Analiza komunikacijskih modela za opis rada TCP protokola i njegovih mehanizama

Veverec, Filip

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

Filip Veverec

KOMUNIKACIJSKI MODELI ZA OPIS RADA TCP PROTOKOLA I NJEGOVIH MEHANIZAMA

DIPLOMSKI RAD

Zagreb, 2020.

SVEUŠILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI POVJERENSTVO ZA DIPLOMSKIISPIT

Zagreb, 1. rujna 2020.

Zavod: **Zavod za informacijsko komunikacijskipromet**

Predmet: **Analiza i modeliranje prometnihsustava**

DIPLOMSKI ZADATAK br. 5906

Pristupnik: **Filip Veverec(0135223731)**

Studij: Promet

Smjer: Informacijsko-komunikacijskipromet

Zadatak: **AnalizakomunikacijskihmodelazaopisradaTCPprotokolainjegovih mehanizama**

Opis zadatka:

Prezentirati upotrebljivost UML jezika u modeliranju rada različitih sustava poput mrežnih elemenata (klijenata i poslužitelja). Opisati značajke TCP protokola i različite scenarije upotrebe tog protokola. Analizirati mehanizme TCP protokola koji se upotrebljavaju prilikom uspostave, održavanja i raskidanja sesija. Razviti komunikacijske modele, koristeci UML jezik, koji ce opisivati redoslijed razmjene poruka izmebu mrežnih elemenata koji sudjeluju u izvobenju TCP mehanizama za uspostavu, održavanje i raskidanjesesija.

Mentor: Predsjednik povjerenstvaza diplomski ispit:

doc. dr. sc. Marko Matulin

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

DIPLOMSKI RAD

KOMUNIKACIJSKI MODELI ZA OPIS RADA TCP PROTOKOLA I NJEGOVIH MEHANIZAMA

COMMUNICATION MODELS FOR DESCRIPTION OF TCP PROTOCOL AND ITS MECHANISMS

Mentor: doc.dr. sc. Marko Matulin

Student: Filip Veverec, 0135223731

Zagreb,rujan 2020.

Sažetak

U ovom radu objašnjene su osnove UML (eng. *UnifiedModelingLanguage*) jezika te pripadajući dijagrami koji se koriste za opisivanje sustava. Rad se u trećem poglavlju dotiče i TCP (eng. *TransmissionControlProtocol*) protokola, jer mehanizmi i algoritmi koji su objašnjeni u ovom radu se vežu uz TCP protokol. Četvrto, peto, šesto i sedmo poglavlje opisuju mehanizme i algoritme koje koristi TCP protokol kako bi ostvario siguran prijenos podataka. U svakom od tih poglavlja su pojedini mehanizmi i algoritmi objašnjeni putem UML dijagrama i to dijagramom međudjelovanja, suradnje i dijagramom stanja i prijelaza.

Ključne riječi: UML jezik; TCP protokol; 3-way handshake mehanizam; slow-start mehanizam; algoritmi za izbjegavanje zagušenja; mehanizmi brze retransmisije

Summary

This paper explains the basics of the UML (UnifiedModelingLanguage) and the associated diagrams used to describe asystem. The paper in the third chapter also discusses the TCP (TransmissionControlProtocol)because the mechanisms and algorithms explained in this paper are related to the TCP protocol. Chapters four, five, six, and seven describe the mechanisms and algorithms used by the TCP protocol to achieve secure data transmission. In each of these chapters, individual mechanisms and algorithms are explained via UML diagrams, namely statechart diagram, collaboration diagram and the sequence diagram.

Keywords: UML language; TCP protocol; 3-way handshake mechanism; slow-start mechanism; congestion avoidance algorithms; mechanisms of fast retransmission

Sadržaj

1. Uvod

Od ranih dana čovječanstva postoji potreba za komunikacijom. U početku je ona bila primitivna te nije uključivala nikakva pomagala za ostvarenje iste, međutim kako su se ljudi udaljavali jedan od drugoga komunikacija sa drugom stranom nije bila moguća bez upotrebe pomagala. Kako su se tehnika i tehnologija unaprjeđivale došlo je upotrebe komunikacijskih uređaja pa je termin „komunikacija“ zamijenjen terminom „telekomunikacija“.

Da bi se ostvarila komunikacija na daljinu, odnosno telekomunikacija, bilo je potrebno razviti način kako to ostvariti. U početku, a i danas osnova telekomunikacije je bazirana na telefonskim bakrenim žicama, a u novije doba sve se više teži upotrebi radio-valova.

