TCP performanse prilikom prenosa WWW/HTTP saobraćaja

Goran Martić

***Sadržaj* — Saobraćaj baziran na TCP protokolu danas čini više od 80% saobraćaja koji se prenosi na Internet globalnoj računarskoj mreži. Male TCP konekcije, iako količinski ne čine većinu TCP saobraćaja, su veoma zastupljene u svakodnevnom radu. Koriste se prilikom prenosa WWW surf saobraćaja, udaljenog pristupa računarima, igranja preko mreže itd. U radu su testirane performanse najkorišćenijih TCP algoritama u kontekstu malih konekcija, konkretno prilikom prenosa WWW/HTTP saobraćaja. Cilj teze je postizanje što veće relevantnosti i praktičnosti rezultata. U skladu sa tim, izvršeno je snimanje i karakterizacija ostvarenog saobraćaja prilikom tipičnog surfa Internetom na osnovu čega je definisan simulacioni model. Simulacije su vršene koristeći produkcioni TCP stek GNU/Linux operativnog sistema. Hibridna platforma koja omogućava kontrolisanost eksperimenta i korišćenje produkcionog TCP steka je ostvarena koristeći savemeni mrežni simulator NS3.**

***Ključne reči* — TCP kontrola zagušenja, thin flow, NS3.**

I. UVOD

Saobraćaj baziran na TCP protokolu danas čini više od 80% saobraćaja koji se prenosi na Internet globalnoj računarskoj mreži [1]. Na osnovu veličine i trajanja konekcije postoje dva tipa TCP (*Transmission Control Protocol*) saobraćaja:

1. Velike, dugačke konekcije – u engleskoj literaturi poznate kao *thick flow*. Koriste se prilikom prenosa velike količine podataka, tipično prilikom preuzimanja sadržaja sa Interneta.
2. Male, kratke konekcije – u engleskoj literaturi poznate kao *thin flow*. Koriste se prilikom WWW (*World Wide Web*) surfovanja, interaktivnih alikacija (telnet/SSH (*Secure Shell*), udaljena administracija računara, igranje preko mreže), itd.

G. Martić, Nil Data Communications, Metohijska 23/9, 11000 Beograd, Srbija; (e-mail: gmartic@nil.co.rs).

Glavnica istraživačkih napora u evoluciji TCP-a je bila u kontekstu prvog tipa saobraćaja. Konstantno povećanje kapaciteta Internet linkova pratio je i brz razvoj mreža za pristup, čime su krajnjem korisniku omogućene brzine reda nekoliko desetina Mbps, pa čak i stotina u nekim delovima sveta. Cilj je bio usavršiti TCP algoritme kako bi se ispratio trend vrtoglavog rasta brzina. Performanse drugog tipa saobraćaja nisu tretirane skoro uopšte, ili su tretirane kao sekundarni cilj ili nuspojava neke nove tehnike.

Cilj rada je testiranje performansi postojećih TCP algoritama u kontekstu malih/kratkih konekcija, konkretno prilikom prenosa WWW/HTTP (*Hyper Text Transfer Protocol*) saobraćaja koji se ostvaruje prilikom surfovanja Internetom. Ideja je da se testiranje izvrši sa tipičnim svakodnevnim scenariom, u najrealnijim mogućim uslovima u cilju postizanja što veće relevantnosti i praktičnosti rezultata. Na osnovu dobijenih rezultata biće date preporuke za podešavanje parametara TCP steka operativnog sistema. Karakteristike analize prezentovane u radu su sledeće:

1. Korišćenje produkcionog TCP steka. U radu se koristi TCP stek GNU/Linux operativnog sistema (u daljem tekstu Linux). Većina drugih simulacija koristi generički TCP stek samog simulacionog softvera.
2. Testiranje najpopularnijih TCP algoritama kontrole zagušenja (CC - *Congestion Control*). I u ovom slučaju testiranje je izvršeno korišćenjem produkcionih algoritama koji su implementirani pod Linux-om.
3. Za sve izvedene testove u radu korišćen je realan scenario. Naime, izvršeno je snimanje saobraćaja ostvarenog prilikom surfovanja Internetom. Poreg toga, izvršena je karakterizacija ostvarenog saobraćaja, na osnovu koje su definisani parametri simulacije.
4. Kontrolisani uslovi. Koristi se savremeni mrežni simulator NS3 koji omogućava sve pomenuto.

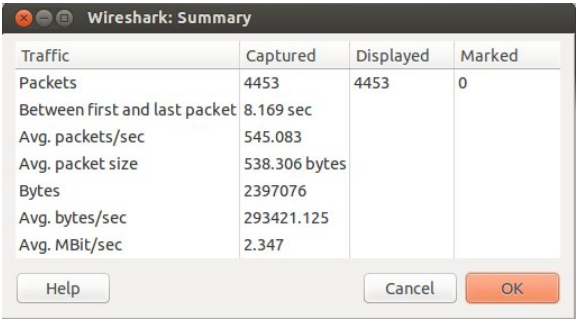
U drugom delu je izvršena analiza i karakterizacija tipičnog surf saobraćaja. U trećem delu je na osnovu podataka dobijenih u drugom delu definisan simulacioni model i parametri simulacije. U četvrtom delu je data analiza rezultata simulacije. U petom delu su dati zaključci na osnovu analiziranih rezultata simulacije. Dalje, date su preporuke za modifikaciju TCP parametara Linux operativnog sistema radi optimizacije prenosa malih konekcija.

II. ANALIZA TIPIČNOG SURF SAOBRAĆAJA

Kao primer za analizu saobraćaja je izabrana popularna web lokacija b92.net. Cilj je da se stekne uvid u dinamiku saobraćaja i saznaju karakteristike kao što su broj TCP sesija, količina podataka, veličina paketa i slično.

