PRIJENOSNI SLOJ INTERNETA

Stjepan Groš

07. 09. 2006.

Sadržaj

1. Uvod

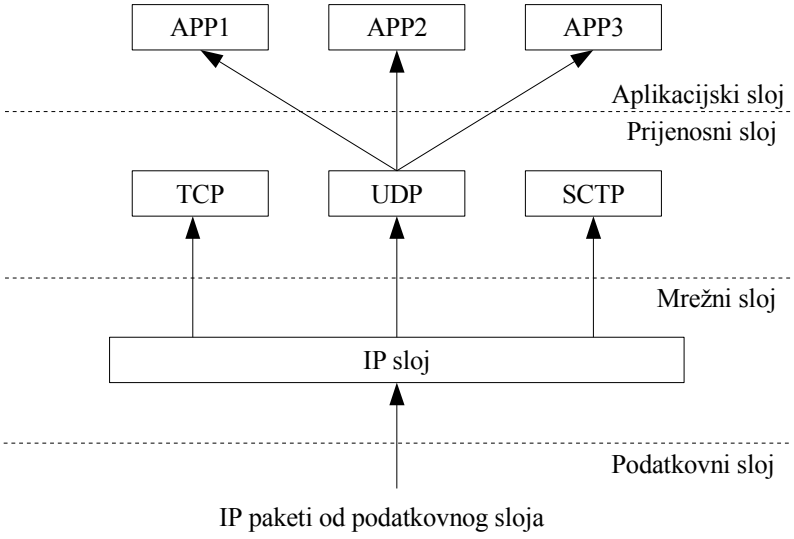
Evolucija Interneta.

Karakteristike prijenosnog protokola:

2. Bespojne usluge Interneta

2.1. UDP

UDP je vrlo jednostavan protokol prijenosnog sloja koji efektivno omogućava aplikacijama direktno korištenje usluga mrežnog sloja. Razlog uvođenja UDP-a umjesto direktnog korištenja IP protokola je multipleksiranje između različitih aplikacija. Shematski, problem čije rješenje zahtijeva demultipleksiranje prikazuje slika 1. Već smo upoznati s načinima demultipleksiranja na relaciji podatkovni-mrežni sloj i mrežni-prijenosni sloj.



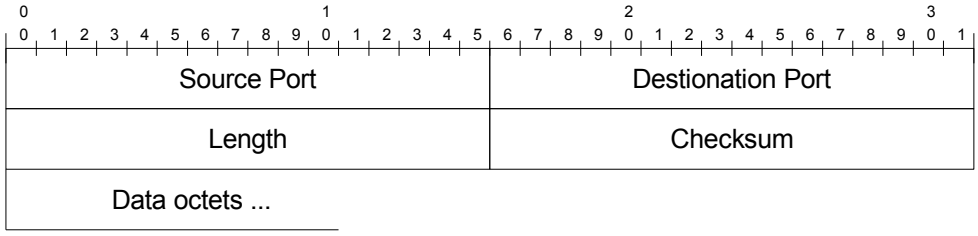
Slika 1. Demultipleksiranje UDP prometa za pojedine aplikacije

Demultipleksiranje na granici UDP-Aplikacije obavlja se uz pomoć sljedeće četvorke:

(lokalna IP adresa, udaljena IP adresa, lokalni pristup, udaljeni pristup)

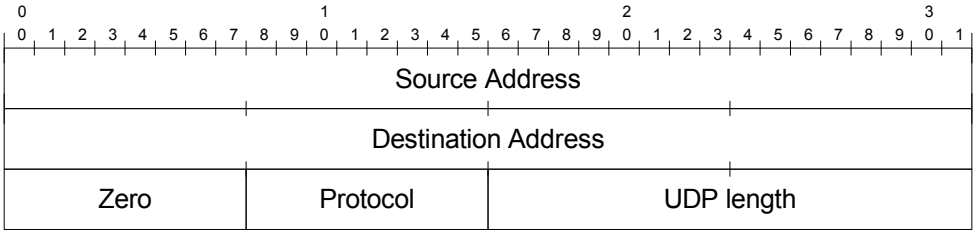
Uz pomoć lokalnog pristupa i lokalne IP adrese identificirana je jedna aplikacija (primjerice APP2), dok je uz pomoć udaljene IP adrese i udaljenog pristupa omogućeno paralelno ostvarivanje više UDP "veza" na istu aplikaciju. Drugim riječima, jedna aplikacija, preko jednog pristupa može istovremeno komunicirati s dvije ili više aplikacija na udaljenim strojevima!

UDP protokol definira svoje zaglavlje u koje su upisani izvorišni i odredišni pristupi. Format tog zaglavlja prikazuje slika 2. Dakle, ako je paket odlazni tada se lokalni pristup nalazi upisan u izvorišnom polju, dok ako je dolazni tada se nalazi u odredišnom polju. Slično vrijedi i za udaljeni pristup. IP adrese UDP protokol preuzima od mrežnog sloja, tj. iz IP paketa. Treba primjetiti da se time krši princip nezavisnosti slojeva.



Slika 2. Format paketa UDP protokola

Osim izvorišnog i odredišnog pristupa, u UDP zaglavlju nalazi se i ukupna duljina paketa te zaštitna suma. Zaštitna suma uključuje i IP adrese odredišta i izvorišta, a za njeno izračunavanje koristi se pseudo-zaglavlje, dakle zaglavlje koje se privremeno kreira kako bi se izračunala navedena suma, ali se to zaglavlje ne koristi u daljnjem procesiranju, tj. ne šalje na mrežu.



Slika 3. Pseudo zaglavlje UDP protokola za izračun zaštitne sume (checksum)

RFC768, RFC3838

2.1.1. UDP Lite

2.1.2. DCCP

Datagram Congestion Control Protocol, u vrijeme pisanja ovog teksta još uvijek u statusu drafta,

3. Spojne usluge Interneta

3.1. TCP

Transmission Control Protocol, ili skraćeno TCP, dominantan je prijenosni protokol Interneta. Svoju popularnost može zahvaliti primarno Webu iako se koristi i za mnoge druge primjene.

Osnovna svojstva usluge koju nudi TCP su sljedeća:

- pouzdanost,
- veza od točke-do-točke,
- dvosmjerni prijenos podataka,
- svi podaci tretiraju se kao niz okteta, bez ikakvih granica u smislu poruka, paketa i slično.

TCP je nasljednik NCP protokola nastao 1981. godine i od tada je u nekoliko navrata poboljšavan. Primarna namjena protokola je ostvarivanje pouzdane, dvosmjerne veze između bilo koja dva entiteta na Internetu.

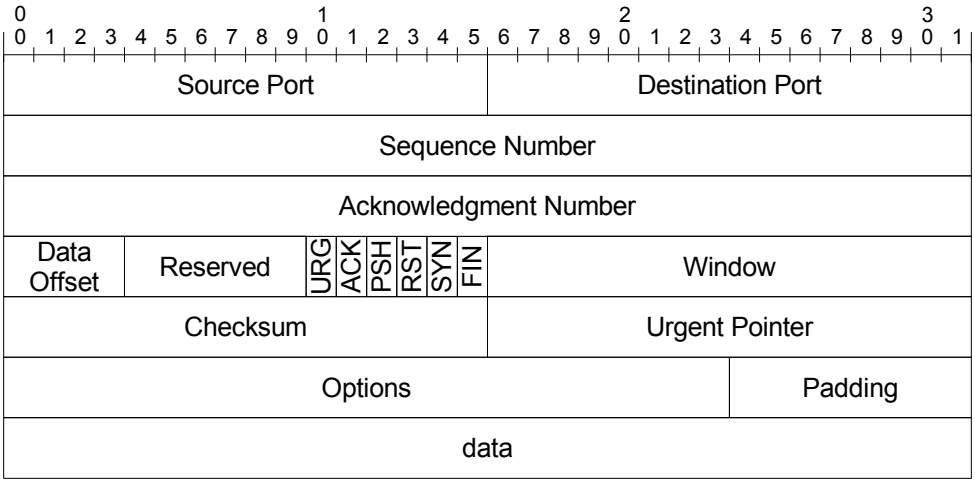
U nastavku teksta će se po pojmom *TCP entitet* smatrati konkretna implementacija standarda na nekom operacijskom računalu. Budući da na jednom računalu više aplikacija može istovremeno koristiti usluge TCP entiteta uvedeno je multipleksiranje putem *pristupa* (engl. port). Pristup je 16-bitni cijeli broj u rasponu od 1 do 65535. Svaka aplikacija koja želi koristiti usluge TCP protokola odabire ili joj se dodjeljuje jedan slobodan broj iz tog raspona. Tada je ta strana veze jedinstveno identificirana s parom (IP adresa, port). Ponekad se to piše i u obliku ipadresa.port, primjerice 10.0.0.1.1024. Svaka veza potpuno je određena s dva para (IP adresa, port), po jedan za svaku stranu.

