7. ALGORITMI TCP PROTOKOLA - SIMULACIJA

Standardizacija protokola sastoji se od vanjske i unutrašnje specifikacije. Vanjska specifikacija definira oblik i značenje pojedinih dijelova PDU. Teško se mijenja bez poremećaja funkcionalnosti mreže, jer sva računala istovremeno moraju preći na novi format PDU. Unutrašnja specifikacija odnosi se na razne postupke na osnovu primljenih PDU. Lakše se mijenjaju jer često postoji kompatibilnost između različitih varijanti istog protokola.

Efekti promjena unutrašnje specifikacije protokola ispituju se najprije na simuliranim, a kasnije i na stvarnim mrežama. Primjena simulatora optimalna je zbog niskih troškova i jednostavnog upravljanja uvjetima eksperimenta. Rezultati simulacija verificiraju se mjerenjima na stvarnoj mreži.

TCP (Transmission Control Protocol) je najčešće korišteni, spojevni protokol prijenosne (transportne) razine Interneta. U okviru ove vježbe simulacijom će se pokazati ponašanje različitih varijanti TCP protokola.

7.1. STANDARDIZACIJA TCP PROTOKOLA

TCP sučelje okrenuto je s jedne strane korisničkim procesima, a s druge strane IP protokolu mrežne razine. Sučelje prema korisniku podržava niz funkcijskih poziva. To su pozivi za uspostavljanje i zatvaranje veze (OPEN, CLOSE), te za slanje i primanje podataka (SEND, RECEIVE). Sučelje prema protokolu mrežne razine se ne specificira, već ga određuje proizvođač komunikacijskih programa.

Kada podaci stignu na prijenosnu razinu, TCP ih može odmah poslati, ili pohraniti u spremnik s ciljem sakupljanja veće količine podataka kako bi ih poslao odjednom. Korisnik može zahtijevati momentalnu isporuku podataka postavljanjem PUSH bita.

TCP se mora oporaviti od gubitka, oštećenja ili dupliciranja podataka, kao i od slučaja dostave paketa pogrešnim redoslijedom. To je omogućeno pridruživanjem rednog broja (sequence number) svakom oktetu koji se prenosi, te očekivanjem pozitivne potvrde (ACK) od prijemnika. Ako se pozitivna potvrda ne primi do isteka vremena retransmisije, podatak se automatski ponovo šalje. Na prijemnoj strani redni broj služi da bi se segmenti poredali u pravilnom redoslijedu, te radi eliminacije duplikata.

TCP omogućuje prijemniku upravljanje količinom podataka koje može odaslati predajnik. Sa svakom potvrdom (ACK), prijemnik šalje i veličinu prozora (RWIN, receiver window) koja označava koliko okteta prijemnik može obraditi. Najveći prozor je 65536 okteta (16 bita).

Glavna je značajka TCPa uspostavljanje veze od krajnje točke pošiljatelja do krajnje točke primatelja. Ove krajnje točke nazivamo priključnicama (socket). Svaka priključnica jedinstveno je određena IP adresom i priključnom točkom (port). Veza je jedinstveno određena parom priključnica. Priključnice mogu sudjelovati u mnogim vezama istodobno (multipleksiranje).

TCP segmenti se šalju unutar IP paketa. TCP zaglavlje, slika 7.1., i podatkovni okteti slijede iza IP zaglavlja.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Source Port								Destination Port
Sequence Number								
Acknowledgment Number								
Data Offset	Reserved	URG	ACK	PSH	RST	SYN	FIN	Window
Checksum								Urgent Pointer
Options								Padding
Data								

Slika 7.1. Format TCP zaglavlja

SOURCE PORT, DESTINATION PORT (po 16 bita)

Identifikator izvorišne i odredišne priključne točke.

SEQUENCE NUMBER (32 bita)

Ako je SYN bit u nuli, ovo polje predstavlja redni broj prvog okteta podataka u segmentu. Ako je SYN u jedinici, redni broj je inicijalni redni broj (ISN), a prvi podatkovni oktet je ISN + 1.

ACKNOWLEDGMENT NUMBER (32 bita)

Ako je ACK bit u jedinici, ovo polje sadrži vrijednost slijedećeg rednog broja kojeg prijemnik očekuje primiti. Time se potvrđuju (kumulativno) svi prethodni podaci.

DATA OFFSET (4 bita)

Kazuje koliko je 32-bitnih riječi sadržano u TCP zaglavlju.

CONTROL BITS (6 bita)

URG: (Urgent Pointer): Pokazivač hitnosti, paket sadrži hitnu poruku.
ACK: Bit potvrde u jedinici znači da polje broj potvrde sadrži potvrdni broj.
PSH: Bit kojim predajnik zahtijeva trenutnu isporuku pristiglih podataka na prijemnoj strani.
RST: Bit za resetiranje veze.
SYN: Ovaj se bit koristi pri uspostavljanju veze. Zahtjev za uspostavljanje veze označava se sa SYN = 1, ACK = 0; dok su SYN = 1, ACK = 1 oznake prihvaćanja zahtjeva za vezu.
FIN: Ovaj bit označava da pošiljatelj nema više podataka.

WINDOW (16 bita)

Predstavlja broj okteta podataka, počevši potvrdnog broja koje je pošiljatelj spreman prihvatiti.

CHECKSUM (16 bita)

Ovo polje ima vrijednost jediničnog komplementa sume jediničnih komplemenata 16-bitnih riječi zaglavlja i podataka. Kontrolna suma pokriva i 96-bitno pseudo-zaglavlje, dio zaglavlja IP protokola koji se ne mijenja prolaskom kroz mrežu, slika 7.2.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Source Address		
Destination Address		
Zero	Protocol	TCP Length

Slika 7.2. Pseudo zaglavlje

URGENT POINTER (16 bita)

Sadržaj ovog polja pokazuje na redni broj posljednjeg okteta hitnih podataka. Ovo vrijedi samo ako je URG bit u jedinici. TCP mora podržavati sve duljine sekvenci hitnih podataka.

OPTIONS (promjenljive duljine)

Opcije se mogu dodati na kraj TCP zaglavlja. Sve se uključuju u kontrolnu sumu zaglavlja. Dva su formata opcija, jednooktetni i višeoktetni.
Jedna od najvažnijih opcija je ona koja omogućuje odrediti maksimalnu veličinu segmenta (MSS. Maximum Segment Size). Ova opcija se šalje za vrijeme uspostave veze.

