na ovom hostu je nešto veća nego u slučaju ostalih hostova, jer je uočeno da u slučaju nedovoljne količine memorije dolazi do usporenja rada algoritma za raspoređivanje opterećenja (tačnije, proces zamjene postojećih pravila na sviču traje znatno duže i ometa rad algoritma). Komunikacione interfejse čine logička

veza prema spoljašnjoj mreži (za testno okruženje to znači da su kreirana dva interfejsa koji povezuju svič sa generatorima saobraćaja), po jedan interfejs prema procesnim elementima i jedan interfejs kojim je direktno spojen sa izlaznim svičem. Ni za jedan od navedenih interfejsa nije bilo potrebno konfigurisati IP adresu.

Procesni elementi predstavljaju virtuelne mašine koje funkcionišu na istom virtuelizacionom hostu kao i SDN svič, uz manju količinu radne memorije (1-2 GB). Komunikacione interfejse na procesnim elementima čine dva neadresirana interfejsa, jedan kojim je procesni element spojen sa ulaznim svičem i jedan kojim je spojen za izlaznim svičem. Saobraćaj dolazi na procesni element samo sa ulaznog sviča, dok je interfejs prema izlaznom sviču rezervisan za prosljeđivanje paketa prema unutrašnjoj mreži.

Konačno, izlazni svič je implementiran u vidu virtuelne mašine relativno slabih karakteristika, jer se od ovog hosta ne očekuju kompleksna izračunavanja koja bi zahtijevala veću količinu resursa. Sva pravila su definisana statički, i sastoje se od prosljeđivanja saobraćaja koji dođe preko bilo kojeg interfejsa od procesnih elemenata prema unutrašnjoj mreži. Saobraćaj koji dolazi iz unutrašnje mreže kao odgovor se prosljeđuje prema ulaznom sviču korištenjem direktne konekcije.

Virtuelizaciono okruženje je implementirano korištenjem VMWare ESXi rješenja [131], putem kojeg su kreirane virtuelne mašine i konekcije između njihovih mrežnih interfejsa. Da bi se izbjegao gubitak paketa zbog validacije adresa koja je ugrađena u VMWare, svi interfejsi funkcionišu u tzv. promiskuitetnom načinu rada u kojem se ne provjerava vjerodostojnost pošiljaoca, već se svi paketi automatski prihvataju na obradu.

5.2 Opis setova podataka korištenih za testiranje

Kako je već objašnjeno, kod testiranja su korištena dva različita seta podataka. Prvi set se sastojao od tzv. pozadinskog saobraćaja, koji predstavlja saobraćaj prikupljen na nekoj aktivnoj mreži koja trenutno nije pogođena nekim (D)DoS napadom. Većina setova podataka koji su dostupni na Internetu je prije objavljivanja podvrgnuta procesu anonimizacije prefiksa. Anonimizacija predstavlja uklanjanje podataka o korisnicima iz sadržaja paketa (u prvom redu, IP adrese i MAC adrese korisnika koji su originalno učestvovali u komunikaciji). Postoje različite tehnike anonimizacije [132], ali im je zajednička osobina da zadržavaju validnost paketa i međusobne odnose između različitih mrežnih prefiksa kakvi su postojali između originalnih paketa. Kao rezultat, dobijaju se setovi podataka koji ne sadrže osjetljive informacije o korisnicima, ali zadržavaju statističke osobine koje su potrebne za testiranje.

Svi korišteni setovi podataka sadrže i određene vrste napada ugrađene u sam sadržaj (engl. *trace*). Međutim, najčešće se radi o drugačijim vrstama napada koji uključuju posebno formirane pakete, a ne veliki broj poslatih paketa u maloj jedinici vremena. Ovo je tipično za sve setove koji sadrže napade koji se zasnivaju na posebno formiranim paketima ili paketima koji se moraju poslati u naročitoj sekvenci, pri čemu brzina slanja nije od presudnog značaja. Međutim, takvo označavanje paketa nema značaj za napade koje detektuje TIDS, jer se bilo

kakav set podataka može posmatrati kao napad ako se pošalje dovoljnim intenzitetom. Iz tog razloga su prikazani setovi korišteni kao pozadinski saobraćaj, jer sadrže pakete sa različitim protokolima i uglavnom su pravilno raspoređeni po odredišnim adresama.

Primarni set podataka korišten u ovu svrhu je *Simpleweb* set dobijen prikupljanjem saobraćaja na rezidencijalnoj mreži studentskog kampusa Univerziteta u Tventeu [133]. Prikupljeni saobraćaj je podvrgnut anonimizaciji prije objavljivanja i sadrži preko 10000 različitih IP adresa. Radi lakšeg preuzimanja, kompletan set je rastavljen na segmente koji obuhvataju period manji od jednog dana. Za potrebe testova, paketi iz setova su generisani alatom *tcpreply* [134] onim redoslijedom kojim su originalno i prikupljeni. Korištena je multiplikacija brzine slanja u odnosu na originalni set kako bi se postigla brzina od oko 80Mbps. Ovi setovi su korišteni za testiranje oba algoritma.

Dodatni testovi su izvršeni korištenjem CAIDA seta podataka dostupnog u okviru PREDICT repozitorijuma [135]. Međutim, originalna brzina paketa u ovom setu je vrlo mala, pa su oni bili neadekvatni za testiranje. Da bi se postigla odgovarajuće brzina, slanje bi moralo biti ubrzano više od 100 puta, pri čemu bi se narušila prirodna dinamika promjene obima saobraćaja (tj. porast u broju paketa koji se redovno dešava sredinom dana u odnosu na rane jutarnje časove bi bio sažet u nekoliko minuta, što bi moglo biti detektovano kao napad).

Drugi dio saobraćaja je korišten samo kod testiranja algoritma za detekciju napada i predstavlja pakete koji su generisani alatom za simulaciju DDoS napada. Sadržaj tih paketa nije bitan, ali predstavljaju opasnost zbog ukupne količine koja se generiše u kratkom vremenu. Za generisanje je korišten alat pod nazivom BoNeSi [136], koji daje mogućnost specifikovanja brzine slanja i broja izvornih adresa koje će biti korištene. Ako broj adresa nije specifikovan, podrazumijeva se da će se slanje vršiti sa stvarne adrese mrežnog interfejsa. U tom slučaju će biti generisan nedistribuirani DoS napad. Za generisanje distribuiranog napada potrebno je navesti datoteku u kojoj su navedene IP adrese čije će vrijednosti biti upisane u polje rezervisano za adresu pošiljaoca u IP paketu. BoNeSi dolazi sa unaprijed definisanim spiskom od 50000 IPv4 adresa za potrebe simuliranja distribuiranog DoS napada.

5.3 Metodologija testiranja

Testiranje je provedeno u dva segmenta. Prvi dio testiranja se odnosi na efikasnost sistema za detekciju (D)DoS napada. Obzirom da sistem radi sa velikim brojem konfiguracionih parametara koji utiču na njegovu efikasnost, testovi su uključivali izvršavanje napada za sve kombinacije parametara koji mogu imati bitan uticaj na efikasnost detekcije. Na osnovu dobijenih rezultata je izvršena analiza rada sistema i date su preporučene vrijednosti testiranih parametara. Sve vrijednosti parametara su testirane za različite brojeve aktivnih procesnih elemenata, u cilju poređenja distribuirane i nedistribuirane arhitekture sistema.

Svaki test se, bez obzira na vrijednosti parametara, odvija istom sekvencom. Sistem je izložen samo pozadinskom saobraćaju u prvih 5 minuta testa. Nakon

toga, pokreće se generisanje dva napada u razmaku od 30 sekundi. Jedan test traje ukupno 20 minuta. U tabeli 5.1 je dat pregled vrijednosti parametara za koje je testiran rad algoritma za detekciju napada.

Testovi algoritma za raspoređivanje imaju pojedinačno trajanje od 15 minuta, jer je testirano ponašanje sistema koji je izložen samo pozadinskom saobraćaju.

Redni broj	Dužina radnog intervala i *keepalive* intervala (SLD i PKI)	Dužina prefiksa (DP)	Broj aktivnih procesnih elemenata	Slike
1	15 sekundi	/16	1	7.1 i 7.2
2	15 sekundi	/16	2	7.3 i 7.4
3	15 sekundi	/16	3	7.5 i 7.6
4	15 sekundi	/16	4	7.7 i 7.8
5	30 sekundi	/16	1	7.9 i 7.10
6	30 sekundi	/16	2	7.11 i 7.12
7	30 sekundi	/16	3	7.13 i 7.14
8	30 sekundi	/16	4	7.15 i 7.16
9	60 sekundi	/16	1	7.17 i 7.18
10	60 sekundi	/16	2	7.19 i 7.20
11	60 sekundi	/16	3	7.21 i 7.22
12	60 sekundi	/16	4	7.23 i 7.24
13	15 sekundi	/24	1	7.25 i 7.26
14	15 sekundi	/24	2	7.27 i 7.28
15	15 sekundi	/24	3	7.29 i 7.30
16	15 sekundi	/24	4	7.31 i 7.32
17	30 sekundi	/24	1	7.33 i 7.34
18	30 sekundi	/24	2	7.35 i 7.36
19	30 sekundi	/24	3	7.37 i 7.38
20	30 sekundi	/24	4	7.39 i 7.40
21	60 sekundi	/24	1	7.41 i 7.42
22	60 sekundi	/24	2	7.43 i 7.44
23	60 sekundi	/24	3	7.45 i 7.46
24	60 sekundi	/24	4	7.47 i 7.48
25	15 sekundi	/28	1	7.49 i 7.50
26	15 sekundi	/28	2	7.51 i 7.52
27	15 sekundi	/28	3	7.53 i 7.54
28	15 sekundi	/28	4	7.55 i 7.56
29	30 sekundi	/28	1	7.57 i 7.58
30	30 sekundi	/28	2	7.59 i 7.60
31	30 sekundi	/28	3	7.61 i 7.62
32	30 sekundi	/28	4	7.63 i 7.64
33	60 sekundi	/28	1	7.65 i 7.66
34	60 sekundi	/28	2	7.67 i 7.68

35	60 sekundi	/28	3	7.69 i 7.70
36	60 sekundi	/28	4	7.71 i 7.72

Tabela 5.1 *Testirane vrijednosti parametara za algoritam za detekciju napada*

Redni broj	Dužina *holddown* (DHP) intervala	Dužina prefiksa (DP)	Broj aktivnih procesnih elemenata	Nazivi slika
1	5 sekundi	/12	3	7.73 i 7.74
2	5 sekundi	/12	4	7.75 i 7.76
3	5 sekundi	/24	3	7.77 i 7.78
4	5 sekundi	/24	4	7.79 i 7.80
5	5 sekundi	/28	3	7.81 i 7.82
6	5 sekundi	/28	4	7.83 i 7.84
7	15 sekundi	/12	3	7.85 i 7.86
8	15 sekundi	/12	4	7.87 i 7.88
9	15 sekundi	/24	3	7.89 i 7.90
10	15 sekundi	/24	4	7.91 i 7.92
11	15 sekundi	/28	3	7.93 i 7.94
12	15 sekundi	/28	4	7.95 i 7.96
13	45 sekundi	/12	3	7.97 i 7.98
14	45 sekundi	/12	4	7.99 i 7.100
15	45 sekundi	/24	3	7.101 i 7.102
16	45 sekundi	/24	4	7.103 i 7.104
17	45 sekundi	/28	3	7.105 i 7.106
18	45 sekundi	/28	4	7.107 i 7.108

Tabela 5.2 *Testirane vrijednosti parametara za algoritam za raspoređivanje opterećenja*

5.4 Diskusija

5.4.1 Analiza rezultata testiranja

5.4.1.1 Analiza rada algoritma za detekciju napada

Rezultati koji su dati u poglavlju 7 su iskorišteni za analizu uticaja vrijednosti korištenih parametara na rad sistema (date u sekciji 4.5). Kod detekcije napada, najznačajniji parametri su dužina radnog (SLD) i *keepalive* (DKI) intervala, te dužina prefiksa (DP), dok je kod algoritma za raspoređivanje opterećenja osim dužine prefiksa od presudnog značaja i dužina *holddown* perioda (DHP).

Različite vrijednosti parametara za testiranje algoritma za detekciju (D)DoS napada su date u tabeli 5.1, pri čemu je vrijednost DKI parametra bila jednaka vrijednosti SLD parametra kako bi se postigla sinhronizacija podataka o opterećenju

i entropiji u bilo kojem trenutku rada sistema. Kako je već rečeno u sekciji 4.5, veće vrijednosti SLD parametra mogu uticati na smanjenje osjetljivosti sistema na kraće promjene saobraćaja, jer će se ekstremne vrijednosti izgubiti usrednjavanjem u toku dužeg vremenskog perioda. S druge strane, prekratak radni interval će dovesti do toga da takve kratke promjene budu detektovane kao napadi, iako se radi o redovnim događajima na mreži koje mogu nastati kao posljedica čitavog niza situacija, bilo da se radi o privremenim prekidima ili povećanom obimu saobraćaja usljed normalnih dnevnih aktivnosti korisnika. Veličina prefiksa posredno utiče na algoritam za detekciju jer od rasporeda saobraćaja po procesnim elementima zavisi i uticaj napada na ukupnu vrijednost entropije. Veći prefiksi (tj. manje vrijednosti DP parametra) u kombinaciji sa dužim radnim intervalima znače da će sistemu biti potrebno više vremena za stabilizaciju i da će promjene rasporeda prilikom raspoređivanja biti drastičnije, što može uticati i na algoritam za detekciju. Jačina tog uticaja zavisi i od prirode pozadinskog (validnog) saobraćaja u mreži, jer za saobraćaj koji je ravnomjerno raspoređen i bez algoritama za raspoređivanje neće biti potrebno vršiti korekcije, pa ni vrijednosti ovih parametara nisu u tolikoj mjeri značajne. Navedeno je posebno vidljivo u testovima kod kojih dužina radnog intervala ima vrijednost 60 sekundi, a veličina prefiksa ima vrijednost 16 (slike 7.17, 7.19, 7.21, 7.23). Slično je i sa vrijednošću entropije za iste testove, pa se ovim opravdava odluka da se u trenutku prelaska u prvu fazu alarma za detekciju napada privremeno suspenduje preraspodjela saobraćaja kako ne bi uticala na detekciju. Nakon završetka napada (bez obzira da li je u pitanju bio stvarni napad ili je sistem prešao iz neke od faza alarma u bezbjedno stanje), sistem ponovo mora izbalansirati opterećenja po procesnim elementima. Obzirom da je od prethodne preraspodjele proteklo dovoljno vremena da se trenutna pravila koja upravljaju raspoređivanjem mogu smatrati nevalidnim, kontroler zadržava samo osnovna pravila najnižeg prioriteta (bazirana na nižim bitima odredišnih IP adresa) i odbacuje sva ostala koja su definisana u toku dotadašnjeg rada. Sa slika 7.29, 7.31, 7.53 i 7.55 je jasno da se konvergencija postiže u relativno malom broju *keepalive* intervala, te da se u tom periodu vrijednosti opterećenja i entropije po procesnim elementima vraćaju u stanje u kojem su se nalazili neposredno prije nego što je napad počeo.

Posebna pažnju u analizi testova je posvećena uticaju broja procesnih elemenata na rad sistema. Iz rezultata testiranja se dolazi do zaključka da je detekcija napada jednostavnija što je broj aktivnih procesora veći, čime je dokazano da skalabilnost sistema ne utiče negativno na njegov rad već da, naprotiv, doprinosi efikasnosti algoritma za detekciju. Naime, da bi se ilustrovala ova tvrdnja, prag entropije za detekciju (parametar PENT) je za sve testove u prilogu postavljen na 0.6. Sa slika koje prikazuju vrijednosti entropije u postavkama sa jednim i dva procesora (slike 7.2, 7.4, 7.10, 7.12 itd.) se vidi da sistem nije detektovao napade jer pad entropije nije dovoljan da bi se vrijednost spustila ispod definisanog praga. Kod postavke sa jednim procesnim elementom, pad entropije je tek oko 10%, što je daleko ispod potrebnih 40%. S obzirom da napad nije detektovan, maliciozni paketi se propuštaju u unutrašnju mrežu i nakon 3 sljedeća radna intervala srednja vrijednost (bezbjedne) entropije će biti jednaka entropiji u toku napada, nakon čega detekcija više i nije moguća. Ovo je posljedica činjenice da kod manjeg broja procesora svaki procesor obrađuje veću količinu saobraćaja, pa je i broj odredišnih adresa znatno veći nego u slučaju većeg broj procesnih elemenata. Povećanje broja

paketa u korist malog broja odredišnih hostova svakako ima uticaj na vrijednost entropije, ali je taj uticaj manje vidljiv jer je broj faktora koji učestvuje u sumi datoj u formuli 4.1 u tom slučaju veći. Sa druge strane, što je broj procesnih elemenata veći, broj odredišnih adresa je proporcionalno manji, pa je i smanjenje entropije lakše detektovati (slike 7.54, 7.56 itd.). Iz rezultata datih na ovim grafovima se vidi da se vrijednost entropije u toku napada spušta i za 60% originalne vrijednosti u toku samo jednog intervala, što je više nego dovoljno za uspješnu detekciju napada. Ako se pritom koriste manji prefiksi (tj. veće vrijednosti DP parametra), sistemu se pruža mogućnost da kroz preraspodjelu opterećenja rasporedi ukupan broj odredišnih adresa po procesnim elementima tako da potencijalni napad ima isti efekat na bilo kojem od njih. Nema garancije da će jednak broj odredišnih prefiksa na svakom procesnom elementu odgovarati broju obrađenih paketa, ali su šanse za to veće ako se koriste manji prefiksi.

Konfigurisanje praga entropije predstavlja jedini aspekt sistema u kojem se može iskoristiti poznavanje načina funkcionisanja mreže od strane mrežnog administratora, dok je u svim ostalim slučajima sistem u stanju da se adaptira na promjene u mreži koje odstupaju od naučenog normalnog stanja. Za niži prag se sistemu omogućava da funkcioniše i sa neoptimalno odabranim vrijednostima konfiguracionih parametara, jer se očekuje da će napad imati toliki intenzitet da će izazvati drastičan pad entropije bez obzira na trenutne vrijednosti. Time se otvara mogućnost neispravne detekcije (tzv. *false negative*), gdje sistem ignoriše aktivan napad (kao u slučaju rezultata testova sa jednim i dva procesna elementa). Posljedice nedetektovanja napada se mogu odraziti i na detekciju kasnijih napada jer se izmjerena vrijednost entropije uzima kao nova bezbjedna vrijednost ako u tom intervalu ništa nije detektovano, pa će se sistem adaptirati na novonastalo stanje i tretirati ga kao regularno ponašanje mreže. To dalje znači da će za uspješnu detekciju vrijednost entropije morati pasti ispod praga koji se sad utvrđuje u odnosu na vrijednost koja je već snižena u odnosu na normalnu (kao posljedica nedetektovanja napada), čime je potrebno da napad ima još veću snagu da bi bio detektovan.

Iz prethodno navedenog slijedi da je dobra praksa za izbor praga entropije postavljanje vrijednosti koja će predstavljati granicu između pada entropije koji dolazi kao posljedica skoka u broju paketa koji odredišni sistem može obraditi i one količine saobraćaja za koju se očekuje da će imati negativne posljedice po funkcionisanje samog servisa. Efikasnost (D)DoS napada koji nastaju zbog velike količine saobraćaja zavisi od mogućnosti odredišnog sistema u većoj mjeri nego što je to slučaj sa nekim drugim napadima. Primjera radi, za efikasno izvođenje napada kakvi su *SQLInjection* i *Cross-Site Scripting* dovoljan je jedan pažljivo formiran zahtjev i njihovo uspješno izvođenje ne zavisi od procesnih mogućnosti sistema koji taj zahtjev obrađuje. Sa druge strane, postavlja se pitanje da li se volumetrijski napad čije je pakete sistem u stanju obraditi bez pomoći sistema za detekciju može uopšte smatrati napadom ili se može svrstati u klasu nevalidno formiranih zahtjeva koji su redovna pojava na svakoj mreži i bez (D)DoS napada. Ovaj problem se svrstava u istu grupu sa problemom detekcije *flash crowd*-a i detaljnije je obrađen u sekciji 5.4.2.

Konačno, na osnovu rezultata testiranja je moguće potvrditi ranije preporuke za izbor optimalnih vrijednosti parametara za algoritam za detekciju napada.

Testovi pokazuju da smanjenje dužine radnog intervala utiče pozitivno na rad algoritma, ali da minimalna vrijednost u određenoj mjeri zavisi od količine saobraćaja koja u toku intervala protiče kroz mrežu. Za prosječne brzine komunikacije do 100 Mbps, ovaj interval bi trebalo odabrati u opsegu od 10 do 15 sekundi. Sve manje vrijednosti donose rizik proglašavanja napada na osnovu nedovoljne količine podataka, dok veće vrijednosti povećavaju robusnost algoritma, ali produžavaju vrijeme potrebno za reakciju na napad.

Sa druge strane, veće vrijednosti DP parametra uvijek daju bolje rezultate po pitanju detekcije, ali je njihova upotreba ograničena procesnim kapacitetima SDN kontrolera i sviča S_1. Ako sistem ima dovoljno radne memorije i procesorske snage, nema ograničenja po pitanju izbora vrijednosti ovog parametra. Za testove u prilogu, zadovoljavajući rezultati su dobijeni korištenjem prefiksa veličine 24 i 28, dok je za prefiks veličine 16 uočena nešto manja stabilnost sistema u smislu jednakog opterećenja i preciznosti algoritma za detekciju. Ovi rezultati su dodatno potvrđeni testovima algoritma za raspoređivanje bez detekcije čiji su rezultati analizirani u nastavku ovog poglavlja. U tabeli 5.3 su sumirani efekti promjena vrijednosti parametara za rad ovog algoritma.

Parametar	Opis	Efekti povećavanja vrijednosti	Efekti smanjivanja vrijednosti
KPI	Dužina *keepalive* intervala	Manje opterećenje sistema, ali i manja mogućnost detekcije	Češće prosljeđivanje informacija o potencijalnom napadu, brži odziv sistema
SLD	Dužina radnog intervala	Rjeđe računanje entropije, neosjetljivost na manje kratkotrajne fluktuacije	Veće opterećenje, preciznije reakcije sistema na promjene (ako je i vrijednost KPI parametra odgovarajuća)
PENT	Prag entropije za detekciju	Osjetljiviji sistem, veća mogućnost lažnih detekcija (engl. *false positive*)	Sposobnost detekcije napada veće magnitude, veća mogućnost lažnih negativnih detekcija (engl. *false negative*)

Tabela 5.3 *Efekti promjena vrijednosti parametara za algoritam za detekciju napada*

5.4.1.2 Analiza rada algoritma za raspoređivanje opterećenja

Kod testiranja algoritma za raspoređivanje opterećenja, za vrijednosti parametara koji ne utiču direktno na njegov rad su odabrane vrijednosti koje su u prvoj seriji testova označene kao optimalne za rad algoritma za detekciju napada. Konkretno, dužine radnog i *keepalive* intervala su postavljene na 10 sekundi.

U slučaju algoritma za raspoređivanje, broj procesnih elemenata se ne može posmatrati kao konfiguracioni parametar, jer algoritam mora ispravno funkcionisati u bilo kojoj implementaciji (tj. sa bilo kojim brojem elemenata). Ispravno ponašanje sistema se dokazuje činjenicom da procesni elementi trpe proporcionalno manje opterećenje za porastom njihovog broja, a da je sistem u svakom slučaju u stanju da ujednači to opterećenje.

Parametri koji imaju najveći uticaj na rad ovog dijela sistema su dužina prefiksa (DP) i dužina *holddown* intervala (DHP). Različite vrijednosti ovih parametara za koje su provedeni testovi su date u tabeli 5.2.

Kako je objašnjeno u sekciji 4.5, DHP parametar označava minimalno vrijeme koje mora proteći od završetka jedne iteracije algoritma za raspoređivanje do početka izvršavanja sljedeće. Ovaj parametar pruža mogućnost sistemu da se stabilizuje nakon preraspodjele mrežnih prefiksa i da prikupi dovoljno podataka o uticaju prethodne raspodjele na opterećenje svakog procesnog elementa. Kraće vrijednosti omogućavaju bržu reakciju, ali često nisu pogodne za korištenje u mrežama koje imaju kratke tokove i kod kojih se u odredišnoj mreži nalazi veliki broj hostova jer će sistem reagovati na kratke promjene prečesto i time čak uticati i na algoritam za detekciju napada. S druge strane, duži interval znači da će se preraspodjele vršiti rjeđe (time se u određenoj mjeri smanjuje zahtjevnost po pitanju resursa), ali i da je veća šansa da su pri svakoj preraspodjeli potrebne drastičnije izmjene na trenutno aktivnim pravilima za raspoređivanje. Naime, očekivano je da se sa protokom vremena narušava uspostavljeni raspored saobraćaja zbog pojave novih i završetka starijih tokova. Ako se algoritam izvršava češće, biće potrebne manje korekcije na postojećim pravilima jer je stepen disbalansa opterećenja između procesora direktno proporcionalan vremenu koje je proteklo od prethodne preraspodjele. Rezultati testiranja pokazuju da se bolje performanse sistema dobijaju izborom kraćeg intervala jer se postiže manji disparitet između opterećenja procesnih elemenata u bilo kojem trenutku, dok se veća potrošnja resursa na kontroleru lako rješava povećavanjem njegovih procesnih mogućnosti.

Slike 7.101, 7.103, 7.105 ilustruju efekte izbora dužih intervala, pri čemu je konvergencija vrijednosti opterećenja znatno sporija zbog rjeđeg pokretanja algoritma za raspoređivanje opterećenja (do potpunog ujednačavanja opterećenja protekne oko 2 minuta). Na ovim slikama je prikazan broj obrađenih paketa, ali je razlika u brzini konvergencije vidljiva i analizom statistike za ukupnu količinu obrađenog saobraćaja (slike 7.102, 7.104, 7.106). Sa druge strane, sa slika 7.91 i 7.95 se vidi da se optimalni rezultati dobijaju izborom dužine intervala iz opsega [10-15] (pri čemu se ujednačavanje opterećenja postiže već nakon jednog minuta), dok je (kao i u slučaju algoritma za detekciju) poželjno da DP parametar ima najveću vrijednost dopuštenu dostupnom količinom sistemskih resursa.

Analiziranjem rezultata se može zaključiti i da algoritam za raspoređivanje funkcioniše bolje u slučaju parnog broja procesora (n>=2), jer za neparan broj procesora već u startu nije moguće prema najnižim bitima odredišne adrese izvršiti ravnomjernu raspodjelu. Međutim, pokazuje se da je sistem u stanju za vrlo kratko vrijeme neutralisati negativne efekte nejednake raspodjele (slike 7.85, 7.89).

U većini slučajeva, vrijednosti dobijene testiranjem različitih parametara se ne razlikuju u ogromnoj mjeri, što dokazuje da je algoritam u stanju da održi performanse sistema za bilo koji odabrani skup parametara. Međutim, izbor optimalnih parametara doprinosi jednostavnijoj detekciji i povećanoj stabilnosti kompletnog sistema. U tabeli 5.4 su sumirani efekti promjena vrijednosti parametara za rad ovog algoritma. Parametar DP je naveden u ovoj tabeli, iako njegova vrijednost posredno utiče i na algoritam za detekciju usljed promjene rasporeda opterećenja koje može izazvati i promjenu entropije.

Parametar	Opis	Efekti povećavanja vrijednosti	Efekti smanjivanja vrijednosti

DHP	Dužina *holddown* intervala	Manje opterećenje sistema, manja osjetljivost na nejednako opterećenje	Veća fleksibilnost i osjetljivost na nejednako opterećenje, ali potencijalno češće izvršavanje algoritma
DP	Dužina prefiksa	Veća granularnost, preciznije balansiranje opterećenja	Manja potrošnja memorije za čuvanje pravila
-	Broj procesnih elemenata	Manja količina saobraćaja po procesnom elementu, potencijalno kraće iteracije algoritma	Veća količina saobraćaja po procesnom elementu, pa time i potencijalno češća potreba za izvršavanjem algoritma

Tabela 5.4 *Efekti promjena vrijednosti parametara za algoritam za raspoređivanje opterećenja*

5.4.2 Ostala razmatranja

Posebnu pažnju u okviru ove diskusije je potrebno posvetiti analizi procenta uspješnosti detekcije (D)DoS napada i faktorima koji utiču na taj procenat. Da bi se mogla analizirati uspješnost algoritma za detekciju, potrebno je jasno definisati osobine koje određeni saobraćaj čine (D)DoS napadom. Međutim, kako je već rečeno u prethodnoj sekciji, kod volumetrijskih (D)DoS napada efikasnost zavisi od performansi odredišnog sistema i nije moguće u opštem slučaju postaviti prag koji će napraviti tačnu granicu između validnog i nevalidnog saobraćaja. Bilo koji pokušaj slanja neželjenog saobraćaja treba tretirati kao napad, ali napadi koji ne predstavlja problem za prijemnu stranu će biti obrađeni i odbačeni od strane krajnjeg primaoca, čak i u slučaju da ne budu prepoznati od strane procesnih elemenata, pa se ovo može smatrati manje bitnim problemom. Analogno, za odredišni sistem slabih procesnih mogućnosti i malo povećanje obima (validnog) saobraćaja može dovesti do prekida servisa.