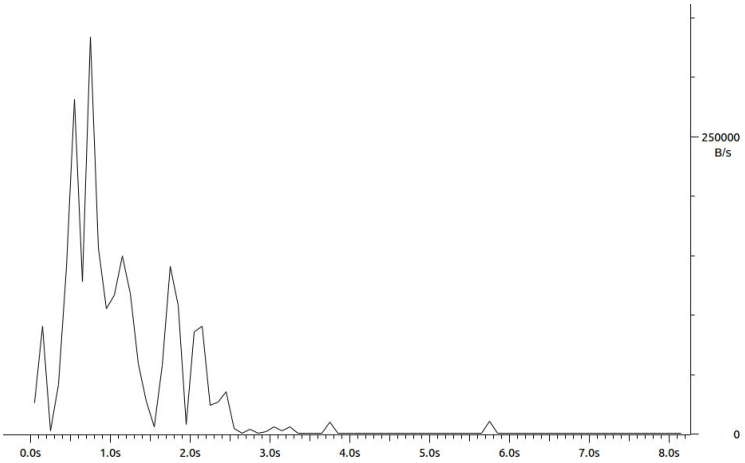
Sa klijentske strane se koristi pretraživač Google Chrome v28 pod operativnim sistemom Linux Ubuntu 13.04, verzija kernela 3.8.0-30. Za snimanje saobraćaja se koristi alat tcpdump 4.3.0 koji snima pakete u pcap formatu pomoću biblioteke libcap 1.3.0. Za parsiranje i analizu pcap datoteka se koriste alati: Wireshark 1.8.2, tcptrace 6.6.7. Za iscrtavanje grafikona na osnovu pcap datoteka se koristi alat captcp 1.4.

Ključne informacije iz Sl. 1 su ukupna količina prenetih podatka (~2,4MB), broj paketa (4453) i ukupno vreme transfera (8,169s).



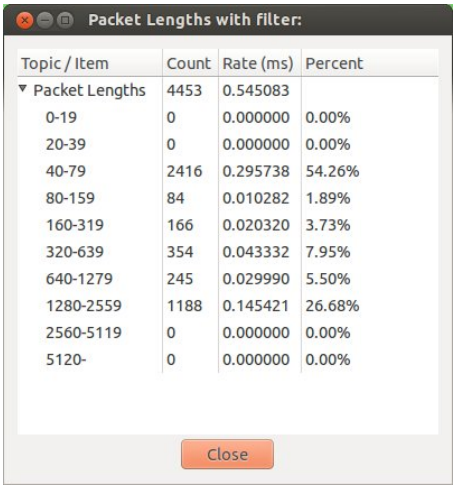
Sl. 1. Statistika ostvarenog saobraćaja prilikom otvaranja B92.net stranice

Na Sl. 2 je prikazana agregirana brzina prenosa svih sesija tokom perioda učitavanja stranice.



Sl. 2. Agregirana brzina prenosa svih TCP sesija prilikom otvaranja B92.net stranice

Informacija o prosečnoj veličini paketa (~538B) sa Sl. 1 nije od velikog značaja jer uglavnom predstavlja kombinaciju dva dominanta tipa paketa: velikih paketa (~1500B) koji nose podatke, i malih (~60B) ACK paketa. Na Sl. 3 je data raspodela veličine paketa u odnosu na ukupan broj paketa.

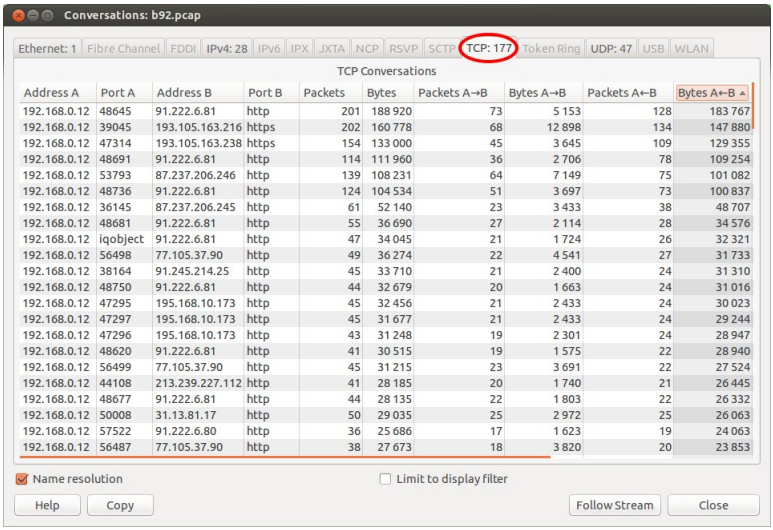


Sl. 3. Raspodela veličine paketa prilikom otvaranja B92.net stranice

Kategorije paketa po veličini:

1. **40-79B: 54,26%**. Vidimo da su malo preko polovine (~54%) paketi veličine 40-79B - radi se o ACK paketima i zanemarljivo malom broju DNS zahteva (DNS standard query) čina je veličina manja od 80B.
2. **1280-2559B: ~26,68%**.Paketi podataka maksimalne veličine - radi se o paketima koji nose korisne informacije, tj. objekte (slike, animacije, tekst) sa sajta.
3. **320-639B: 7,95%**. Većinu ovih paketa čine HTTP zahtevi (HTTP-GET komanda). Manji udeo imaju završni paketi sesija koje nose korisne podatke (HTTP-OK komanda). U ovom opsegu se javi i poneki neuobičajeno veliki paketi DNS odgovora.
4. **640-1279B: 5,50%**. Većinu ovih paketa čine završni paketi sesija koje nose korisne podatke. Manji udeo imaju HTTP zahtevi. U ovom opsegu se obično nalaze i fragmentirani TCP segmenti.
5. **160-319B: 3,73%**. U ovom opsegu se nalaze paketi standardni paketi DNS odgovora, završni paketi sesija koje nose korisne podatke, i poslednji fragmenti podeljenih TCP segmenata.