PADDING (promjenljive duljine)

U ovo polje se dodaju nule dok se ne popuni 32-bitna riječ.

Održavanje TCP veze zahtijeva pamćenje više varijabli. One se pohranjuju u strukturi nazvanoj TCB (Transmission Control Block). TCB sadrži informacije o identifikatorima lokalne i udaljene priključnice, prioritetima i sigurnosti veze, pokazivačima na red za retransmisiju i tekući segment, te o varijablama vezanim za redne brojeve predaje i prijema podataka.

Varijable sekvenci za slanje podataka su:

SND.UNA –redni broj prvog odaslanog, a nepotvrđenog okteta;
SND.NXT – redni broj slijedećeg okteta za slanje;
SND.WND – veličina predajnog prozora; maksimalni broj okteta koji se mogu odaslati bez primitka potvrde. Kada je SND.UNA = SND.NXT nema nepotvrđenih podataka, te se može poslati puni prozor SND.WND).
SND.UP –pokazivač na hitne podatke;

SND.WL1 –redni broj okteta kod zadnje promjene veličine prozora;
SND.WL2 –potvrda kod zadnje promjene veličine prozora;
ISS –početni predajni redni broj okteta.

Varijable sekvenci za prijem podataka:
RCV.NXT –redni broj sljedećeg okteta;
RCV.WND –veličina prijemnog prozora;
RCV.UP –pokazivač na hitne podatke;
IRS –početni prijemni redni broj okteta.

Varijable koje uzimaju vrijednosti iz tekućeg segmenta:
SEG.SEQ –redni broj prvog okteta u segmentu;
SEG.ACK –redni broj okteta koji se očekuje;
SEG.LEN –dužina segmenta;
SEG.WND –trenutna veličina prozora prijemnika;
SEG.UP –pokazivač na kraj hitnih podataka u segmentu;
SEG.PRC –okteti koji imaju prednost.

U tablici 7.1. prikazana su moguća stanja TCP veze:

STANJE:	**ZNAČENJE:**
CLOSED	Veza je neaktivna (raskinuta)
LISTEN	Stanje čekanja zahtjeva za vezu
SYN-SENT	Poslan je zahtjev za vezu; čeka se da druga strana odgovori zahtjevom za vezu
SYN-RECEIVED	Primljen je zahtjev za vezu; čeka se potvrda zahtjeva za vezu
ESTABLISHED	Stanje normalnog prijenosa podataka
FIN-WAIT-1	Odaslan je zahtjev za raskidanje veze
FIN-WAIT-2	Čeka se zahtjev za raskidanje veze od udaljenog TCPa
CLOSE-WAIT	Čekanje na zahtjev za raskidanje veze od lokalnog korisnika
CLOSING	Čeka se potvrda na poslani zahtjev za prekid veze
LAST-ACK	Čeka se potvrda na zahtjev za raskid veze od udaljenog TCPa
TIME-WAIT	Čeka se dva maksimalna vremena života segmenta (MSL), kako bismo se uvjerili da su svi zaostali segmenti stigli do udaljenog TCPa. Za to vrijeme se ne može ponovo uspostaviti veza između ove dvije priključnice.

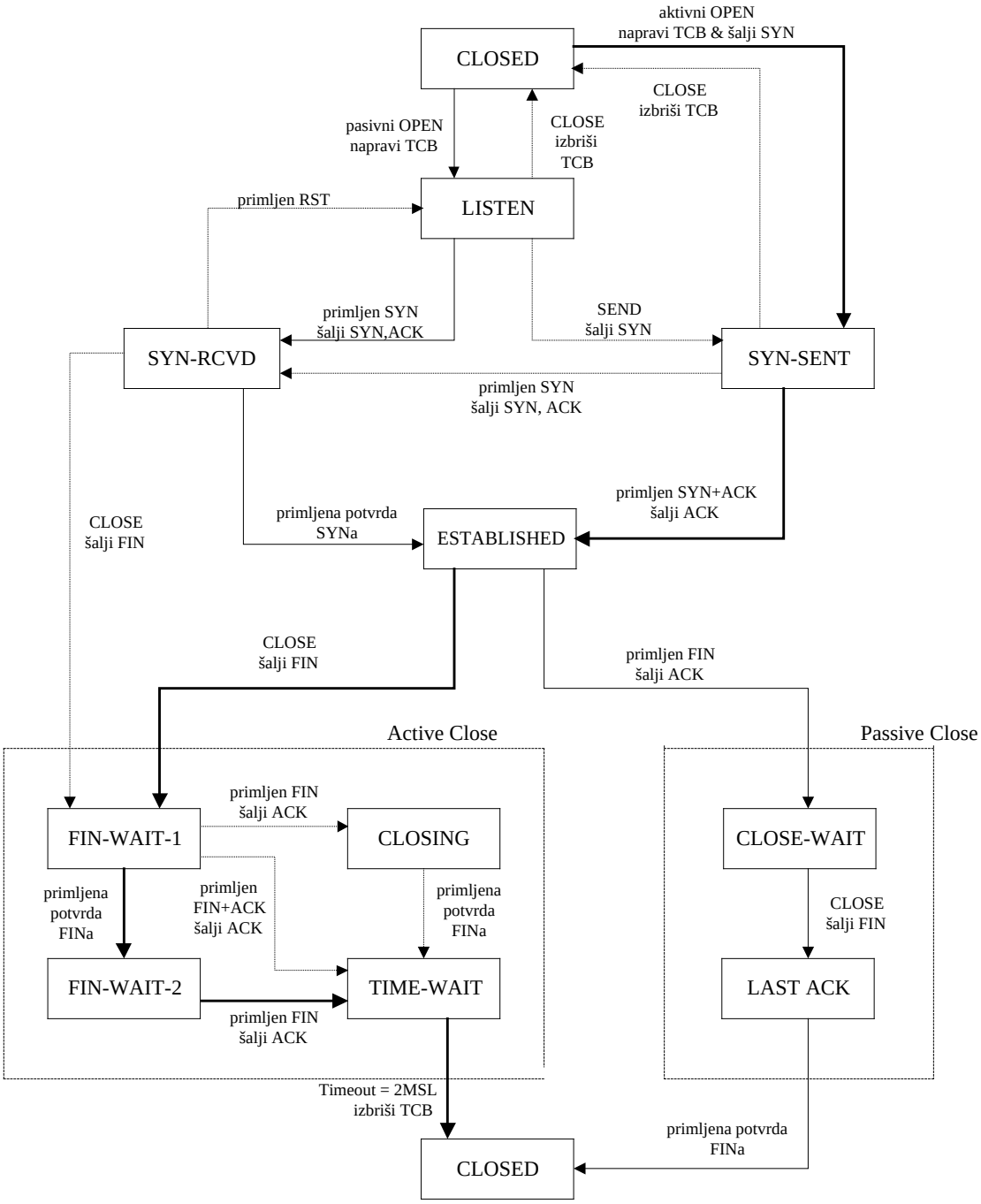
Tablica 7.1.Moguća stanja TCP veze

Dijagram stanja veze na slici 7.3. prikazuje promjene stanja u skladu s događajima koji ih uzrokuju i rezultirajućim akcijama, ali se ne bavi greškama ni akcijama koje nisu povezane s promjenama stanja.