Na Internetu poslužitelji nude određene usluge klijentima, primjerice uslugu elektroničke pošte ili Web. Poslužitelji koji nude usluge su aplikacije koje na određenom pristupu čekaju zahtijeve klijenata. Kako svaka aplikacija koja koristi usluge TCP protokola može čekati na bilo kojem pristupu u rasponu od 1 do 65535 klijentu je teško pronaći uslugu na pojedinom poslužitelju. Kako bi klijent pristupio poslužitelju, osim naziva ili adrese poslužitelja, mora znati i pristup. No, može se lako desiti da je poslužiteljska aplikacija u međuvremenu pokrenuta ponovo te da čeka na nekom drugom pristupu. Kako bi se riješio taj problem i olakšalo korištenje standardnih usluga rezervirani su određeni *poznati pristupi* (engl. well known ports). Popis pristupa i usluga koje se nalaze na tim pristupima dostupne su preko udruge [IANA04] Primjerice, elektronička pošta ima dodjeljen pristup 25, dok je Web poslužiteljima dodjeljen pristup 80. Na računalima koja izvršavaju Unix operacijski sustav popis standardnih usluga nalazi se u datoteci `/etc/services`.

U TCP protokolu jedinica podataka je segment, tj. razmjena podataka obavlja se u segmentima koji se potom pakiraju u IP pakete i šalju preko mreže. Minimalna veličina segmenta je 20 okteta i to je segment koji sadrži samo zaglavlje TCP paketa. Maksimalna veličina je .

Problemi s kojima se susreće TCP su: velik raspon vrijednosti RTT parametra na vrlo maloj vremenskoj skali, mogućnost dolaska segmenata u izmjenjenom redoslijedu i varijacije u količini resursa koju svaka veza ima na raspolaganju.

Struktura segmenta prikazana je na slici 4.

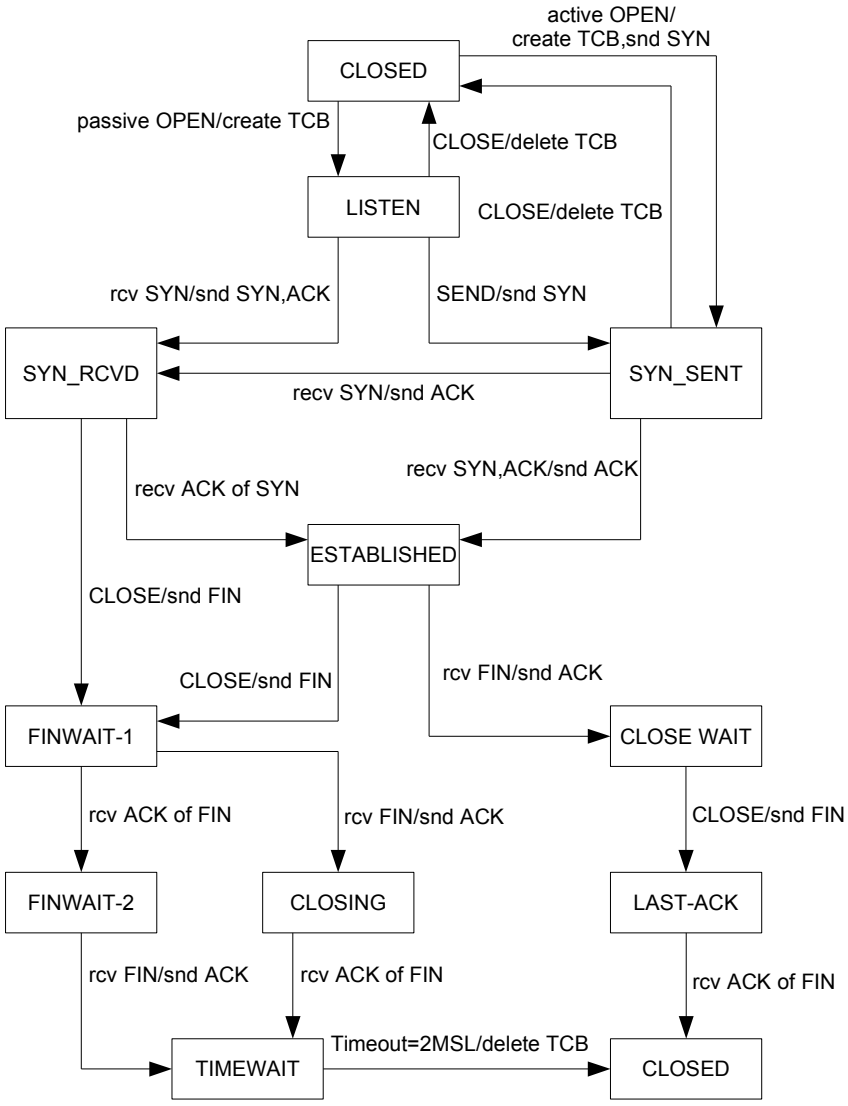


Slika 4. Format TCP segmenta

Tablica 1. Popis polja u zaglavlju TCP segmenta s kratkim opisom

Polje	***Opis***
Source Port	Izvorišni pristup, tj. pristup s kojega se šalje segment.
Destination Port	Odredišni pristup, tj. pristup na koji se šalje segment.
Sequence Number	Slijedni broj prvog okteta podataka u segmentu.
Acknowledgment Number	Sljedeći očekivani oktet podataka. Polje je važeće samo ako je postavljena zastavica ACK.
Data Offset	Broj 32-bitnih riječi u zaglavlju TCP segmenta. Efektivno označava početak podataka.
Reserved	Polje rezervirano za buduću upotrebu. Mora biti postavljeno na 0.
Flags	Zastavice URG, ACK, PSH, RST, SYN i FIN.
Window	Broj okteta započevši sa slijednim brojem navedenim u Acknowledgment Number polju koje je pošiljatelj segmenta spreman prihvatiti od druge strane.
Checksum	Kontrolna suma pseudo zaglavlja segmenta.
Urgent Pointer	Pokazivač na kraj hitnih podataka u TCP segmentu. Polje je važeće samo ako je postavljena zastavica URG.
Options	Opcije.

TCP za svoj rad koristi samo jedno zaglavlje te je zbog toga to zaglavlje dosta kompleksno. Ipak, ne koriste se stalno svi elementi zaglavlja.



Slika 5. Automat stanja TCP entiteta

RFC793, RFC872, RFC879, RFC896, RFC962, RFC1078, RFC1106, RFC1110, RFC1144, RFC1146, RFC1263, RFC1323, RRC1337, RFC2140, RFC2398, RFC2415, RFC2416, RFC2488, RFC2525, RFC2581, RPC2582, RFC2760, RFC2861, RFC2873, RFC2923, RFC2988, RFC3042, RFC3360, RFC3448, RFC3449, RFC3465, RFC3481, RFC3517, RFC3522, RFC3649, RFC3708, RFC3742, RFC3782,

Ekstenzije: RFC1693

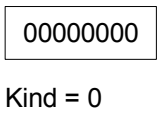
Opcije: RFC2018, RFC2385, RFC2883, RFC3562

3.1.1. Opcije

TCP protokol definira mehanizam opcija koje omogućavaju transparentno proširenje protokola. U originalnoj specifikaciji protokola [RFC0793] definirane su tri opcije koje se moraju implementirati,

to su *End of Options List*, *No-Operation* i *Maximum Segment Size*. Svaka opcija započinje sa kodom pri čemu iza koda opcije, ako postoje parametri, tada dolazi još jedan oktet koji sadrži ukupnu duljinu opcije, te parametri opcije.

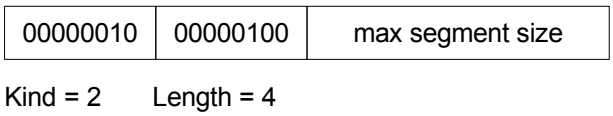
Format opcije *End of Options List* prikazan je na slici 6. Ta opcija označava kraj opcija i početak podataka.



Slika 6. Format opcije End of Option List

Opcija No-Operation nema nikakvo posebno značenje. Promjerice, može koristiti u slučajevima kada pošiljatelj želi poravnati iduću opciju na početak 32-bitne riječi. Ipak, primatelj TCP segmenta nema nikakve garancije da će opcije biti poravnate te mora biti spreman i na takve situacije.

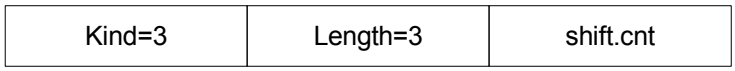
Opciju *Maximum Segment Size* koristi TCP entitet kako bi drugoj strani objavio maksimalnu veličinu segmenta koju je spreman prihvatiti. U maksimalnu veličinu segmenta uračunati su samo podaci, bez zaglavlja TCP protokola [RFC0879]! Svaka strana slobodna je odrediti svoju maksimalnu veličinu segmenta [RFC0879], a podrazumijevana vrijednost u slučaju da opcija nije navedena je 536 (minimalna veličina IP paketa od 576 okteta umanjena za 40 okteta zaglavlja IP i TCP protokola). Format opcije prikazan je na slici 7. Ukupna veličina opcije je 32 bita pri čemu prvih 8 bita predstavlja kod opcije, drugih 8 bita sadrži veličinu opcije i potom dolazi 16 bita u kojima je upisana



Slika 7. Format opcije Maximum Segment Size

maksimalna veličina segmenta koju je pošiljatelj spreman prihvatiti. Ova opcija smije se pojaviti samo u inicijalnim TCP segmentima.