Na Sl. 4 vidimo da su podaci preneti pomoću 177 TCP sesija, što deluje prilično mnogo imajući u vidu da je preneto samo ~2,4 MB podataka. Razlog leži u načinu implementacije savremenih pretraživača - otvaraju više paralelnih sesija kojima se preuzimaju objekti sa iste stranice kako bi se skratilo vreme učitavanja stranice. Često se objekti ne opslužuju sa lokalnog servera, nego predstavljaju samo referencu na druge servere sa kojih klijent mora da preuzme podatke.



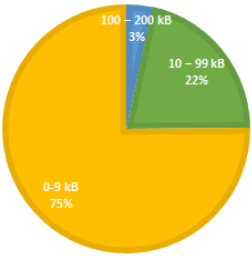
Sl. 4. B92.net: Lista TCP sesija

Trenutno aktuelna verzija HTTP protokola (HTTP v1.1) podrazumevano koristi perzistente TCP konekcije - više HTTP objekata se prenosi kroz istu TCP konekciju. Perzistentni HTTP smanjuje količinu posla koja je asocirana sa uspostavom TCP konekcije (procesorski ciklusi, memorija, vreme potrebno za uspostavu). Ova činjenica posebno dobija na značaju ako je generalno količina podataka mala, što je upravo slučaj sa HTTP objektima. Perzistenti HTTP se postiže održavanjem TCP konekcije pomoću kontrolnih paketa (engl. *keepalive*). Perzistente TCP konekcije se prepoznaju po pojavi više od jednog para HTTP-GET+HTTP-OK komandi.

U tabeli 1 je data raspodela veličine TCP sesija, odnosno raspodela u odnosu na količinu prenetih podataka po TCP sesiji.

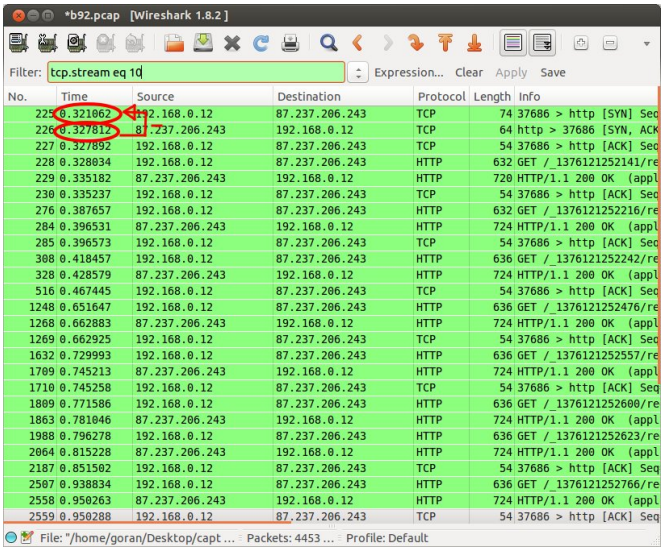
TABELA 1: B92.NET: RASPODELA VELIČINE TCP SESIJA

Kategorija konekcija	***Broj sesija***	***Procenat od ukupnog broja sesija***	***Količina podataka po sesiji***
Velike konekcije	6	3,39 %	100-200 kB
Srednje konekcije	38	21,47 %	10-99 kB
Male konekcije	133	75,14 %	0-9 kB



Sl. 5. B92.net: Raspodela veličine TCP sesija

Na osnovu razmene SYN paketa na početku sesije se može dati gruba procena RTT-a (*Round Trip Time*). Za sesiju prikazanu na Sl. 6 možemo da vidimo da je RTT = 0,327182 - 0,321062 = 0,00612 s (6,12 ms).



Sl. 6. B92.net: RTT proračun

Dalje, zanimljivo je primetiti da klijent generiše ACK paket za svaki primljeni paket (pogledati Sl. 6). Po TCP specifikaciji [2] kako bi se smanjio broj ACK paketa dovoljno je slati ACK za svaki drugi primljeni segment. U Linux-u je implementirana *quick-ack* optimizacija koja šalje ACK nakon

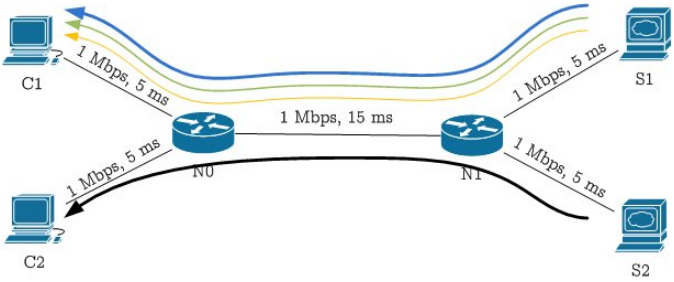
svakog primljenog paketa ali samo u *slow-start* fazi, onda kada su ACK paketi (odnosno informacija o stanju na mreži) i najpotrebniji predajniku.

III. SIMULACIONI MODEL I METODOLOGIJA

Za modelovanje i simulaciju koristi se program NS3 v3.16. NS3 je mrežni simulator diskretnih događaja. NS3 predstavlja nastavak izuzetno popularnog i sada već vremešnog NS2 simulatora. Međutim, NS3 je novi proizvod, kompletno „napisan od nule“ u programskom jeziku C++. Podržani jezici za pisanje korisničkih modela su C++ i Python.

Jedna od izuzetno zanimljivih mogućnosti NS3 simulatora je NSC (*Network Simulation Cradle*), u bukvalnom prevodu kolevka za mrežnu simulaciju. Radi se o mogućnosti NS3-a da se u modelu koristi pravi Linux mrežni stek. Do sada su mrežni simulatori implementirali svoje verzije mrežnog steka i verzije TCP CC algoritama. Ako ste želeli da izvršite simulaciju koristeći realne sisteme, morali ste da napravite fizičko testno okruženje. Ovaj pristup je skup, nefleksibilan, oduzima puno vremena, i ima ograničenu primenu. NS3 NSC omogućava da u mrežnim čvorovima modela koje pišete, direktno koristite Linux mrežni stek (uključujući i TCP CC algoritme), a ne generički NS3 pandan. Ovim se dobija najbolje iz oba sveta: „realna“ simulacija fizičkog testnog okruženja, i praktičnost i fleksibilnost mrežnog simulatora.