U slučaju prekida veze uz gubitak nepotvrđenih podataka i pokušaja njenog ponovnog uspostavljanja, javlja se problem raspoznavanja zaostalih segmenata iz prijašnje veze. Zato se koriste slučajno odabrani početni redni brojevi ISN (Initial Sequence Number). Kod uspostave veze, generator početnih rednih brojeva odabire 32-bitni ISN, a najniži se bit uvećava svake 4 μs. Stoga ISN ima period od približno 4.55 sata. Kako pretpostavljamo da segmenti ne mogu ostati u mreži duže od MSL (Maximum Segment Life), a to je obično oko dvije minute, možemo smatrati da je ISN jedinstven.

Za uspostavu veze dva TCPa moraju sinkronizirati početne redne brojeve. To se radi razmjenom segmenata za uspostavu veze, koji nose kontrolni bit SYN i ISN. Prilikom sinkronizacije obje strane moraju poslati svoj ISN, te primiti potvrdu za to od druge strane.

Prilikom uspostave veze koristi se proces sinkronizacije u tri koraka (three-way handshake). Obično ovaj postupak pokreće jedno računalo, dok drugo odgovara na njega (slika 7.4.).



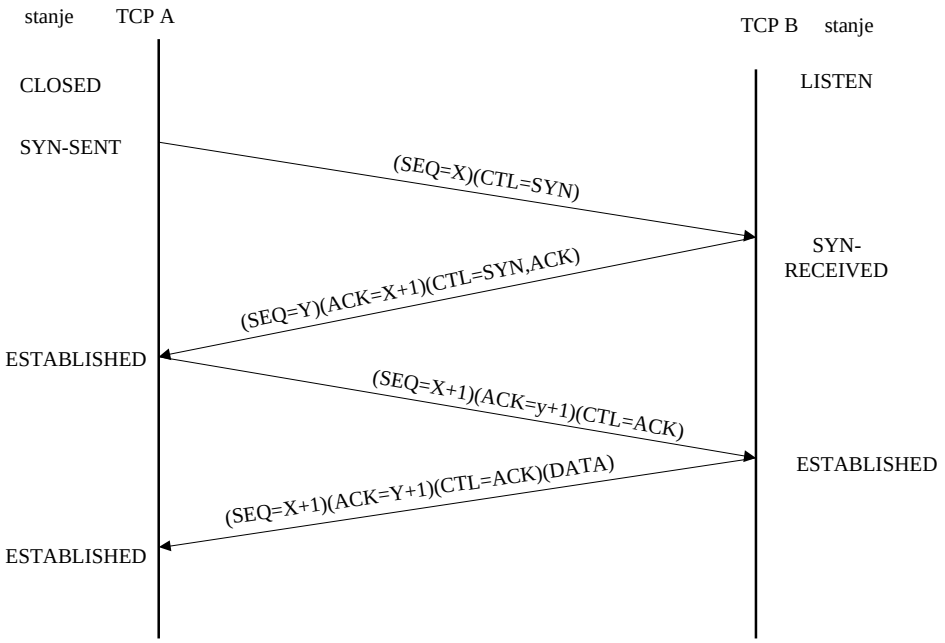
Slika 7.3. Dijagram stanja TCP veze

Računalo B nalazi se u stanju osluškivanja te čeka na zahtjev za uspostavu veze. Računalo A šalje zahtjev za uspostavu veze, SYN =1 i ACK =0 te broj sekvence od koje želi započeti prijenos, X. Nakon toga računalo B šalje SYN bit i potvrdu primitka SYN bita, tj. zahtjev za slanjem segmenta X+1. Računalo A tada odašilje segment X+1 i veza je uspostavljena.

Ukoliko se posumnja da je stigao paket zaostao iz prethodne veze (iste priključnice, ali redni broj daleko izvan prozora, šalje se upravljačka poruka RESET. Općenito je pravilo da se RESET mora poslati uvijek kada stigne segment koji očito nije namijenjen tekućoj vezi.

Stanica šalje segment s bitom FIN u trenutku kada nema više podataka za slanje, tj. kada korisnik izda naredbu CLOSE. TCP koji pošalje FIN nastavlja primati sve dok ne stigne poruka da je i druga strana raskinula vezu, tj. da nema podataka za slanje.

- zatvaranje veze započinje korisnik naredbom CLOSE,
- zatvaranje veze započinje udaljeni TCP slanjem upravljačkog signala FIN,
- oba korisnika istovremeno šalju CLOSE signal.



Slika 7.4.Uspostava veze TCP protokola

U stanju uspostavljene veze prijenos podataka se obavlja razmjenom podatkovnih segmenata. Uslijed greške ili zagušenja na mreži može doći do gubitka segmenta. Stoga TCP koristi mehanizam retransmisije kako bi osigurao dostavu svakog segmenta. Potvrde su kumulativne, što znači da potvrda X-tog okteta podrazumijeva i potvrdu svih prethodnih.

Ako u određenom vremenu (RTO, Retransmission Timeout) ne dobije potvrdu, TCP ponovno šalje segment, računajući da je izgubljen. Zbog raznolikosti mreža u sustavu i širokog raspona uporabe TCP veza, ovo vrijeme se računa dinamički. Kvalitetan proračun tog vremena od ključnog je značenja za učinkovitost TCP veze. Proračuni se zasnivaju na vremenu potrebnom da stigne potvrda za odaslani paket. To se vrijeme naziva vrijeme obilaska, RTT (Round Trip Time). RTT se stalno mijenja i ovisi o trenutnoj opterećenosti mreže.