Za ostvarenje prijenosa zvuka nije dovoljna samo fizička konekcija (kabel), nego su potrebna pravila po kojima će se komunikacija ostvariti, a ta pravila se nazivaju telekomunikacijski protokoli. Protokoli određuju pravila po kojima se podaci prenose od izvora do odredišta.

Jedan od protokola koji se dominantno koristi za pouzdan prijenos podataka je TCP (eng. *TransmissionControlProtocol*) protokol koji će u nastavku rada biti detaljnije objašnjen.

Mehanizmi koji se koriste prilikom uspostave, održavanja i raskida TCP konekcije uključuju *3-way handshake*i*Slow-start* mehanizam te razne algoritme za izbjegavanje zagušenja te mehanizam brze retransmisije u slučaju gubitka paketa koji nisu stigli na odredište ili su stigli oštećeni.

Svrha ovog rada jest analiza različitih mehanizama i algoritama koji se koriste prilikom uspostave, održavanja i raskida TCP sesije u paketnoj mreži, poput Internet mreže.

Cilj istraživanja jest identifikacija objekata u pojedinoj komunikaciji mrežnih elemenata na strani poslužitelja te na strani klijenta. Korištenjem UML (eng. *UnifiedModelingLanguage*) jezika izraditi komunikacijske modele koji će opisati 3-*way handshake* te *Slow-start*mehanizame te algoritme za izbjegavanje zagušenja i mehanizme brze retransmisije.

Naslov diplomskog rada je: Komunikacijski modeli za opis TCP protokola i njegovih mehanizama, a sam rad je podijeljen na osam poglavlja:

1. Uvod
2. Primjena UML jezika u modeliranju sustava
3. Opis značajki TCP protokola i različitih scenarija upotrebe
4. Opis *3-way handshake* TCP mehanizma
5. *Slow-start* mehanizam i njegove značajke
6. Algoritmi za izbjegavanje zagušenja
7. Mehanizmi brze retransmisije
8. Zaključak.

U drugom poglavlju će biti opisan UML jezik kao i najvažniji dijagrami koji se koriste za opisivanje rada sustava.

Treće poglavlje detaljno objašnjava značajke TCP protokola s obzirom da je to jedan od dominantnih protokola koji se danas koriste, a također je i glavna tema ovog rada.

Četvrto poglavlje opisuje i objašnjava *3-way handshake* mehanizam koji TCP protokol koristi kako bi uspostavio i potvrdio vezu između dva telekomunikacijska sustava među kojima se događa prijenos podataka. Baš iz tog razloga se koristi TCP protokol, odnosno onda kada je bitno da paketi sigurno stignu na odredište.

U petom poglavlju opisan je *Slow-start* mehanizam kojeg također koristi TCP protokol kako bi osigurao optimalan prijenos podataka. Ovaj mehanizam ograničava brzinu prijenosa podataka kako bi prijenos bio neprekinut, odnosno kako bi se izbjeglo zagušenje u mreži. To je bitno zbog kontinuiranog prijenosa podataka, a samim time i točnosti isporuke poslanih podataka.

Šesto poglavlje navodi sve relevantne algoritme za izbjegavanje zagušenja koji se koriste u TCP sesiji. Također neki od njih su analizirani i detaljnije objašnjeni. Algoritmi za izbjegavanje zagušenja su vrlo bitna komponenta TCP sesije jer je TCP protokol namijenjen brzom i primarno točnom slanju podataka, a ukoliko dođe do zagušenja, podaci neće biti isporučeni na vrijeme te će se prijenos morati ponoviti.

U sedmom poglavlju su opisani mehanizmi brze retransmisije. Ovi mehanizmi su vrlo bitni jer podaci koji nisu stigli ili su stigli oštećeni trebaju ponoviti i poslati ponovno kako bi poruka na primateljevoj strani bila logična.

Osmo poglavlje je zaključak, a u tom poglavlju bit će ukratko objašnjen cijeli rad te zaključak koji je donesen iz analiza koje su obrađene u poglavljima četiri, pet, šest i sedam.