Sa Sl. 7 se vidi da se model sastoji od dva klijenta (C1, C2), dva servera (S1, S2), i dva rutera (N0, N1). Propusna moć svih linkova je 1Mbps. Kašnjenja linkova su prikazana na Sl. 7. RTT između klijenta i servera je dakle 50ms. MTU (*Maximum Transmission Unit*) svih linkova je 1500B. Veličina paketa je 1450B. Veličina reda čekanja na mrežnim interfejsima je 4 paketa, radi se o FIFO redu, tj. koristi se *drop-tail* logika odbacivanja paketa.



Sl. 7. Simulacioni model

Koristi se Linux TCP mrežni stek iz verzije kernela 2.6.26. On je instaliran na sve čvorove u mreži. Sledi primer instalacije Linux mrežnog steka u NS3 čvor C1:

```
std::string nscStack = "liblinux2.6.26.so";
internet_nsc.SetTcp("ns3::NscTcpL4Protocol","Library",Str
ingValue (nscStack));
  internet_nsc.Install (node_c1n0.Get (0));
```

NS3 NSC podržava Linux *sysctl* mehanizam preko svojih biblioteka - potrebno je pozvati odgovarajuću funkciju i proslediti joj dva argumenta: naziv parametra i novu vrednost parametra. To se postiže na sledeći način:

```
std::string tcpCong = "reno";
Config::Set("/NodeList/*/$ns3::Ns3NscStack<linux2.6.26>/n
et.ipv4.tcp_congestion_control", StringValue (tcpCong));
```

Analogno se upravlja i drugim TCP parametrima. Spisak podržanih parametara zavisi od Linux-a, ne od NS3 NSC. Na primer, aktivacija SACK opcije se postiže sa komandom:

```
Config::Set("/NodeList/*/$ns3::Ns3NscStack<linux2.6.26>/n
et.ipv4.tcp_sack", StringValue ("1"));
```

Kontrolom četiri parametra (CC algoritam: NewReno [3] i Cubic [4], SACK [5], FACK [6], ABC [7]), i njihovim kombinacijama postiže se izolacija različitih TCP funkcionalnosti i omogućava njihovo poređenje. U tabeli 2 su dati CC algoritmi koji će biti testirani, i vrednosti parametara pomoću kojih su oni realizovani u simulacionom modelu.

TABELA 2: VERZIJE TCP CC ALGORITAMA ZA SIMULACIJU

CC algoritam za simulaciju	*CC algoritam*	*SACK*	*FACK*	*ABC*
NewReno	NewReno	0	0	0
NewReno-SACK	NewReno	1	0	0
NewReno-FACK	NewReno	1	1	0
NewReno-ABC	NewReno	0	0	1
NewReno-SACK-ABC	NewReno	1	0	1
NewReno-FACK-ABC	NewReno	1	1	1
Cubic	Cubic	0	0	0
Cubic-SACK	Cubic	1	0	0
Cubic-FACK	Cubic	1	1	0
Cubic-ABC	Cubic	0	0	1
Cubic-SACK-ABC	Cubic	1	0	1
Cubic-FACK-ABC	Cubic	1	1	1

Parametri simuliranog saobraćaja su dati u tabeli 3, i definisani su na osnovu snimljenog realnog saobraćaja datog u tabeli 1. Raspodela veličine TCP sesija varira od sajta do sajta. Iz tog razlog relacija između realnog i simuliranog saobaraćaja nije direktna "jedan na jedan", već je raspodela veličine TCP sesija realnog saobraćaja iskorišćena kao osnova za definisanje simuliranog saobraćaja.

TABELA 3: SIMULACIONI MODEL: PARAMETRI SAOBRAĆAJA

Kategorija konekcija	*Broj sesija*	*Procenat od ukupnog broja sesija*	*Količina podataka po sesiji*	*Tok u modelu*
Velike kon.	2	11,17 %	120 kB	S1->C1
Srednje kon.	5	29,41 %	21 kB	S1->C1
Male kon.	10	58,82 %	4 kB	S1->C1
Pozadinska kon.	1	/	15 MB	S2->C2

Definisana su dva toka saobraćaja: S1->C1 (gornji tok na Sl. 7) i S2->C2 (donji tok na Sl. 7). Tok S1->C1 je posmatrani tok. Sastoji se od 2 velike (120kB), 5 srednjih (21kB), i 10 malih (4kB) sesija. Tok S2->C2 se sastoji od jedne pozadinske sesije koja prenosi veću količinu podataka (15MB). Svrha ovog toka je da emulira realno okruženje krajnjeg korisnika gde se tipično kroz deljeni ruter kod samog korisnika (čvor N0) prenose paralelno veliki (npr. preuzimanje fajla) i mali strimovi (npr. surfovanje). Takođe, pozadinska

sesija (S2->C2) će se nadmetati za resurse (propusni opseg) sa posmatranim S1->C1 tokom, i omogućiće zbog relativno malih redova čekanja u čvorovima (4 paketa) generisanje gubitaka, tj. poslužiće kao okidač za pokretanje rada TCP CC algoritama.

Rezultat izvršavanja simulacije su pcap datoteke, jedna datoteka po mrežnom interfejsu. Za potrebe analize, posmatra se interfejs koji je najbliži izvoru podataka - mrežni interfejs čvora S1.

IV. ANALIZA REZULTATA SIMULACIJE

Parametri koji će biti posmatrani su vreme prenosa, procenat retransmitovanih paketa, i brzina prenosa. Vreme prenosa je najrelevantniji od pomenuta tri parametra, iz razloga što smo posmatramo performanse malih/kratkih TCP sesija. Zanima nas koji TCP CC algoritam za najkraće vreme prikazuje web stranicu korisniku. Da za cilj imamo poređenje performansi TCP CC algoritama pri prenosu velikih fajlova, brzina prenosa bi nam bila najrelevantniji parametar. Procenat retransmitovanih paketa nam služi više kao indikativan parametar - govori koliko je agresivan određeni CC algoritam, a ne koliko je performantan. Isto važi i za brzinu prenosa.