Kod setova podataka koji se koriste za testiranje IDS/IPS sistema, najčešće postoje posebne oznake (tzv. labele) kojima su označeni maliciozni paketi, kako bi se kasnije mogla izvršiti validacija dobijenih rezultata. Sa druge strane, volumetrijski (D)DoS napadi mogu biti u potpunosti sastavljeni od validnog saobraćaja, pa je teže izvršiti analizu graničnih slučajeva. Za potrebe testiranja su generisani napadi koji se svojom snagom izdvajaju iznad pozadinskog saobraćaja, čime su simulirani jaki napadi koji bi sigurno predstavljali problem za obradu na prijemnoj strani.

Sličan problem se manifestuje i kod pojave *flash crowd*-a, o kojem je bilo riječi u sekciji 3.1. TIDS ne pravi razliku između pravog (D)DoS napada i *flash crowd*-a, jer i jedna i druga situacija mogu imati negativne posljedice po sistem, pa nema razloga za drugačiji tretman zbog činjenice da se radi o validnim zahtjevima korisnika.

Kad je riječ o tehnologijama anonimizacije adresa (NAT, VPN i sl.), njihov efekat je bitno uzeti u obzir samo kod izolovanja napadača. Uspješan napad bi mogao na isti način biti detektovan i bez adresa pošiljalaca, jer paketi moraju biti upućeni prema validnom odredištu (u protivnom će biti odbačeni ranije na putanji od strane nekog mrežnog uređaja).

Drugi značajan aspekt koji je potrebno analizirati u ovoj sekciji su mehanizmi prevencije napada. U slučaju nedistribuiranog napada, napadači se izoluju praćenjem broja poslatih paketa za svakog korisnika u toku druge faze alarma. Broj paketa koji pošalje napadač je drastično veći od broja paketa koje pošalje običan korisnik, pa izolovanje i kreiranje odgovarajućih pravila ne predstavlja problem, što je vidljivo i na slikama u sekciji 7.1.

6. Zaključak

U ovom radu je predstavljen sistem za detekciju (D)DoS napada u računarskim mrežama. Kod bilo kojeg sistema za detekciju intruzije u računarskim mrežama, potrebno je pronaći rješenje za dva osnovna problema: efikasnu detekciju u realnom vremenu i skaliranje sistema za detekciju na veće količine saobraćaja i veće brzine komunikacije.

Kroz analizu postojećih rješenja i literature pokazano je da razvoj rješenja sposobnog za detekciju napada na svim slojevima OSI modela nije moguć iz praktičnih razloga. Napadi se razlikuju prema načinu izvođenja i manifestovanja u dovoljnoj mjeri da bi upravljanje takvim sistemom bilo veoma kompleksno i da bi unosilo kašnjenje u komunikaciji koje se ne može tolerisati na konvergiranim mrežama. Razvijeni sistem rješava problem detekcije napada koji nastaju kao posljedica slanja ogromne količine (često potpuno validnog) saobraćaja prema malom broju primalaca kako bi se izazvao prekid funkcionisanja kompletnog sistema ili jednog njegovog dijela.

Efikasnost sistema se postiže korištenjem pristupa baziranog na entropiji za detekciju napada i SDN komponenti mreže za upravljanje pravilima za prosljeđivanje i blokiranje napada u situacijama gdje je to moguće. Rezultati izvršenih testova pokazuju da entropija predstavlja metriku koja je otporna na promjene obima saobraćaja koje dolaze kao posljedica redovnog dnevnog funkcionisanja svake računarske mreže. Prikazani sistem je razvijen tako da bude adaptivan kako bi se izbjegla zavisnost od predznanja administratora o potencijalnim napadima i izbjegla mogućnost da se mehanizmi odbrane zaobiđu pažljivom modifikacijom načina na koji je napad izveden. Obzirom da je većina postojećih rješenja zasnovana upravo na potpisu (odnosno, specifikaciji načina manifestovanja napada), ovakav pristup predstavlja očigledan doprinos u polju detekcije napada [137].

Dodatno, sistem funkcioniše transparentno za ostatak mreže, što značajno otežava mogućnost izvođenja bilo kakvog napada usmjerenog prema samom sistemu za detekciju.

Skaliranje sistema je realizovano kroz namjenski razvijen algoritam za raspoređivanje opterećenja. Razvijeni algoritam funkcioniše u kombinaciji sa algoritmom za detekciju napada tako da ne utiče negativno na rad sistema, a pritom omogućava proširenje kapaciteta na proizvoljan broj procesnih elemenata u toku rada.

Razvijeni sistem je evaluiran kroz čitav niz testova koji simuliraju rad računarske mreže velike brzine u realnom okruženju. Testirano je ponašanje sistema za različite vrijednosti parametara koji bitno utiču na njegovo funkcionisanje, te su predložene podrazumijevane vrijednosti koje bi trebalo da pruže zadovoljavajuće

performanse u većini mrežnih okruženja. Dodatno, iskustvo mrežnog administratora može doprinijeti poboljšanju performansi kroz prilagođavanje određenih vrijednosti za specifične uslove funkcionisanja mreže.

Razvijeni sistem može imati naročitu primjenu u okruženjima sa jako velikim brojem hostova u okviru zaštićene mreže, odnosno okruženjima kod kojih su brzina komunikacije i broj prenesenih paketa u jedinici vremena u tački agregacije na ulazu u mrežu jako veliki. Tipičan primjer ovakvih okruženja su *cloud* infrastrukture, koje se u posljednje vrijeme suočavaju sa rastom popularnosti zbog jednostavnosti i relativno niske cijene implementacije servisa sa stanovišta krajnjeg korisnika.

6.1 Poređenje sa postojećim rješenjima

Kako je već objašnjeno u sekciji 3.1, u literaturi preovladavaju rješenja bazirana na specifikaciji potpisa napada, prvenstveno zbog jednostavnosti implementacije. TIDS funkcioniše na bazi detekcije anomalija u komunikaciji na mreži, pri čemu se kao osnovna metrika za detekciju koristi entropija broja primljenih paketa po adresama primalaca u zaštićenoj mreži. U poređenju sa rješenjima koja svoj rad u potpunosti baziraju na specifikovanju potpisa napada [28], TIDS omogućava administratoru da djelimično prilagodi njegovo funkcionisanje konfigurisanjem graničnih vrijednosti, ali je sistem i dalje u stanju da reaguje adaptivno na promjene u mreži. Zajednička karakteristika TIDS-a i velikog broja prikazanih rješenja [78], [93], [94], [95], [96] je činjenica da najveću upotrebnu vrijednost imaju u *cloud* okruženjima sa velikim brojem hostova i velikom brzinom komunikacije.

Izbor metrike je presudan za efikasnost rješenja za detekciju, jer osim mogućnosti detekcije utiče i na performanse sistema. Na primjer, rješenje predloženo u [43] koristi brzinu (frekvenciju) pristizanja paketa, pri čemu se pretpostavlja da će u slučaju napada ta frekvencija biti znatno veća nego u slučaju normalnog toka saobraćaja. Ovakav način detekcije je teško zaobići prilagođavanjem načina izvođenja napada bez gubitka njegove efektivnosti. Rješenje koristi hardversku implementaciju, što mu povećava brzinu rada, ali zbog cijene umanjuje mogućnosti skaliranja na velike mreže. Mjerenje frekvencije pristizanja uključuje mjerenje proteklog vremena između svaka dva susjedna paketa, što i u slučaju hardverske implementacije može predstavljati veliko opterećenje za sistem. Dizajn TIDS-a omogućava implementaciju korištenjem potrošačkog hardvera, pa se izbjegava vezivanje za namjenske hardverske komponente koje bi mu povećale cijenu implementacije.

Problem performansi i skaliranja sistema je potrebno razmotriti i u kontekstu SDN mreža. U prvom redu, u literaturi se ističu potencijalni problemi sa performansama koji dolaze kao posljedica samog dizajna SDN tehnologija [53]. Paket koji ne podliježe postojećim pravilima (najčešće prvi paket svakog toka) se uvijek šalje kontroleru, što može dovesti do latencije u obradi. TIDS ovaj problem rješava proaktivnim instaliranjem pravila niskog prioriteta u tabelu prosljeđivanja koja pokrivaju slučajeve u kojima nema specifičnijih pravila višeg prioriteta koja nastaju radom algoritama za detekciju napada i raspoređivanje. U takvoj konfiguraciji, kontroleru se šalju samo paketi koji su dio protokola za komunikaciju sa

procesnim elementima, a koji su izuzeti iz navedenih pravila zbog činjenice da koriste drugačiju vrijednost ToS polja u zaglavlju drugog sloja. TIDS sve pakete tretira na isti način (bez uvođenja prioriteta za pojedine tipove saobraćaja), pa se efikasno korištenje propusnog opsega i skalabilnost mogu posmatrati kao osnovne metrike za kvantifikovanje rada sistema.

Proaktivni pristup konfiguraciji ulaznog SDN sviča omogućava brži prelazak sistema u funkcionalno stanje, što je sa stanovišta sigurnosti bitnije od jednakog opterećenja elemenata u početnim fazama rada. Kontroler koji koristi TIDS nema statičkih konfiguracionih parametara, već se kompletna konfiguracija šalje od strane procesnih elemenata u okviru svake *keepalive* poruke. Provedeni testovi pokazuju da je za trenutnu implementaciju dovoljan samo jedan kontroler, pa nema potrebe za implementacijom naprednih tehnika za upravljanje politikama i konfiguracije koji su u velikoj mjeri zastupljeni u literaturi [62], [63], [64], [65].

Ako bi bilo potrebno izvršiti distribuciju kontrolera, morala bi se osigurati razmjena informacija o vrijednostima entropije na pojedinim procesnim elementima kako bi se omogućila detekcija napada. Jedan od pristupa koji bi se u ovom slučaju mogao iskoristiti je hijerarhijska organizacija kontrolera i njihovo grupisanje u klastere, kako bi se minimizovala količina informacija koja se šalje između različitih elemenata [138], [139]. U tom slučaju ne bi bilo potrebno implementirati punu povezanost između svih kontrolera (engl. *full mesh*), već samo unutar svakog klastera (engl. *intracluster*) i između samih klastera (engl. *intercluster*).

Konačno, za potrebe ovog rada su posebno značajna istraživanja koja se bave upotrebom SDN tehnologija u domenu sigurnosti i detekcije napada na računarske sisteme, jer zbog programabilnosti i jednostavnosti implementacije predstavlja pogodnu platformu za razvoj rješenja za detekciju napada baziranih na specifikaciji [71]. Kako je naglašeno u sekciji 3.2, rješenje predstavljeno u radu [72] koristi sličan mehanizam detekcije baziran na entropiji, ali ne uključuje mehanizme za raspoređivanje opterećenja i koristi kontroler kako bi izvršio izračunavanja. Izdvajanje kompleksnih računskih operacija na namjenske hostove (pritom zasnovane na nespecifičnom hardveru) nudi bolje performanse i veću stabilnost kompletnog sistema.

6.2 Kritički osvrt na rad i pravci daljeg istraživanja

Zaštita računarskih sistema je kontinualan zadatak i bilo kakav sistem ovakve vrste je potrebno konstantno unapređivati, bez obzira u kojoj mjeri je on uspješan u bilo kojem datom trenutku. Iz tog razloga se može očekivati dodatni porast popularnosti sistema koji su bazirani na programabilnim komponentama, gdje se u prvom redu izdvajaju SDN-bazirane mreže. Ovakva programabilnost omogućava laku proširivost i promjene karakteristika sistema bez potrebe za značajnim ulaganjima u novi hardver. Tehnologije koje omogućavaju logičku decentralizaciju i virtuelizaciju kao rješenje za skalabilnost daju potrebnu fleksibilnost sistemu koja mu je potrebna kako bi odgovorio na sve veće sigurnosne prijetnje kojima su računarske mreže i servisi svakodnevno izloženi. Imajući navedeno u vidu, identifikovano je

nekoliko oblasti u okviru kojih se može nastaviti istraživanje predstavljeno u ovom radu.

U prvom redu, transparentno funkcionisanje sistema i mogućnost obrade velike količine podataka bez unošenja kašnjenja u komunikaciju se može iskoristiti u čitavom nizu primjena koje ne moraju biti vezane za sigurnosne aspekte rada sistema. Ovakva implementacija se može upotrebiti za statističku analizu saobraćaja u cilju povećanja kvaliteta servisa ili utvrđivanja karakteristika saobraćaja na bilo kojem nivou OSI modela kako bi se modifikovalo ponašanje mreže u cilju ostvarivanja boljih performansi.

U domenu sigurnosti, ovakav sistem je primjenjiv na bilo koju mrežu velikog kapaciteta. Omogućavanje detekcije drugih tipova napada zahtijeva samo proširenje dijela sistema koji sadrži implementaciju algoritma za detekciju, ali se mehanizmi raspoređivanja opterećenja mogu iskoristiti u istom obliku, bez obzira na podatak iz zaglavlja PDU-a koji se koristi u detekciji.

Najveći problem kod implementacije bilo kojeg sistema za detekciju i prevenciju intruzije su parametri okruženja sistema koji se ne mogu sa potpunom preciznošću predstaviti matematičkim modelom. Za određivanje bezbjedne vrijednosti entropije, kao i vrijednosti bilo kojeg parametra na osnovu kojeg bi bila donesena odluka da napad nije u toku ili da je uspješno spriječen gotovo uvijek potrebno konsultovati administratora mreže koji poznaje funkcionisanje mreže u dovoljnoj mjeri da bi mogao ispravno konfigurisati sistem. Parametri koji za jednu mrežu imaju ispravne vrijednosti nisu primjenjivi na neku drugu mrežu, pa se svakoj implementaciji bilo kakvog sistema ove vrste mora pristupiti zasebno.

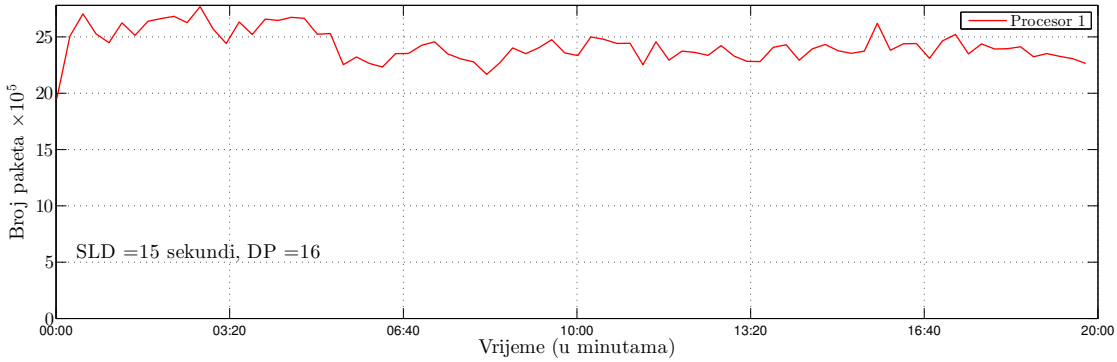
Oblast u kojoj je moguće unaprijediti postojeći sistem je mehanizam sprečavanja distribuiranih DoS napada. Jedan od pravaca kojim bi se moglo odvijati takvo unapređenje je iskorištenje podrške za BGP protokol koje je ugrađeno u novije verzije Ryu SDN kontrolera koji je korišten za implementaciju u ovom radu. Takav mehanizam bi uključivao promjenu načina analiziranja saobraćaja kako bi se iskoristili podaci koji su dostupni putem BGP protokola kako bi se uticalo na slanje paketa iz autonomnih sistema za koje sistem procijeni da sadrže izvore DoS napada. Dodatno, novija proširenja BGP protokola [140], [141] dozvoljavaju modifikaciju tabela rutiranja specifikovanjem tokova (engl. Flow Specification), što bi omogućilo filtriranje saobraćaja od strane napadača u tački koja je tom napadaču najbliža (kod njegovog provajdera), čime bi se značajno smanjilo opterećenje mrežnih elemenata TIDS-a, jer bi se sve informacije o blokiranju paketa direktno prosljeđivale odgovarajućim ruterima iza kojih se izvor napada nalazi. S obzirom da je mehanizam specifikovanja tokova sličan mehanizmu koji već koristi OpenFlow protokol - potrebno je navesti kombinaciju vrijednosti polja iz zaglavlja različitih PDU-ova, potrebna modifikacija samog sistema bi bila relativno trivijalna.

Kod komplesnih računarskih mreža sa većim brojem ulaznih tačaka bi za uspješnu detekciju bilo potrebno implementirati više instanci TIDS-a. U takvom okruženju od koristi može biti implementacija mehanizma komunikacije između pojedinih procesnih elemenata (posebno iz različitih instanci) ili komunikacija između različitih kontrolera. U trenutnoj implementaciji nema komunikacije između procesora, jer nije potrebna razmjena informacija (svaki procesor ima sve

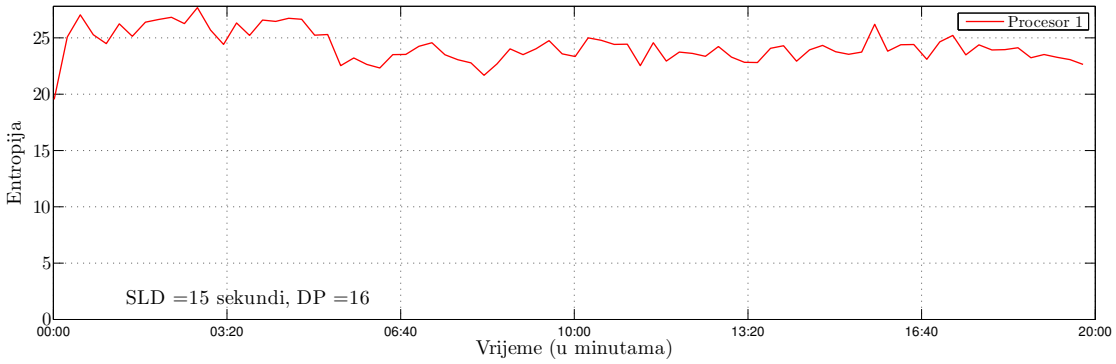
informacije koje su mu potrebne), a ujedno se izbjegavaju problemi sa sinhronizacijom stanja. Kod većeg broja ulaza isti napad bi mogao biti raspoređen tako da pojedinačne instance vide manji intenzitet saobraćaja, a da se on agregira tek na krajnjem prijemu. Komunikacijom između elemenata različitih instanci bi se omogućilo stvaranje kompletne slike stanja entropije, ali bi to za sobom povlačilo postojanje centralnog procesnog elementa (koji bi sadržavao sve informacije), što bi dalje zahtijevalo implementaciju redundanse i mehanizma za podjelu opterećenja između tih centralnih elemenata TIDS-a.

7. Prilozi

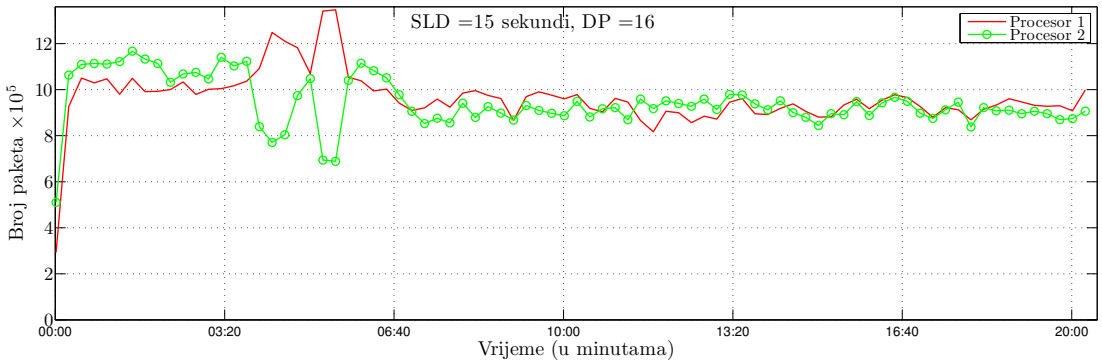
7.1 Rezultati testiranja algoritma za detekciju (D)DoS napada



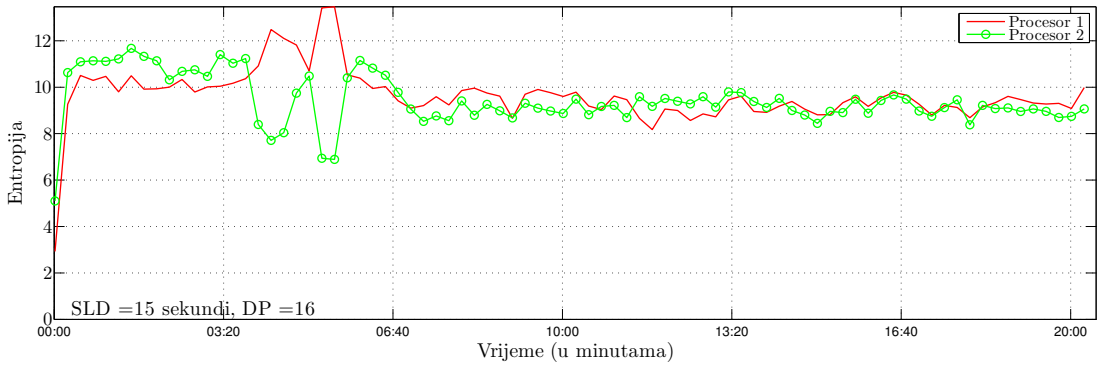
Slika 7.1 *Opterećenje sistema (1 procesor, SLD=15 sekundi, DP=16)*



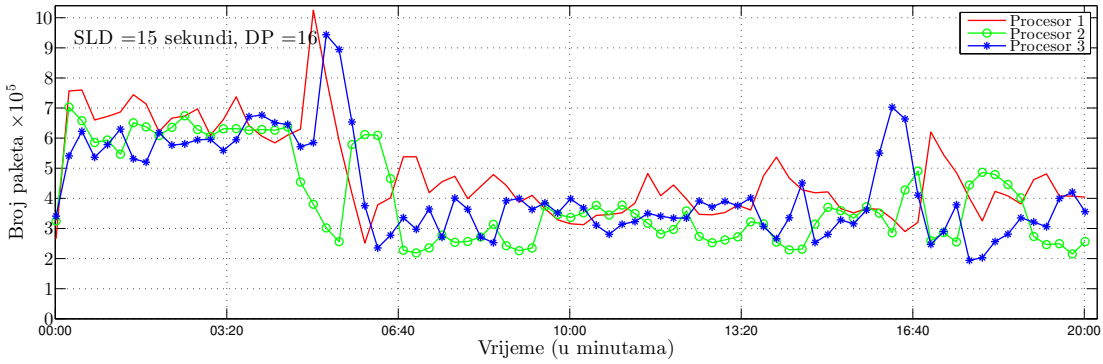
Slika 7.2 *Entropija po intervalima (1 procesor, SLD=15 sekundi, DP=16)*



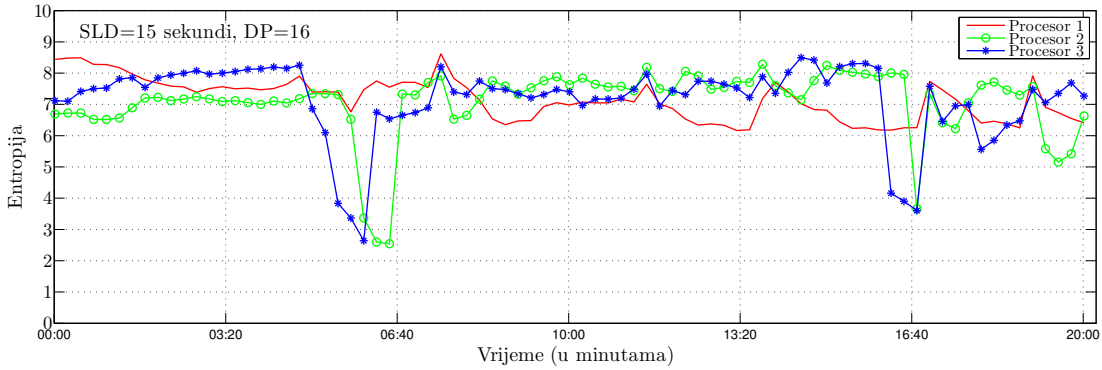
Slika 7.3 *Opterećenje sistema (2 procesora, SLD=15 sekundi, DP=16)*



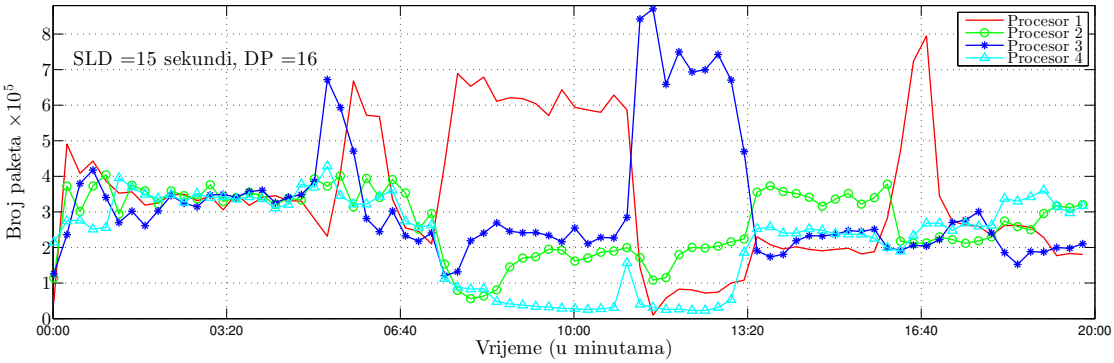
Slika 7.4 *Entropija po intervalima (2 procesora, SLD=15 sekundi, DP=16)*



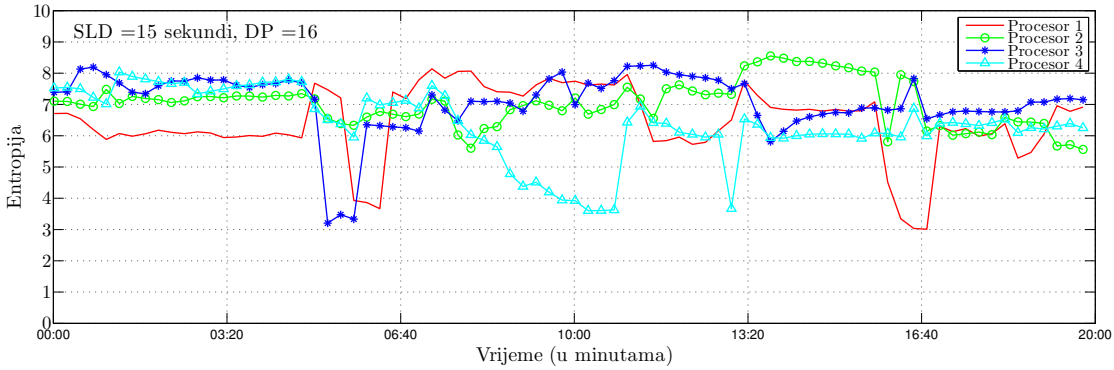
Slika 7.5 *Opterećenje sistema (3 procesora, SLD=15 sekundi, DP=16)*



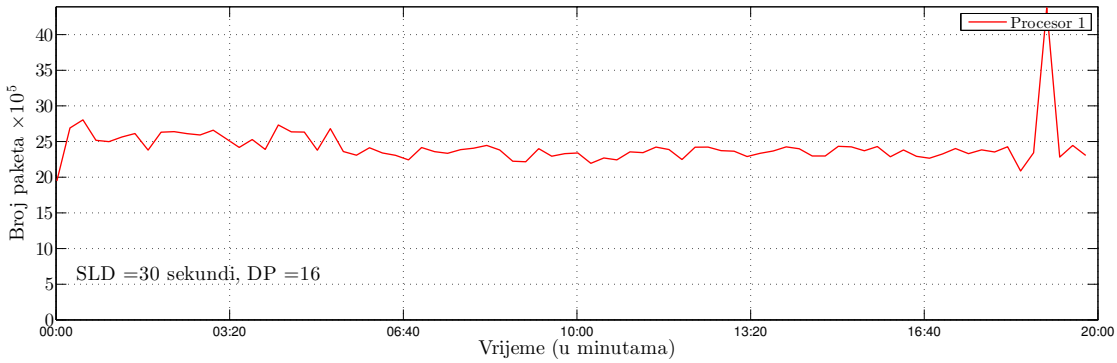
Slika 7.6 *Entropija po intervalima (3 procesora, SLD=15 sekundi, DP=16)*