2. Primjena UML jezika u modeliranju sustava

UML jezik pruža mehanizme za planiranje i modeliranje telekomunikacijskih sustava i procesa u njima. Tvorci UML jezika su krajem 80-ih godina biliGradyBooch, James Rumbaughi IvarJacobson te je od tada izašlo nekoliko verzija UML jezika, posljednja i trenutno aktualna verzija je UML 2.2.[1]

UML je vizualni jezik jer je pomoću njega moguće „vidjeti“ kako će sustav izgledati te kako će se ponašati. UML koristi dijagrame u kojima su sintaksa i semantika strogo definirani.

Životni ciklus programskog sustava se može prikazati nizom ciklusa, a svaki od ciklusa ima četiri faze, a to su: početna faza, faza razrade, faza izrade i faza prijelaza.

U početnoj fazi se definiraju ciljevi određenog sustava, definira se što će se nalaziti u sustavu, a što neće. Predlaže se početna arhitektura (eng. *CandidateArhitecture*) te se prepoznaju potencijalni rizici. U ovoj fazi se procjenjuju troškovi, definira se plan rada.

U svakoj fazi se dosežu prekretnice (eng. *Milestone*) putem kojih se može pratiti napredak. Ukoliko se prekretnice ne dostižu ili je potrebno previše resursa za njihovo dostizanje, često se smjer u kojem razvoj sustava ide može i promijeniti.

U drugoj fazi koja se naziva faza razrade se definiraju financijski plan i plan rada, a sve u cilju da se ispitaju mogućnosti izgradnje sustava. U ovoj fazi se definira arhitektura sustava te je to jedna od prekretnica u ciklusu izrade sustava.

U fazi izrade se izgrađuje sustav koji ima karakteristiku da uspješno radi u testnim uvjetima odnosno da je održiv. Također u ovoj fazi glavna prekretnica je uspješna operativna sposobnost.

Zadnja faza je faza prijelaza gdje se sam sustav predaje na korištenje korisniku koji ga je i zatražio odnosno kupio. Izvode se razna testiranja u realnom okruženju kako bi se otklonili kvarovi i poteškoće prije kompletne implementacije.

[1] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

UML jezik opisuje sustav putem dijagrama od kojih svaki ima svoju određenu funkciju, odnosno pokazuje kako sustav izgleda s obzirom na pogled koji određeni dijagram ima. U nastavku rada su navedene i objašnjene najkorištenije vrste UML dijagrama.

Generalna podjela UML dijagrama je na dijagrame ponašanja (eng. *BehavioralDiagram*) i strukturne dijagrame (eng. *StructureDiagram*)[2].

Dijagrami ponašanja:

1. Dijagram aktivnosti (eng. *ActivityDiagram*),
2. Dijagram slučaja uporabe (eng. *Use CaseDiagram*),
3. Dijagram stanja i prijelaza (eng. *StatechartDiagram*),
4. Dijagram suradnje (eng. *CollaborationDiagram*),
5. Dijagram međudjelovanja (eng. *SequenceDiagram*).

Strukturni dijagram:

1. Dijagram klasa (eng. *ClassDiagram*)

2.1.Dijagram aktivnosti

Dijagram aktivnosti je dijagram koji kronološki opisuje razmjenu poruka unutar sustava, odnosno izvršavaju se aktivnosti kako vrijeme odmiče. Aktivnost je radnja koju objekt neprestano izvodi, ona se može prekinuti te se može rastaviti u druge aktivnosti.[3]

Dijagram pokazuje aktivnosti koje objekti izvodi, a objekti mogu biti razdvojeni u plivačke staze(eng. *Swimline*), one odvajaju akcije koje se događaju kod određenog objekta, a one pomažu samo za lakše razumijevanje dijagrama, te ne odvajaju komunikaciju između staza.

Na vrhu dijagram se nalazi crni krug koji označava početak dijagrama, a izdužene elipse označavaju aktivnost i akciju koja se u određenom trenutku odvija. Veza između aktivnosti/akcija je označena strelicom u smjeru u kojem se odvija akcija pa tako ta veza ne može biti dvosmjerna.

[2]https://support.microsoft.com/hr-hr/office/uml-diagrams-in-visio-ca4e3ae9-d413-4c94-8a7a-38dac30cbed6#OfficeVersion=Office_2013_-_Office_2019
[3]Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram aktivnosti, FPZ, svibanj, 2018, str. 4.

Crna linija koja se proteže vodoravno označava mjesto spajanja ili račvanja. Spajanje je usklađivanje dva ili više tijeka izvođenja u jedan, a račvanje je razdvajanje u dva ili više tijekova izvođenja.