Iduća bitna opcija je *Window Scale*. Naime, maksimalna veličina prozora prema temeljnoj TCP specifikaciji je 65536 okteta. To je dovoljno za veze čiji produkt propusnosti i kašnjenja nije velik. Međutim, porastom vrijednosti tog produkta raste i neefikasnost TCP protokola. Primjerice, na vezi čija propusnost je 1Gb/s i kašnjenje 1ms u svakom trenutku u prometu može biti oko 130KB. To je daleko više od prozora koji nudi TCP te bi prema tome veza pola vremena bila prazna. Problem se rješava navedenom opcijom Window Scale. Format te opcije prikazuje slika 8.



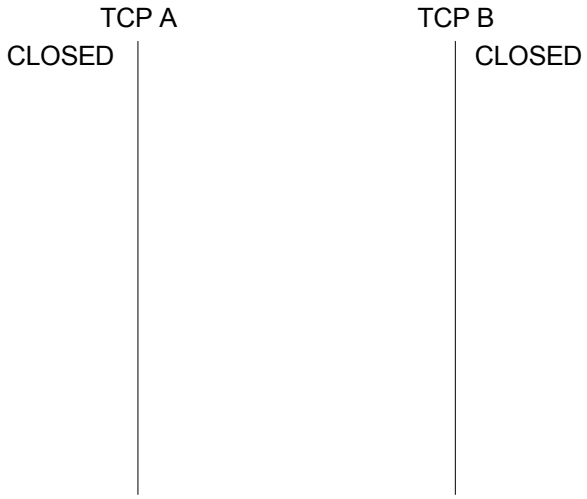
Slika 8. Format opcije Window Scale

3.1.2. Faze u korištenju TCP usluge

U nastavku teksta opisuju se razmjene paketa za pojedine faze korištenja usluge TCP protokola. Razmjena paketa bit će grafički prikazana uz pomoć dijagrama danog na slici 9.

Na slici vrijeme teče prema dolje. Linije na slici označavaju mjesto s kojega se šalju i na koje se primaju TCP segmenti. Prostorna udaljenost između klijenta i poslužitelja simbolizirana je razmakom

između dviju linija. S vanjskih strana obiju linija u većini slučajeva će biti označeno stanje u kojemu se nalazi odgovarajući TCP entitet. Pod izrazom TCP entitet podrazumijevamo konkretnu TCP implementaciju, primjerice implementaciju unutar Linux operacijskog sustava.



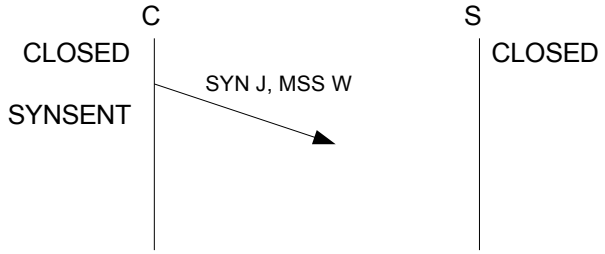
Slika 9. Dijagram za prikaz razmjene TCP segmenata

Za segmente koji se razmjenjuju između entiteta pretpostavlja se da uvijek sadrže postavljene ispravne vrijednosti izvorišnih i odredišnih pristupa, međutim, na slikama se ti pristupi neće pojavljivati. Također, treba uzeti u obzir kako jedan konkretan dijagram vrijedi isključivo za jednu vezu.

Prilikom korištenja TCP usluge odgovarajući entitet prolazi kroz tri osnovne faze: (i) uspostava veze, (ii) razmjena podataka i (iii) prekid veze. Svaka od tih faza detaljnije je objašnjena u idućim potpoglavljima.

3.1.2.1. Uspostava veze

Prije uspostave veze poslužitelj ulazi u stanje LISTEN, dok je klijent u stanju CLOSED. Za uspostavu veze u TCP protokolu potrebno je razmijeniti tri segmenta. Prvi segment šalje klijent, odnosno strana koja inicira vezu. U tom segmentu postavljena je zastavica SYN (*synchronize*). Također, u tom segmentu polje *sequence number* sadrži inicijalnu vrijednost slijednog broja koju će klijent koristiti za numeriranje podatka koje šalje. Odmah nakon slanja klijent prelazi u stanje SYNSENT. Na slici 10 je simbolički prikazan segment koji klijent šalje poslužitelju i to u nekom trenutku dok segment još nije stigao do poslužitelja.



Slika 10. Trenutak između slanja SYN segmenta i prijema od strane poslužitelja

Na slici su uz poslani segment označeni i osnovni podaci koji se šalju. U slučaju SYN segmenta podaci su inicijalni slijedni broj (J) i maksimalna veličina segmenta koju klijent prihvaća (W). Treba primjetiti kako nakošenost strelice predstavlja vrijeme potrebno da segment stigne na odredište. Što je

vrijeme propagacije dulje to je strelica koja označava segment nakošenija, tj. više naginje prema dolje.

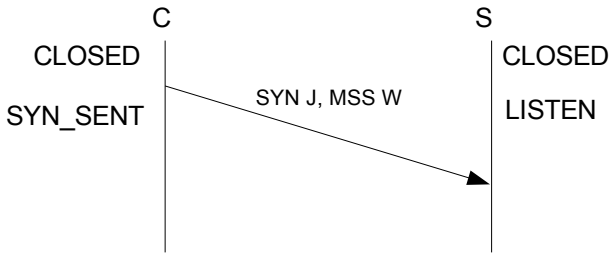
Za tekstualni prikaz TCP segmenta koristiti ćemo sljedeći zapis:

TCP(S_{PORT}, C_{PORT}, SYN|FIN|PUSH|<NULL>,
SEQ(N), ACK(N), URG(N), OPTS(LIST), DATA(N))

Prva dva parametra u zapisu predstavljaju odredišni pristup (S_{PORT}) i izvorišni pristup (C_{PORT}). Nakon toga dolazi popis postavljenih zastavica. U ovom slučaju vodi se evidencija samo o SYN i FIN zastavicama budući da se iz ostatka paketa ne može razlučiti da su postavljene. Nakon zastavica dolazi slijedni broj te potvrda. Ako nije postavljena zastavica ACK tada se slijedni broj potvrde ne piše. Iza potvrde je pokazivač na količinu hitnih podataka. I kod tog pokazivača njegova prisutnost implicira postavljenu zastavicu URG, a ako tog podatka nema tada ni zastavica nije postavljena. Popis opcija nalazi se unutar OPTS liste, a na kraju dolazi oznaka količine podataka. Ako je ta oznaka izostavljena tada znači da veličina podataka nije bitna ili da je nula.

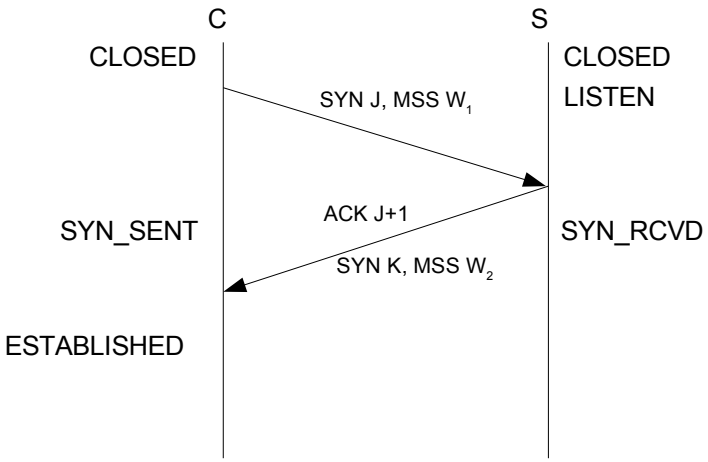
Korištenjem navedene notacije situacija sa slike 10 bila bi napisana na sljedeći način:

TCP(S_{PORT}, C_{PORT}, SYN,SEQ(J), OPTS(MSS(W)))



Slika 11. Trenutak primanja SYN segmenta od strane poslužitelja

Trenutak kada poslužitelj primi navedeni segment prikazan je na slici 11. Odmah pošto primi zahtijev za uspostavom veze poslužitelj šalje segment sa postavljenim SYN i ACK zastavicama. U poslanom segmentu polje slijednog broja sadrži vrijednost inicijalnog slijednog broja poslužitelja, polje potvrde sadrži vrijednost inicijalnog slijednog broja kojeg je primio od klijenta uvećan za jedan i definiranu maksimalnu veličinu segmenta koju je spreman prihvatiti. Odgovor poslužitelja grafički je prikazan na slici 12.