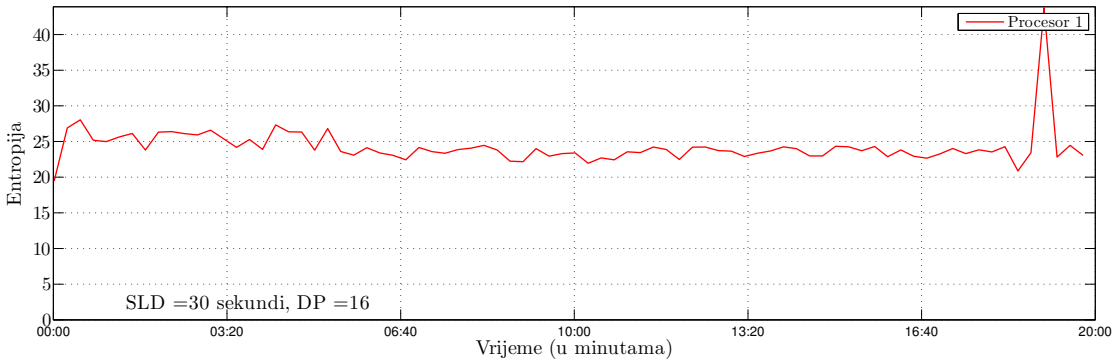
Slika 7.7 *Opterećenje sistema (4 procesora, SLD=15 sekundi, DP=16)*



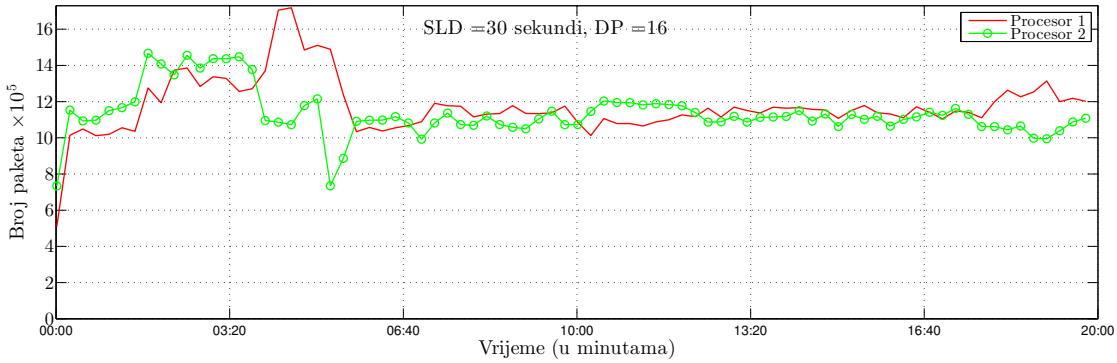
Slika 7.8 *Entropija po intervalima (4 procesora, SLD=15 sekundi, DP=16)*



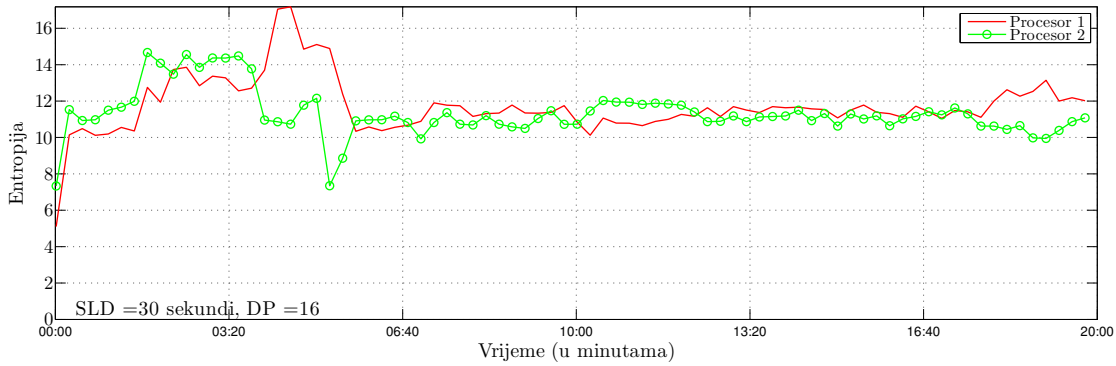
Slika 7.9 *Opterećenje sistema (1 procesor, SLD=30 sekundi, DP=16)*



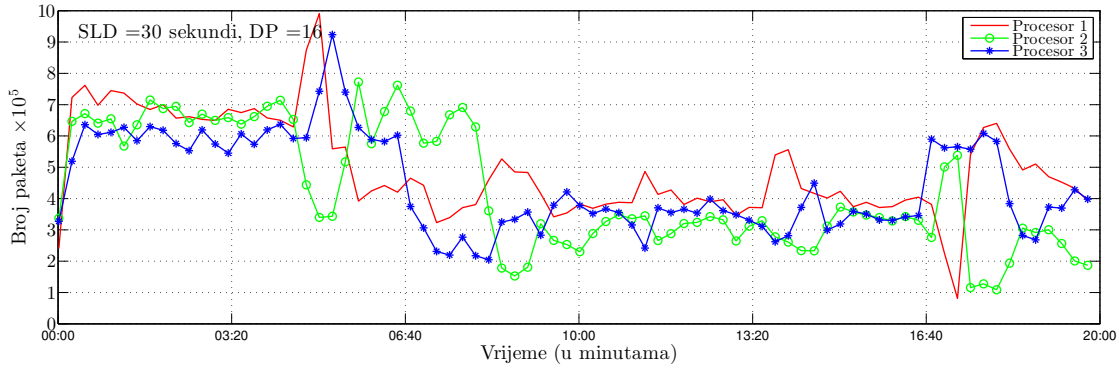
Slika 7.10 *Entropija po intervalima (1 procesor, SLD=30 sekundi, DP=16)*



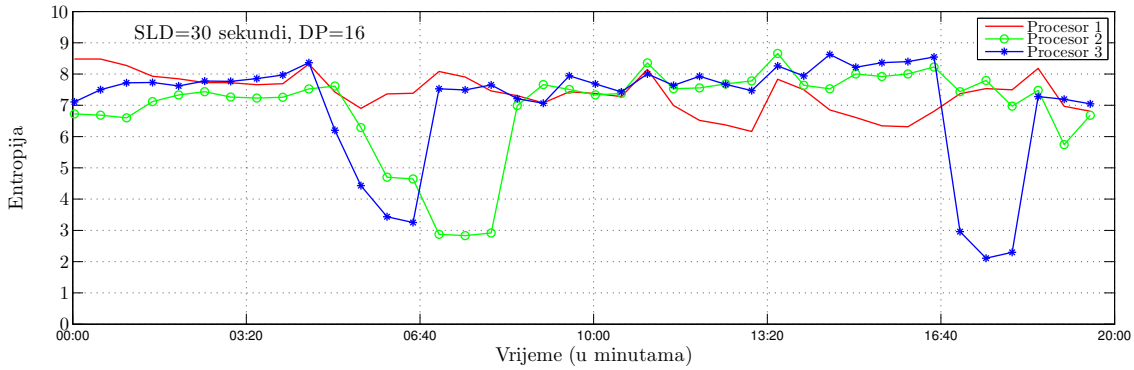
Slika 7.11 *Opterećenje sistema (2 procesora, SLD=30 sekundi, DP=16)*



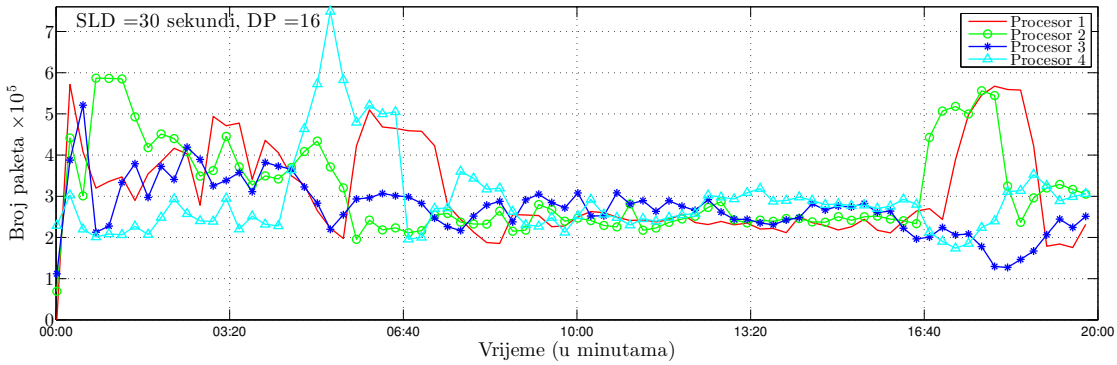
Slika 7.12 *Entropija po intervalima (2 procesora, SLD=30 sekundi, DP=16)*



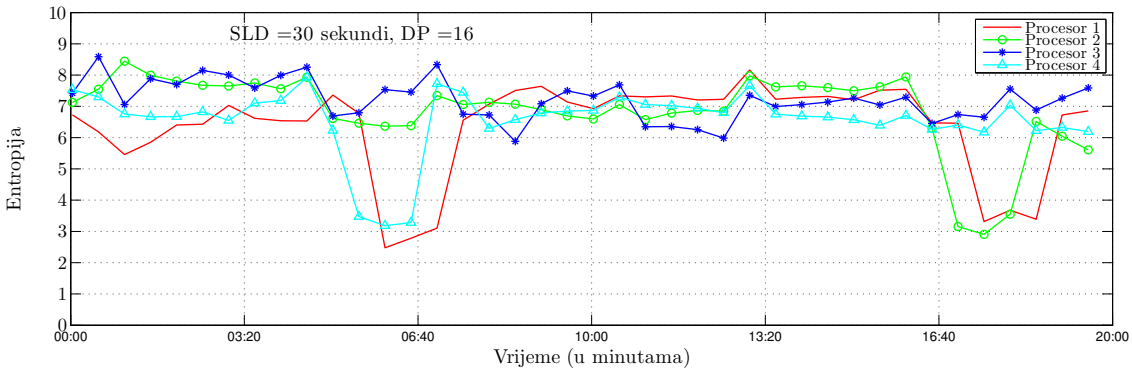
Slika 7.13 *Opterećenje sistema (3 procesora, SLD=30 sekundi, DP=16)*



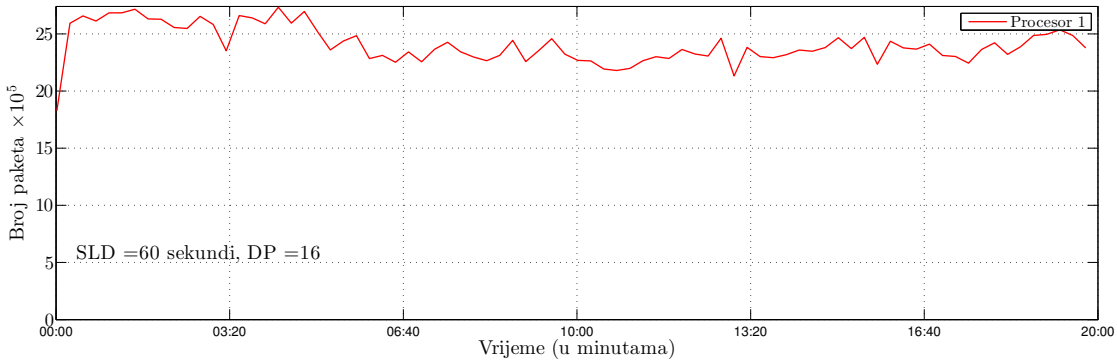
Slika 7.14 *Entropija po intervalima (3 procesora, SLD=30 sekundi, DP=16)*



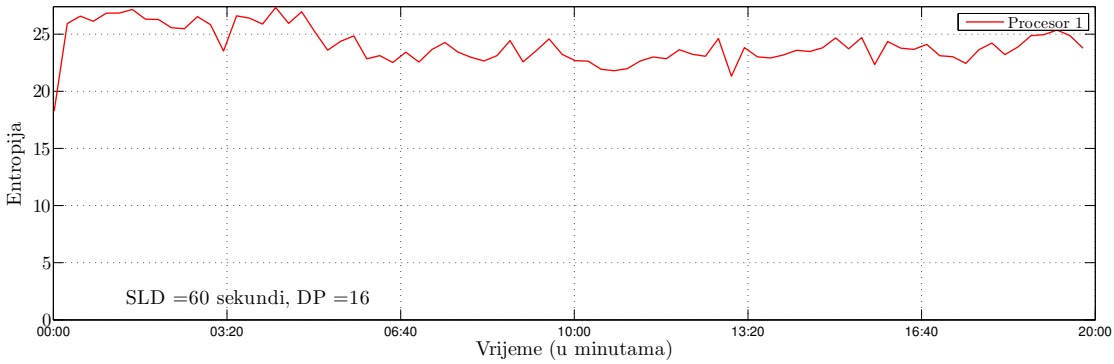
Slika 7.15 *Opterećenje sistema (4 procesora, SLD=30 sekundi, DP=16)*



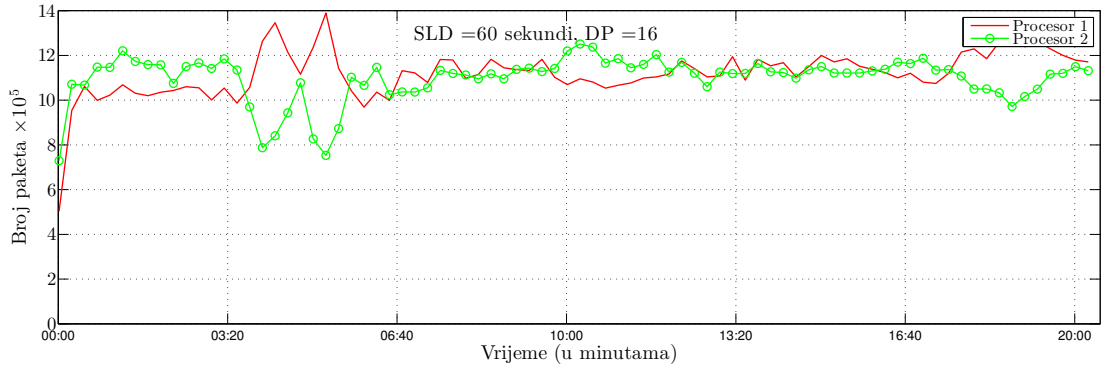
Slika 7.16 *Entropija po intervalima (4 procesora, SLD=30 sekundi, DP=16)*



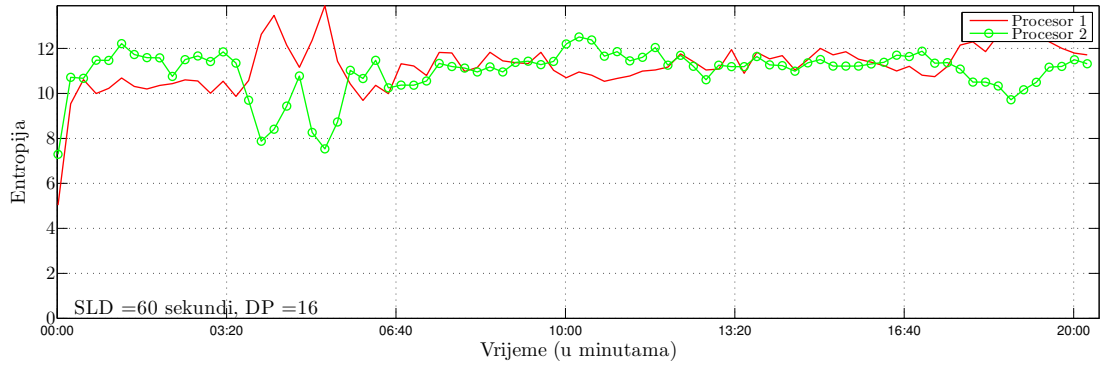
Slika 7.17 *Opterećenje sistema (1 procesor, SLD=60 sekundi, DP=16)*



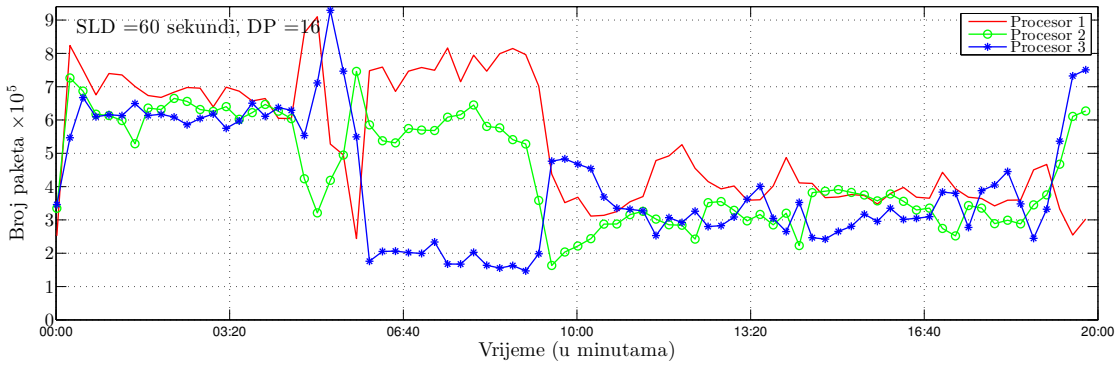
Slika 7.18 *Entropija po intervalima (1 procesor, SLD=60 sekundi, DP=16)*



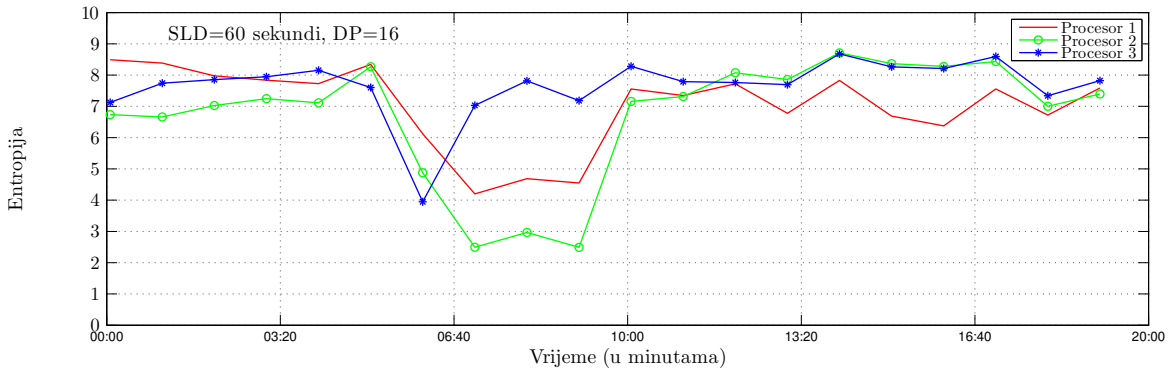
Slika 7.19 *Opterećenje sistema (2 procesora, SLD=60 sekundi, DP=16)*



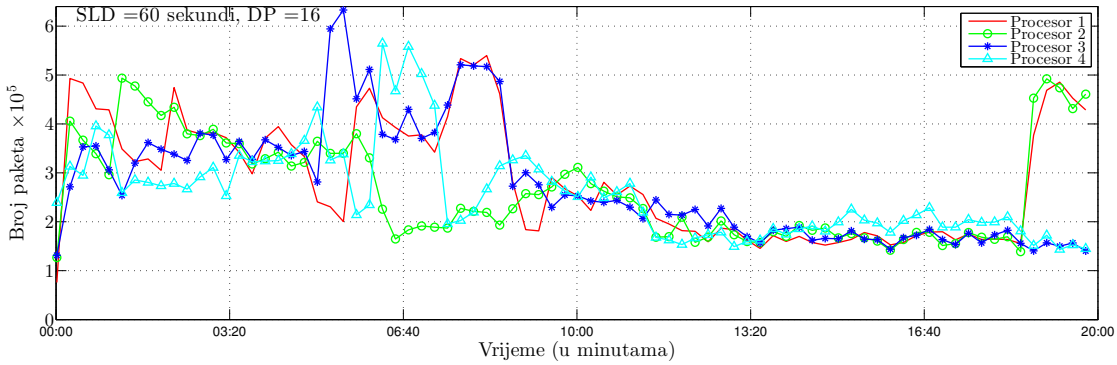
Slika 7.20 *Entropija po intervalima (2 procesora, SLD=60 sekundi, DP=16)*



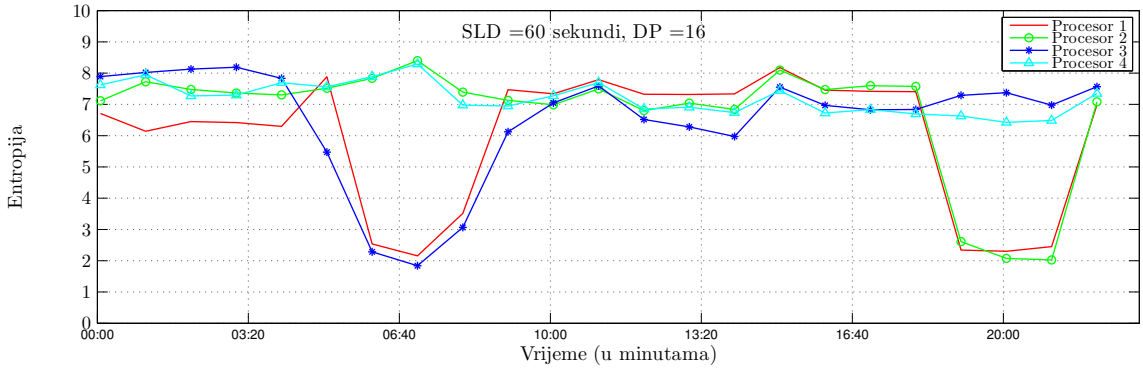
Slika 7.21 *Opterećenje sistema (3 procesora, SLD=60 sekundi, DP=16)*



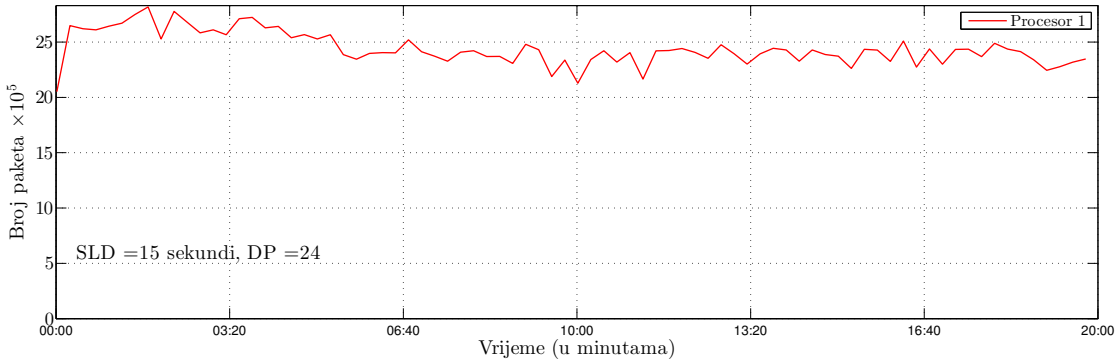
Slika 7.22 *Entropija po intervalima (3 procesora, SLD=60 sekundi, DP=16)*



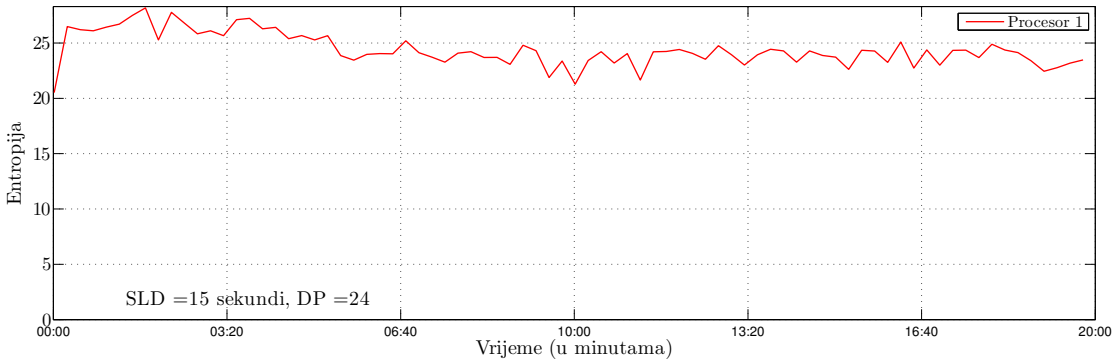
Slika 7.23 *Opterećenje sistema (4 procesora, SLD=60 sekundi, DP=16)*



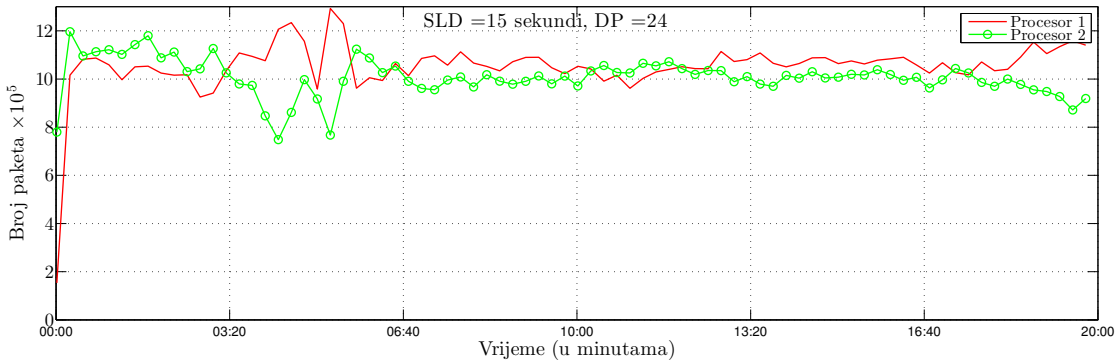
Slika 7.24 *Entropija po intervalima (4 procesora, SLD=60 sekundi, DP=16)*



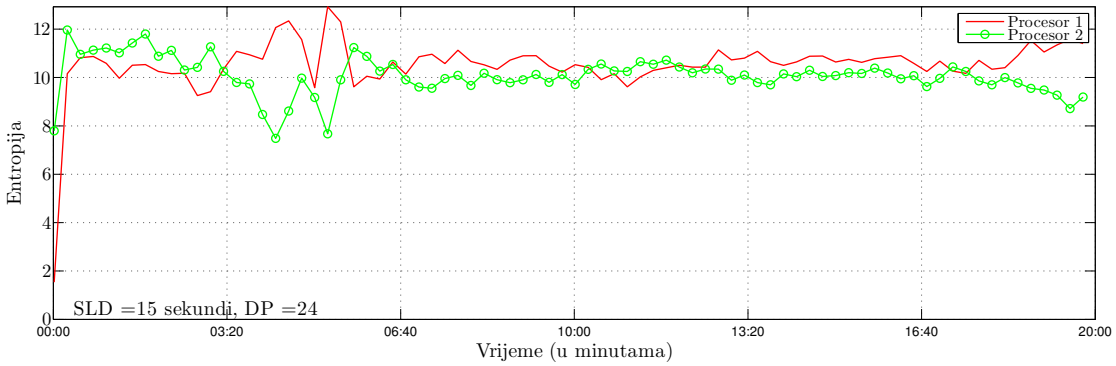
Slika 7.25 *Opterećenje sistema (1 procesor, SLD=15 sekundi, DP=24)*



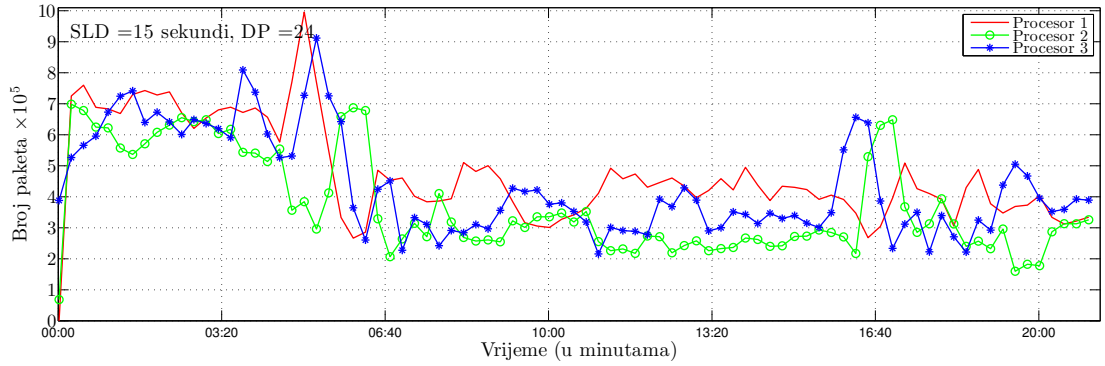
Slika 7.26 *Entropija po intervalima (1 procesor, SLD=15 sekundi, DP=24)*