Računanje vremena retransmisije obavlja se od 1988. Jacobsonovim algoritmom. Za svaku vezu TCP promatra vrijednost RTT koja predstavlja filtriranu vrijednost stare vrijednosti i nove vrijednosti M:

$$RTT = \alpha RTT + (1-\alpha)M$$

gdje je α faktor koji odlučuje kolika će se težina pridati starom vremenu. Obično α ima vrijednost 7/8.

Da bi se spriječile neželjene retransmisije (kratak RTO) i predugo čekanje na detekciju gubitka (dugačak RTO), računa se devijacija D:

$$D = \alpha D + (1-\alpha)\,|RTT-M|$$

gdje α može, ali i ne mora imati istu vrijednost kao i kod proračuna vremena obilaska (RTT). Konačno se izračuna vrijeme retransmisije (čekanja na potvrdu) prema formuli:

$$RTO = RTT + 4*D$$

Problemi pri računanju vremena RTT javljaju se kod ponovo odaslanog segmenta. Kada dođe signal potvrde, nije jasno odnosi li se on na retransmitirani ili na izvorni segment. Krivi odgovor može unijeti velike pogreške kod računanja vremena obilaska. Ovaj problem je riješen u Karnovom algoritmu (Phil Karn), ili slanjem vremena odašiljanja (Timestamp Option).

Karn je jednostavno predložio: ne računati RTT kod prijema potvrde, ako se ona odnosi na segment koji je bio ponovo odaslan. Umjesto toga udvostručiti RTO kod svake ponovljene retransmisije istog segmenta (tzv. “backoff mehanizam”). Za slijedeći odaslani paket (bez retransmisije) računaju se novi RTT i RTO koji odgovaraju novom stanju.

7.2. RAZVOJ KONTROLE TOKA TCP PROTOKOLA

Nazivni kapacitet mreže predstavlja maksimalnu količinu podataka koju mreža može prenijeti, a da ne nastupi zagušenje (congestion). Relativno opterećenje mreže opisuje faktor ρ:

$$\rho = \frac{ponudjeni\_promet}{kapacitet\_mreze}$$

Zagušenje nastaje kada faktor ρ u nekom vremenskom intervalu $[t_1, t_2]$ poprimi vrijednost $\rho \geq 1$. U ovisnosti o periodu trajanja, razlikujemo trajna, periodična, privremena i trenutna zagušenja. Kontrola toka je osnovni mehanizam za izbjegavanje zagušenja kod mreža s prospajanjem paketa. Usmjeravanje prometa je za paketne mreže pomoćni postupak izbjegavanja zagušenja.

Prozorska kontrola toka ograničava broj paketa u mreži, ali dozvoljava slanje praskova (burst) paketa. Nasuprot tome, kontrola brzine garantira jednoliki tok paketa, ali ne garantira broj paketa u mreži. Kontrola toka TCP protokola postiže se mehanizmom prozorske kontrole.

Suština mehanizma prozorske kontrole toka je u tome da se ograniči broj paketa koje izvorište može poslati. Kod starijih protokola prozor je bio jednak modulu numeracije paketa. Kod novijih, širina prozora određuje se mogućnostima prijemnika (manje od modula numeracije) i konačno mogućnostima mreže (manje od mogućnosti prijemnika).

Paketi podataka putuju od izvorišta ka odredištu, dok potvrde putuju kroz mrežu u suprotnom smjeru. Ukupno vrijeme potrebno da paket prođe kroz mrežu (tj. da se njegova potvrda vrati u izvorište) jednako je vremenu obilaska RTT. Tek kada je jedan paket napustio mrežu, izvorište može poslati sljedeći. Za velike mreže slanje pojedinog paketa nema utjecaja, pa su varijacije RTT dovoljno male. Odavde slijedi važna pretpostavka o konstantnom vremenu obilaska. Ako je W- širina prozora, a RTT- vrijeme obilaska, brzina predaje predajnika dana je formulom:

$$R = \frac{W}{RTT} \ [broj\ paketa/sek]$$

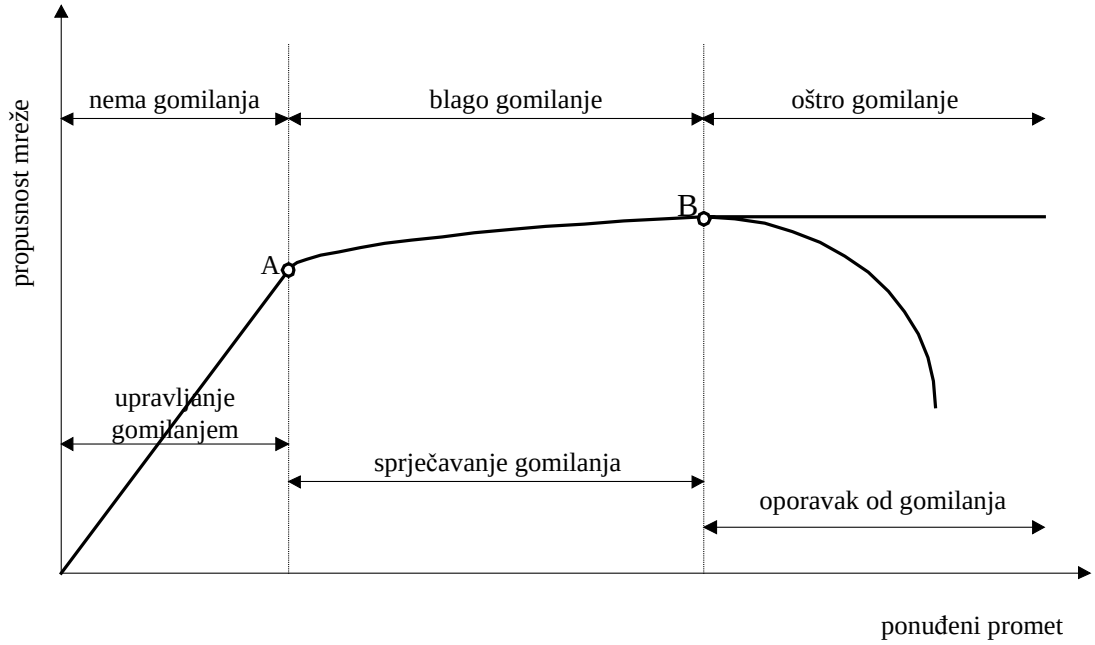
jer kroz to vrijeme mrežom prođe cijela širina prozora W. Ograničavanjem širine prozora W može se kontrolirati brzina predaje R, uz pretpostavku da je RTT približno konstantno.