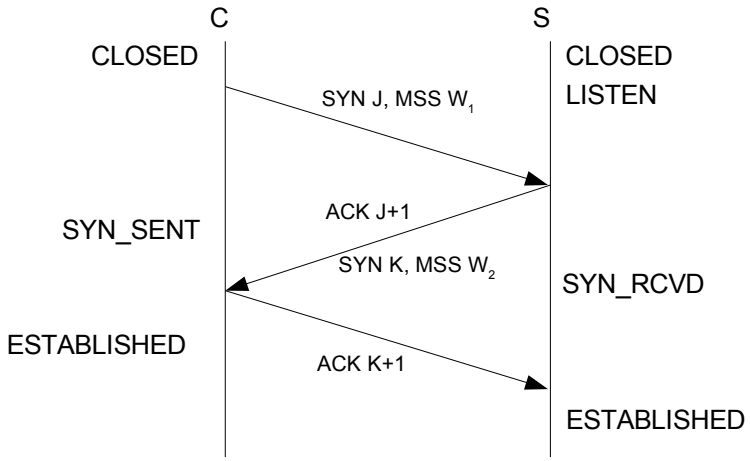


Slika 12. Završetak druge faze uspostave TCP veze

Tekstualni zapis ogovora je

TCP(S_{PORT}, C_{PORT}, SYN,SEQ(K), ACK(J+1), OPTS(MSS(W_2)))

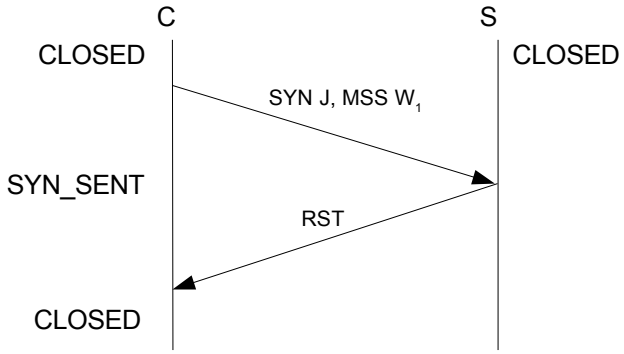
Zadnja faza uspostave veze je nastupa kada klijent pošalje ACK segment i potom ulazi u stanje ESTABLISHED. To znači da je spreman za slanje iza primanje podataka preko upravo uspostavljene veze. I poslužitelj nakon što primi ACK segment ulazi u stanje ESTABLISHED i spreman je za korištenje veze. Ukupna razmijena segmenata koja vodi do uspostave veze prikazana je na slici 13.



Slika 13. Kompletna razmjena segmenata tijekom uspostave TCP veze

Razlog razmjene inicijalnih slijednih brojeva je smanjenje vjerojatnosti interferencije sa nekom starom vezom. TCP specifikacija zahtijeva da obje strane odaberu slučajne brojeve za početne slijedne brojeve.

Moguće je da poslužitelj odbije uspostavljanje veze. Primjerice, takva situacija može se pojaviti ako poslužitelj nije u LISTEN stanju, ako mu je red zahtijeva za vezom prevelik, itd. Odbijanje veze obavlja se slanjem segmenta kojemu je postavljena RST (*reset*) zastavica. Taj slučaj shematski je prikazan na slici 14.



Slika 14. Razmjena segmenata u slučaju odbijanja veze

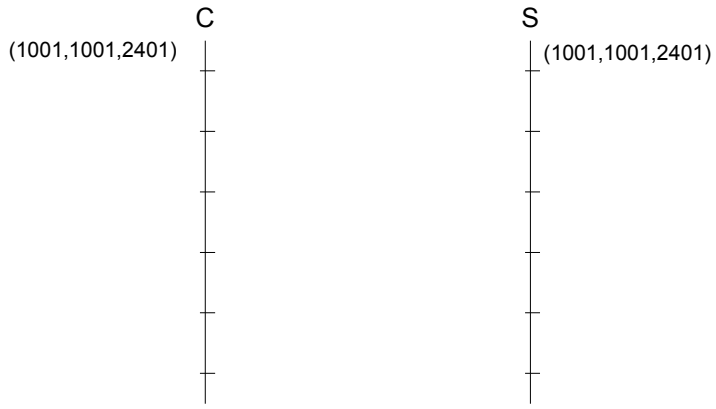
3.1.2.2. Razmjena podataka

Razmjena podataka obavlja se razmjenom segmenata. Segment može sadržavati samo podatke. Tada polje slijednog broja sadrži slijedni broj prvog okteta u segmentu. Također, segment može sadržavati i samo potvrdu već primljenih podataka. Tada je postavljena zastavica ACK, a u polju potvrde nalazi se slijedni broj prvog idućeg očekivanog okteta. Osim ta dva slučaja, moguća je i kombinacija, tj. jedan segment sadrži i podatke i potvrdu primljenih podataka.

Veličina segmenta ograničena je sa dva faktora. Prvi je maksimalna veličina segmenta koju je suprotna strana odredila prilikom uspostavljanja veze. Drugo ograničenje je trenutna veličina prozora koju prijemna strana dopušta. Odlučujuća je manja od te dvije vrijednosti.

U idućem primjeru biti će prikazan prijenos 800 okteta podataka od klijenta do poslužitelja. Podaci dogovoreni tijekom uspostave veze su: ISN (Inicijalni slijedni broj) je 1000, obje strane su odredile MSS (maksimalna veličina segmenta) 200 i WIN (maksimalna veličina prozora) je 1400. Klijent šalje maksimalan broj okteta bez primitka potvrde, a poslužitelj potvrđuje prvih 600 okteta i pri tome smanjuje veličinu prozora na 500 okteta. Za sada će biti zanemarivani utjecaji TCP zaglavlja na veličinu segmenta.

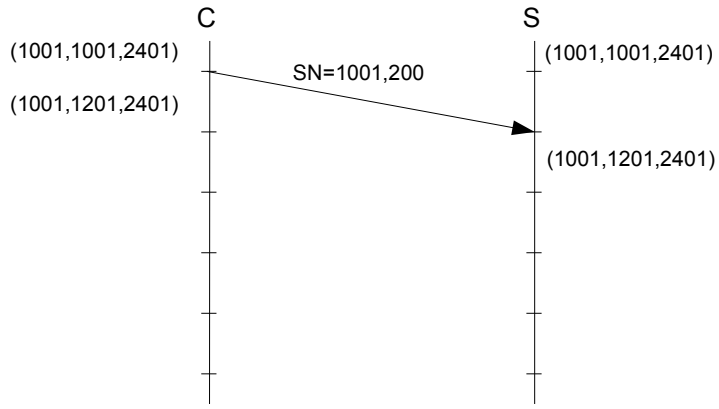
Inicijalno stanje, odmah nakon uspostave veze a neposredno prije početka slanja prikazano je na slici 15.



Slika 15. Inicijalno stanje TCP entiteta prije početka slanja

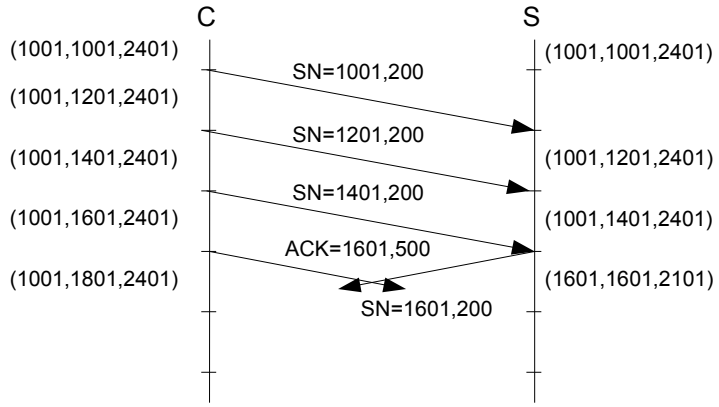
I na primjenoj i na predajnoj strani koriste se uređene trojke kako bi se opisalo stanje pojedinih entiteta. Te trojke odnose se samo na jedan smijer slanja podataka, tj. u ovom slučaju promatramo samo slučaj kada klijent šalje podatke. Ako bi htjeli istovremeno voditi evidenciju i o podacima koje šalje poslužitelj trebali bi još dvije uređene trojke, za smjer od poslužitelja do klijenta. Na predajnoj strani prvi član pokazuje na prvi nepotvrđeni oktet, drugi član pokazuje na prvi neposlani oktet i treći član pokazuje na gornju granicu prozora, tj. na prvi oktet koji se ne smije slati. Na prijemnoj strani prvi član uređene trojke pokazuje na prvi nepotvrđeni oktet, drugi član pokazuje na slijedni broj idućeg očekivanog okteta i treći član pokazuje na gornju granicu prozora, tj. na prvi oktet koji neće biti prihvaćen. Nakon slanja prvog segmenta situacija je prikazana na slici 16. Treba primjetiti kako je slijedni broj prvog neposlanog okteta 1001, a dogovoreni inicijalni slijedni broj je 1000! To je zbog toga što se i za uspostavljanje veze "potroši" jedan oktet.

Odmah nakon slanja segment klijent je povećao pokazivač na prvi neposlani oktet (drugi član uređene trojke), a poslužitelj je nakon primanja segmenta povećao svoj pokazivač na idući očekivani oktet (također drugi član uređene trojke). Na strelici koja označava poslani segment nalazi se podatak o prvom oktetu u segmentu (1001), te duljina podataka (200).