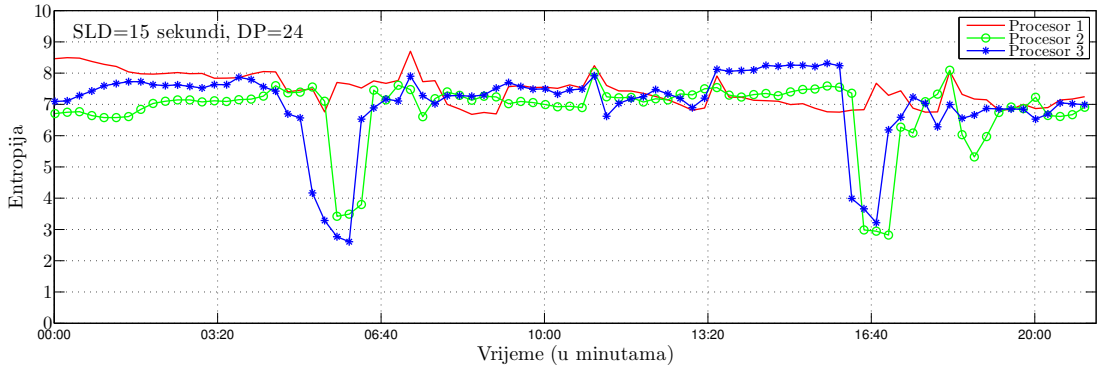
Slika 7.27 *Opterećenje sistema (2 procesora, SLD=15 sekundi, DP=24)*



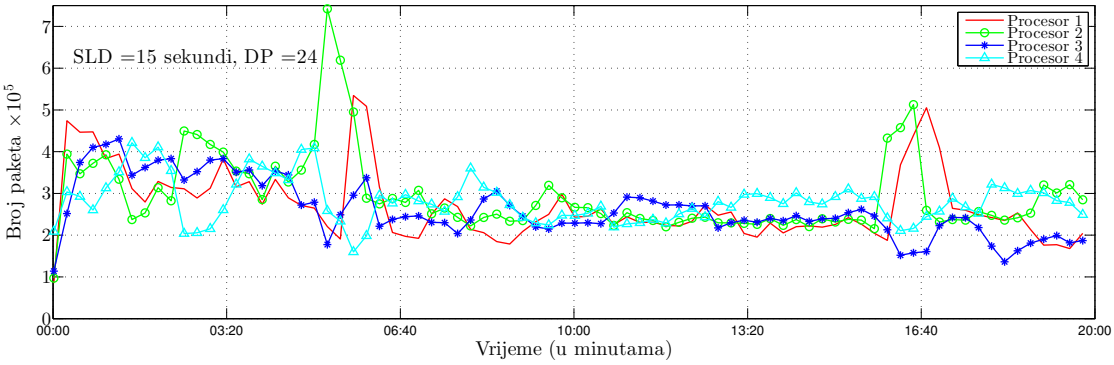
Slika 7.28 *Entropija po intervalima (2 procesora, SLD=15 sekundi, DP=24)*



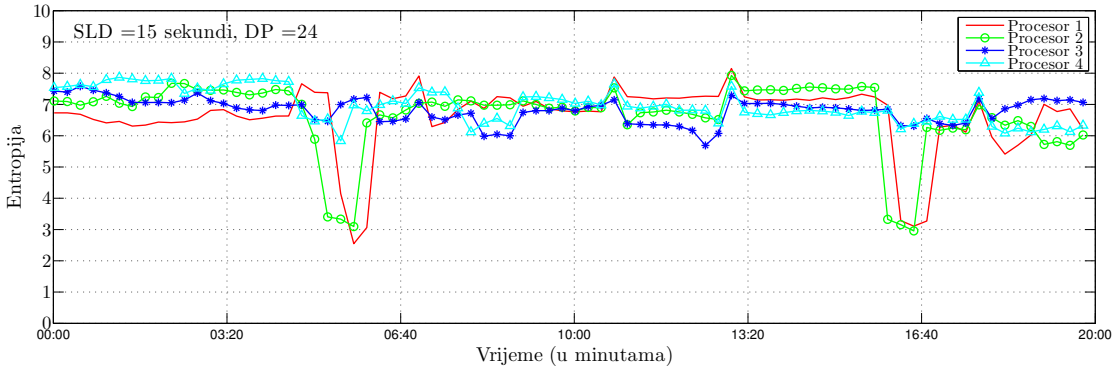
Slika 7.29 *Opterećenje sistema (3 procesora, SLD=15 sekundi, DP=24)*



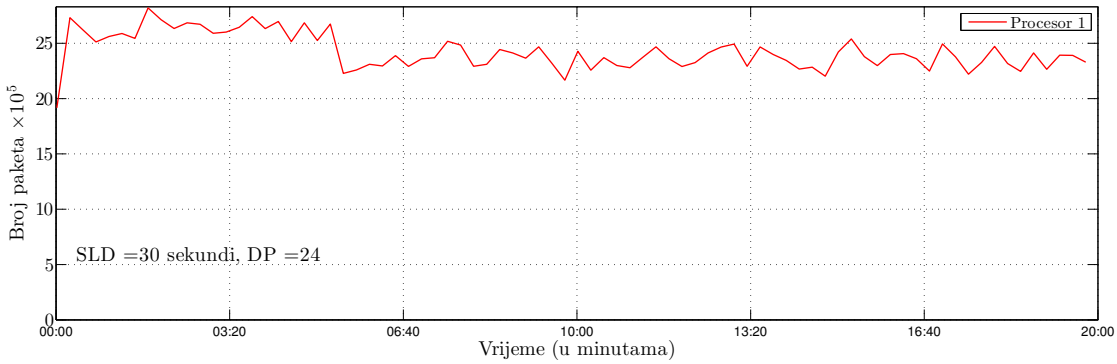
Slika 7.30 *Entropija po intervalima (3 procesora, SLD=15 sekundi, DP=24)*



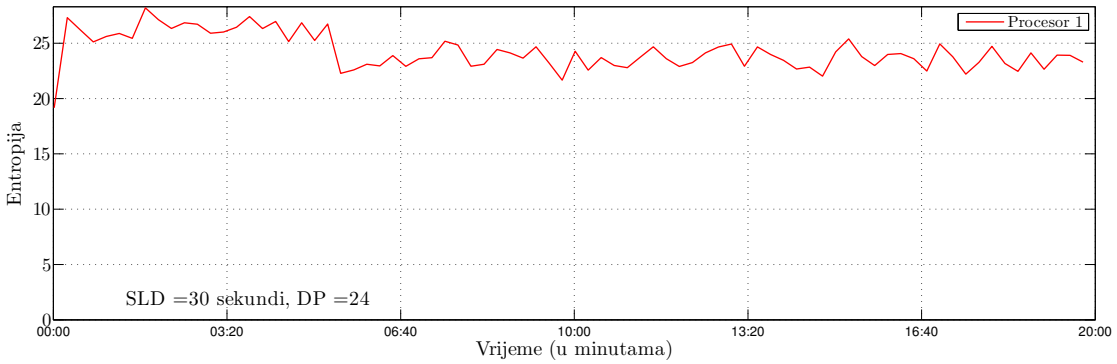
Slika 7.31 *Opterećenje sistema (4 procesora, SLD=15 sekundi, DP=24)*



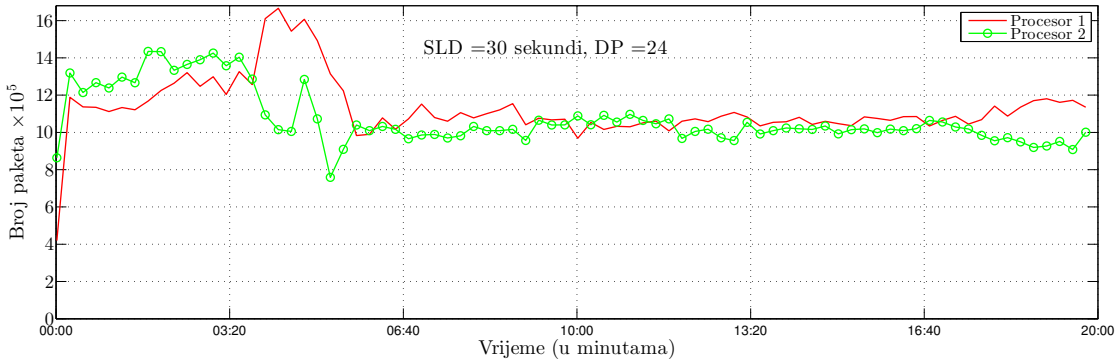
Slika 7.32 *Entropija po intervalima (4 procesora, SLD=15 sekundi, DP=24)*



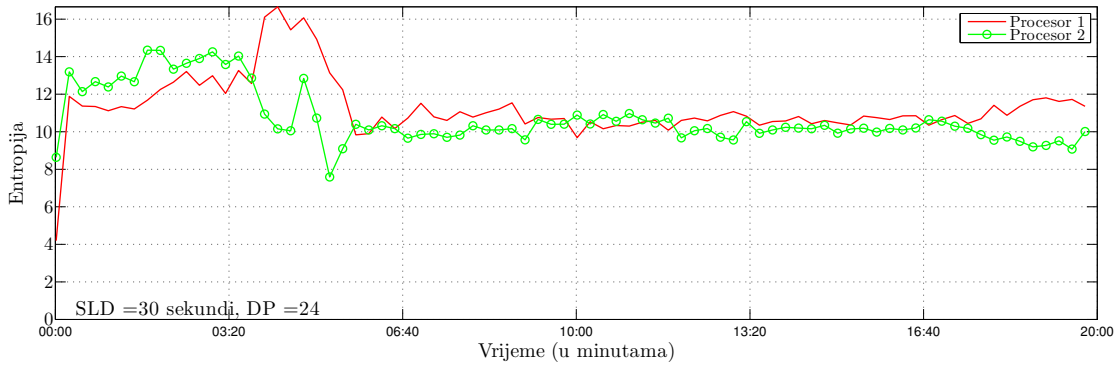
Slika 7.33 *Opterećenje sistema (1 procesor, SLD=30 sekundi, DP=24)*



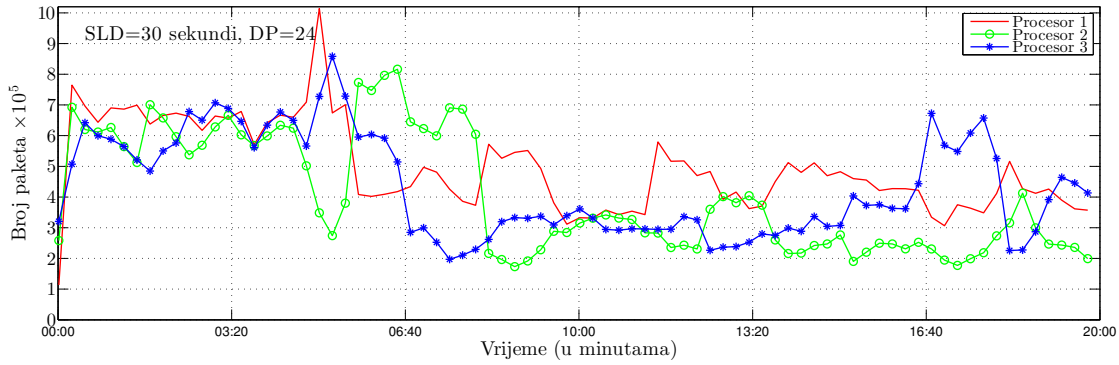
Slika 7.34 *Entropija po intervalima (1 procesor, SLD=30 sekundi, DP=24)*



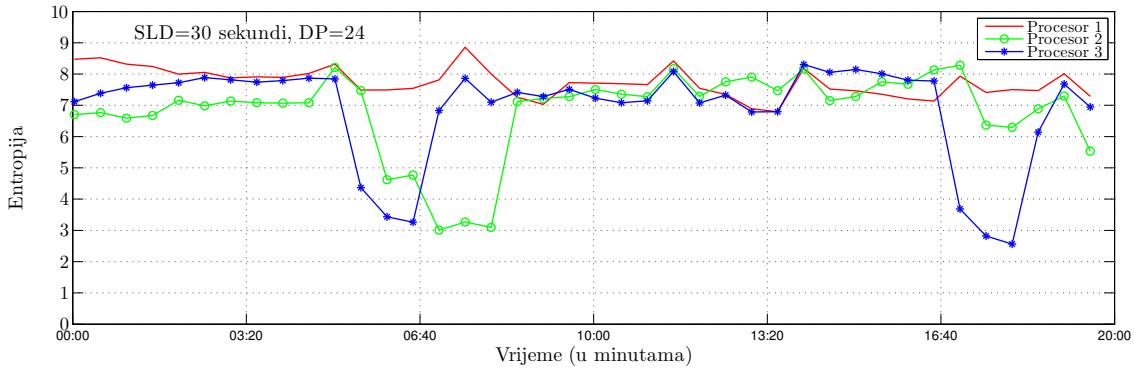
Slika 7.35 *Opterećenje sistema (2 procesora, SLD=30 sekundi, DP=24)*



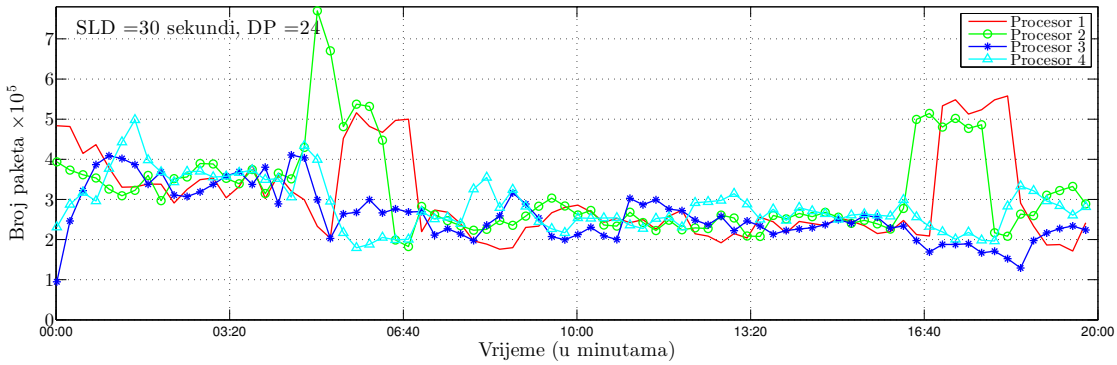
Slika 7.36 *Entropija po intervalima (2 procesora, SLD=30 sekundi, DP=24)*



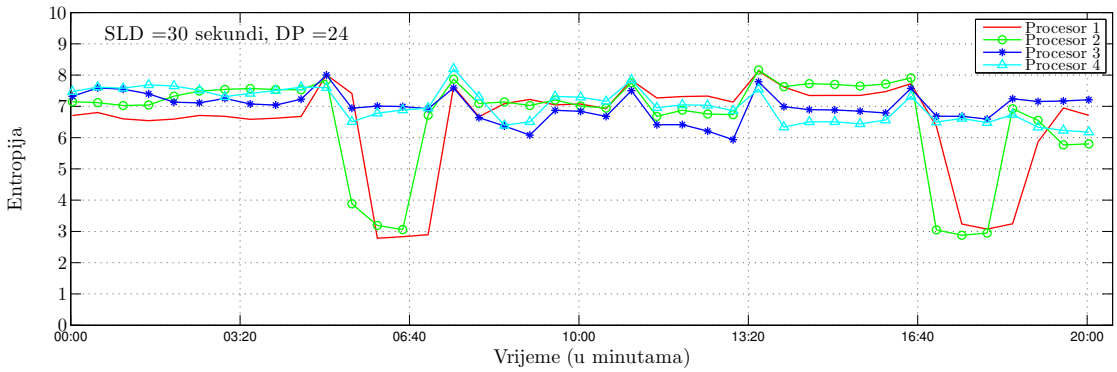
Slika 7.37 *Opterećenje sistema (3 procesora, SLD=30 sekundi, DP=24)*



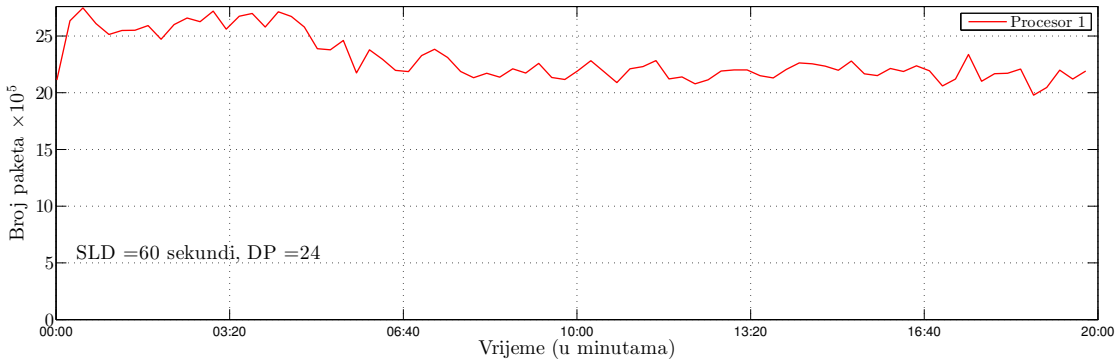
Slika 7.38 *Entropija po intervalima (3 procesora, SLD=30 sekundi, DP=24)*



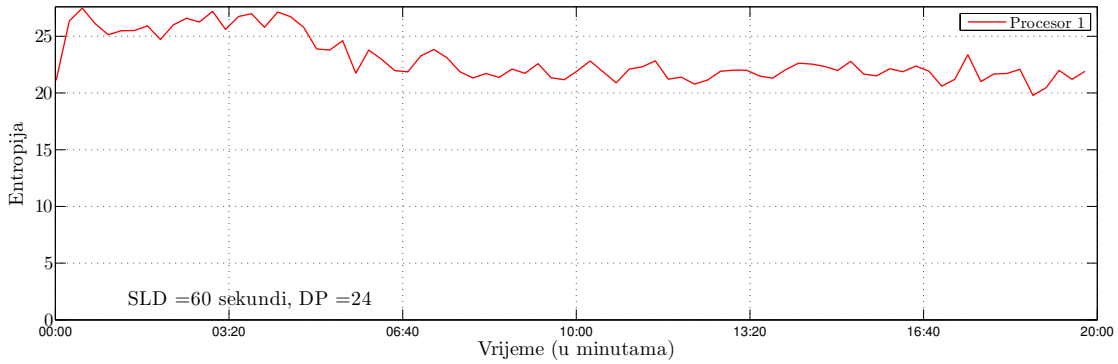
Slika 7.39 *Opterećenje sistema (4 procesora, SLD=30 sekundi, DP=24)*



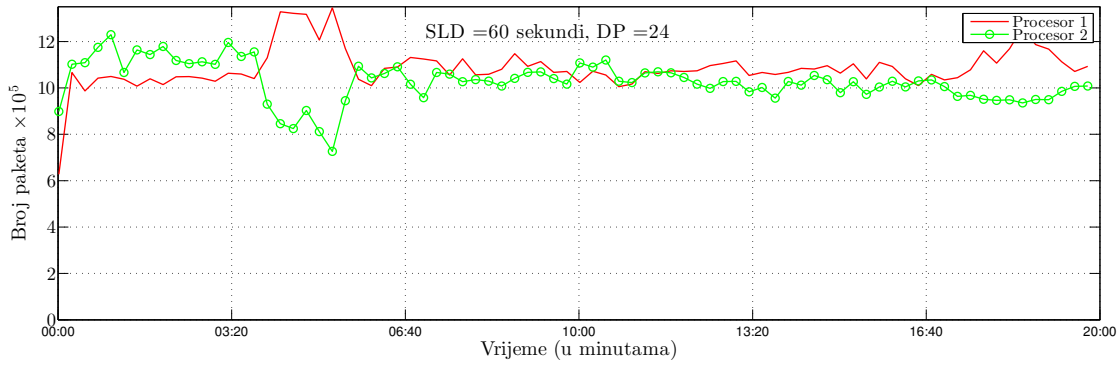
Slika 7.40 *Entropija po intervalima (4 procesora, SLD=30 sekundi, DP=24)*



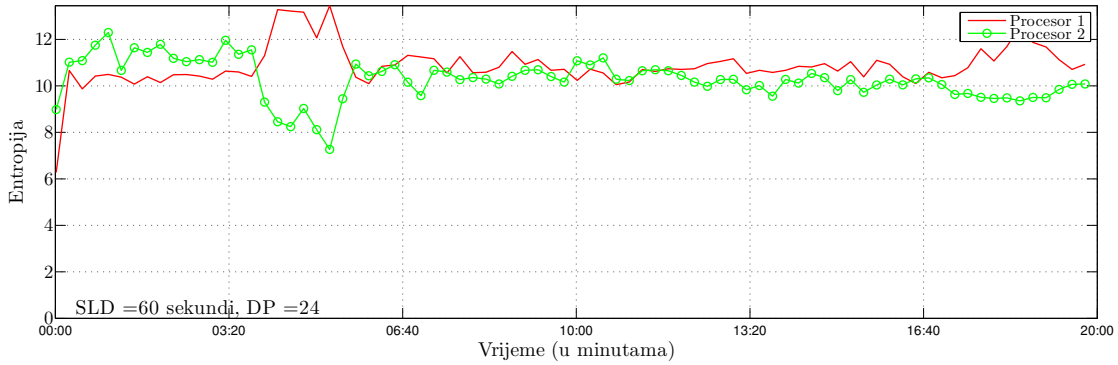
Slika 7.41 *Opterećenje sistema (1 procesor, SLD=60 sekundi, DP=24)*



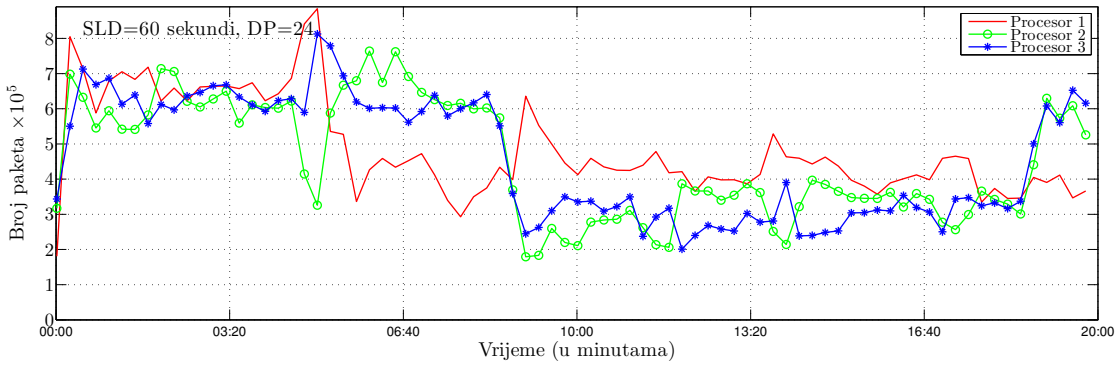
Slika 7.42 *Entropija po intervalima (1 procesor, SLD=60 sekundi, DP=24)*



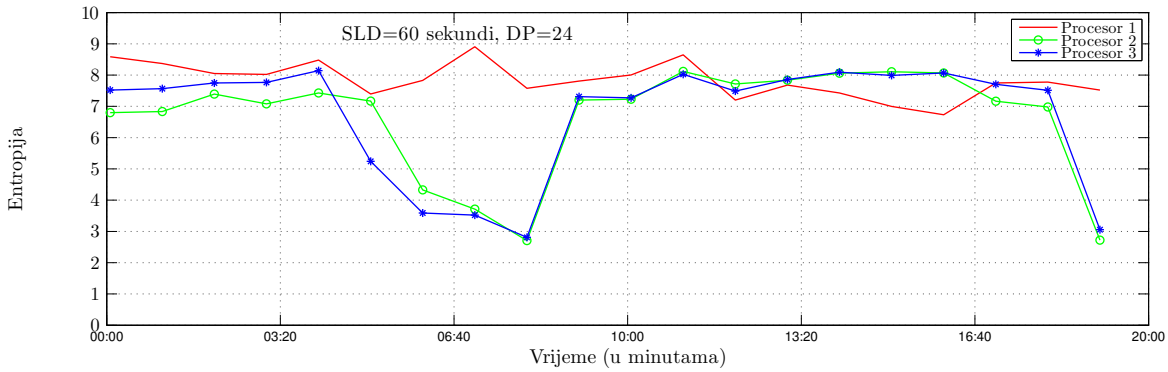
Slika 7.43 *Opterećenje sistema (2 procesora, SLD=60 sekundi, DP=24)*



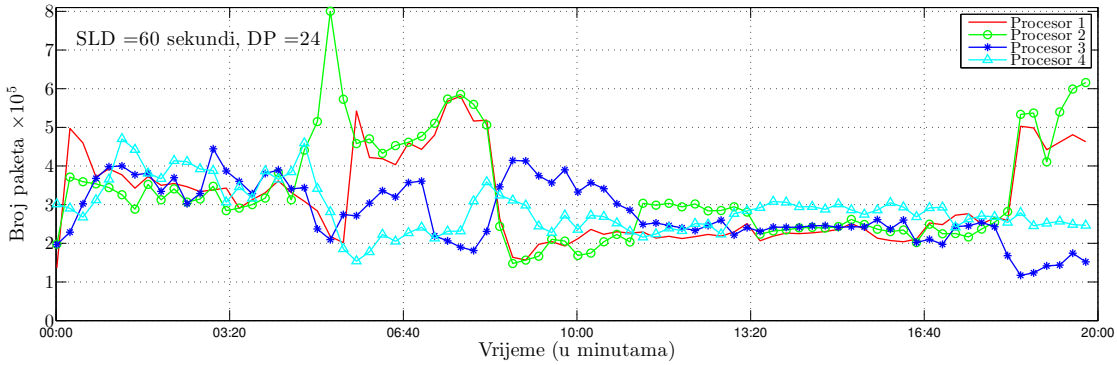
Slika 7.44 *Entropija po intervalima (2 procesora, SLD=60 sekundi, DP=24)*



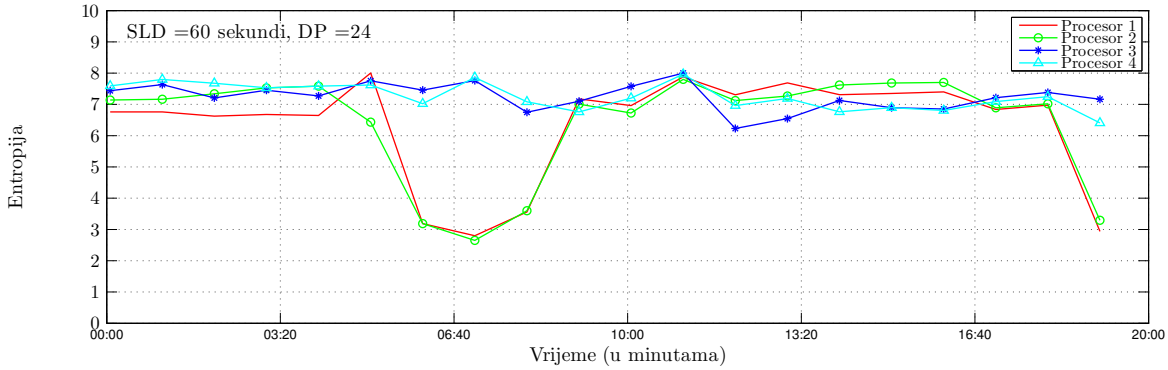
Slika 7.45 *Opterećenje sistema (3 procesora, SLD=60 sekundi, DP=24)*