Posljedice zagušenja vidljive su iz dijagrama na slikama 7.5. i 7.6. Ovi dijagrami prikazuju funkcijske ovisnosti propusnosti mreže i prosječnog kašnjenja/gubitka o ponuđenom prometu.

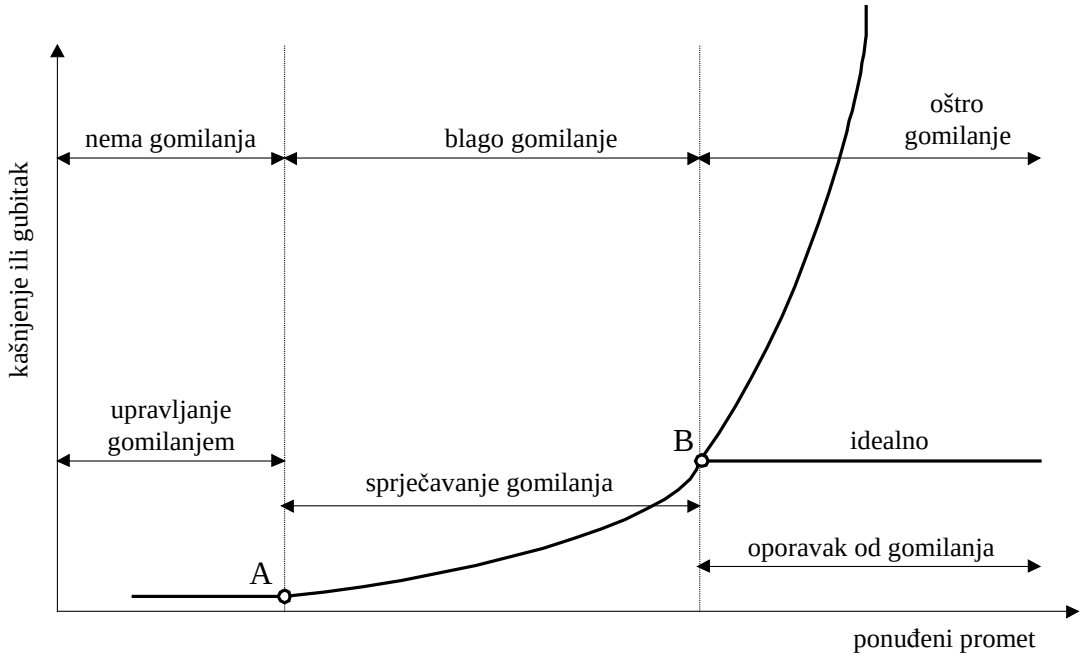
Iz dijagrama na slici 7.5. vidljivo je da propusnost, a time i iskorištenje mreže, pri relativno malom opterećenju mreže, raste s povećanjem ponuđenog opterećenja do točke A (tzv. koljeno, eng. knee). Pri daljnjem povećanju opterećenja povećanje propusnosti mreže je sporije, jer mreža ulazi u stanje blagog gomilanja. Povećava se i kašnjenje.

Kada ponuđeni promet dostigne vrijednost koja odgovara točki B (tzv. greben, eng. cliff), mreža ulazi u stanje oštrog gomilanja. Propusnost mreže se naglo smanjuje pri daljnjem povećanju opterećenja, jer zbog konačne veličine međuspremnika u čvorovima mreže dolazi do gubitaka paketa i retransmisija.. Tako u području oštrog gomilanja dolazi do degradacije kvalitete usluge, što se ogleda u povećanju kašnjenja ili gubitaka.

Djelotvornost neke sheme kontrole zagušenja može se procijeniti na osnovi kašnjenja ili gubitka do kojega dolazi kada je ponuđeni promet veći od najvećeg dozvoljenog prometa. Ovu funkcijsku ovisnost prikazuje dijagram na slici 7.6.



Slika 7.5. Funkcijska ovisnost propusnosti mreže o ponuđenom prometu



Slika 7.6. Funkcijska ovisnost kašnjenja ili gubitka o ponuđenom prometu

Optimalna radna točka mreže je točka A. U teoriji bismo nastojali održati mrežu što bliže toj točki. No u praksi algoritmi koji se primjenjuju otkrivaju raspoloživi kapacitet mreže upravo uvođenjem u zagušenje (tj. do točke B), te naknadnim smanjivanjem ponuđenog prometa.

TCP koristi gubitak segmenta kao indikaciju zagušenja, pa održava mrežu u radnoj točki oko točke B. Za postupke upravljanja prometom i sprječavanje zagušenja TCP raspolaže s nekoliko algoritama koji su često puta mijenjani i nadograđivani, u skladu s razvojem protokola TCP.

Nagle-ov algoritam: Kod nekih aplikacija uočeno je vrlo neučinkoviti iskorištenje kanala. Primjer za to je TELNET veza, gdje se šalje znak po znak s tipkovnice. To znači da svaki pojedinačni znak putuje u svom paketu, dakle 1 oktet podataka i 40 okteta zaglavlja (20 B TCP zaglavlja + 20 B IP zaglavlja). Učinak se svodi na samo 2.4% korisne informacije po paketu, što nepotrebno opterećuje mrežu.

J. Nagle je 1984. godine ponudio adaptivno rješenje. On je predložio zadržavanje slanja novih korisnikovih podataka dok se ne potvrde svi prethodno odaslani paketi ili dok se ne skupi dovoljno podataka za slanje segmenta maksimalne veličine (MSS – Maximum Segment Size). Nadolazeći korisnikovi podaci mogu se odmah proslijediti ili zadržati za kasnije slanje.

Silly Window Syndrome: Sljedeći problem koji se može pojaviti je sindrom besmislenih prozora. Ovaj se problem javlja kada se podaci proslijede u velikim blokovima, a aplikacija na prijamnoj strani čita samo jedan po jedan oktet.

U početnom stanju spremnik prijemnika je pun i pošiljatelj to zna (ima oglašenu veličinu prozora jednaku nuli). Kada se iščita jedan oktet na prijamnoj strani, odmah se oglašava prozor veličine jednog okteta. Pošiljatelj šalje jedan oktet podataka, te dolazi do ponavljanja ciklusa. Ovo rezultira slanjem svih podataka u paketima koji sadrže samo po 1 oktet podataka.