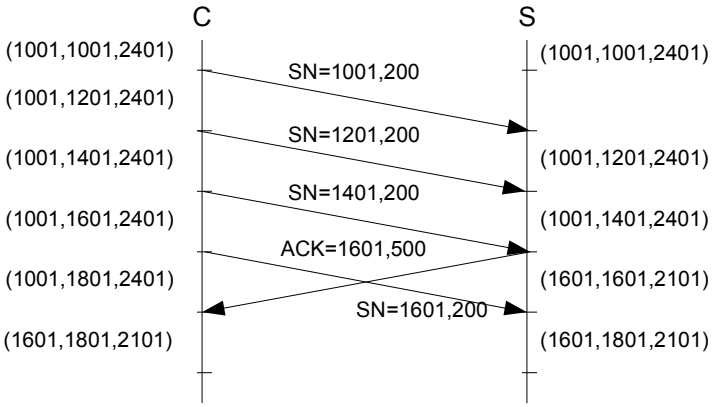
Slika 16. Stanje nakon slanja prvog segmenta podataka

Situacija nakon slanja prva četiri podatkovna segmenta i segmenta sa potvrdom od strane poslužitelja prikazana je na slici 17.



Slika 17. Stanje veze nakon slanja prvi 800 okteta i potvrde poslužitelja

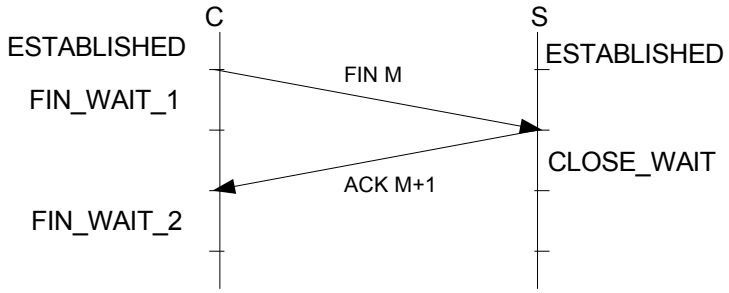
Na strelici koja označava segment što ga šalje poslužitelj nalazi se oznaka ACK=1601,500. Ta oznaka označava da se radi o potvrdi (ACK), da se potvrđuju svi okteti uključivo sa 1600-tim, te da se veličina prozora mijenja na 500 okteta. Promjena veličine okvira vidljiva je na poslužitelju koji je u trećem članu zapisao vrijednost 2101 što znači da je to slijedni broj prvog okteta kojega neće prihvatiti. Istovremeno kada je poslužitelj poslao segment i klijent je poslao dodatni podatkovni segment. Konačna situacija, nakon što oba segmenta dostignu odredište, prikazana je na slici 14.



Slika 18. Shematski prikaz zadane razmjene podataka

3.1.2.3. Prekid veze

Prekid veze obavlja se razmjenom tri ili četiri segmenta, ovisno o situaciji. U trenutku kada neka strana želi prekinuti vezu šalje FIN segment, tj. segment sa postavljenom FIN zastavicom. Shematski će biti prikazana situacija kada klijent želi prekinuti vezu, a kasnije i server prekida vezu. Na slici 18 prikazana je razmjena segmenata koja uzrokuje prekid na klijentskoj strani. Nakon zatvaranja klijentske strane poslužitelj i dalje može slati podatke. U trenutku kada i poslužitelj želi zatvoriti vezu ponavlja istovjetan postupak, tj. šalje FIN segment. Ta situacija prikazana je na slici 19.



Slika 19. Klijent zatvara svoju stranu TCP veze

Preporuke kada generirati ACK potvrde:

Događaj	***Akcija TCP prijemnika***
Dolazak *in-order* segmenta sa očekivanim slijednim brojem. Svi podaci do očekivanih već potvrđeni. Nema praznina u primljenim podacima.	Odgođeni ACK (delayed ACK). Pričekati do 500ms za idući segment. Ako idući segment ne pristigne u tom periodu tada pošalji ACK.
Dolazak *in-order* segmenta sa očekivanim slijednim brojem. Jedan *in*-order segment već čeka ACK. Nema praznina u primljenim podacima.	Odmah pošalji jedan kumulativni ACK, čime se potvrđuju oba segmenta.
Dolazak segmenta van redoslijeda s većim od očekivanog slijednog broja. Detektirane praznine u primljenim podacima.	Odmah pošalji duplicirani ACK sa idućim očekivanim slijednim brojem okteta.
Dolazak segmenta koji djelomično ili potpuno popunjava praznine u primljenim podacima.	Odmah pošalji ACK uz pretpostavku da segment započinje na doljnjem dijelu praznine.

Triple ACK

3.1.3. Proširenja TCP protokola

3.1.3.1. Selektivna potvrda

3.1.4. Parametri TCP veze

3.1.4.1. Vrijeme do povratka potvrde

Vrijeme do povratka povrde (engl. Round Trip Time, RTT) je parametar čiju točnu vrijednost je dosta teško utvrditi. U specifikaciji TCP protokola predložen je jednostavan način izračunavanja RTT-a. Ako je *SampleRTT* izmjereno vrijeme od slanja segmenta do dolaska potvrde, a *EstimatedRTT*

dosadašnja procijenjena vrijednost parametra RTT, tada je nova vrijednost procjene dana sljedećim izrazom:

$$EstimatedRTT = \alpha \times EstimatedRTT + (1-\alpha) \times SampleRTT$$

Parametar α određuje kako se ponaša procjena. Ako parametar α ima malu vrijednost tada će stare vrijednosti imati veći utjecaj te će procjena biti neosjetljiva na kratkotrajne varijacije RTT-a. Međutim, to znači i da će se procjenjena vrijednost RTT-a slabije prilagođavati trenutnoj u slučaju brzih promjena. S druge strane, veliki parametar α znači da novija mjerenja imaju veći utjecaj na procjenu, ali i da će kratkotrajne velike fluktuacije RTT-a imati velik utjecaj. Prema specifikaciji [RFC0793] vrijednost parametra je između 0.8 i 0.9. Na osnovu procjenjene vrijednosti RTT parametra postavlja se vremensko ograničenje na sljedeću vrijednost:

$$TimeOut = 2 \times EstimatedRTT$$

Dobivena vrijednost vrlo je konzervativna, a tijekom upotrebe iskazao se je i jedan vrlo bitan problem, naime ACK potvrđuje prihvat segmenata, ali u slučaju retransmisije ne može se utvrditi kojega. Ako se pretpostavi da se potvrda odnosi na prvo slanje segmenta tada je moguće dobiti preveliki RTT, a samim tim i vrlo velika vremenska ograničenja pa će sustav sporo reagirati na izgubljene segmente u mreži. S druge strane, ako se pretpostavi da je pristigla potvrda za ponovno poslani segment moguće je dobiti premalo procjenjeno vrijeme RTTa, a samim tim i prekratko vremensko ograničenje zbog kojega će se nepotrebno slati segmenti na mrežu.

Rješenje navedenog problema je jednostavno, a naziva se *Karn/Partridge algoritam*, prema svojim autorima. Za sve segmente koji su poslani više od jednom prestaje se sa mjerenjem vremena potrebnog da stigne ACK. Osim te promjene, isti autori predložili su da se prilikom ponovnog slanja nekog segmenta upotrijebi dvostruko prethodno vremensko ograničenje bez obzira na trenutnu procjenu, slično kao kod Ethernet mreže.

Iduće poboljšanje TCP protokola predstavlja Jacobson/Karels algoritam. Primarna namjena tog algoritma je kontrola zakrčenja – što je objašnjeno u idućem potpoglavlju, međutim, određene promjene vezane su i za procjenu RTTa.

Glavni problem izračuna vremenskog ograničenja predloženog u TCP specifikaciji je činjenica da ne uzima u obzir podatak o varijaciji. Kako bi se i taj podatak uzeo u obzir predložena je sljedeća promjena. Pošiljatelj i dalje mjeri *SampleRTT* kao i prije, ali sada je izračun nešto drugačiji:

$$Difference = SampleRTT - EstimatedRTT$$

$$EstimatedRTT = EstimatedRTT + (\delta \times Difference)$$

$$Deviation = Deviation + \delta(|Difference| - Deviation)$$

Vremensko ograničenje se računa na sljedeći način:

$$TimeOut = \mu \times EstimatedRTT + \phi \times Deviation$$

Parametar delta ima vrijednost između 0 i 1, μ je 1, a φ je 4.