Sledi komparativno poređenje rezultata simulacije po kategorijama. Za svaku kategoriju biće prikazani:

1. Tabela - Komparativno poređenje algoritama, sortirano po rastućem vremenu prenosa.
2. Grafik - Poređenje po algoritmu, poređenje NewReno i Cubic algoritama sa ekvivalentnim opcijama.
3. Grafik - Agregirano poređenje svih varijanti NewReno naspram svih varijanti Cubic algoritama.

A. Komparativno poređenje - Velike konekcije

Iz tabele 4 vidimo da je razlika između algoritama minimalna - svi algoritmi su slični po performantnosti prilikom prenosa velikih konekcija. Jedino je NewReno-FACK-ABC vidno sporiji od ostalih algoritama: Cubic-FACK (prvi algoritam) je za 33,84% efikasniji od NewReno-FACK-ABC (poslednji). Ne računajući njega, Cubic-FACK je za samo 17,93% efikasniji od NewReno-SACK-ABC (pretposlednji algoritam).

TABELA 4: REZULTATI SIMULACIJE - KOMPARATIVNO POREĐENJE: VELIKE KONEKCIJE

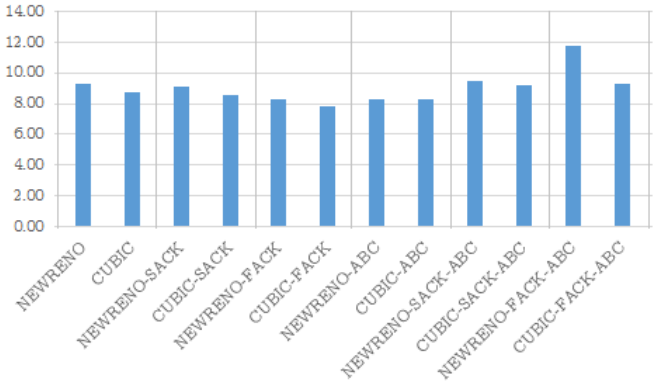
CC algoritam	*Vreme [sec]*	*Retransmisije [%]*	*Brzina [kbps]*
Cubic-FACK	7,79	7,14	255,21
NewReno-ABC	8,27	12,44	251,89
Cubic-ABC	8,28	12,08	241,44
NewReno-FACK	8,32	9,66	244,70
Cubic-SACK	8,59	7,80	234,98
Cubic	8,71	9,71	241,69
NewReno-SACK	9,06	10,48	227,88
Cubic-SACK-ABC	9,17	11,76	221,46
Cubic-FACK-ABC	9,27	15,10	225,59
NewReno	9,32	12,50	234,01
NewReno-SACK-ABC	9,48	8,99	210,89
NewReno-FACK-ABC	11,76	15,58	182,18

Vredi zapaziti da su prva četiri algoritma imaju ili FACK ili ABC opciju, što daje na značaju ovim opcijama kod prenosa velikih konekcija. Kada uporedimo performanse ovih algoritama sa performansama algoritama bez pomenutih opcija, npr. Cubic-FACK (7,78s) u odnosu na Cubic (8,71s), vidimo da je u svim slučajevima prva grupa algoritama ostvarila bolje rezultate.

Slično važi i za „čistu“ SACK opciju, tj. NewReno-SACK (9,06s) je bolji od NewReno (9,3278s) i Cubic-SACK (8,59s) je bolji od Cubic (8,71s).

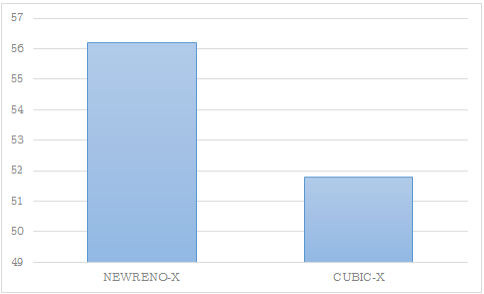
Dakle, kod prenosa velikih konekcija opcije SACK, FACK, i ABC daju očekivana pobošljanja kod oba algoritma.

Generalno, kada poredimo NewReno i Cubic sa ekvivalentnim opcijama (pogledati Sl. 8), vidimo da je u svim slučajevima Cubic performantniji, osim u slučaju ABC opcije gde je razlika zanemarljiva (8,28 - 8,27 = 0,01s). Ovo je i očekivano, s obzirom da je Cubic algoritam namenski dizajniran za veliku brzinu prenosa što dolazi do izražaja kod prenosa veće količine podataka.



Sl. 8. Rezultati simulacije - Poređenje po algoritmu: velike konekcije

Konačno, kada uporedimo performanse svih varijanti NewReno naspram svih varijanti Cubic algoritama (pogledati Sl. 9), vidimo da su Cubic algoritmi efikasniji za 7,85% kod prenosa velikih konekcija.