1982. godine je Clark predložio rješenje ovog problema. Potrebno je spriječiti prijemnik da oglašava prozor veličine jednog okteta. Prijemnik treba oglasiti prozor veličine maksimalnog segmenta koji je oglašen kod uspostave veze. Do tog trenutka on bi morao biti u stanju čekanja da se isprazne međuspremnici koji bi mu dozvolili da napravi taj korak.

Slow start: Algoritam usporenog starta služi za otkrivanje raspoloživog kapaciteta mreže, a koristi se na početku prijenosa ili kod oporavka od gubitka. Jednostavno se implementira pomoću dvije varijable. To su prag Slow Starta SSTHRESH i prozor zagušenja CWND (Congestion Window). SSTHRESH označava vrijednost na kojoj veza izlazi iz Slow Start faze i ulazi u fazu izbjegavanja zagušenja. SSTHRESH ima početnu vrijednost od 64 kB (toliko iznosi maksimalna veličina prijemnog prozora udaljenog TCPa).

Nakon uspostave veze, prozor zagušenja se postavlja na jedan segment. Prijenos započinje odašiljanjem jednog paketa i čekanjem potvrde za taj paket. Kada se primi potvrda, prozor zagušenja se poveća s jednog na dva segmenta. To znači da se sada mogu poslati dva segmenta. Proces se nastavlja dalje tako da se primitkom svake potvrde CWND uvećava za jedan. Kako je jedna potvrda upravo primljena, za jednu primljenu potvrdu šalju se dva nova paketa. Na taj način CWND se udvostručuje primitkom svih potvrda iz prethodnog prozora. Ukoliko je vrijeme obilaska konstantno, prozor zagušenja raste eksponencijalno.

Rast prozora zagušenja CWND ograničen je s dva mehanizma. Prvo, manjom od dvije vrijednosti RWIN i SSTHRESH, nakon čega se automatski prolazi u fazu izbjegavanja zagušenja. Drugo, eksponencijalni rast prometa može dovesti do gubitka segmenta.

Ukoliko je u fazi usporenog starta došlo do gubitka, ažuriraju se vrijednosti:

$$SSTHRESH = \max(2; CWND/2) \quad , \quad CWND = 1$$

Time nanovo započinje usporeni start, ali sada do polovičnog prozora CWND/2 u odnosu na prozor koji je uzrokovao zagušenje. Postupak se ponavlja sve dok faza usporenog starta ne prođe bez gubitaka. Postignuti CWND smatra se optimalnim u tom trenutku, te TCP prelazi u vazu izbjegavanja zagušenja.

Congestion avoidance: Izbjegavanje zagušenja je faza u kojoj TCP treba ispitivati mogućnost povećanja prozora kako bi iskoristio kapacitet mreže oslobođen eventualnim završetkom prijenosa drugih korisnika. Stoga se u ovoj fazi CWND povećava za 1 svakih RTT vremena, odnosno po algoritmu:

$$CWND = CWND + 1/CWND$$

Ova faza traje sve do ponovnog gubitka segmenta. Tada se ponavlja faza usporenog starta, pri čemu se koristi SSTHRESH = max(2; CWND/2) i CWND = 1, kao gore. U obje faze, prijemnik mora slati potvrdu za svaki primljeni paket.

TCP s ugrađenim usporenim startom i izbjegavanjem zagušenja poznat je kao **osnovni TCP**.

Fast retransmit: Čekanje na istek vremena retransmisije RTO je dugotrajno i traje RTT+4D. Za to vrijeme će svi paketi napustiti mrežu i ona ostaje neiskorištena. Brza detekcija gubitka moguća je ako

prijemnik za svaki prekoredno primljeni segment (nakon gubitka) ponavlja posljednju poslanu kumulativnu potvrdu. Predajnik nakon duge duplicirane potvrde još nije siguran da li se radi o gubitku ili samo poremećaju redoslijeda isporuke. Međutim, nakon što primi tri duplicirane potvrde, predajnik zaključuje da je došlo do gubitka segmenta i obavlja ponovno slanje daleko prije isteka RTO. To je algoritam brze retransmisije.

TCP s ugrađenom brzom retransmisijom poznat je od 1989. kao **TAHOE TCP**.

Fast recovery: Algoritam brzog oporavka je uveden kako bi se što bolje iskoristile prednosti brze retransmisije. Naime, nakon gubitka paketa TCP normalno mora ići u fazu usporenog starta. Algoritam brzog oporavka izbjegava ovu fazu na način da se kod brze retransmisije parametri postave:

$$SSTHRESH = CWND/2 \quad ; \quad CWND = SSTHRESH+3$$

čime se uzimaju u obzir paketi koji su izašli iz mreže (tri duplicirane potvrde). Primitkom potvrde novih podataka izlazi ulazi se u fazu izbjegavanja zagušenja s polovičnim prozorom:

$$CWND = SSTHRESH.$$

Algoritam brzog oporavka efikasan je samo za jednostruke pogrješke.

TCP s ugrađenom brzom retransmisijom i brzim oporavkom poznat je od 1990. kao **RENO TCP**.

Djelomične potvrde: Da bi se ubrzao izlazak iz faze brzog oporavka za slučaj višestrukog gubitka segmenta, uvedeno je razlikovanje novih i djelomičnih (parcijalnih) potvrda. Djelomična potvrda se bazira na karakteristikama kumulativne potvrde: kod jednostrukog gubitka, prijemnik će, nakon što primi brzom retransmisijom ponovljeni segment, potvrditi sve segmente poslane do popune nedostajućih podataka.

Ukoliko prijemnik potvrdi smo dio podataka, predajnik može zaključiti da se radi o višestrukom gubitku segmenata. Takva potvrda se zove djelomična potvrda. Ona istovremeno znači da je određeni broj segmenata izašao iz mreže, te da je još neki od ranije poslanih segmenata izgubljen. Taj segment je moguće odmah ponovno poslati.

TCP s ugrađenim djelomičnim potvrdama poznat je kao eksperimentalni **NEW-RENO TCP**.