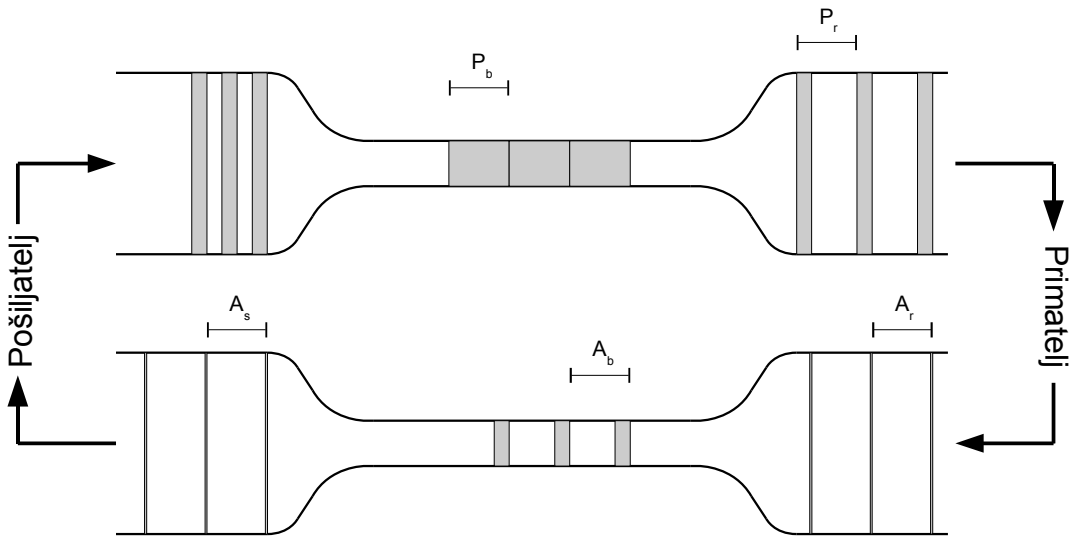
FIXME: *Implementacija cjelobrojnim brojevima!*

3.1.5. Kontrola zakrčenja

U 10. mjesecu 1986. godine na Internetu je došlo do prve ozbiljnije pojave zakrčenja [JACOB88]. Tada je efektivna brzina pojedinih veza opadala i za faktor 1000. Istraživanjem te pojave kao glavni uzrok označeni su protokol TCP i ponašanje implementacije u BSD operacijskom sustavu. Kako bi se izbjeglo ponavljanje zakrčenja predložene su odgovarajuće promjene TCP protokola koje su od onda dosta razrađene.

Ovom prilikom biti će ponovo napomenuta razlika između kontrole toka i kontrole zakrčenja. Algoritmi kontrole toka štite prijemnik od preopterećenja koje može uzrokovati predajnik. Primjerice, predajnik može biti nekoliko puta jače i brže računalo od prijemnika te se može desiti da prijemnik ne stigne prihvatiti i obraditi sve pristigle podatke. U tom slučaju prijemnik korištenjem kontrole toka može privremeno zaustaviti predajnik. S druge strane, namjena algoritama kontrole zakrčenja je zaštita same mreže, tj. sprečavanje svih predajnika od slanja previše podataka u mrežu.

Temelj razvijenih algoritama za izbjegavanje zakrčenja predstavlja zapažanje kako TCP na osnovu ACK segmenata može regulirati količinu podataka koju šalje u mrežu. To je objašnjeno uz pomoć slike 20.



Slika 20. Shematski prikaz djelovanja ACK segmenata za samokontrolu slanja TCP protokola

Na slici je propusnost pojedinih dijelova mreže simbolički prikazana kao odgovarajuća širina "cijevi". I pošiljatelj i primatelj nalaze se na brzim dijelovima mreže pa im je prema tome širina cijevi dosta velika. Međutim, mreža sadrži i spore (ili zakrčene) dijelove koji su dosta sporiji, a simbolički je to prikazano sužavanjem cijevi. Neka pošiljatelj pošalje 3 segmenta velikom brzinom (mali razmak između segmenata), neke veličine. Podaci koje segment prenosi prikazani su sjenčano pri čemu za konstantnu veličinu segmenta zbog brže veze će biti uži prikaz segmenta. Obratno, što je manja brzina mreže, za istu veličinu segmenta on će zauzimati veću površinu budući da dulje traje njegov prijenos.

Kada tri poslana segmenta stignu u uži dio cijevi (sporiji dio mreže) kreću se sporije zbog čega su širi i efektivno povećava se razmak između njih na P_b. Nakon izlaska iz sporog dijela mreže dolaze ponovo u brži dio gdje imaju malu širinu zbog velike brzine, ali im razmak P_r ostaje očuvan, tj. jednak P_b! Ako pretpostavimo najgori slučaj za mrežu, tj. da primatelj odmah odgovara s potvrdom za svaki primljeni segment, tada će razmak između potvrda ostati očuvan i identičan razmaku s kojim su primljeni podatkovni segmenti. U slučaju da primatelj ne odgovara na svaki segment odmah tada će se razmak između potvrda samo povećati.

Dakle, očito je kako primatelj na osnovu primljenih potvrda ima uvid u stanje mreže! Prikazana situacija vrijedi za stabilno stanje veze, tj. nakon što je prijenos podataka trajao određeno vrijeme. Problem je trenutak kada kreće prijenos podataka, bilo zbog tek uspostavljene veze ili zbog njene dulje neaktivnosti. U takvim situacijama predajnik nema informaciju o stanju u mreži i može se desiti da pošalje previše podataka što dovodi do zakrčenja, nepotrebnih retransmisija i opadanja efikasnosti prijenosa. Kako bi se riješio taj problem uvedene su neke tehnike kao što je *polagani start* (engl. slow-start), fast-retransmit i fast-recovery.

3.1.5.1. Additive Increase/Multiplicative Decrease

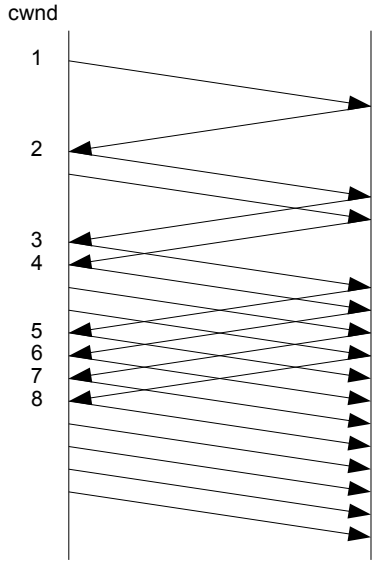
3.1.5.2. Polagani start

Algoritam *polaganog starta* (engl. slow start) služi za izbjegavanje zakrčenja i predstavlja dodatak na algoritam kontrole toka. Temeljna modifikacija koju uvodi taj algoritm predstavlja varijabla cwnd koja se prilikom uspostavljanja veze inicijalizira na vrijednost jednog maksimalnog segmenta. Kada TCP treba poslati podatke tada varijabla awnd određuje maksimalnu količinu koja se šalje bez čekanja potvrde. Varijabla awnd izračunava se pomoću sljedećeg izraza:

$$awnd = min(cwnd, credit) \quad (1)$$

U tom izrazu, credit je količina podataka koju pošiljatelju može poslati prema pravilima kontrole toke. U slučaju da do trenutka kada se računa awnd još nije ništa poslano tada je to puna vrijednost prozora, a u suprotnom to je vrijednost prozora umanjena za poslane i još nepotvrđene podatke.

Iako se naziva algoritam polaganog starta ipak se dosta brzo dolazi do maksimanog prozora. Na slici 21 prikazana je vrijednost varijable cwnd čija vrijednost očito raste eksponencijalno kako prijenos odmiče.



Slika 21. Porast varijable cwnd nakon početka prijenosa podataka

U [RFC3390] razmatra se mogućnost inicijalnog postavljanja varijable cwnd na vrijednost zadanu sljedećim izrazom:

$$min(4 \cdot MSS, max(2 \cdot MSS, 4380\,bytes)) \quad (1)$$

U jednom trenutku broj segmenata koje je moguće poslati ograničava ili raspoloživi prozor ili gubitak neke potvrde. Gubitak potvrde uzima se kao indikacija zakrčenja u mreži te se

3.1.5.3. Fast Retransmit and Fast Recovery

3.1.6. Utjecaj elemenata mrežnog sloja na TCP

3.2. T/TCP

RFC1379, RFC1644

3.3. SCTP

TCP, kao dominantan protokol Interneta, ne zadovoljava potrebe koje se javljaju u telekomunikacijskim uslugama. Zbog nedostataka TCP-a određen broj aplikacija postizao je pouzdanu uslugu prijenosa upotrebom vlastitih mehanizama nad UDP-om. Neke od mana TCP protokola su:

s

Sve većom upotrebom Internet protokola u telekomunikacijama javila se potreba za uvođenjem novog protokola koji bi nadomjestio neke od nTCP

RFC3286 – An introduction to SCTP protocol

RFC3309 – Stream Control Transmission Protocol (SCTP) Checksum Change

RFC2960 – Stream Control Transmission Protocol

3.4. RTP

RFC3550, RFC3551, RFC3552

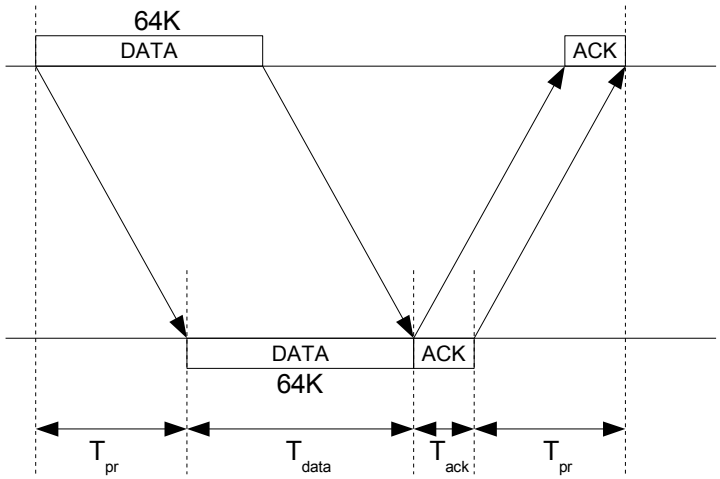
4. Primjeri

PRIMJER 1. Propusnost TCP-a sa standardnom veličinom prozora

Uz pretpostavku da TCP za prijenos koristi standardnu veličinu prozora odrediti maksimalnu propusnost i postotak teoretskog maksimuma u slučaju veze brzine 1Gb/s i kašnjenja 1ms. Kolika je potrebna veličina prozora za postizanje maksimalne propusnosti?