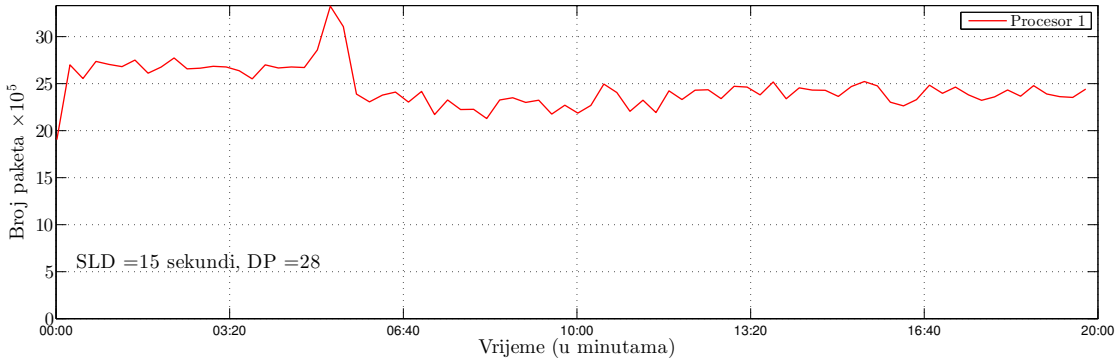
Slika 7.46 *Entropija po intervalima (3 procesora, SLD=60 sekundi, DP=24)*



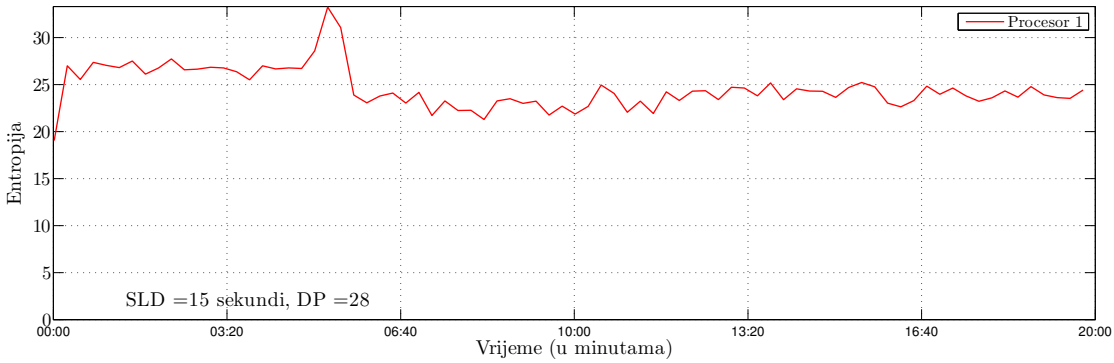
Slika 7.47 *Opterećenje sistema (4 procesora, SLD=60 sekundi, DP=24)*



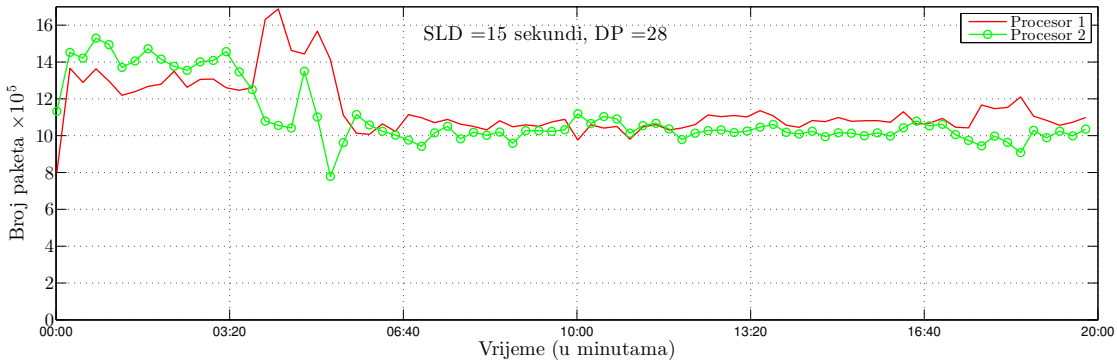
Slika 7.48 *Entropija po intervalima (4 procesora, SLD=60 sekundi, DP=24)*



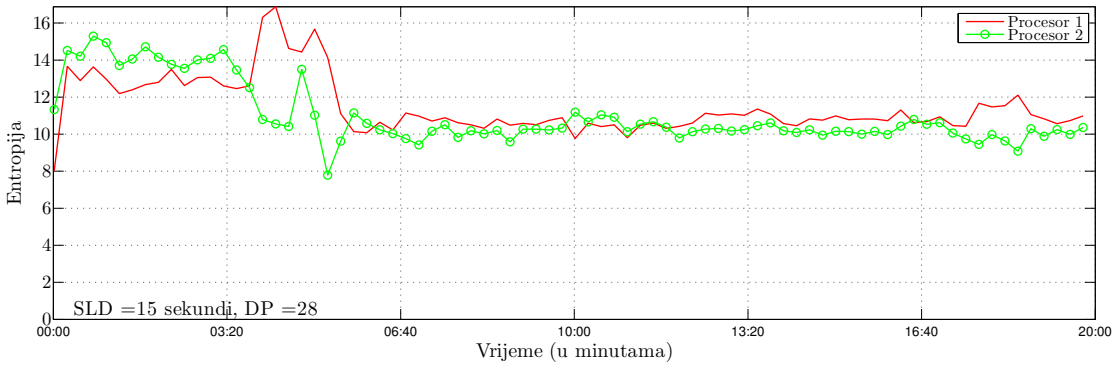
Slika 7.49 *Opterećenje sistema (1 procesor, SLD=15 sekundi, DP=28)*



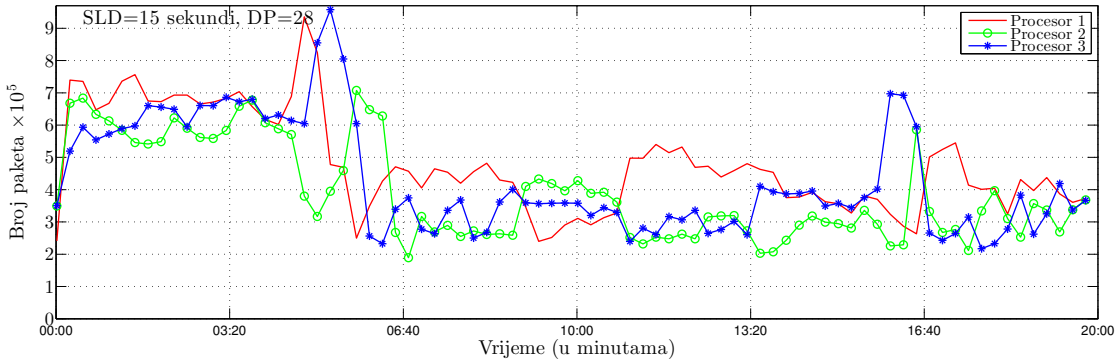
Slika 7.50 *Entropija po intervalima (1 procesor, SLD=15 sekundi, DP=28)*



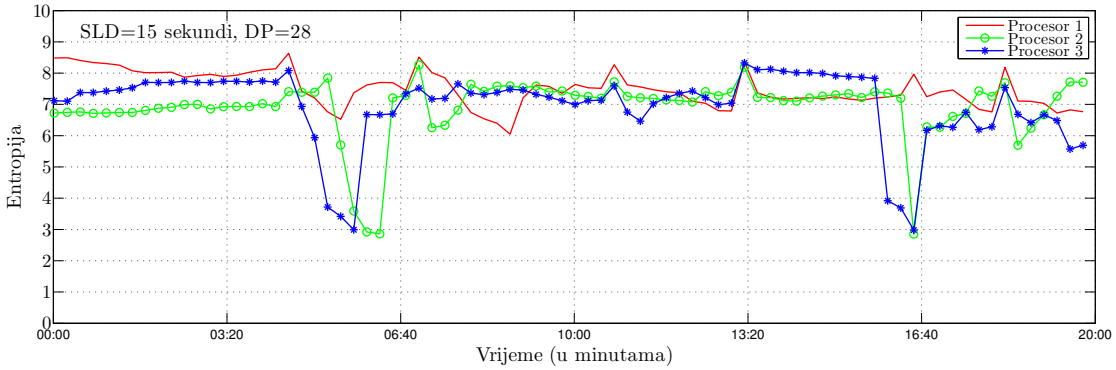
Slika 7.51 *Opterećenje sistema (2 procesora, SLD=15 sekundi, DP=28)*



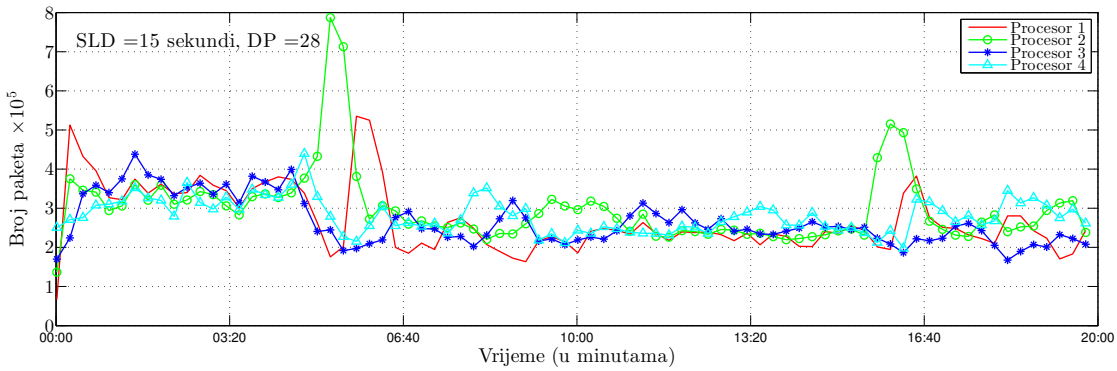
Slika 7.52 *Entropija po intervalima (2 procesora, SLD=15 sekundi, DP=28)*



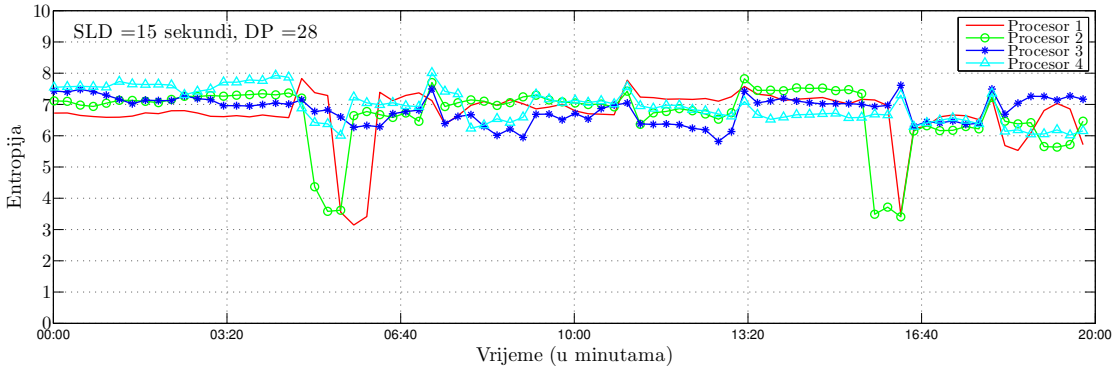
Slika 7.53 *Opterećenje sistema (3 procesora, SLD=15 sekundi, DP=28)*



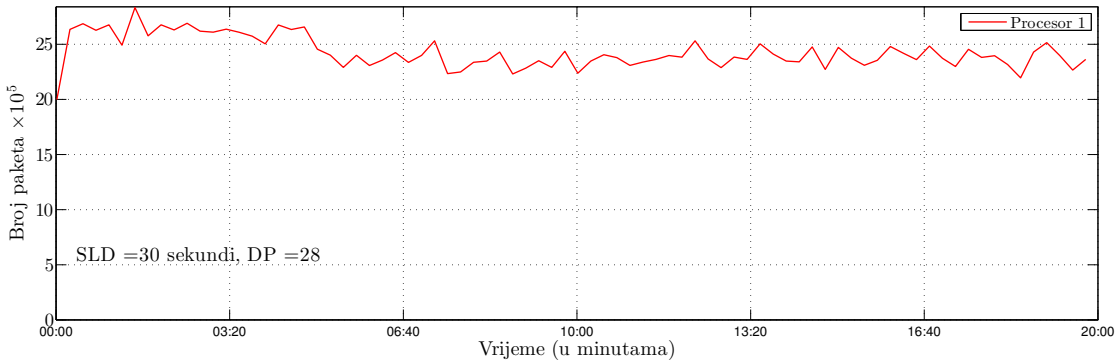
Slika 7.54 *Entropija po intervalima (3 procesora, SLD=15 sekundi, DP=28)*



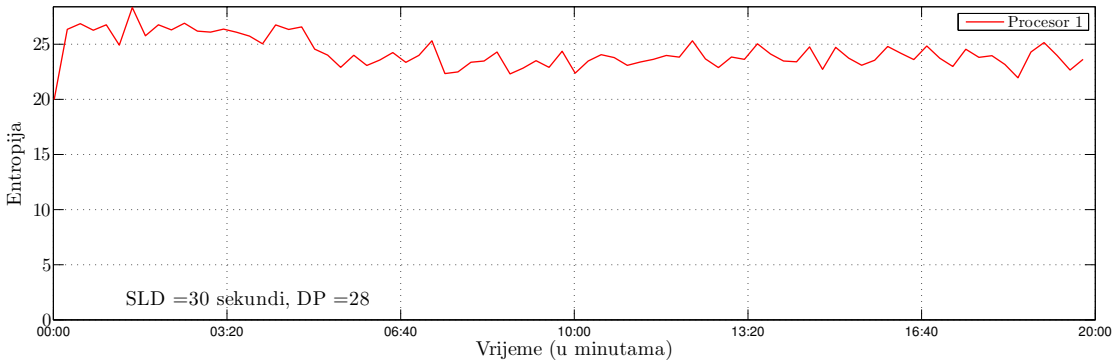
Slika 7.55 *Opterećenje sistema (4 procesora, SLD=15 sekundi, DP=28)*



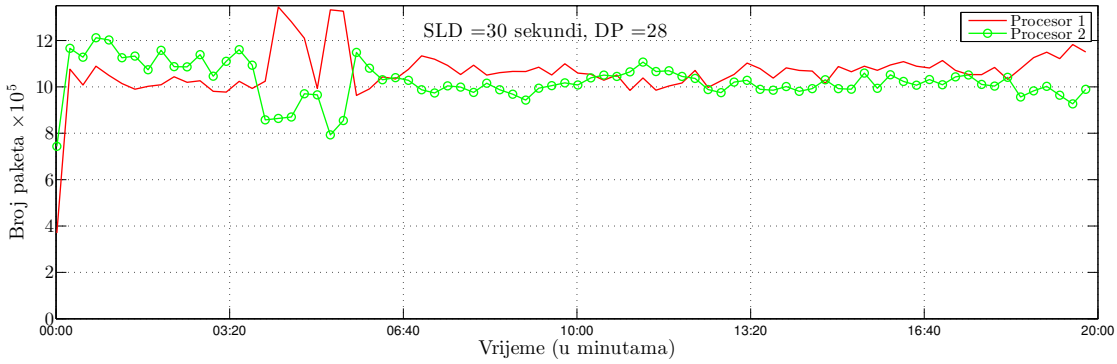
Slika 7.56 *Entropija po intervalima (4 procesora, SLD=15 sekundi, DP=28)*



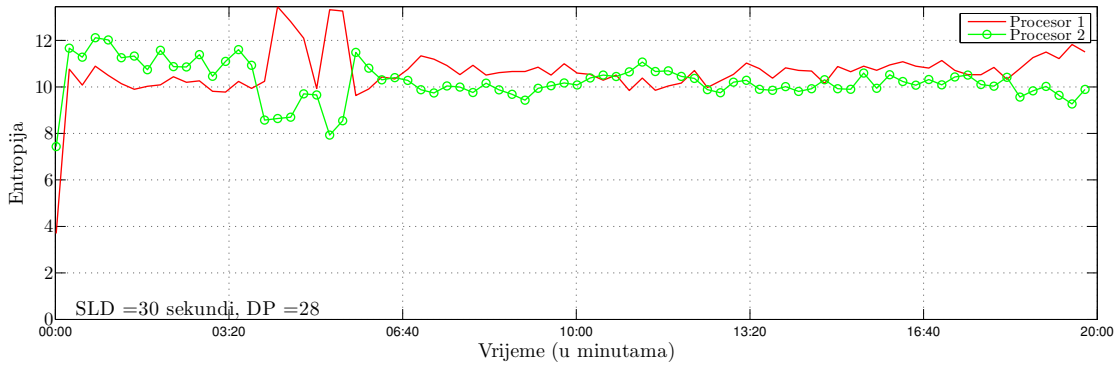
Slika 7.57 *Opterećenje sistema (1 procesor, SLD=30 sekundi, DP=28)*



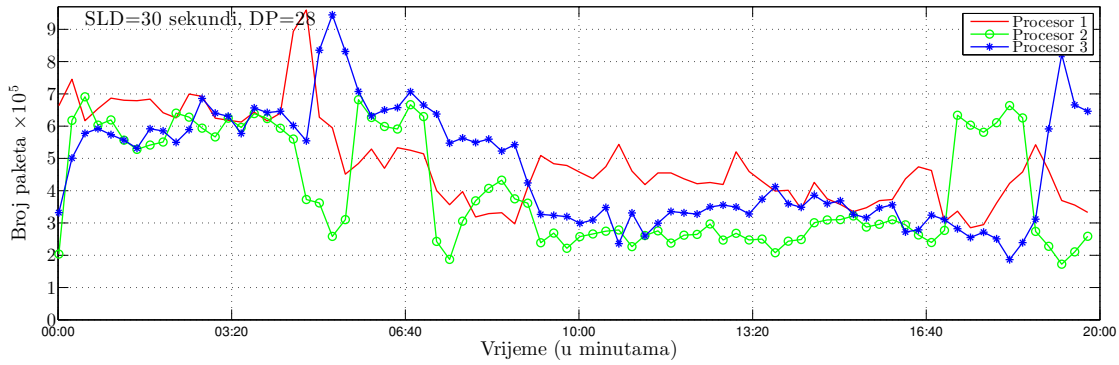
Slika 7.58 *Entropija po intervalima (1 procesor, SLD=30 sekundi, DP=28)*



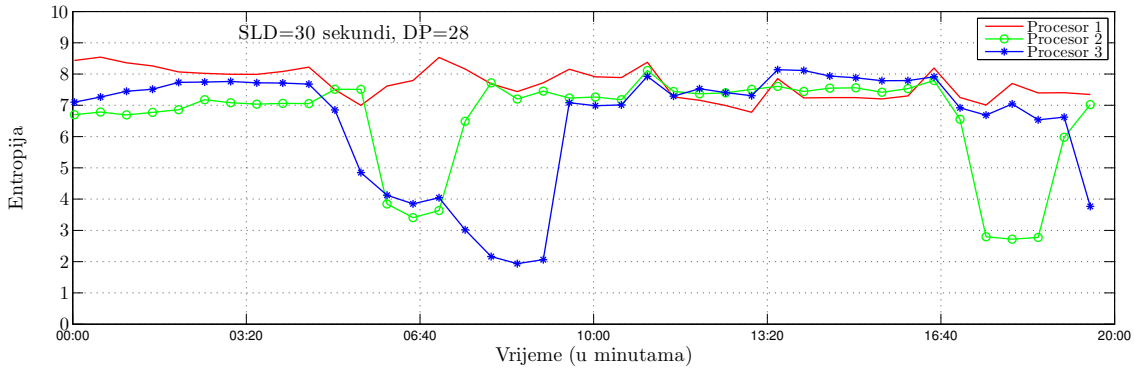
Slika 7.59 *Opterećenje sistema (2 procesora, SLD=30 sekundi, DP=28)*



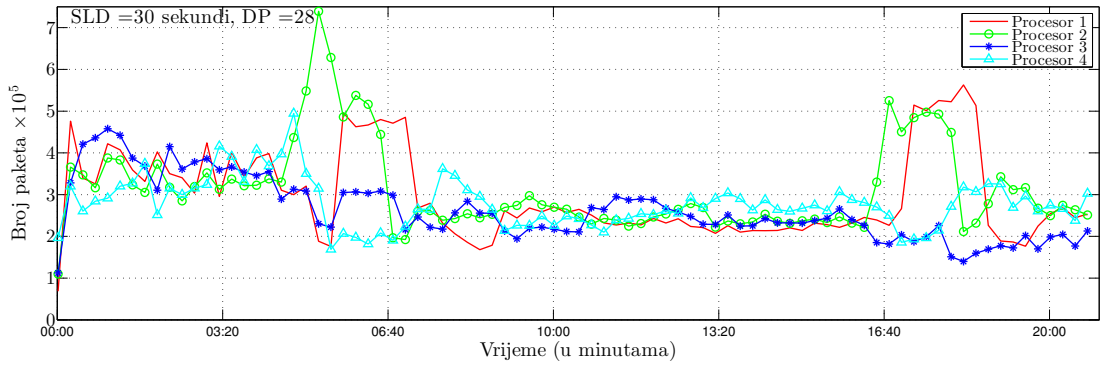
Slika 7.60 *Entropija po intervalima (2 procesora, SLD=30 sekundi, DP=28)*



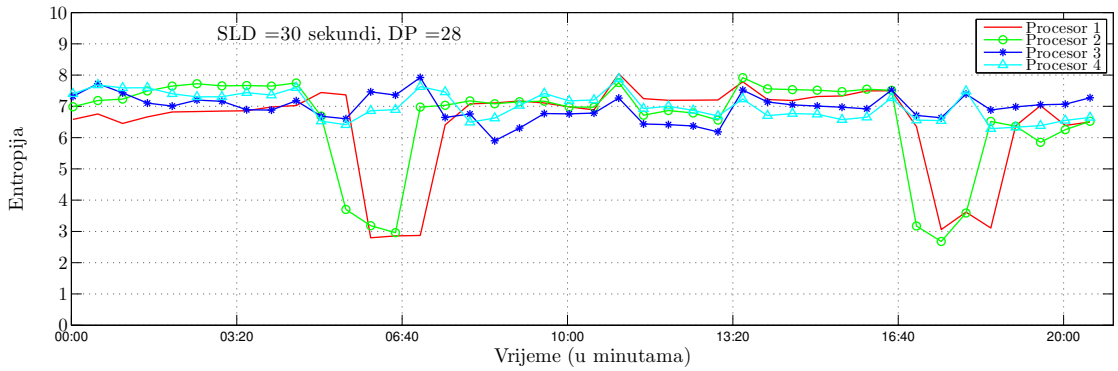
Slika 7.61 *Opterećenje sistema (3 procesora, SLD=30 sekundi, DP=28)*



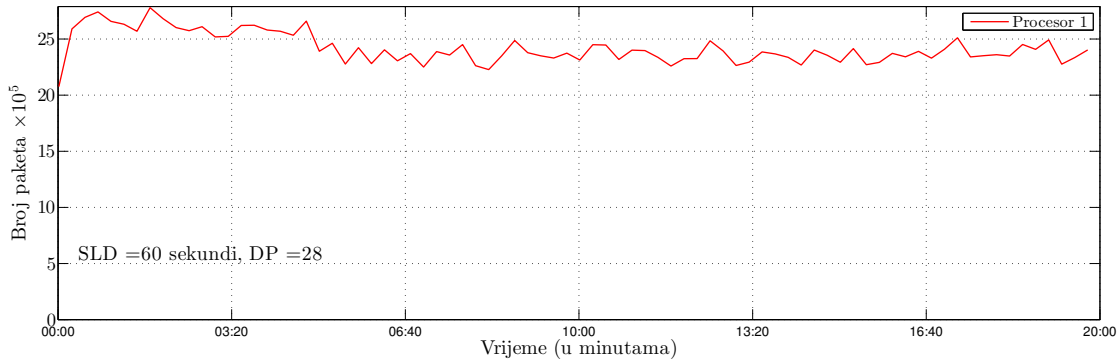
Slika 7.62 *Entropija po intervalima (3 procesora, SLD=30 sekundi, DP=28)*



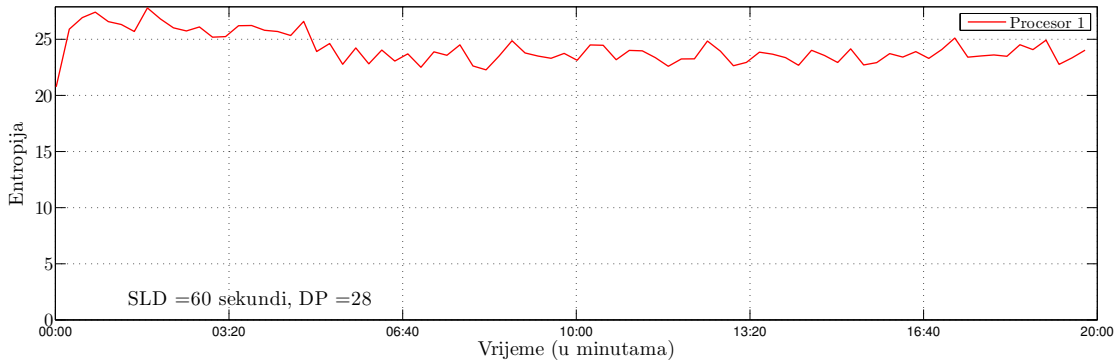
Slika 7.63 *Opterećenje sistema (4 procesora, SLD=30 sekundi, DP=28)*



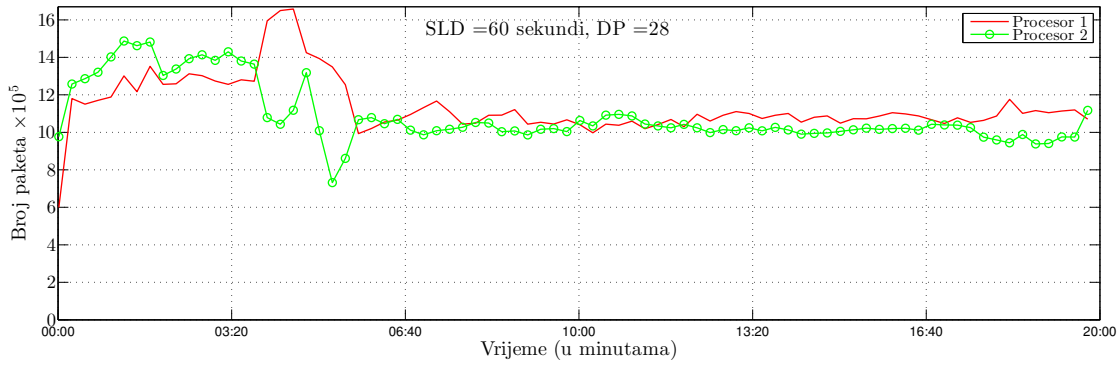
Slika 7.64 *Entropija po intervalima (4 procesora, SLD=30 sekundi, DP=28)*



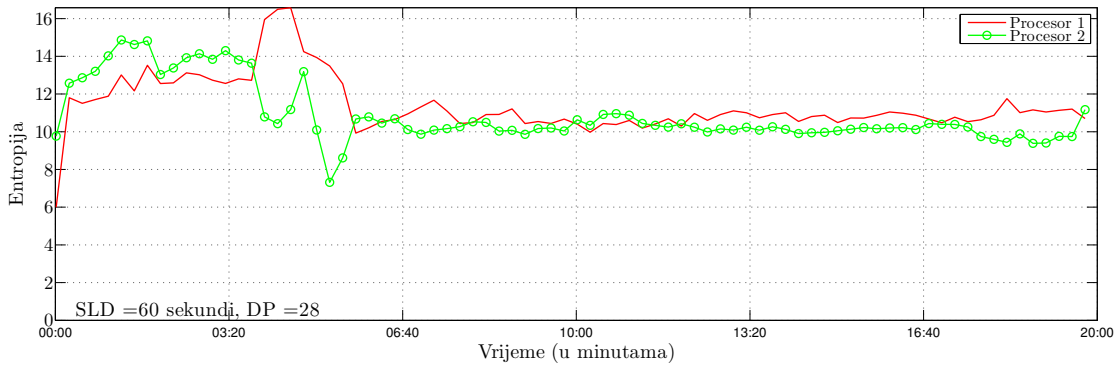
Slika 7.65 *Opterećenje sistema (1 procesor, SLD=60 sekundi, DP=28)*