7.3. SIMULACIJSKA MJERENJA NA PAKETNIM MREŽAMA

Simulacija pomoću računala je disciplina određivanja modela nekog stvarnog fizičkog ili teoretskog sustava, pokretanje takvog modela na digitalnom računalu, te analiza dobivenih rezultata. Simulacija je nužna u slučajevima kada je model je kompleksan s mnogo međusobno povezanih varijabli stanja, model je nelinearan, ili kada model sadrži varijable koje su slučajne prirode.

Značaj simulacije naročito dolazi do izražaja u projektiranju računalnih mreža. Korištenjem simulacije možemo mijenjati parametre pojedinih čvorišta u mreži bez skupih intervencija na samim uređajima.

Simulatori paketnih mreža su jako zahtjevni u pogledu obrade podataka. Kada čvorište prosljeđuje paket, generiraju se najmanje dva događaja: jedan određuje vrijeme dostavljanja paketa fizičkom mediju, a drugi vrijeme u kojem susjedni čvor prihvati paket. Ako je riječ o izvorišnom čvorištu, postoji još i događaj koji određuje slijedeći paket za retransmisiju. Ovi su simulatori jednako tako zahtjevni i u pogledu memorije. Broj paketa koji prolaze kroz mrežu raste linearno s povećanjem umnoška pojasne širine i kašnjenja.

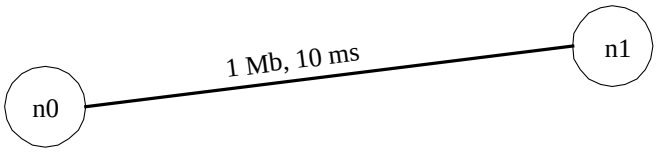
U ovoj vježbi koristi se simulator paketnih mreža NS (Network Simulator), razvijen 1989. godine u Lawrence Berkeley National Laboratory (LBL) SAD kao varijanta već postojećeg simulatora REAL.

NS je mrežni simulator koji radi na principu liste događaja, a ugrađen je u Tool Command Language (Tcl) UNIX sustava. Struktura simulatora je izvedena u programskom jeziku C++, a može se upravljati i definirati pomoću Tcl sučelja. NS je organiziran kao interpreterska ljuska koja razumije naredbe pisane u Tcl jeziku, te naredbe samog simulatora. Naredba *ns* poziva određene funkcije NS simulatora unutar Tcl skripte. Pomoću naredbe *ns* definira se mrežna topologija, podešavaju se izvorišta i odredišta i starta se simulacija.

Mrežna topologija se gradi uporabom triju osnovnih elemenata: čvorišta, veze i agenata. Čvorišta se kreiraju pomoću naredbe *ns node* i ugrađuju se u mrežu naredbom *ns link*. Čvorišta su pasivni objekti koji služe kao spremnici za *agente*, objekte koji aktivno upravljaju simulacijom. Svako čvorište ima jedinstvenu adresu koja se dodjeljuje automatski. Primjeri agenata su izvorišta i odredišta prometa u mreži. Naredbom *ns agent* kreira se agent na određenom čvorištu. Naredbe *ns node, ns agent* i *ns link* stvaraju nove objekte, te vraćaju proceduru pomoću koje se može pristupiti objektima.

Simulacija se pokreće naredbom *ns run* i odvija se dok se ne obrade svi događaji, odnosno dok se ne izvrši naredba *stop*.

Zadavanje topologije simulirane mreže objašnjeno je na jednostavnoj mreži na slici 7.7. Mreža se sastoji od dva međusobno povezana čvorišta. Na slici su dani i podaci o kapacitetu i kašnjenju na kanalu.



Slika 7.7. Jednostavni primjer mrežne topologije

Ovakvu jednostavnu topologiju prikazati ćemo unutar Tcl skripte za NS simulator:

```
set n0 [$ns node]
set n1 [$ns node]
```

Naredba *ns node* vraća objekt koji opisuje čvorište. Taj objekt se pomoću Tcl naredbe *set* sprema u pojedine varijable (n0, n1). Uglate zagrade određuju prioritet izvršavanja Tcl naredbi. Sada stvorena čvorišta moramo povezati komunikacijskim kanalom:

```
ns_duplex $n0 $n1 1Mb 10ms drop-tail
```

Tcl procedura *ns_duplex* ima slijedeću sintaksu:

```
ns_duplex node1 node2 bandwidth delay type
```

Ova procedura stvara dvosmjernu vezu (tj. dvije jednosmjerne veze) između čvorišta n0 i n1, pojasne širine 1Mb/s, s kašnjenjem 10 ms. Varijabla *type* određuje metodu posluživanja spremnika čvorišta. Tri su osnovna načina:

- drop-tail,
- red (random-early drop),
- cbq (class-based queuing).

Ovdje su obje veze tipa drop-tail. To je najjednostavnija metoda po kojoj čvorište poslužuje paket koji prije dođe, a odbacuje pristigle pakete nakon popunjavanja reda čvorišta. Red poslužuje pakete kako dođu, ali odbacuje slučajno izabrani paket unutar reda. Cbq poslužuje pakete prema prioritetu.

Nakon zadavanja fizičke topologije mreže, definiramo slanje podataka iz čvorišta n0 u n1. U NS simulatoru podaci se uvijek šalju od jednog “agenta” drugome. Agenti su procesi koji stvaraju promet. Pomoću naredbe *ns agent* svakom čvorištu pridijeliti ćemo procese koji šalju ili primaju pakete. Sintaksa naredbe *ns agent* je slijedeća:

```
ns agent type node
```

Čvorište *node* mora prethodno biti stvoreno i povezano u mrežu naredbom *ns link*. Parametar *type* definira tip stvorenog agenta:

tcp	BSD Tahoe TCP
tcp-reno	BSD Reno TCP
tcp-newreno	modificirana verzija BSD Reno TCP
tcp-sack1	BSD Reno TCP sa selektivnim potvrdama

tcp-fack	BSD Reno TCP sa unaprijednim porukama
tcp-sink	standardno TCP odredište
tcp-sink-da	TCP odredište koje stvara kašnjene potvrde
sack1-tcp-sink	TCP odredište koje stvara selektivne potvrde
sack1-tcp-sink-da	TCP odredište koje stvara selektivne potvrde s kašnjenjem
cbr	izvorište prometa konstantne brzine prijenosa
loss-monitor	CBR odredište koje izvješćuje o gubicima