Rješenje

U standardu koji opisuje TCP maksimalna veličina prozora je 64K, bez upotrebe opcije za povećanje prozora. Ignorirati ćemo činjenicu kako je maksimalna veličina jednog segmenta 64K te da je u tu veličinu uključeno i zaglavlje. Bez te pretpostavke morali bi računati prijenos dodatnih 40 okteta za svaki segment što bi dodatno utjecalo na rezultat. Dakle, predajna strana može poslati maksimalno 65K i potom mora čekati na potvrdu kako bi poslala idućih 64K. Na slici je prikazan vremensko-prostorni dijagram za slanje jednog bloka od 64K.



Vrijeme potrebno za prijenos jednog bloka podataka i povratak potvrde primitka iznosi:

$$T_{uk} = T_{pr} + T_{data} + T_{ack} + T_{pr}$$

Odnosno, ako se uvrste veličine segmenta s podacima i potvrdom:

$$T_{uk} = 2\mathrm{T}_{pr} + 8\frac{F_{data} + F_{ack}}{B}$$

S F je označena veličina okvira u oktetima, a s B je označena brzina veze u b/s. U navedenom vremenu se dakle prenese 64KB pa je efektivan propusnost

$$B' = \frac{F_{data}}{2\mathrm{T}_{pr} + 8\frac{F_{data} + F_{ack}}{B}} = \frac{B\,F_{data}}{2\mathrm{T}_{pr}B + 8\mathrm{F}_{data} + 8\mathrm{F}_{ack}}$$

Uvrštavanjem zadanih iznosa dobijamo sljedeću vrijednost propusnosti

$$B' = \frac{1\mathrm{Gb}/s \cdot 65536}{2 \cdot 1\mathrm{ms} + 8 \cdot 65536\mathrm{B} + 8 \cdot 20\mathrm{B}} = 24.76\mathrm{MB}/s$$

Ili otprilike 20% teorijskog maksimuma.

Kako bi se postigla maksimalna moguća propusnost pošiljatelj nakon poslanih 64K mora imati mogućnost slanja sve dok ne pristigne potvrda. Dakle, veličina prozora mora biti

$$W = B \cdot T_{uk} = 2\mathrm{T}_{pr} + 8\frac{F_{data} + F_{ack}}{B} = 2\mathrm{BT}_{pr} + 8(F_{data} + F_{ack})$$

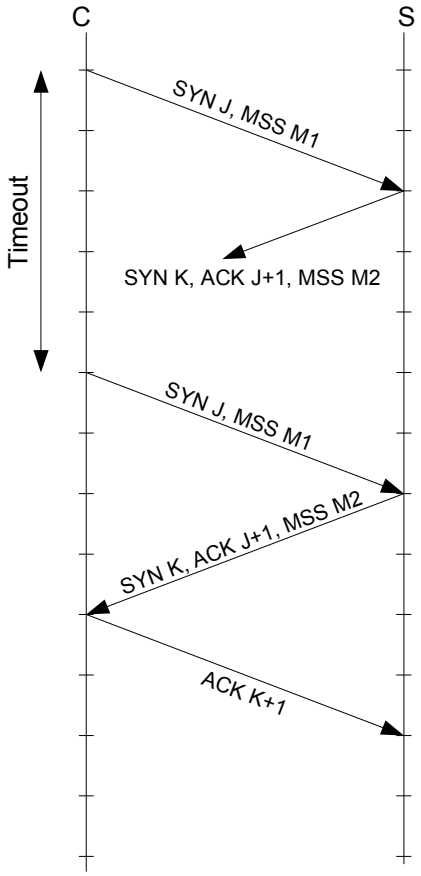
Ili kada se uvrste zadane vrijednosti

$$W = 2 \cdot 1\mathrm{Gb}/s \cdot 1\mathrm{ms} + 8(65536 + 20) = 315556\,okteta$$

PRIMJER 2. Razmjena segmenata pri uspostavi veze s gubitkom segmenata

Prikažite grafički uspostavu veze između dva TCP entiteta pri čemu se je 1. paket sa SYN i ACK zastavicama izgubio prije no što je stigao na odredište. Vremensko ograničenje na klijentu je 5 vremenskoh jedinica, isto kao i na poslužitelju. RTT iznosi 4 vremenske jedinice.

Rješenje

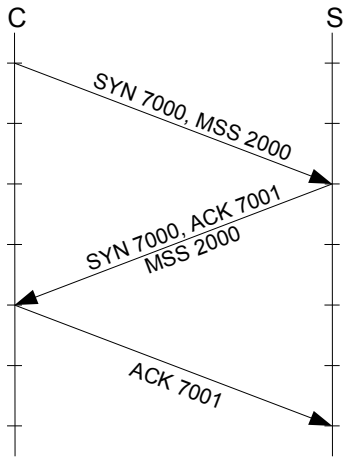


PRIMJER 3. Razmjena segmenata pri uspostavi veze s gubitkom segmenata

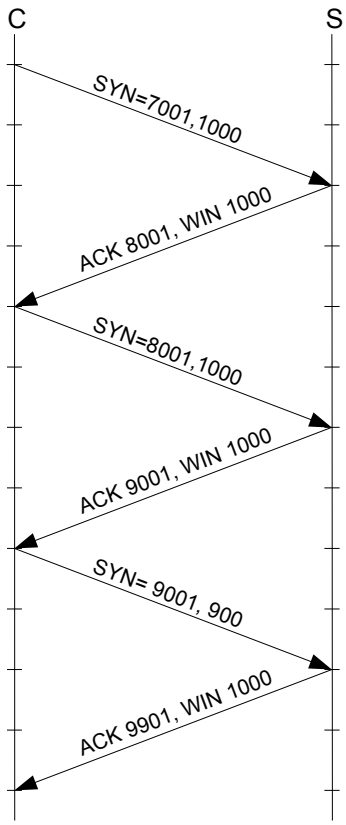
Klijent treba poslati poslužitelju 2900 okteta koristeći TCP. Tijekom uspostave veze poslužitelj i klijent definiraju istu maksimalnu veličinu segmenta od 2000 okteta te početne slijedne brojeve od 7000. Tijekom cijele komunikacije maksimalna veličina prozora je 1000 okteta. Nacrtati vremenski dijagram razmjene segmenata prilikom uspostavljanja veze, prijenosa podataka i raskidanja veze. Pretpostaviti da poslužitelj potvrđuje segment odmah nakon primitka, klijent šalje maksimalnu moguću količinu podataka po segmentu i da je RTT 4 vremenske jedinice. Zanemariti djelovanje algoritama upravljanja zakrčivanjem u protokolu.

Rješenje

Prilikom uspotavljanja veze razmjenjuju se tri segmenta koja definiraju parametre veze. Prema uvijetima u zadatku ta razmjena ima sljedeći oblik:



Nakon uspostavljanja veze klijent odmah započinje sa slanjem. Kako mu je ograničenje na veličinu segmenta 2000 okteta, a ograničenje prozora 1000 to znači da smije slati maksimalno 1000 okteta te potom mora čekati na potrvrdu. Odmah po promitku potvrde može slati idućih 1000 okteta i tako do kraja.