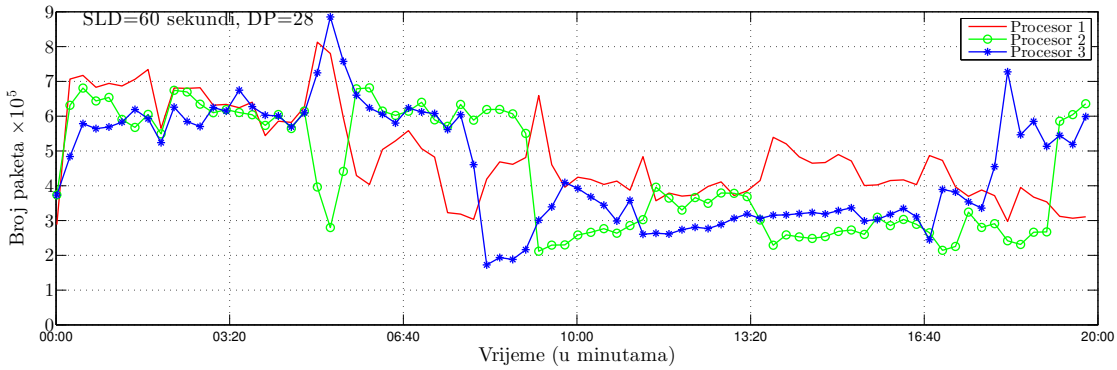
Slika 7.66 *Entropija po intervalima (1 procesor, SLD=60 sekundi, DP=28)*



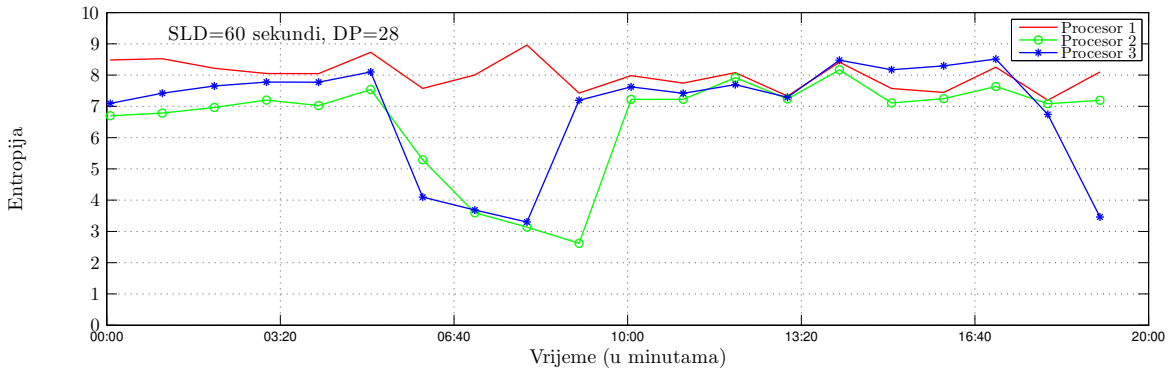
Slika 7.67 *Opterećenje sistema (2 procesora, SLD=60 sekundi, DP=28)*



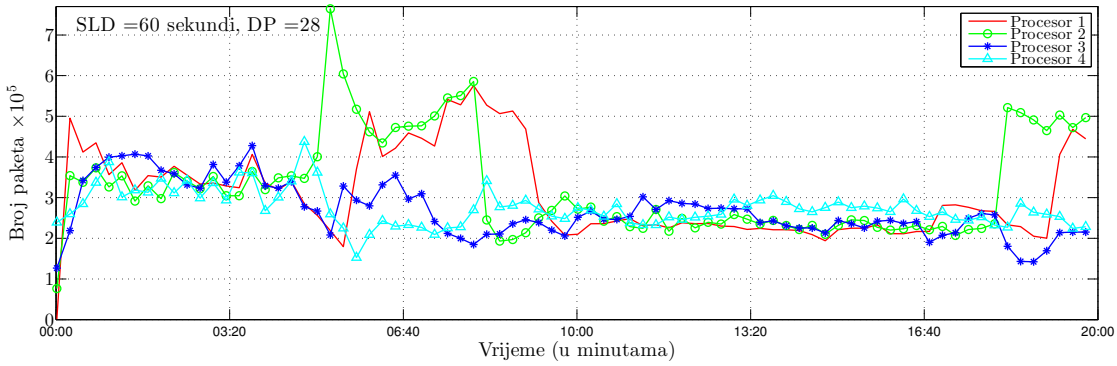
Slika 7.68 *Entropija po intervalima (2 procesora, SLD=60 sekundi, DP=28)*



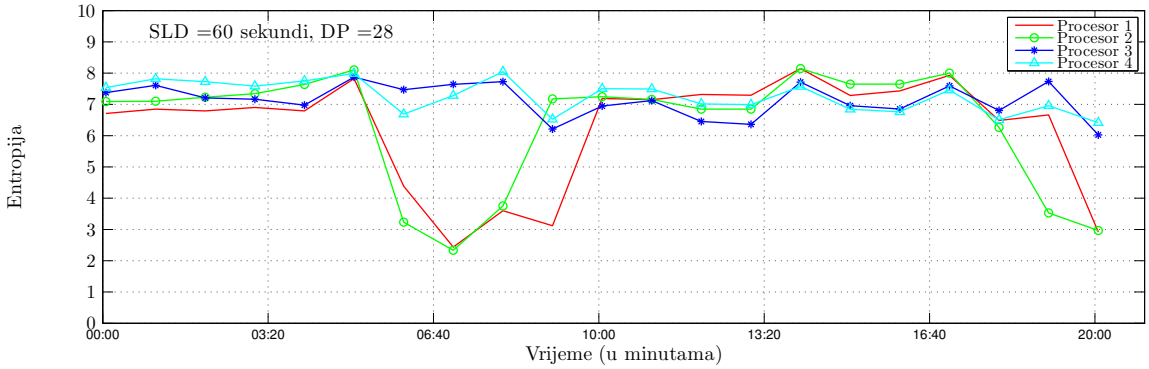
Slika 7.69 *Opterećenje sistema (3 procesora, SLD=60 sekundi, DP=28)*



Slika 7.70 *Entropija po intervalima (3 procesora, SLD=60 sekundi, DP=28)*

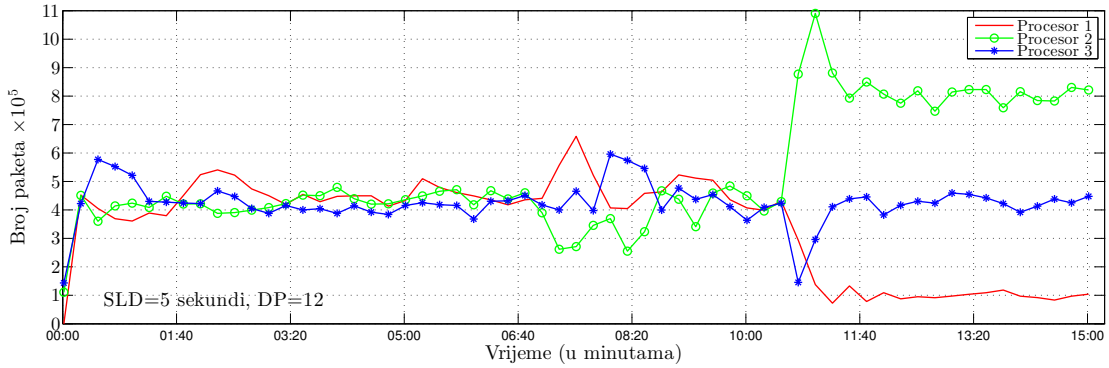


Slika 7.71 *Opterećenje sistema (4 procesora, SLD=60 sekundi, DP=28)*

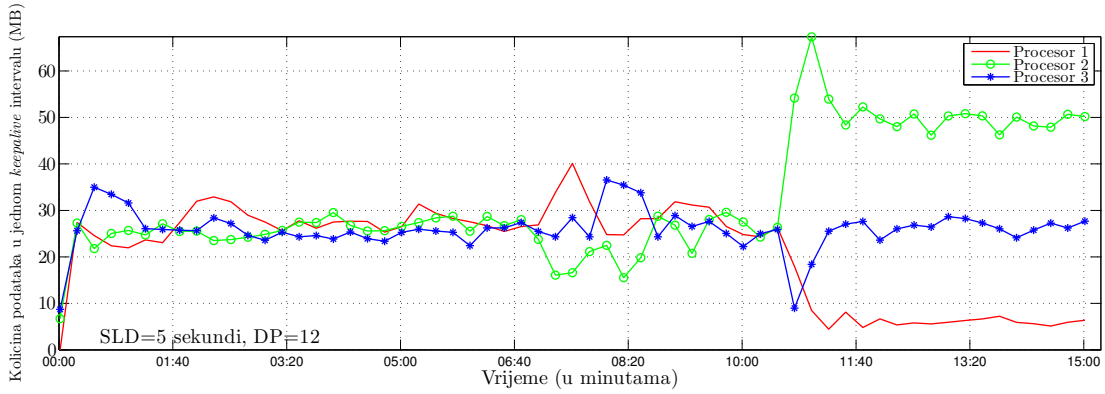


Slika 7.72 *Entropija po intervalima (4 procesora, SLD=60 sekundi, DP=28)*

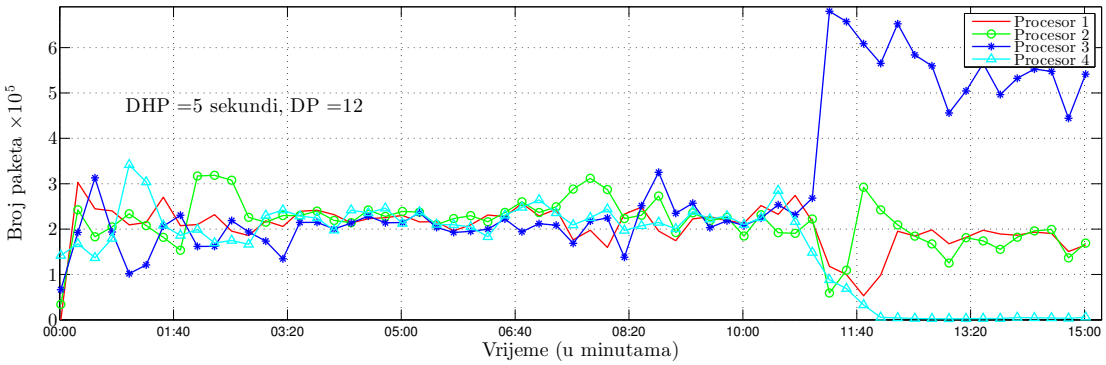
7.2 Rezultati testiranja algoritma za raspoređivanje opterećenja



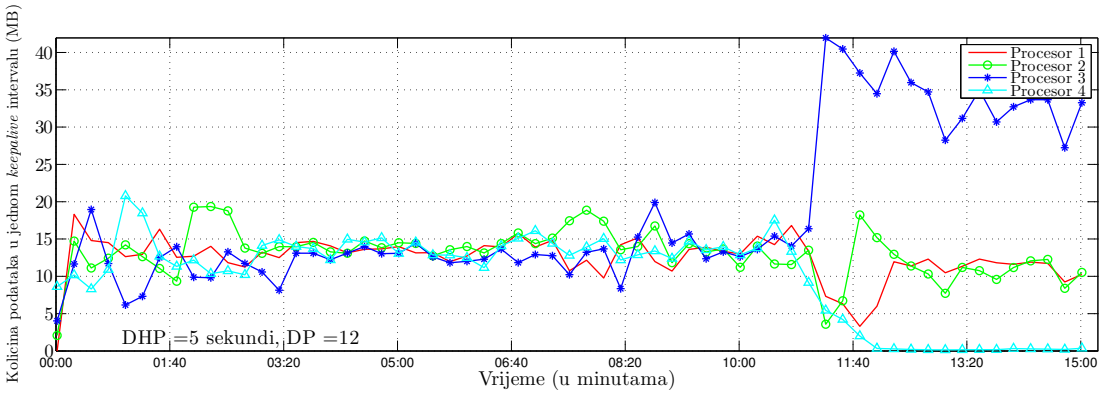
Slika 7.73 *Opterećenje sistema prema broju paketa u jednom intervalu (3 procesora, DHP=5 sekundi, DP=12)*



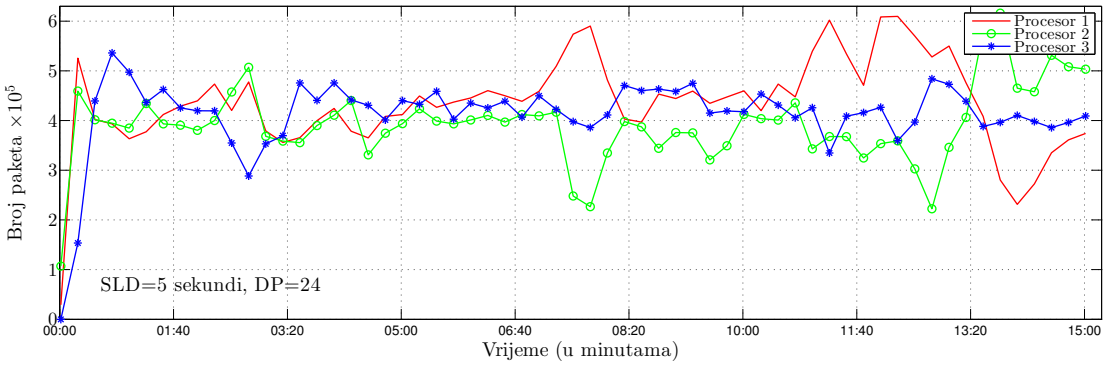
Slika 7.74 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (3 procesora, DHP=5 sekundi, DP=12)*



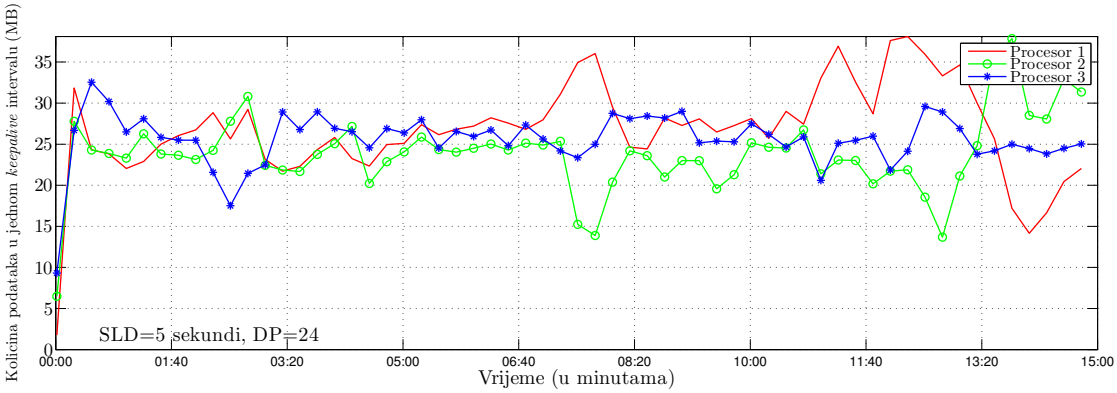
Slika 7.75 *Opterećenje sistema prema broju paketa u jednom intervalu (4 procesora, DHP=5 sekundi, DP=12)*



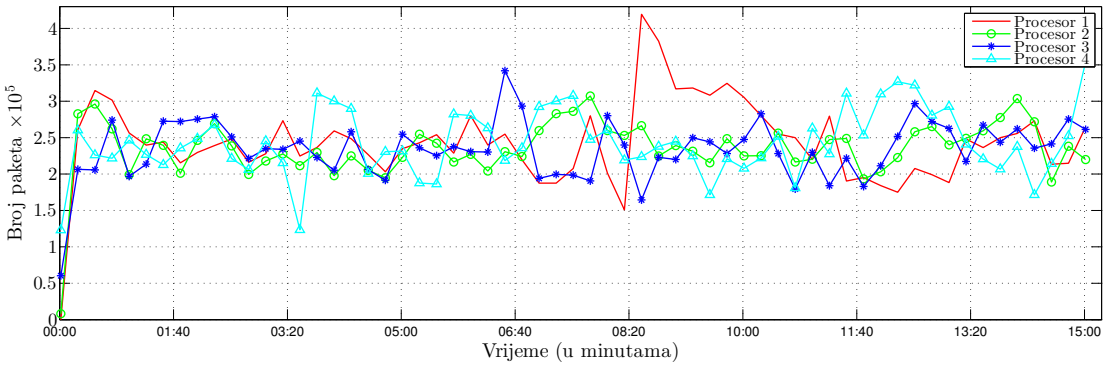
Slika 7.76 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (4 procesora, DHP=5 sekundi, DP=12)*



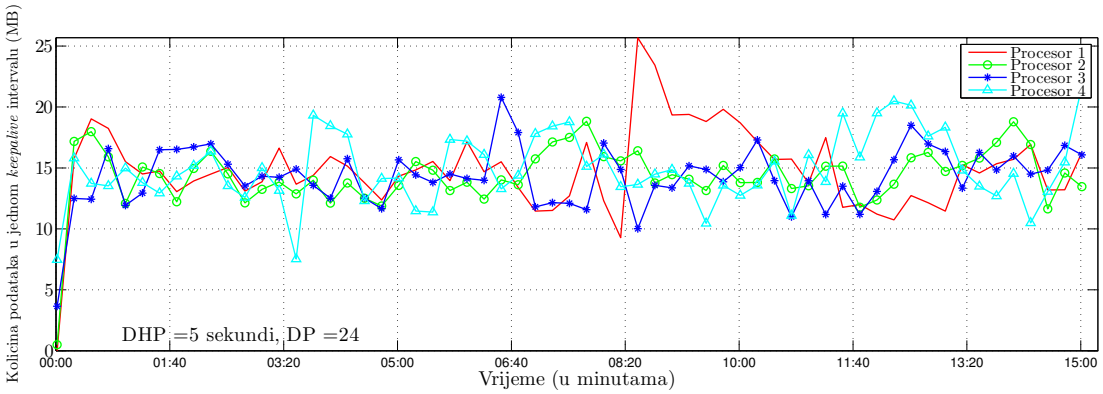
Slika 7.77 *Opterećenje sistema prema broju paketa u jednom intervalu (3 procesora, DHP=5 sekundi, DP=24)*



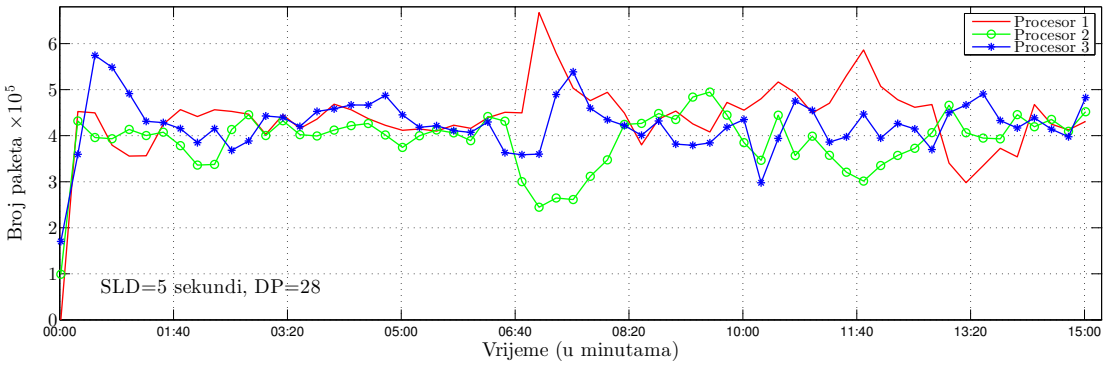
Slika 7.78 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (3 procesora, DHP=5 sekundi, DP=24)*



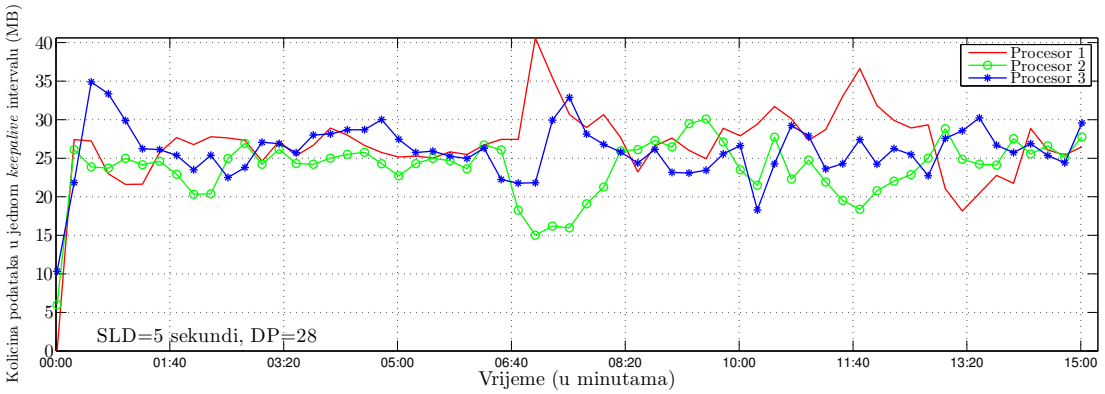
Slika 7.79 *Opterećenje sistema prema broju paketa u jednom intervalu (4 procesora, DHP=5 sekundi, DP=24)*



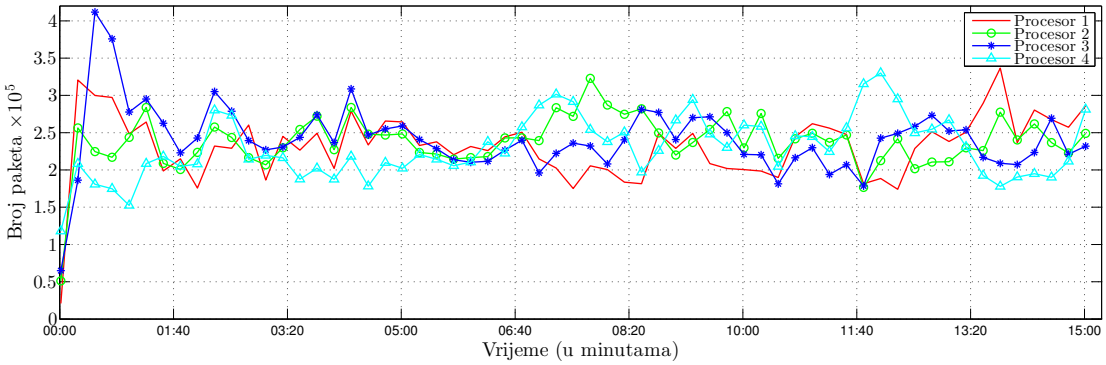
Slika 7.80 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (4 procesora, DHP=5 sekundi, DP=24)*



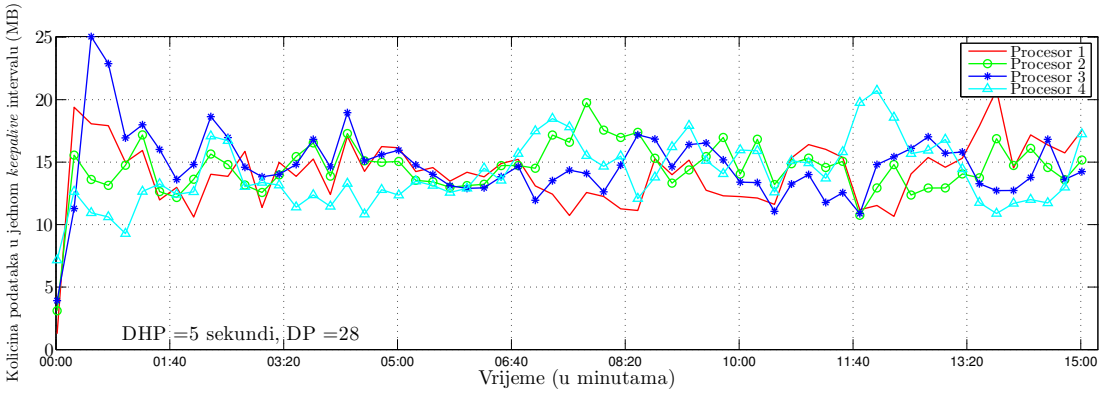
Slika 7.81 *Opterećenje sistema prema broju paketa u jednom intervalu (3 procesora, DHP=5 sekundi, DP=28)*



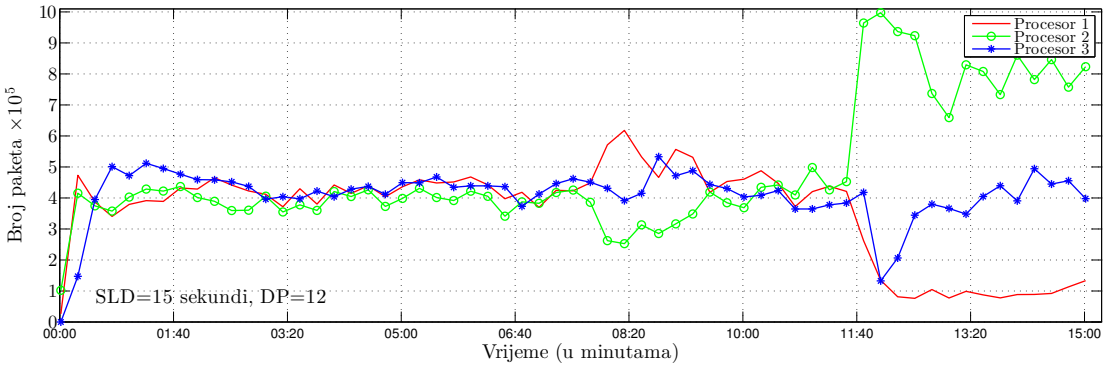
Slika 7.82 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (3 procesora, DHP=5 sekundi, DP=28)*



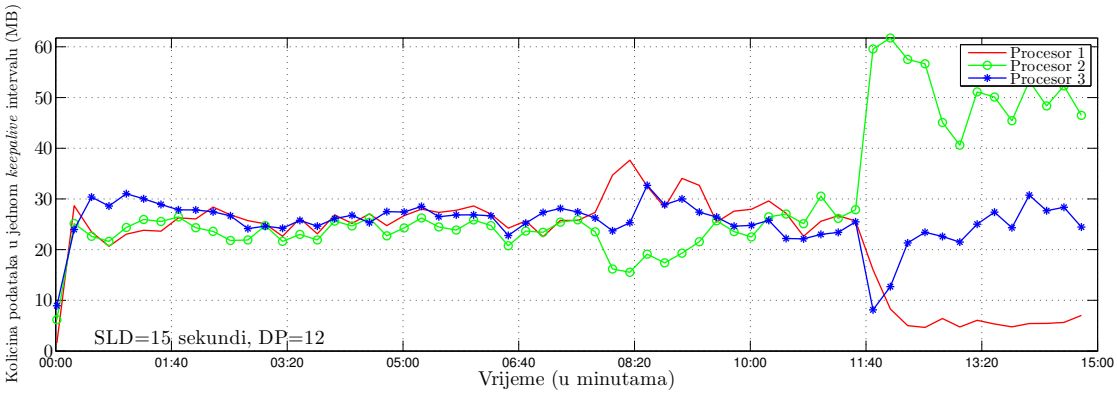
Slika 7.83 *Opterećenje sistema prema broju paketa u jednom intervalu (4 procesora, DHP=5 sekundi, DP=28)*



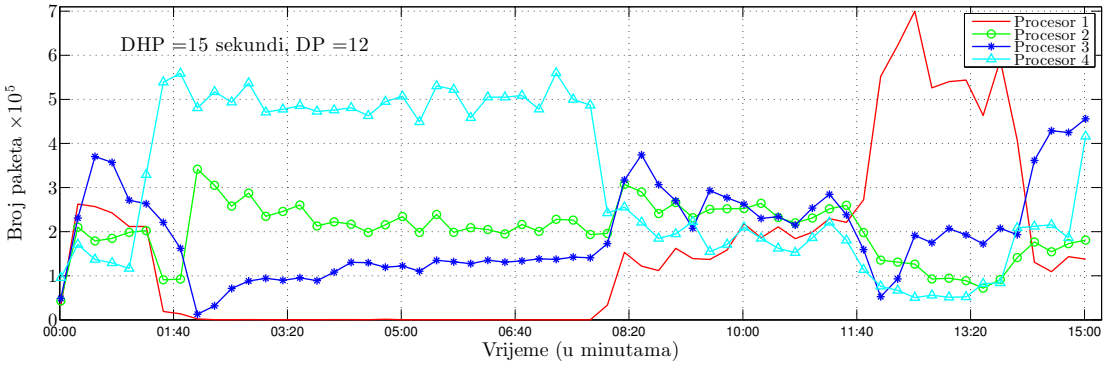
Slika 7.84 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (4 procesora, DHP=5 sekundi, DP=28)*