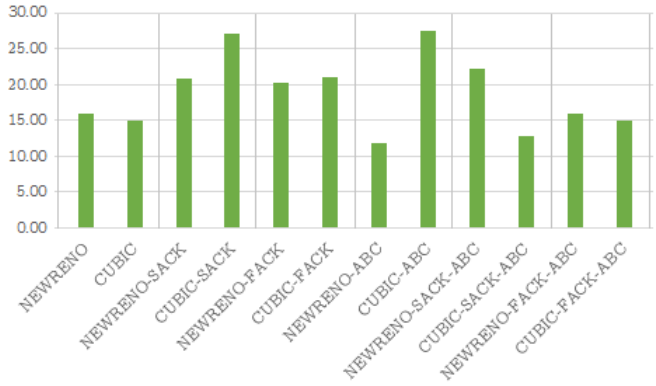
Sl. 9. Rezultati simulacije - Poređenje agregirano: velike konekcije

Kod parametra retransmisije uočavamo duplo veću razliku: 7,14% (Cubic-FACK) u odnosu na 15,58% (NewReno-FACK-ABC). Dakle NewReno-FACK-ABC je retransmitovao skoro duplo više paketa od Cubic-FACK algoritma. Razlog za ovo leži u agresivnoj FACK opciji. Ovo možemo da potvrdimo tako što primetimo da je NewReno-SACK-ABC retransmitovao samo 8,99% paketa. Dodatno, NewReno-ABC koji takođe nema agresivni FACK algoritam je retransmitovao 12,44% - više nego NewReno-SACK-ABC. Razlog je taj što NewReno-ABC nema SACK opciju pomoću koje bio znao koji paketi su stigli do prijemnika, što znači je taj višak retransmisija bio nepotreban.

B. Komparativno poređenje - Srednje konekcije

TABELA 5: REZULTATI SIMULACIJE - KOMPARATIVNO POREĐENJE: SREDNJE KONEKCIJE

CC algoritam	*Vreme [sec]*	*Retransmisije [%]*	*Brzina [kbps]*
NewReno-ABC	11,88	12,82	145,03
Cubic-SACK-ABC	12,83	20,00	82,93
Cubic-FACK-ABC	14,89	16,98	90,22
Cubic	15,05	11,11	82,99
NewReno-FACK-ABC	15,85	23,89	70,15
NewReno	15,93	13,93	81,45
NewReno-FACK	20,18	15,63	56,43
NewReno-SACK	20,82	9,01	54,22
Cubic-FACK	21,05	11,30	51,77
NewReno-SACK-ABC	22,15	13,59	48,09
Cubic-SACK	27,02	14,84	36,78
Cubic-ABC	27,38	15,15	40,44



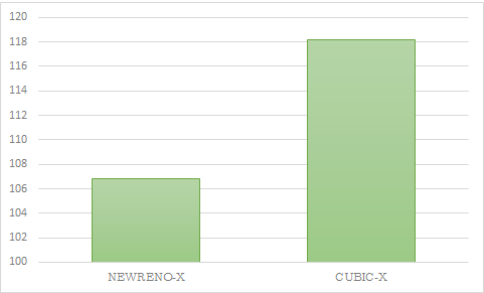
Sl. 10. Rezultati simulacije - Poređenje po algoritmu: srednje konekcije

Razlika u vremenu prenosa srednjih konekcija između prvog i poslednjeg CC algoritma je izraženija nego kod velikih konekcija: NewReno-ABC je 2,3 puta efikasniji od Cubic-ABC.

Kod detaljnijeg proučavanja tabele 5 uočava se zanimljiv obrazac za oba algoritma. Naime, NewReno ≤ NewReno-FACK ≤ NewReno-SACK, i Cubic ≤ Cubic-FACK ≤ Cubic-SACK. Drugim rečima, kod prenosa srednjih konekcija SACK opcija je usporila prenos, a FACK opcija uspeva donekle da poboljša situaciju.

Dalje, zanimljivo je primetiti da ABC opcija unosi drastične promene u performanse oba originalna algoritma: kod NewReno na bolje (NewReno-ABC je prvi), a kod Cubic na lošije (Cubic-ABC je poslednji). NewReno-ABC je za 25,42% efikasniji od NewReno, dok je Cubic-ABC za 45,03% manje efikasan od Cubic algoritma.

Konačno, kada uporedimo performanse svih varijanti NewReno naspram svih varijanti Cubic algoritama (pogledati Sl. 11), vidimo da su NewReno algoritmi efikasniji za 9,65% kod prenosa srednjih konekcija.



Sl. 11. Rezultati simulacije - Poređenje agregirano: srednje konekcije

Razlika u procentu retransmitovanih paketa između prvog i poslednjeg CC algoritma je manje izražena nego kod velikih konekcija: 15,15% (Cubic-ABC) - 12,82% (NewReno-ABC) = 2,33%.

C. *Komparativno poređenje - Male konekcije*

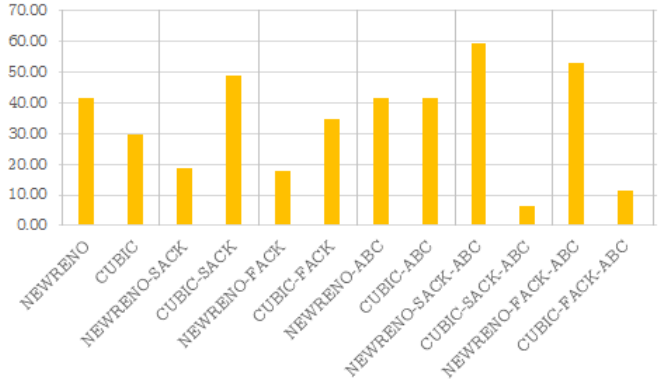
Kod prenosa malih konekcija smo dobili najzanimljivije rezultate.

TABELA 6: REZULTATI SIMULACIJE - KOMPARATIVNO POREĐENJE: MALE KONEKCIJE

CC algoritam	*Vreme [sec]*	*Retransmisije [%]*	*Brzina [kbps]*
Cubic-SACK-ABC	6,18	30,99	78,21
Cubic-FACK-ABC	11,59	29,58	65,28
NewReno-FACK	17,85	23,40	121,70
NewReno-SACK	18,67	24,24	119,05
Cubic	29,66	27,68	78,42
Cubic-FACK	34,76	23,16	125,91
NewReno-ABC	41,50	26,17	151,04
Cubic-ABC	41,54	26,79	116,07
NewReno	41,68	26,42	122,60
Cubic-SACK	48,84	25,00	107,94
NewReno-FACK-ABC	53,02	24,32	34,22
NewReno-SACK-ABC	59,12	29,33	31,19

Razlika u vremenu prenosa između prvog i poslednjeg CC algoritma je ubedljivo najizraženija kod prenosa malih konekcija: Cubic-SACK-ABC je čak 9,57 puta efikasniji od NewReno-SACK-ABC.