C
S
SYN=7001,1000
ACK 8001, WIN 1000
SYN=8001,1000
ACK 9001, WIN 1000
SYN= 9001, 900
ACK 9901, WIN 1000

PRIMJER 4. Analiza razmjene segmenata

Upotrebom `tcpdump` programa snimljena je komunikacija između dva TCP entiteta. Analizirati dobijeni trag i odrediti da li je ispravan te u slučaju da nije prikazati ispravan trag.

```
10:40:42.244503 B > A: S 1168512000:1168512000(0) win 32768 <mss
1460,nop,wscale 0> (DF) [tos 0x8]
10:40:42.259908 A > B: S 3688169472:3688169472(0) ack 1168512001 win 32768 <mss
1460>
10:40:42.389992 B > A: . ack 1 win 33580 (DF) [tos 0x8]
10:40:42.664975 A > B: P 1:513(512) ack 1 win 32768
10:40:42.700185 A > B: . 513:1973(1460) ack 1 win 32768
10:40:42.718017 A > B: . 1973:3433(1460) ack 1 win 32768
10:40:42.762945 A > B: . 3433:4893(1460) ack 1 win 32768
10:40:42.811273 A > B: . 4893:6353(1460) ack 1 win 32768
10:40:42.829149 A > B: . 6353:7813(1460) ack 1 win 32768
10:40:42.853687 B > A: . ack 1973 win 33580 (DF) [tos 0x8]
10:40:42.864031 B > A: . ack 3433 win 33580 (DF) [tos 0x8]
```

Rješenje

Svaka linija ispisa predstavlja jedan segment. U općem slučaju svaka linija sadrži u prvoj koloni informaciju o trenutku kada je segment pristigao na računalo, odnosno kada je otišao s računala. Potom u drugoj koloni dolazi IP adresa računala koje šalje segment i, nakon znaka veće, IP adresa računala kojemu je segment namijenjen. U ovom ispisu IP adrese zamijenjene su simboličkim oznakama A i B. Potom dolazi popis postavljenih zastavica u TCP segmentu. Ostatak linije čine ostali podaci iz zaglavlja TCP segmenta ili IP paketa koji mogu varirati.

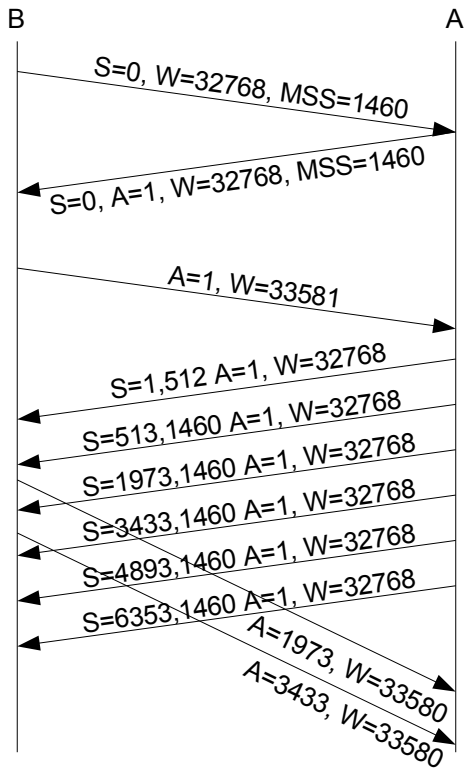
Prva linija predstavlja zahtijev za uspostavljanjem veze što se vidi po postavljenoj zastavici SYN (označeno kao S). Odabrani početni slijedni broj je 1168512000, a količina podataka u segmentu je 0. Veličina rozora postavljena je na 32k. U segmentu se također nalazi određen broj opcija od kojih jedinu efektivnu vrijednost ima opcija za maksimalnu veličinu segmenta (mss) koja iznosi 1460 okteta. Opcija nop se ignorira, dok opcija za skaliranje veličine prozora ima vrijednost 0, tj. nema skaliranja. Zadnja dva podatka prve linije odnose se na IP paket. Naime DF znači da je postavljena zastavica *Don't Fragment* u zaglavlju IP paketa, a također da je vrijednost polja *TOS* postavljena na 0x8.

U drugoj liniji vidimo odgovor računala A koji pristiže nakon nešto više od 15ms. U tom odgovoru A bira svoj inicijalni slijedni broj (3688169472), potvrđuje segment od B s idućim očekivanim slijednim brojem (ack 1168512001), veličinom prozora od također 32k i maksimalnom veličinom segmenta od 1460 okteta.

U trećoj liniji, nakon 130ms, završava uspostavljanje veze. Također, od te linije slijedni brojevi su relativni u odnosu na inicijalne slijedne brojeve.

Od četvrte do devete linije vidimo kako računalo A šalje šest segmenata bez čekanja potvrde. Četvrti segment ima postavljenu zastavicu PUSH što znači da se od TCP entiteta na računalu B traži trenutna isporuka podataka aplikaciji. Svi segmenti, osim četvrtog, maksimalne su veličine. Tek nakon toga pristiže potvrda prvog segmenta s podacima, a nakon njega i potvrda druga dva segmenta.

Grafički, ta razmjena može se prikazati na sljedeći način:



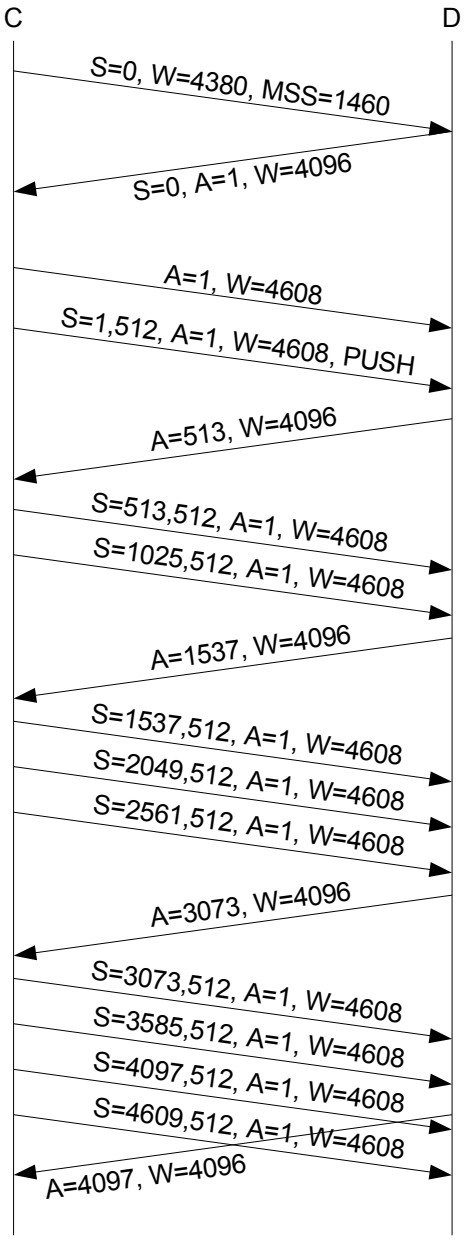
Problem navedene razmjene podataka je u tome da A ne koristi algoritam polaganog starta već naglo započinje sa slanjem podataka.

Sljedeći izlaz TCP programa prikazuje ispravan redoslijed segmenata koji poštiva pravilo polaganog starta:

```
12:35:31.914050 C > D: S 1448571845:1448571845(0) win 4380 <mss 1460>
12:35:32.068819 D > C: S 1755712000:1755712000(0) ack 1448571846 win 4096
12:35:32.069341 C > D: . ack 1 win 4608
12:35:32.075213 C > D: P 1:513(512) ack 1 win 4608
12:35:32.286073 D > C: . ack 513 win 4096
12:35:32.287032 C > D: . 513:1025(512) ack 1 win 4608
12:35:32.287506 C > D: . 1025:1537(512) ack 1 win 4608
12:35:32.432712 D > C: . ack 1537 win 4096
12:35:32.433690 C > D: . 1537:2049(512) ack 1 win 4608
12:35:32.434481 C > D: . 2049:2561(512) ack 1 win 4608
12:35:32.435032 C > D: . 2561:3073(512) ack 1 win 4608
12:35:32.594526 D > C: . ack 3073 win 4096
12:35:32.595465 C > D: . 3073:3585(512) ack 1 win 4608
12:35:32.595947 C > D: . 3585:4097(512) ack 1 win 4608
12:35:32.596414 C > D: . 4097:4609(512) ack 1 win 4608
12:35:32.596888 C > D: . 4609:5121(512) ack 1 win 4608
12:35:32.733453 D > C: . ack 4097 win 4096
```

Kao što se vidi iz ispisa C nakon uspostavljanja veze šalje segment i čeka na potvrdu što je posljedica inicijalnog postavljanja vrijednosti varijable cwnd na 1. Nakon primitka potvrde C uvećava vrijednost cwnd i može poslati nova dva segmenta. Segmenti se kumulativno potvrđuju, te ta potvrda omogućava povećanje cwnd-a na tri što dozvoljava C-u slanje iduća tri segmenta. Ta tri segmenta opet su potvrđena kumulativno te se cwnd povećava na četiri i zatim se šalju četiri nova segmenta. Zadnja lnija predstavlja potvrdu prva tri segmenta.

Grafički je navedena razmjena prikazana idućom slikom.



5. Literatura

[RFC0791] Postel, John, *Internet protocol – Darpa Internet Protocol Specification*, RFC791, IETF, September 1981.
http://www.ietf.org/rfc/rfc0791.txt?number=791

[RFC0879] Postel, John, *The TCP Maximum Segment Size and Related Topics*, RFC879, IETF, November1983.
http://www.ietf.org/rfc/rfc0879.txt?number=879

[PETER00] Peterson, L. L., Davie, B. S., *Computer Networks: A Systems Approach*, 2nd edition, Morgan Kaufmann Publishers, 2000.

[JAVOB88] Jacobson, V., Karels, M. J., *Congestion Avoidance and Control*, November 1988.