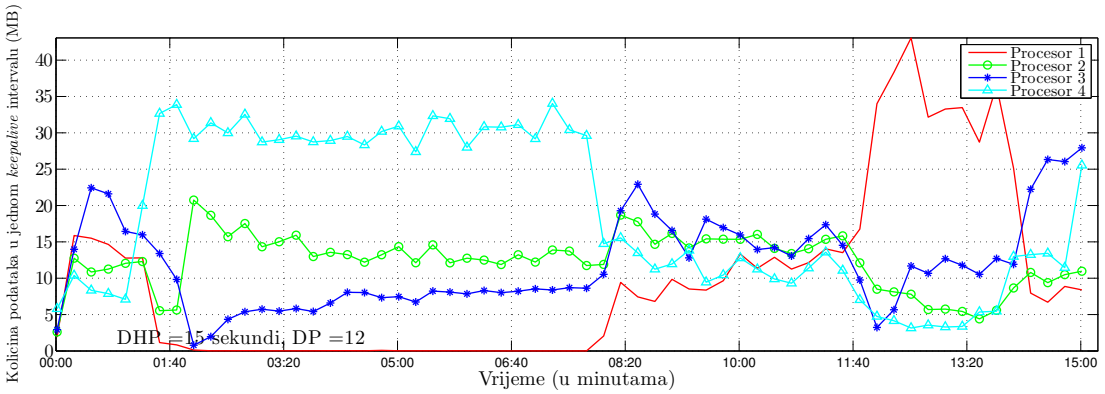
Slika 7.85 *Opterećenje sistema prema broju paketa u jednom intervalu (3 procesora, DHP=15 sekundi, DP=12)*



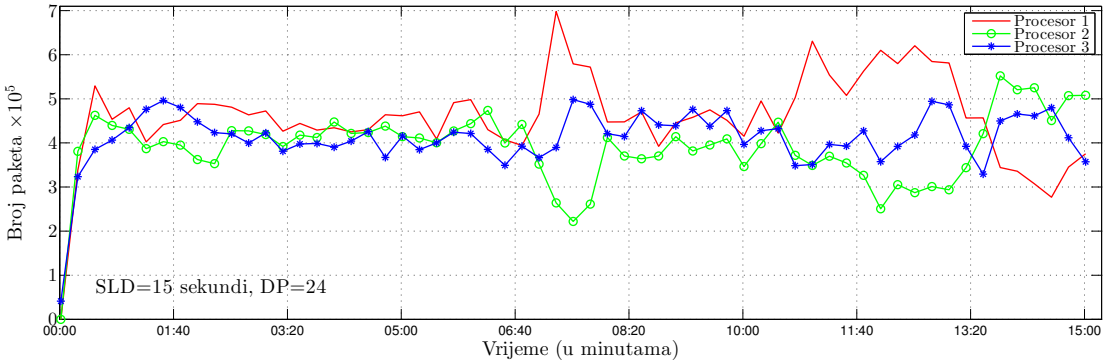
Slika 7.86 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (3 procesora, DHP=15 sekundi, DP=12)*



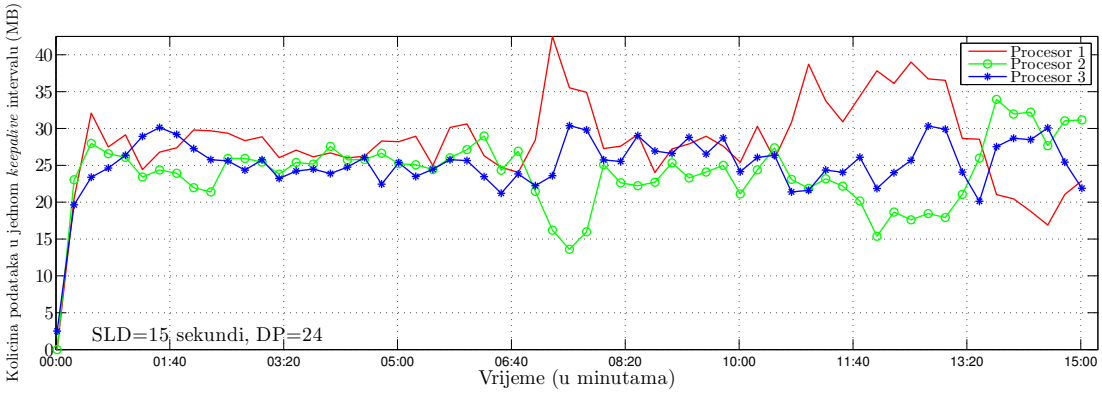
Slika 7.87 *Opterećenje sistema prema broju paketa u jednom intervalu (4 procesora, DHP=15 sekundi, DP=12)*



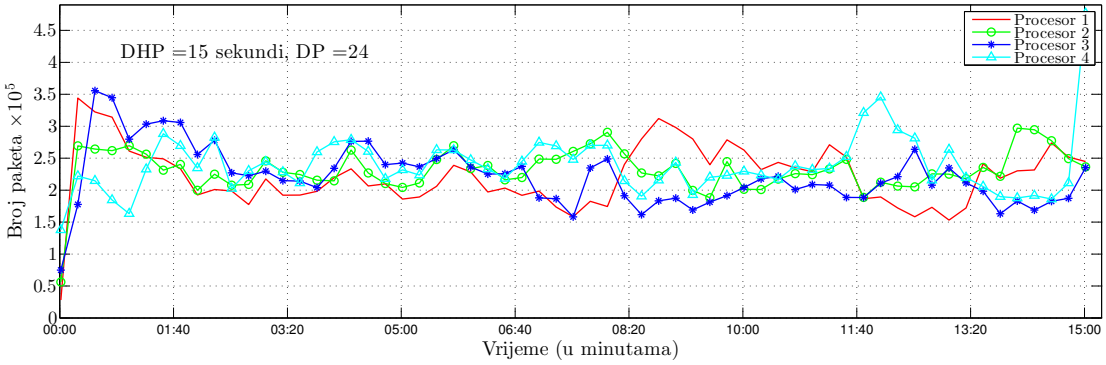
Slika 7.88 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (4 procesora, DHP=15 sekundi, DP=12)*



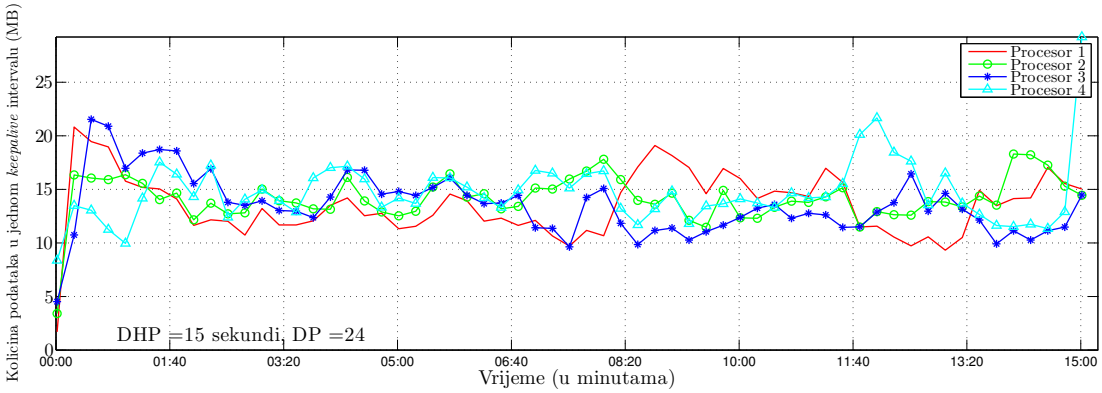
Slika 7.89 *Opterećenje sistema prema broju paketa u jednom intervalu (3 procesora, DHP=15 sekundi, DP=24)*



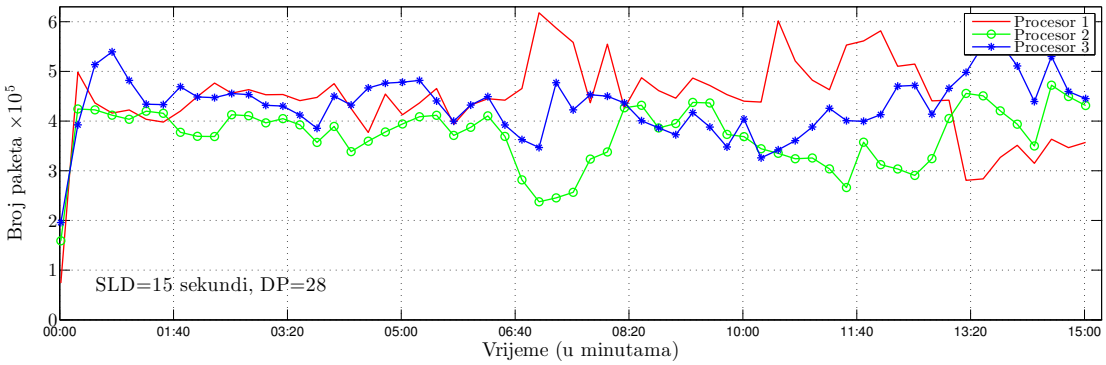
Slika 7.90 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (3 procesora, DHP=15 sekundi, DP=24)*



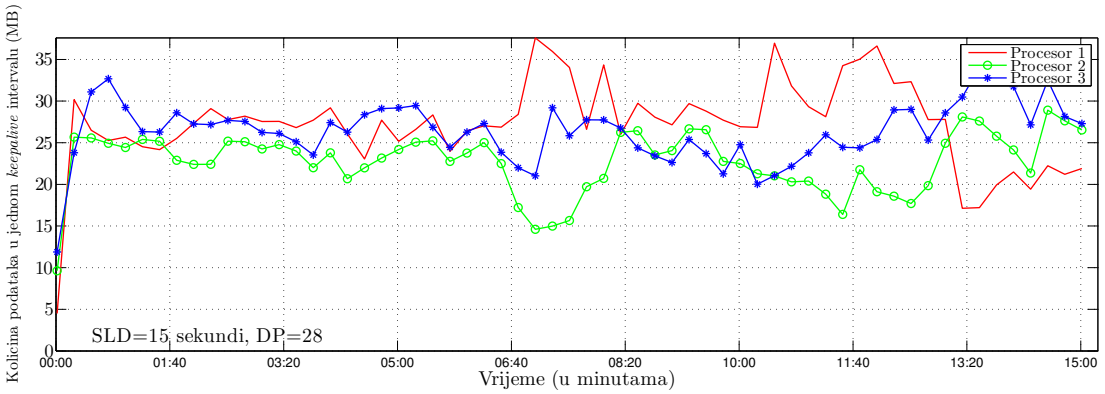
Slika 7.91 *Opterećenje sistema prema broju paketa u jednom intervalu (4 procesora, DHP=15 sekundi, DP=24)*



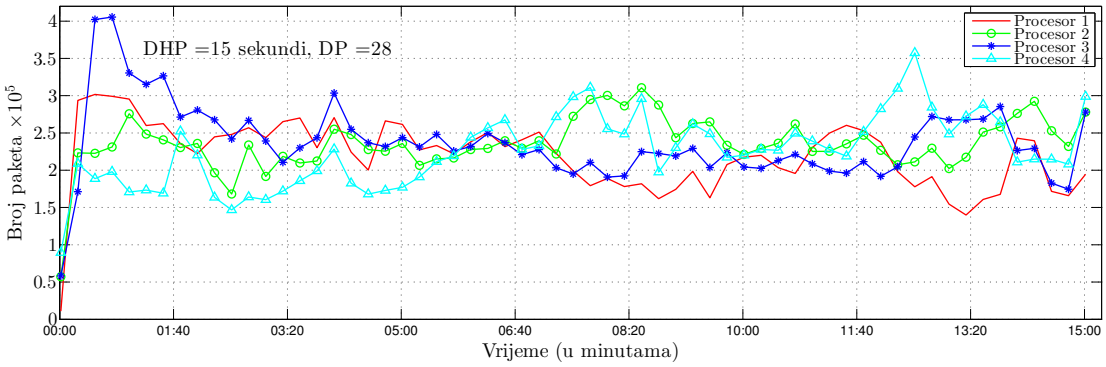
Slika 7.92 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (4 procesora, DHP=15 sekundi, DP=24)*



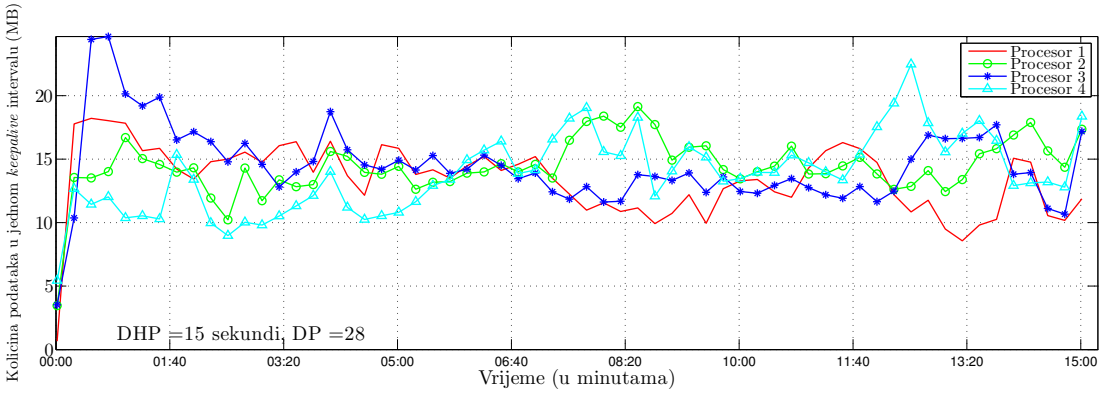
Slika 7.93 *Opterećenje sistema prema broju paketa u jednom intervalu (3 procesora, DHP=15 sekundi, DP=28)*



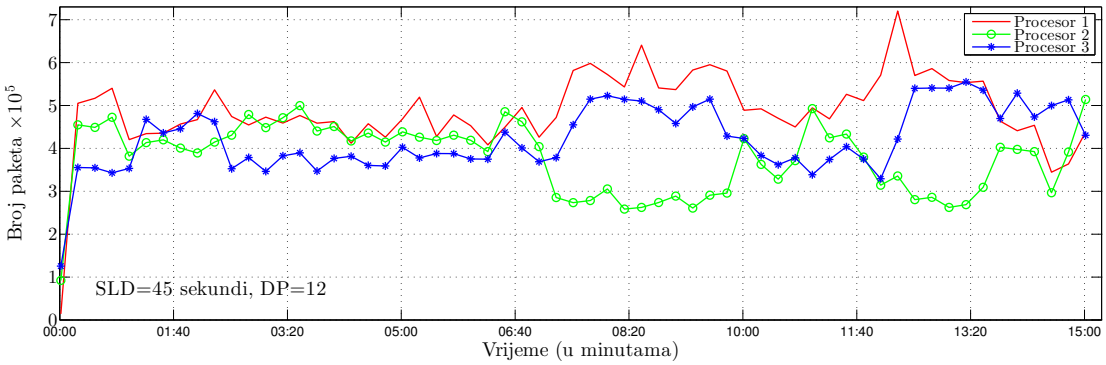
Slika 7.94 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (3 procesora, DHP=15 sekundi, DP=28)*



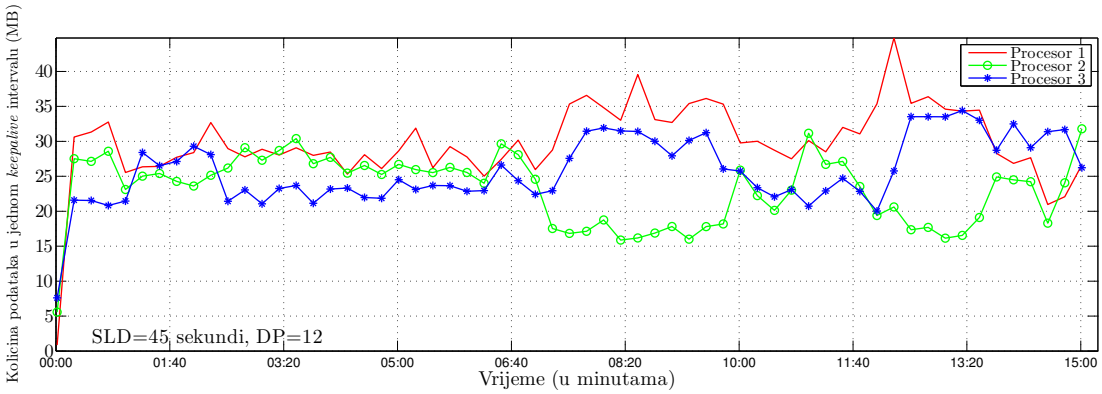
Slika 7.95 *Opterećenje sistema prema broju paketa u jednom intervalu (4 procesora, DHP=15 sekundi, DP=28)*



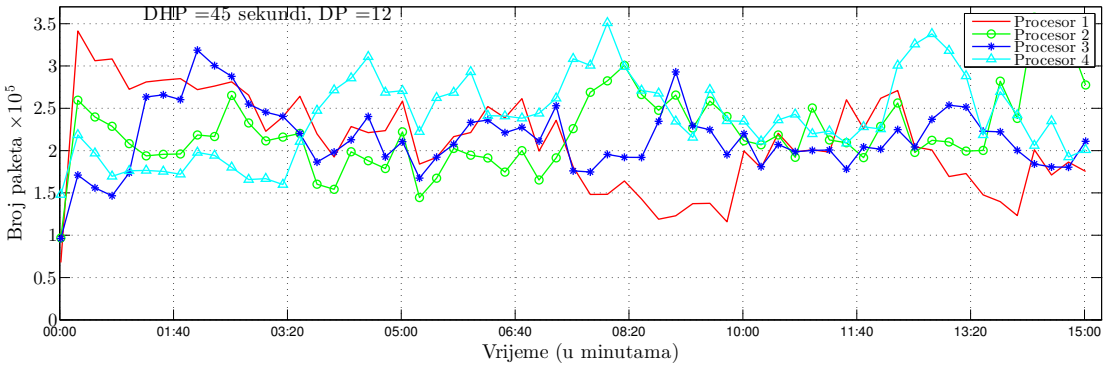
Slika 7.96 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (4 procesora, DHP=15 sekundi, DP=28)*



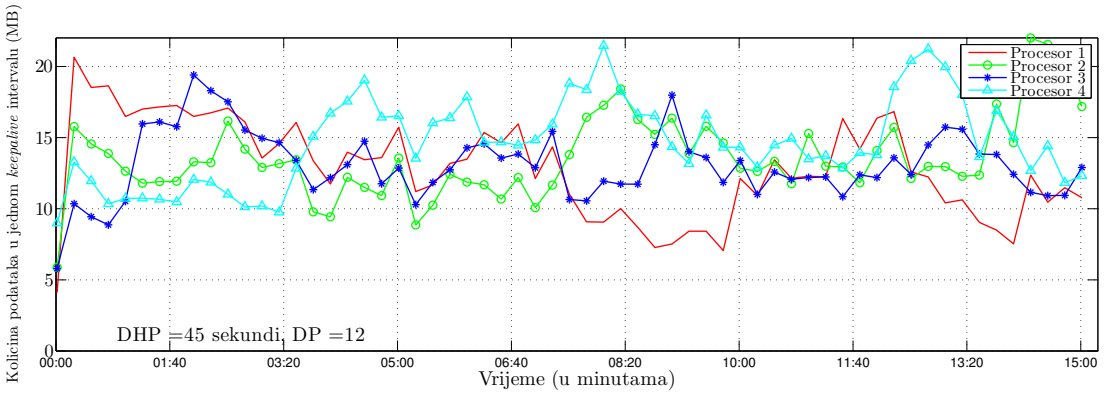
Slika 7.97 *Opterećenje sistema prema broju paketa u jednom intervalu (3 procesora, DHP=45 sekundi, DP=12)*



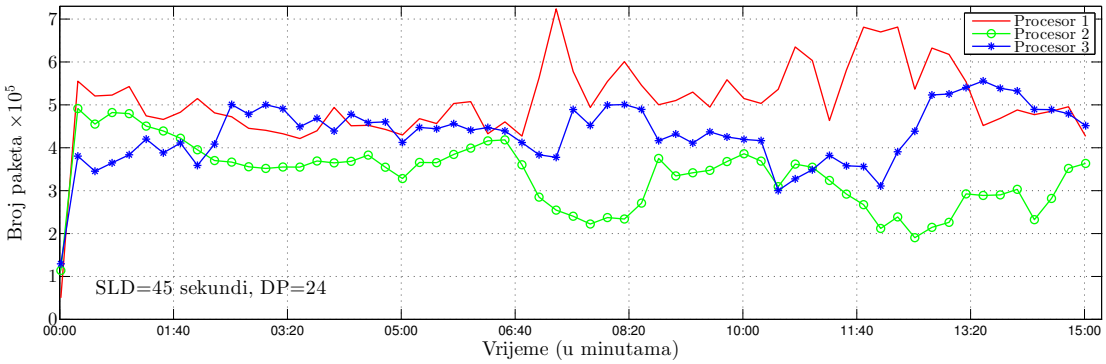
Slika 7.98 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (3 procesora, DHP=45 sekundi, DP=12)*



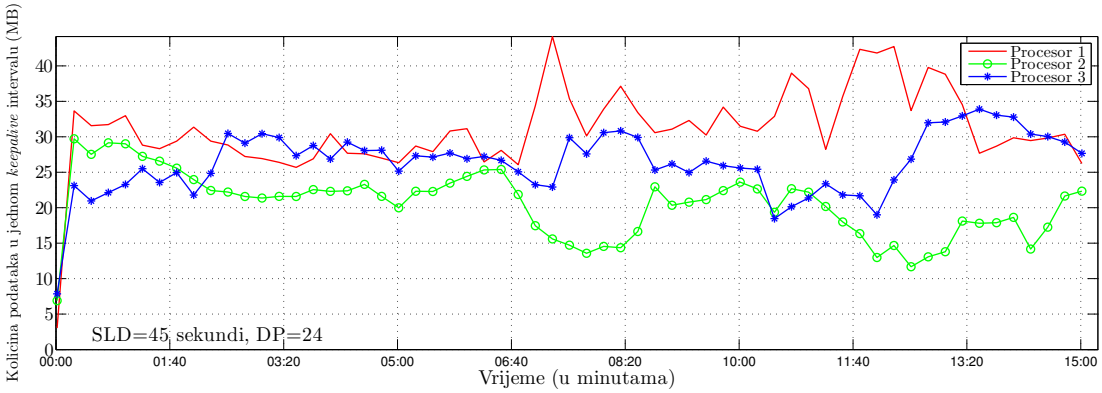
Slika 7.99 *Opterećenje sistema prema broju paketa u jednom intervalu (4 procesora, DHP=45 sekundi, DP=12)*



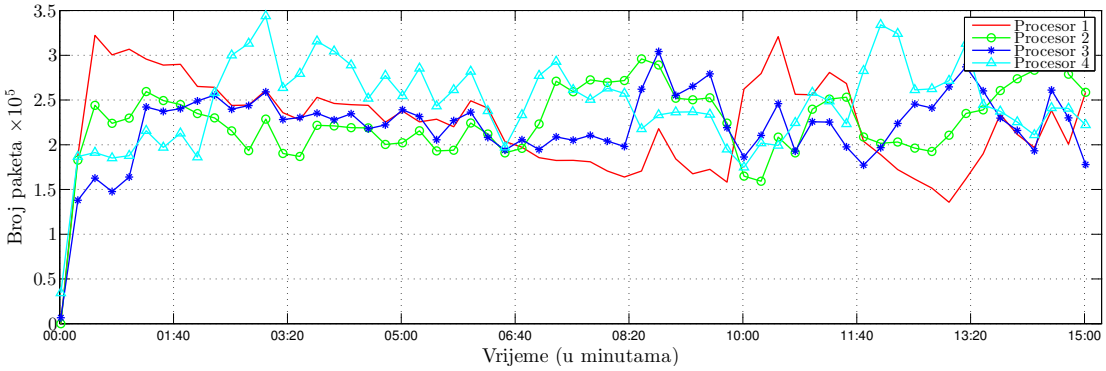
Slika 7.100 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (4 procesora, DHP=45 sekundi, DP=12)*



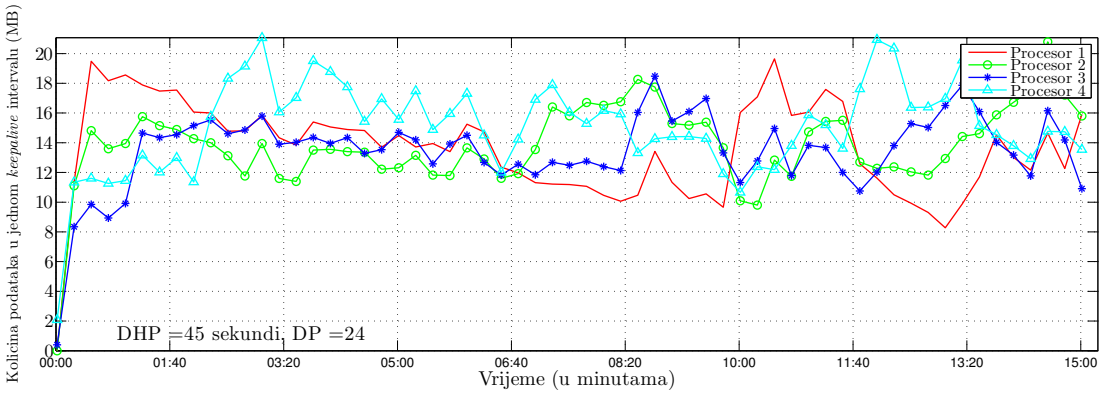
Slika 7.101 *Opterećenje sistema prema broju paketa u jednom intervalu (3 procesora, DHP=45 sekundi, DP=24)*



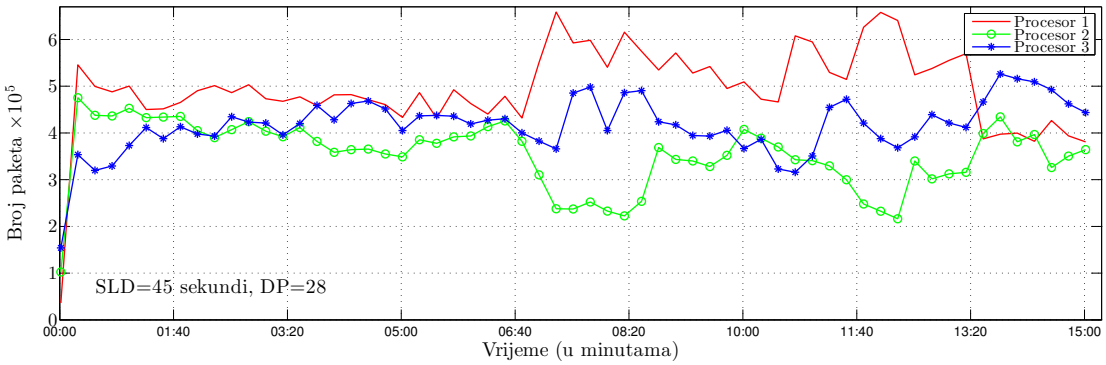
Slika 7.102 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (3 procesora, DHP=45 sekundi, DP=24)*



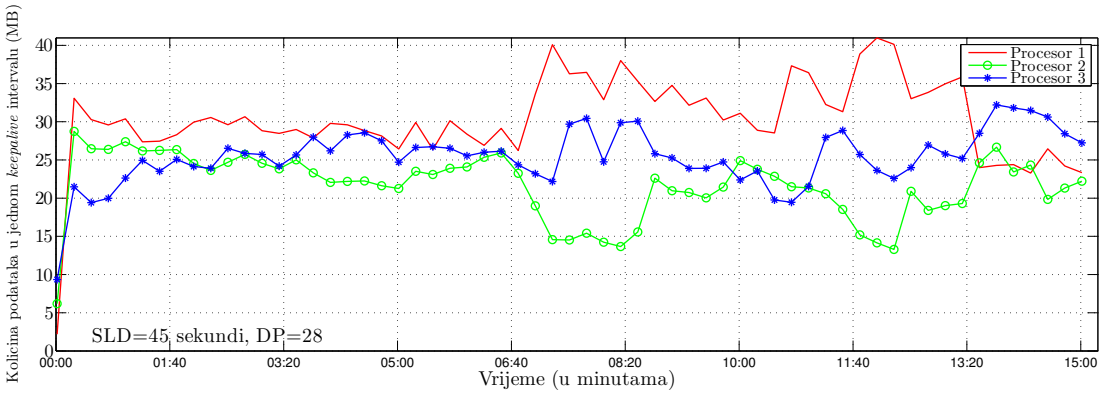
Slika 7.103 *Opterećenje sistema prema broju paketa u jednom intervalu (4 procesora, DHP=45 sekundi, DP=24)*



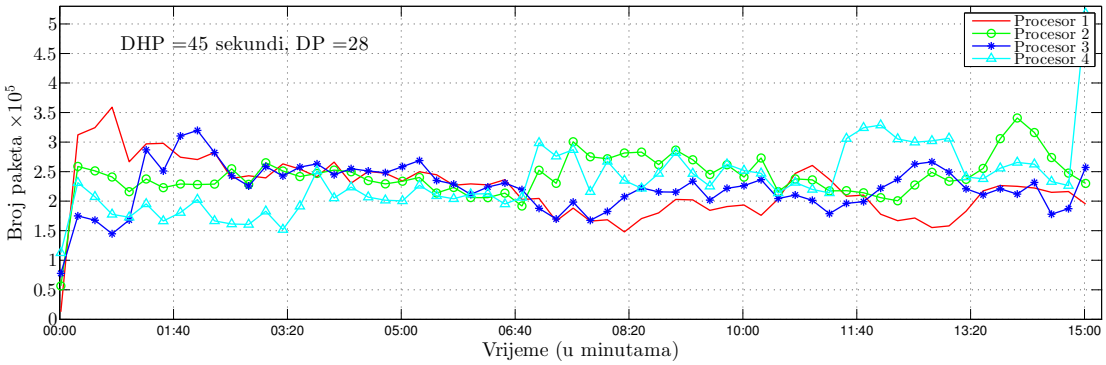
Slika 7.104 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (4 procesora, DHP=45 sekundi, DP=24)*



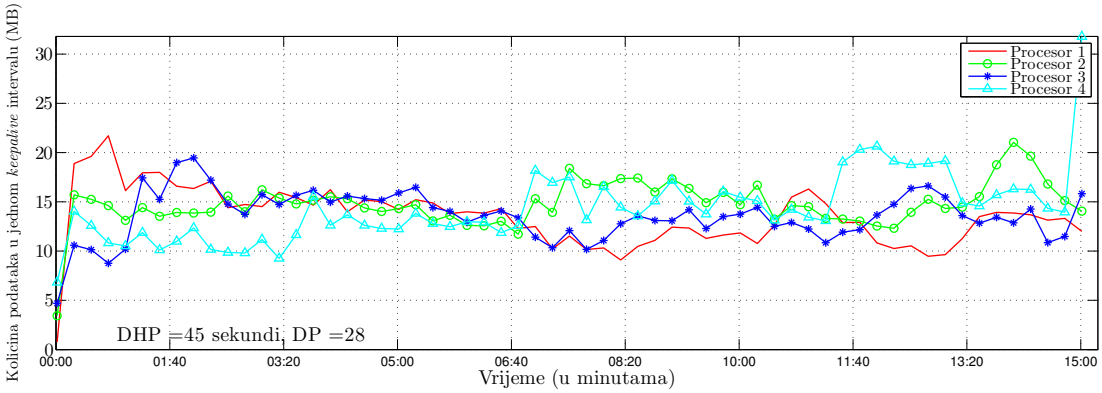
Slika 7.105 *Opterećenje sistema prema broju paketa u jednom intervalu (3 procesora, DHP=45 sekundi, DP=28)*



Slika 7.106 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (3 procesora, DHP=45 sekundi, DP=28)*



Slika 7.107 *Opterećenje sistema prema broju paketa u jednom intervalu (4 procesora, DHP=45 sekundi, DP=28)*



Slika 7.108 *Opterećenje sistema prema količini saobraćaja u jednom intervalu (4 procesora, DHP=45 sekundi, DP=28)*

Bibliografija

[1] S. W. Cadzow, "Security mechanisms in converged networks," in *Proceedings of the The First IEE International Conference on Commercialising Technology and Innovation*, IET, 2005.