Generalno, uočava se značajna razlika u ostvarenim rezultatima između prve četiri i ostalih pozicija. Prva četiri algoritma imaju SACK opciju, na osnovu čega zaključujemo da SACK ima važnu ulogu kod prenosa malih konekcija.

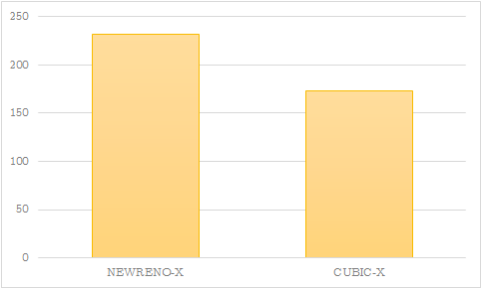


Sl. 12. Rezultati simulacije - Poređenje po algoritmu: male konekcije

Posmatrajući Sl. 12 uočavamo da je klasični Cubic algoritam je bolji od NewReno. Međutim, u kombinaciji sa jednom od SACK/FACK/ABC opcija, NewReno je bolji u sva tri slučaja. Dalje, pomenute opcije poboljšavaju NewReno (što je i očekivano), ali imaju kontraefekat kod Cubic algoritma.

Uočavamo da je ABC opcija, slično kao kod prenosa sredjnih konekcija, opet napravila polarizaciju između NewReno i Cubic algoritama, sada u kombinaciji sa jednom od SACK/FACK opcija ali sa obrnutim rezultatom (Cubic bolji od NewReno).

Konačno, kada uporedimo performanse svih varijanti NewReno naspram svih varijanti Cubic algoritama (pogledati Sl. 13), vidimo da su Cubic algoritmi efikasniji za 25,57% kod prenosa malih konekcija. Mada ovaj rezultat i nije reprezentativan jer su rezultati prva dva algoritma značajno odnela prevagu u korist Cubic algoritama.



Sl. 13. Rezultati simulacije - Poređenje agregirano: male konekcije

Razlika u procentu retransmitovanih paketa između prvog i poslednjeg CC algoritma je najmanje izražena kod prenosa malih konekcija: 30,99% (Cubic-SACK-ABC) - 29,33% (NewReno-SACK-ABC) = 1,66%.

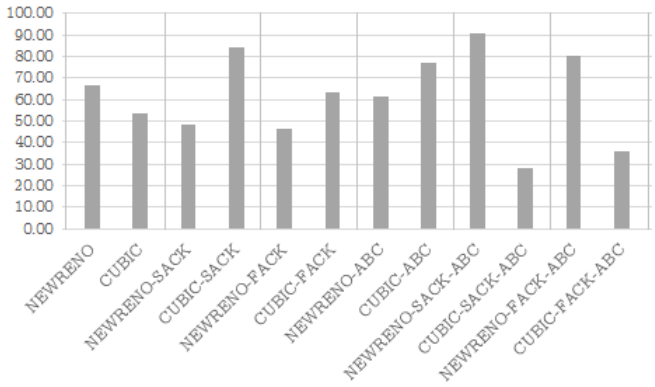
Podatak da svi CC algoritmi kod prenosa malih sesija čine približno jednak procenat retransmisija je izuzetno zanimljiv. Ovo znači da su razlike u vremenu prenosa posledica isključivo implementacije samog CC algoritma. S tim u vezi treba zapaziti da je prva dva mesta zauzeo Cubic algoritam sa SACK i ABC opcijama, i FACK opcijom kao tačkom diferencijacije: Cubic-SACK-ABC 6,18s i Cubic-FACK-ABC 11,59s. Dalje, razlika između trećeg i četvrtog mesta je mala, svega 18,67 - 17,84 = 0,83s. Njih je zauzeo NewReno algoritam sa SACK opcijom, i FACK opcijom kao tačkom diferencijacije: NewReno-FACK: 17,85s i NewReno-SACK: 18,67s. Treće i četvrto mesto su ostvarili znatno bolje rezultate (skoro dva puta) od petog mesta (Cubic: 29,66s).

D. Komparativno poređenje po svim kategorijama

Sledi finalno komparativno poređenje rezultata simulacije po svim kategorijama (pogledati tabelu 7 i Sl. 14).

TABELA 7: REZULTATI SIMULACIJE - KOMPARATIVNO POREĐENJE: SVE KATEGORIJE

CC algoritam	*Vreme [sec]*	*Retransmisije [%]*	*Brzina [kbps]*
Cubic-SACK-ABC	28,18	20,92	127,53
Cubic-FACK-ABC	35,75	20,55	127,03
NewReno-FACK	46,35	16,23	140,94
NewReno-SACK	48,55	14,59	133,72
Cubic	53,42	16,17	134,37
NewReno-ABC	61,65	17,14	182,65
Cubic-FACK	63,59	13,87	144,30
NewReno	66,93	17,62	146,02
Cubic-ABC	77,80	18,00	132,65
NewReno-FACK-ABC	80,63	21,27	95,52
Cubic-SACK	84,45	15,88	126,57
NewReno-SACK-ABC	90,75	17,31	96,72



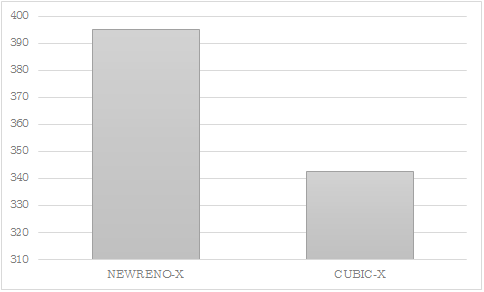
Sl. 14. Rezultati simulacije - Poređenje po algoritmu: sve kategorije

Situacija je slična kao kod rezultata prenosa malih konekcija. Razlog za ovo je razumljiv - videli smo da su upravo kod prenosa malih konekcija ostvarene najveće razlike između algoritama.