Najprije definiramo izvorišta i odredišta

```
set src1 [ns agent tcp $n0]
set snk1 [ns agent tcp-sink $n1]
set src2 [ns agent tcp $n1]
set snk2 [ns agent tcp-sink $n0]
```

pa ih povežemo (umjesto TCP uspostave veze)

```
ns_connect $src1 $snk1
ns_connect $src2 $snk2
```

Da bi lakše pratili pakete različitih veza, smjestit ćemo ih u zasebne klase, iako ne koristimo cbq:

```
$src1 set class1
$src2 set class 2
```

Nakon što smo stvorili odgovarajuće klase, trebamo kreirati ftp izvorišta u svakom čvorištu pomoću TCP objekta *source*:

```
set ftp1 [$src1 source ftp]
set ftp2 [$src2 source ftp]
```

Sada možemo pokrenuti oba ftp izvorišta u željenim trenucima:

```
ns at 0.0 “$ftp1 start”
ns at 1.0 “$ftp2 start”
```

NS omogućuje vrlo jednostavan način pozivanja bilo koje procedure u određenom trenutku naredbom *ns at.* Objekti klase *trace* služe za ispis izlaznih podataka direktno u vanjsku datoteku:

```
set trace [ns trace]
$trace attach [open out.tr w]
$n0 trace $trace
$n1 trace $trace
```

Metoda sintakse *$trace attach fileID* pridružuje datoteku objektu klase *trace* tako da se događaji bilježe u zadanu datoteku. Odabrana datoteka mora biti otvorena za upisivanje.

Simulacija se pokreće naredbom *ns run.* Izvršavanje traje sve dok postoje događaji koji se trebaju izvršiti, ili dok je sami ne prekinemo naredbom *ns stop*. Pretpostavimo da želimo trajanje simulacije od 10 sekundi:

```
ns at 10.0 “exit 0”
ns run
```

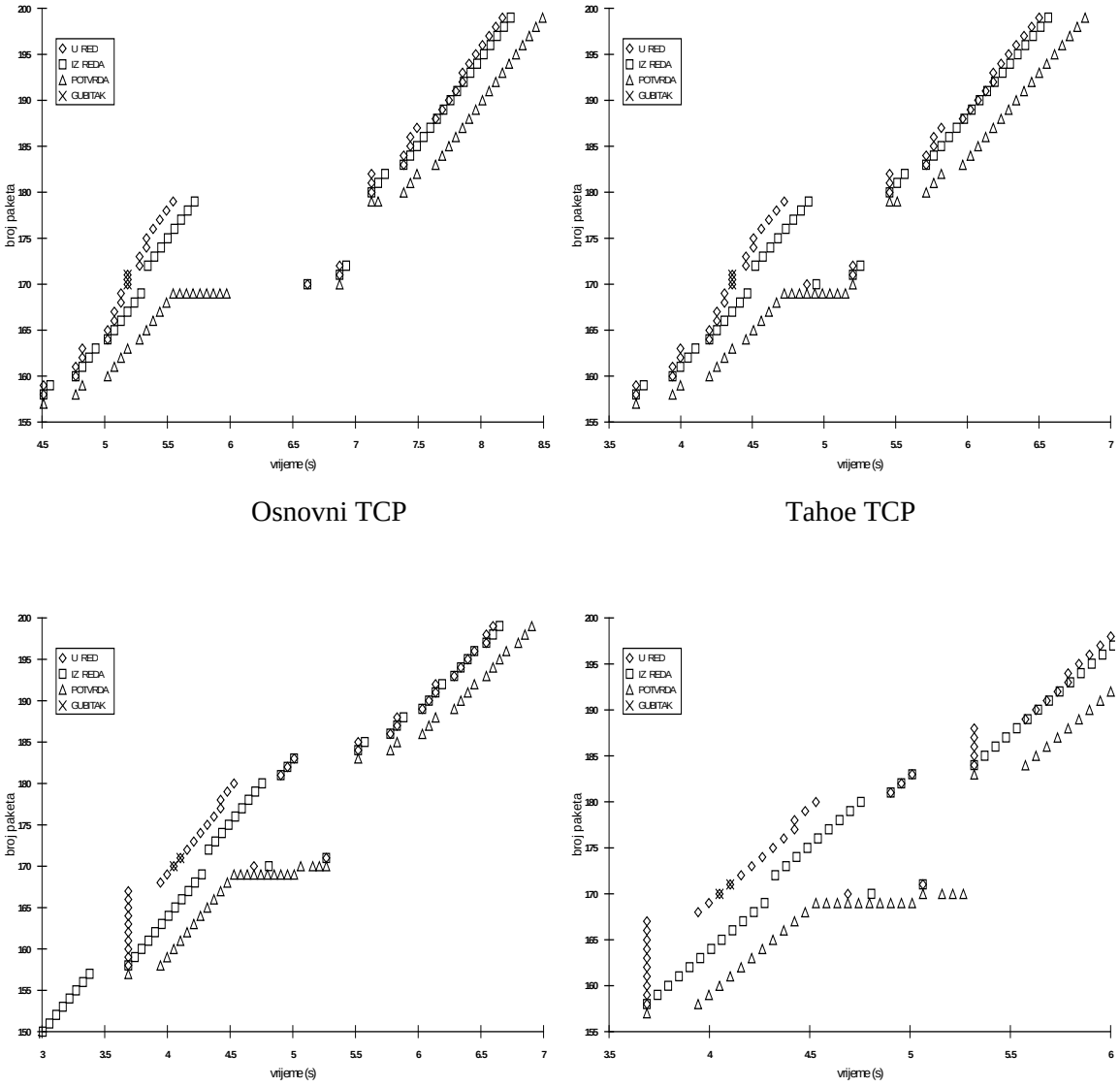
7.4 ZADATAK

Simulirati ponašanje različitih varijanti TCP protokola na jednostavnoj mreži i komentirati rezultate.

7.5 PRIMJER

Na kanalu kapaciteta 150 kb/s i kašnjenja 100 ms izazvan je gubitak segmenata 70 i 71. Snimke slanja segmenta (U RED), emitiranja segmenta (IZ REDA) i prijema potvrde (POTVRDA), te gubitaka

(GUBITAK) dane su na vremensko-prostornim grafovima, gdje je dimenzija prostora redni broj paketa, slika 7.8.



Slika 7.8. Veza 150 kb/s, 100 ms, 70/2 gubitka

Samo NEW-RENO TCP uspijeva retransmitirati izgubljene segmente bez gubitka kapaciteta kanala.