[2] Z. Djuric and O. Joldzic, "Sigurnost web aplikacija," in *XVI Festival informatickih dostignuća Infofest 2009, Budva, Crna Gora*, pp. 204–212, 2009.

[3] O. Joldzic and Z. Djuric, "Tipovi napada na web servise," in *Infoteh 2010, Jahorina, Bosna i Hercegovina*, pp. 482–486, 2010.

[4] O. Joldzic and Z. Djuric, "Prijedlog rješenja za detekciju i klasifikaciju sigurnosnih propusta web aplikacija," *Journal of Information Technology and Multimedia Systems InfoM*, vol. 2010, pp. 42–52, 2010.

[5] O. Joldzic and Z. Djuric, "Mogućnost testiranja sigurnosnih propusta u web aplikacijama sa prijedlogom rješenja sistema za testiranje," in *XVI Festival informatickih dostignuća Infofest 2009, Budva, Crna Gora*, pp. 213–220, 2009.

[6] J. N. T. Library, "Understanding ping of death attacks." https://www.juniper.net/documentation/en_US/junos15.1x49/topics/concept/denial-of-service-os-ping-of-death-attack-understanding.html, 2015.

[7] W. Eddy, "Tcp syn flooding attacks and common mitigations." https://tools.ietf.org/html/rfc4987, 2007.

[8] U. o. S. C. Information Sciences Institute, "Transmission control protocol." https://tools.ietf.org/html/rfc793, 1981.

[9] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems," *Journal of Network and Computer Applications*, vol. 40, pp. 307–324, 2014.

[10] W. John, S. Tafvelin, and T. Olovsson, "Passive internet measurement: Overview and guidelines based on experiences," *Computer Communications*, vol. 33, pp. 533–550, 2010.

[11] A. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd edition*. Wiley, 2008.

[12] Nist and E. Aroms, *NIST Special Publication 800-94 Guide to Intrusion Detection and Prevention Systems (IDPS)*. Paramount, CA: CreateSpace, 2012.

[13] A. S. Ashoor and S. Gore, "Difference between intrusion detection system

(ids) and intrusion prevention system (ips)," in *Advances in Network Security and Applications, Proceedings of the 4th International Conference, CNSA 2011* (D. Wyld, M. Wozniak, N. Chaki, N. Meghanathan, and D. Nagamalai, eds.), Springer, 2011.

[14] R. Antonello, S. Fernandes, C. Kamienski, D. Sadok, J. Kelner, I. GóDor, G. Szabó, and T. Westholm, "Deep packet inspection tools and techniques in commodity platforms: Challenges and trends," *J. Netw. Comput. Appl.*, vol. 35, pp. 1863–1878, Nov. 2012.

[15] A. Ashraf, H. Jamal, S. A. Khan, Z. Ahmed, and M. I. Baig, "A heterogeneous service-oriented deep packet inspection and analysis framework for traffic-aware network management and security systems," *IEEE Access*, vol. 4, pp. 5918–5936, 2016.

[16] C. Xu, S. Chen, J. Su, S. M. Yiu, and L. C. K. Hui, "A survey on regular expression matching for deep packet inspection: Applications, algorithms, and hardware platforms," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 4, pp. 2991–3029, 2016.

[17] S. Garg, A. Garg, A. Kandpal, K. Joshi, R. Chauhan, and R. H. Goudar, "Ontology and specification-based intrusion detection and prevention system," in *Proceedings of the Confluence 2013: The Next Generation Information Technology Summit (4th International Conference)*, pp. 154–159, IET, 2013.

[18] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, pp. 643–666, 2004.

[19] J. P. Anderson, "Computer Security Technology Planning Study," Tech. Rep. ESD-TR-73-51, U.S. Air Force Electronic Systems Division, october 1972.

[20] P. A. Karger and R. R. Schell, "Multics security evaluation: Vulnerability analysis," Tech. Rep. ESD-TR-74-193, HQ Electronic Systems Division, jun 1974.

[21] K. J. Biba, "Integrity considerations for secure computer systems," tech. rep., MITRE Corp., may 1977.

[22] R. Bisbey and D. Hollingworth, "Protection analysis: Final report," tech. rep., Advanced Research Projects Agency, may 1978.

[23] J. Anderson, "Computer security threat monitoring and surveilance," *NIST Technical Report, 1980*, 1980.

[24] G. Nibaldi, "Proposed technical evaluation criteria for trusted computer systems," tech. rep., MITRE Corp., october 1979.

[25] S. T. Zargar, J. B. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE Communications Surveys & Tutorials*, pp. 2046–2069, 2013.

[26] T. Ptacek and T. Newsham, "Insertion, evasion, and denial of service: Eluding network intrusion detection," tech. rep., 1998.

[27] T. S. Research and I. Group, "Snort - network intrusion detection and prevention system." http://manual-snort-org.s3-website-us-east-1.amazonaws.com/, 2016.

[28] U. Oktay and O. K. Sahingoz, "Proxy network intrusion detection system for cloud computing," in *Proceedings of the 2013 International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, pp. 98–104, IEEE, 2013.

[29] T. Xing, D. Huang, L. Xu, C.-J. Chung, and P. Khatkar, "Snortflow: A openflow-based intrusion prevention system in cloud environment," in *Proceedings of the 2013 Second GENI Research and Educational Experiment Workshop*, GREE '13, pp. 89–92, IEEE Computer Society, 2013.

[30] H. Li and D. Liu, "Research on intelligent intrusion prevention system based on snort," in *Proceedings of the 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*, pp. 251–253, IEEE, 2010.

[31] C.-M. Chen, P.-Y. Yang, Y.-H. Ou, and H.-W. Hsiao, "Targeted attack prevention at early stage," in *Proceedings of the 2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 866–870, IEEE, 2014.

[32] C. Xiuqing, Z. Yongping, and G. Yu, "Adaptive intrusion prevention algorithm based on hmm model," in *Proceedings of the 2011 International Conference on E-Business and E-Government (ICEE)*, pp. 1–4, IEEE, 2011.

[33] J. Mirkovic, P. Reiher, G. Prier, S. Michel, and J. Li, "D-ward: Ddos network attack recognition and defense." http://www.lasr.cs.ucla.edu/ddos/, 2002.

[34] J. Mirkovic, G. Prier, and P. Reiher, "Attacking ddos at the source," in *Proceedings of 10th IEEE International Conference on Network Protocols*, pp. 312–321, 2002.

[35] J. Mirkovic, *D-WARD: Source-end Defense Against Distributed Denial-of-service Attacks*. PhD thesis, 2003. AAI3121225.

[36] J. Steinberger, A. Sperotto, H. Baier, and A. Pras, "Collaborative attack mitigation and response: A survey," in *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 910–913, IEEE, 2015.

[37] A. Yaar, A. Perrig, and D. Song, "Siff: A stateless internet flow filter to mitigate DDoS flooding attacks," in *Proceedings of IEEE Symposium on Security and Privacy*, may 2004.

[38] "Senss: Software-defined security service." https://steel.isi.edu/Projects/SENSS/, 2016.

[39] M. Yu, Y. Zhang, J. Mirkovic, and A. Alwabel, "Senss: Software defined security service," in *Presented as part of the Open Networking Summit 2014 (ONS 2014)*, (Santa Clara, CA), USENIX, 2014.

[40] A. Alwabel, M. Yu, Y. Zhang, and J. Mirkovic, "Senss: Observe and control your own traffic in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 349–350, Aug. 2014.

[41] E. Shi, I. Stoica, D. Andersen, and A. Perrig, "Overdose: A generic ddos protection service using an overlay network," tech. rep., 2006.

[42] S. Yu, Y. Tian, S. Guo, and D. O. Wu, "Can we beat ddos attacks in clouds?," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 2245–2254, IEEE, 2014.

[43] E. Chan, H. Chan, K. Chan, V. Chan, S. Chanson, M. Cheung, C. Chong, K. Chow, A. Hui, L. Hui, L. Lam, W. Lau, K. Pun, A. Tsang, W. Tsang, S. Tso, D.-Y. Yeung, and K. Yu, "Idr: An intrusion detection router for defending against distributed denial-of-service (ddos) attacks," in *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks, 2004.*, IEEE, May 2004.

[44] K. Kumar, R. C. Joshi, and K. Singh, "A distributed approach using entropy to detect ddos attacks in isp domain," in *Proceedings of the 2007 International Conference on Signal Processing, Communications and Networking*, pp. 331–337, IEEE, 2007.

[45] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, IMC '05, pp. 32–32, USENIX Association, 2005.

[46] K. Bernsmed and S. Fischer-Hubner, "Denial-of-service mitigation for internet services," in *Proceedings of the NordSec 2014*, pp. 213–228, Springer, 2014.

[47] K. Prasad, A. Reddy, and K. Rao, "Discriminating ddos attack traffic from flash crowds on internet threat monitors (itm) using entropy variations," *African Journal of Computing & ICT*, vol. 6, pp. 53–62, 2013.

[48] K. Li, W. Zhou, P. Li, J. Hai, and J. Liu, "Distinguishing ddos attacks from flash crowds using probability metrics," in *Proceedings of the Third International Conference on Network and System Security, 2009. (NSS '09.)*, pp. 9–17, IEEE, 2009.

[49] P. R. Reddy, R. Siva, and C. Malathi, "Techniques to differentiate ddos attacks from flash crowd," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, pp. 295–299, 2013.

[50] T. Thapngam, S. Yu, W. Zhou, and G. Beliakov, "Discriminating ddos attack traffic from flash crowd through packet arrival patterns," in *Proceedings of the 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 952–957, IEEE, 2011.

[51] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, (Berkeley, CA, USA), pp. 287–300, USENIX Association, 2005.

[52] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *Proceedings of the 22Nd International Conference*

on Theory and Applications of Cryptographic Techniques, EUROCRYPT'03, (Berlin, Heidelberg), pp. 294–311, Springer-Verlag, 2003.

[53] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, p. 63, 2015.

[54] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks," *Queue*, vol. 11, pp. 20–40, Dec. 2013.

[55] C.-W. Tseng, Y.-T. Yang, and L.-D. Chou, "An ipv6-enabled software-defined networking architecture," in *Proceedings of the 2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–3, IEEE, 2013.

[56] C.-W. Tseng, S.-J. Chen, Y.-T. Yang, and L.-D. Chou, "Ipv6 operations and deployment scenarios over sdn," in *Proceedings of the 2014 16th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–6, IEEE, 2014.

[57] K. Govindarajan, S. Setapa, K. C. Meng, and H. Ong, "Interoperability issue between ipv4 and ipv6 in openflow enabled network: Ipv4 and ipv6 transaction flow traffic," in *Proceedings of the 2014 International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*, pp. 58–63, IEEE, 2014.

[58] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 3–14, Aug. 2013.

[59] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation: A slice abstraction for software-defined networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pp. 79–84, ACM, 2012.

[60] R. V. Nunes, R. L. Pontes, and D. Guedes, "Virtualized network isolation using software defined networks," in *Proceedings of the 2013 IEEE 38th Conference on Local Computer Networks (LCN)*, pp. 683–686, IEEE, 2013.

[61] M. Antikainen, T. Aura, and M. Särelä, "Spook in your network: Attacking an SDN with a compromised openflow switch," in *Proceedings of the Secure IT Systems - 19th Nordic Conference, NordSec 2014*, pp. 229–244, 2014.

[62] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: State distribution trade-offs in software defined networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, (New York, NY, USA), pp. 1–6, ACM, 2012.

[63] O. Blial, M. B. Mamoun, and R. Benaini, "An overview on sdn architectures with multiple controllers," *Journal of Computer Networks and Communications*, vol. 2016, 2016.

[64] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "Balanceflow: Controller load balancing for openflow networks," in *Proceedings of the 2012 IEEE*

2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), IEEE, 2012.

[65] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, "Consistent updates for software-defined networks: Change you can believe in!," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pp. 7:1–7:6, ACM, 2011.

[66] L. Schiff, S. Schmid, and P. Kuznetsov, "In-band synchronization for distributed sdn control planes," *SIGCOMM Comput. Commun. Rev.*, vol. 46, pp. 37–43, Jan. 2016.

[67] G. Yao, J. Bi, and L. Guo, "On the cascading failures of multi-controllers in software defined networks," in *Proceedings of the 2013 21st IEEE International Conference on Network Protocols (ICNP)*, IEEE, 2013.

[68] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and D. R. Cheriton, "Policycop: An autonomic qos policy enforcement framework for software defined networks," in *Proceedings of IEEE SDN for Future Networks and Services (SDN4FNS), 11-13 Nov. 2013, Trento*, pp. 1–7, IEEE, 2013.

[69] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, pp. 114–119, 2013.

[70] N. Foster, A. Guha, M. Reitblatt, A. Story, M. J. Freedman, N. P. Katta, C. Monsanto, J. Reich, J. Rexford, C. Schlesinger, D. Walker, and R. Harrison, "Languages for software-defined networks," *IEEE Communications Magazine*, vol. 51, pp. 128–134, 2013.

[71] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson, "Fresco: Modular composable security services for software-defined networks.," in *NDSS*, The Internet Society, 2013.

[72] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," *Computer Networks*, vol. 62, pp. 122–136, 2013.

[73] B. Nunes, M. Mendonca, N. Xuan-Nam, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 1617–1634, 2014.

[74] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 493–512, 2014.

[75] S. Song, L. Ling, and C. Manikopoulo, "Flow-based statistical aggregation schemes for network anomaly detection," in *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, 2006. ICNSC '06*, IEEE, April 2006.

[76] R. de O. Schmidt, L. Hendriks, A. Pras, and R. van der Pol, "Openflow-based link dimensioning," in *Innovating the Network for Data Intensive Science*

Workshop, INDIS 2014, SCinet, November 2014.

[77] J. Mattes, "Traffic measurement on openflow-enabled switches," 2012.

[78] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *Proceedings of the 2013 IEEE Military Communications Conference, MILCOM 2013*, IEEE, 2013.

[79] N. Handigol, S. Seetharaman, M. Flajslik, R. Johari, and N. Mckeown, "Aster*x: Load-balancing as a network primitive," in *Proceedings of the 9th GENI Engineering Conference*, 2010.

[80] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, (New York, NY, USA), pp. 202–208, ACM, 2009.

[81] J.-Y. Jo, Y. Kim, H. J. Chao, and F. L. Merat, "Internet traffic load balancing using dynamic hashing with flow volume," vol. 4865, pp. 154–165, 2002.

[82] V. Sreenivas, M. Prathap, and M. Kemal, "Load balancing techniques: Major challenge in cloud computing - a systematic review," 2014.

[83] B. Radojevic and M. Zagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments," in *Proceedings of the 34th International Convention MIPRO 2011*, IEEE, 2011.

[84] H. Menon, N. Jain, G. Zheng, and L. Kale, "Automated load balancing invocation based on application characteristics," *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, vol. 0, pp. 373–381, 2012.

[85] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari, "Plug-n-Serve: Load-balancing web traffic using OpenFlow," 2009.

[86] S. Yilmaz, A. M. Tekalp, and B. D. Unluturk, "Video streaming over software defined networks with server load balancing," in *Proceedings of 2015 International Conference on Computing, Networking and Communications (ICNC)*, 2015.

[87] P. Kreuger, O. Görnerup, D. Gillblad, T. Lundborg, D. Corcoran, and A. Ermedahl, "Autonomous load balancing of heterogeneous networks," in *VTC*, 2015.

[88] S. Namal, I. Ahmad, A. Gurtov, and M. Ylianttila, "SDN Based Inter-Technology Load Balancing Leveraged by Flow Admission Control," in *Proceedings of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, pp. 1–5, IEEE, November 2013.

[89] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying nox to the datacenter," in *Proceedings of workshop on Hot Topics in Networks (HotNets-VIII)*, 2010.

[90] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pp. 27–38, ACM,

2013.

[91] G. Cheng, H. Chen, Z. Wang, and S. Chen, "Dha: Distributed decisions on the switch migration toward a scalable sdn control plane," in *Networking*, pp. 1–9, IEEE, 2015.

[92] R. Rajavel, "De-centralized load balancing for the computational grid environment," in *Proceedings of the International Conference on Communication and Computational Intelligence, 2010.*, pp. 419–424, December 2010.

[93] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed sdn controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp. 7–12, ACM, 2013.

[94] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Adaptive Resource Management and Control in Software Defined Networks," vol. 12, pp. 18–33, March 2015.

[95] C.-C. Li and K. Wang, "An SLA-aware Load Balancing Scheme for Cloud Datacenters," in *Proceedings of the 2014 International Conference on Information Networking (ICOIN)*, IEEE, April 2014.

[96] I. Bolodurina, D. Parfenov, and A. Shukhman, "Approach to the effective controlling cloud computing resources in data centers for providing multimedia services," in *Proceedings of the 2015 International Siberian Conference on Control and Communications (SIBCON)*, IEEE, May 2015.

[97] R. Wang, D. Butnariu, and J. Rexford, "Openflow-based server load balancing gone wild," in *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, pp. 12–12, USENIX Association, 2011.

[98] J. Postel, "Internet control message protocol." https://tools.ietf.org/html/rfc792, 1981.

[99] D. C. Plummer, "An ethernet address resolution protocol." https://tools.ietf.org/html/rfc826, 1982.

[100] C. Benvenuti, *Understanding Linux Network Internals*. O'Reilly Media, Inc., 2005.

[101] L. Deri, S. P. A. Netikos, V. D. B. Km, and L. L. Figuretta, "Improving passive packet capture: Beyond device polling," in *Proceedings of SANE 2004*, pp. 85–93, 2004.

[102] Intel, "Intel data plane development kit: Programmer's guide." http://www.intel.com/content/www/us/en/intelligent-systems/intel-technology/intel-dpdk-programmers-guide.html, 2013.

[103] Intel, "Dpdk performance report." http://www.intel.com/content/www/us/en/intelligent-systems/intel-technology/intel-dpdk-programmers-guide.html, 2013.

[104] T. D. Nadeau and K. Gray, *SDN: Software Defined Networks*. O'Reilly Media, Inc., 2013.

[105] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing

software-defined networks," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pp. 1–14, USENIX Association, 2013.

[106] A. Vishnoi, R. Poddar, V. Mann, and S. Bhattacharya, "Effective switch memory management in openflow networks," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, DEBS '14, pp. 177–188, ACM, 2014.

[107] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-defined networking (sdn): Layers and architecture terminology." https://tools.ietf.org/html/rfc7426, 2015.

[108] "Openflow message layer." http://flowgrammable.org/sdn/openflow/message-layer.

[109] S. Central, "Sdn controller comparison." https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/.

[110] R. S. F. Community, "Ryu sdn controller." http://osrg.github.io/ryu/, 2014.

[111] I. S. Association, "Ethertype." http://standards.ieee.org/develop/regauth/ethertype/eth.t
2014.

[112] J. Moy, "Ospf version 2." https://tools.ietf.org/html/rfc2328, 1998.

[113] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[114] M. Cotton and L. Vegoda, "Special use ipv4 addresses." https://tools.ietf.org/html/rfc5735, 2010.

[115] R. Hinden and S. Deering, "Ip version 6 addressing architecture." https://tools.ietf.org/html/rfc4291, 2006.

[116] M. Blanchet, "Special-use ipv6 addresses." https://tools.ietf.org/html/rfc5156, 2008.

[117] A. I. E. Point, "Amsterdam internet exchange statistics." https://ams-ix.net/technical/statistics, 2016.

[118] K. C. Claffy, *Internet Traffic Characterization*. PhD thesis, La Jolla, CA, USA, 1994. UMI Order No. GAX94-32912.

[119] kc claffy, G. Miller, and K. Thompson, "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone," in *Proceedings of the 1998 International Networking Conference (INET) '98*, (Geneva, Switzerland), The Internet Society, Jul 1998.

[120] A. Broido, Y. Hyun, R. Gao, and kc claffy, *Their Share: Diversity and Disparity in IP Traffic*, pp. 113–125. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[121] B. Williamson, *Developing IP Multicast Networks*. Cisco Press, 1999.

[122] M. Cotton, L. Vegoda, and D. Meyer, "Iana guidelines for ipv4 multicast address assignments." https://tools.ietf.org/html/rfc5771, 2014.

[123] R. Droms, "Ipv6 multicast address scopes." https://tools.ietf.org/html/rfc7346, 2014.

[124] J. Abley and K. Lindqvist, "Operation of anycast services." https://tools.ietf.org/html/rfc4786, 2006.

[125] N. Kim, H. Lim, H. shik Park, and M. Kang, "Detection of multicast video flooding attack using the pattern of bandwidth provisioning efficiency," *IEEE Communications Letters*, vol. 14, pp. 1170–1172, Jan. 2011.

[126] P. Srisuresh and M. Holdrege, "Ip network address translator (nat) terminology and considerations." https://tools.ietf.org/html/rfc2663, 1999.

[127] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)." https://tools.ietf.org/html/rfc4271, 2006.

[128] S. T. Chou, A. Stavrou, J. Ioannidis, and A. D. Keromytis, "gore: Routing-assisted defense against ddos attacks," in *Information Security* (J. Zhou, J. Lopez, R. Deng, and F. Bao, eds.), vol. 3650 of *Lecture Notes in Computer Science*, pp. 179–193, Springer Berlin Heidelberg, 2005.

[129] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pp. 254–265, ACM, 2011.

[130] T. M. Thomas, *OSPF Network Design Solutions 2nd Edition*. Cisco Press, 2003.

[131] VMware, "Vmware virtual networking concepts." http://www.vmware.com/files/pdf/virtual_networking_concepts.pdf, 2015.

[132] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 29–38, Jan. 2006.

[133] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "Dnssec and its potential for ddos attacks - a comprehensive measurement study," in *Proceedings of the Internet Measurement Conference 2014*, (Vancouver, BC, Canada), ACM Press, 2014.

[134] "Tcpreply." http://tcpreplay.appneta.com/, 2015.

[135] P. R. for the Defense of Infrastructure Against Cyber Threats (PREDICT), "Predict data repository." https://www.predict.org/Default.aspx?tabid=169, 2015.

[136] "Bonesi - the ddos botnet simulator." https://code.google.com/p/bonesi/, 2015.

[137] O. Joldzic, Z. Djuric, and P. Vuletic, "A transparent and scalable anomaly-based dos detection method," *Computer Networks*, vol. 104, pp. 27–42, 2016.

[138] M. Chatterjee and S. Setua, "A New Clustered Load Balancing Approach for Distributed Systems," in *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*, IEEE, March 2015.

[139] Zehua Guo and Mu Su and Yang Xu and Zhemin Duan and Luo Wang and Shufeng Hui and H. Jonathan Chao, "Improving the Performance of Load Balancing in Software-Defined Networks Through Load Variance-Based Synchronization," *Computer Networks*, vol. 68, pp. 95–109, 2014. Communications and Networking in the Cloud.

[140] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch, and D. McPherson, "Dissemination of flow specification rules." https://tools.ietf.org/html/rfc5575, 2009.

[141] J. Haas, "Clarification of the flowspec redirect extended community." https://tools.ietf.org/html/rfc7674, 2015.

Biografija

Ognjen Joldžić je rođen 14.08.1986. u Banjoj Luci, gdje je završio i osnovnu školu 2000. godine, te Gimnaziju 2004. godine. Školovanje je nastavio na Elektrotehničkom fakultetu Univerziteta u Banjoj Luci, smjer Računarstvo i informatika. Diplomirao je 2008. godine, a 2010. godine je, na istom fakultetu, završio i II ciklus studija. Od 2010. godine je zaposlen kao saradnik na Elektrotehničkom fakultetu, prvo u naučnom zvanju asistenta, a od 2014. godine u zvanju višeg asistenta.

Изјава 1

ИЗЈАВА О АУТОРСТВУ

Изјављујем
да је докторска дисертација

Наслов рада Адаптивни систем за детекцију DDoS напада у рачунарским мрежама

Наслов рада на енглеском језику Adaptive System for DDoS Attack Detection in Computer Networks

☒ резултат сопственог истраживачког рада,
☒ да докторска дисертација, у цјелини или у дијеловима, није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
☒ да су резултати коректно наведени и
☒ да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

У Бањој Луци, 29.05.2017. године

Потпис докторанта

Изјава 2

Изјава којом се овлашћује Универзитет у Бањој Луци
да докторску дисертацију учини јавно доступном

Овлашћујем Универзитет у Бањој Луци да моју докторску дисертацију под насловом

Адаптивни систем за детекцију DDoS напада у рачунарским мрежама

која је моје ауторско дјело, учини јавно доступном.

Докторску дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у дигитални репозиторијум Универзитета у Бањој Луци могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (*Creative Commons*) за коју сам се одлучио/ла.

1. Ауторство
2. Ауторство – некомерцијално
3. Ауторство – некомерцијално – без прераде
4. Ауторство – некомерцијално – дијелити под истим условима
5. Ауторство – без прераде
6. Ауторство – дијелити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

У Бањој Луци, 29.05.2017. godine

Потпис докторанта

Изјава 3

Изјава о идентичности штампане и електронске верзије докторске дисертације

Име и презиме аутора Огњен Јолџић

Наслов рада Адаптивни систем за детекцију DDoS напада у рачунарским мрежама

Ментор проф. др Зоран Ђурић

Изјављујем да је штампана верзија моје докторске дисертације идентична електронској верзији коју сам предао/ла за дигитални репозиторијум Универзитета у Бањој Луци.

У Бањој Луци, 29.05.2017. године

Потпис докторанта