Na prva dva mesta odskaču Cubic-SACK-ABC (28,18s) i Cubic-FACK-ABC (35,75s). Treće i četvrto mesto su zauzeli NewReno-FACK (46,35s) i

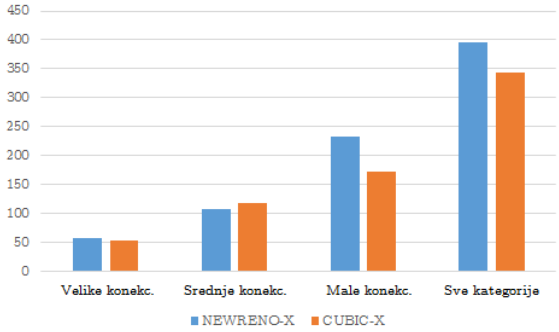
NewReno-SACK (48,55s). Na petom mestu je Cubic (53,42s). Cubic-SACK-ABC (prvi) je 21,17% efikasniji od NewReno-FACK (treći), a 47,25% od Cubic (peti).

Konačno, kada uporedimo performanse svih varijanti NewReno naspram svih varijanti Cubic algoritama (pogledati Sl. 15), vidimo da su Cubic algoritmi efikasniji za 13,24%.



Sl. 15. Rezultati simulacije - Poređenje agregirano: sve kategorije

Na Sl. 16 je dato sumarno poređenje po kategorijama NewReno naspram Cubic varijanti algoritama.



Sl. 16. Rezultati simulacije - Poređenje agregirano: sumarno

Razlika u retransmisijama između prvog i poslednjeg algoritma je mala: 20,92% (Cubic-SACK-ABC) - 17,31% (NewReno-SACK-ABC) = 3,61%.

V. ZAKLJUČCI I PREPORUKE

Prethodno opisana testiranja i analize daju sledeće rezultate:

1. Klasični Cubic je efikasniji od klasičnog NewReno algoritma, za 20,19%.
2. Ukoliko se koristi jedna od opcija SACK/FACK/ABC, NewReno je efikasniji od Cubic algoritma, i to za 42,51%, 27,11%, 20,14%, respektivno.
3. Cubic algoritam u kombinaciji sa SACK-ABC ili FACK-ABC opcijama je ostvario ubedljivo najbolje rezultate - efikasniji su od trećeplasiranog algoritma (NewReno-FACK) za 39,2%, odnosno 22,87%.

Na osnovu dobijenih rezultata i poznavanja podrazumevanih podešavanja popularnih TCP stekova, radi smanjivanja vremena učitavanja tipičnih WWW stranica mogu se definisati sledeće preporuke:

1. Obavezno uključiti SACK opciju, FACK opciono. FACK opcija u svim slučajevima donosi koristi, ali ne velike.
2. Ako se koristi Cubic uključiti ABC tehniku. Ako se koristi NewReno isključiti ABC tehniku.
3. Podrazumevano ponašanje Linux operativnog sistema je korišćenje Cubic algoritma, sa uključenim SACK i FACK opcijama. Ovo ne rezultira optimalnim performansama. Radi optimizacije može se uraditi jedno od sledećeg:
    a. Ostaviti Cubic algoritam, uključiti ABC opciju, i isključiti SACK opciju. Povećava se efikasnost za 55,68%.
    b. Ostaviti Cubic algoritam, i uključiti ABC opciju. Povećava se efikasnost za 43,78%.
    c. Promeniti algoritam na NewReno. Povećava se efikasnost za 27,11%.

Ove preporuke treba implementirati na strani koja šalje podatke, odnosno na serverskoj strani. U zavisnosti od operativnog sistema koji se koristi, zavise i podrazumevana podešavanja, kao i mogućnosti njihove modifikacije.

LITERATURA

[1] M. Fomenkov, K. Keys, D. Moore, K. Claffy, Longitudinal study of Internet traffic from 1998-2003, Trinity College Dublin, 2004, pp. 5.
[2] R. Braden, Requirements for Internet Hosts -- Communication Layers (RFC1122), IETF, Oktobar 1989.
[3] T. Henderson, S. Floyd, A. Gurtov, Y. Nishida, The NewReno Modification to TCP's Fast Recovery Algorithm (RFC6582), IETF, April 2012.
[4] I. Rhee, L. Xu, CUBIC: A New TCP-Friendly High-Speed TCP Variant, New York, ACM, Jul 2008.
[5] E. Blanton, M. Allman, K. Fall, L. Wang, I. Jarvinen, M. Kojo, Y. Nishida, A Conservative Loss Recovery Algorithm based on Selective Acknowledgment (SACK) for TCP (RFC6675), IETF, Avgust 2012.
[6] M. Mathis, J. Mahdavi, Forward Acknowledgment: Refining TCP Congestion Control, New York, ACM, 1996.
[7] M. Allman, TCP Congestion Control with Appropriate Byte Counting (ABC) (RFC3465), IETF, Februar 2003.
[8] K. R. Fall, W. R. Stevens, TCP/IP Illustrated, Volume 1, 2nd ed., New York, Addison Wesley, 2013.

ABSTRACT

The Transmission Control Protocol (TCP) makes up for more than 80% of the total Internet traffic volume. TCP thin flows, although not constituting the majority of total traffic volume, are very common in everyday use. Thin flows are used for WWW surf, remote computer administration, online gaming, etc. In this work thin flow performance of the most commonly used TCP congestion control algorithms are tested, specifically for transport of WWW (HTTP) traffic. Special emphasis si made to make the results relevant and practical, as much as possible. In regard to that, traffic generated during typical WWW surf session is captured and characterized, and based on that the simulation model is made. Simulations are done using production TCP stack of GNU/Linux operating system. Hybrid platform which allows controllability of experiment and usage of production TCP stack is possible because of modern network simulator NS3.

TCP performance for WWW/HTTP traffic transport

Goran Martić