Rombom je označeno grananje i stapanje, a to je vrlo slično označavanje kao i kod dijagrama toka pa je ovo jedna od najkorištenijih oznaka. Koristi se za grananje aktivnosti uz neki uvjet. Ako je uvjet zadovoljen tijek izvođenja kreće u jednom smjeru, međutim ako nije tada tijek izvođenja kreće u drugom smjeru. Uvjeti koji se koriste su uvijek bool-eviizrazi, odnosno istina (eng. *True*) ili laž (eng. *False*). Kod stapanja nema uvjeta nego je moguće da se dva ili više tijekova spoje u jedan tijek.

Treba razlikovati grananje i stapanje od spajanja i račvanja jer su to dvije funkcije koje imaju različite zadaće.

Na kraju dijagrama se nalazi crna točka sa crnim krugom koja označava kraj dijagrama.

2.2.Dijagram slučaja uporabe

Dijagram slučaja uporabe opisuje ponašanje sustava, odnosno što sustav treba raditi, ali ne opisuje način kako treba raditi.[4]

Sa lijeve i desne strane se nalaze sudionici (simbol osobe) koji su uključeni u funkcioniranje sustava. Simbol ne označava određenu osobu nego označava ulogu koju ima u modeliranom sustavu. Slučaj uporabe se označava elipsom u koju se upisuje kratak opis zahtjeva na sustav.

Prikazuju se odnosi između sudionika (simbol osobe) i slučaja uporabe, sudionici su uvijek smješteni sa lijeve i desne strane, a sam slučaj uporabe se smješta u sredinu te se povezuje sa određenim sudionicima koji sudjeluju u nekom procesu.

2.3.Dijagram stanja i prijelaza

Dijagram stanja i prijelaza prati pojedine objekte i njihov životni ciklus unutar sustava. Objekt se može nalaziti u raznim stanjima te može prelaziti iz jednog u drugo stanje. Da bi objekt prešao u neko stanje, često postoji neki uvjet koji mora biti zadovoljen.[5]

[4]Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram slučaja uporabe, FPZ, svibanj, 2015., str. 8.

Postoje slični simboli označavanja kao i kod dijagrama aktivnosti. Crna puna točka označava početak, a crna točka sa crnim krugom označava završno stanje objekta kojeg se prati. Stanja se označavaju pravokutnikom sa oblim rubovima, a u njih se upisuje stanjeu kojem se objekt nalazi.

Sva stanja su povezana te se uvijek stavlja strelica koja označava prijelaze između stanja.

Prijelaz objekta iz jednog u drugo stanje započinje kada se pojavi neki događaj. Kako bi se dogodio prijelaz iz jednog u drugo stanje često treba biti zadovoljen neki uvjet, ali je moguće da objekt pređe u neko drugo stanje bez uvjeta, a uvjet se postavlja u obliku Boole-ovih izraza.

2.4.Dijagram suradnje

Dijagram suradnje je vrsta dijagrama koja prikazuje interakciju objekata te slijed kojim se poruke razmjenjuju, ovaj dijagram nema vremensku komponentu, ali se može utvrditi redoslijed u kojem se izvršavaju akcije između objekata.[6]

Pravokutnicima su označeni objekti koji se nalaze u sustavu te koji imaju neku funkciju. Ti objekti su povezani, ali veze između njih mogu biti različite, razliku se:

1. Asinkrone veze i
2. Zahtjev za postupkom.

Asinkrone veze se koriste onda kada objekt koji ih šalje ne očekuje neku povratnu radnju od objekta prema kojemu ta veza ide. Dok kod veza koje zahtijevaju postupak, kako samo ime i govori, zahtijevaju neku radnju od objekta prema kojemu ta veza ide. Uz svaku vezu se nalazi i redni broj, taj broj govori po kojem redoslijedu se izvode akcije.

2.5.Dijagram međudjelovanja

Dijagram međudjelovanja je usmjeren na vremenski redoslijed razmjene poruka između objekata. Sastoji se od četiri osnovna elementa, a to su:[7]

---

[5]Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram stanja i prijelaza, FPZ, svibanj, 2015., str. 4.
[6]Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram suradnje, FPZ, studeni, 2015., str. 3.

1. Oznake objekta,
2. Crte života,
3. Težište nadzora,
4. Poruke.

Oznake objekta smještene su na vrhu dijagrama te se označavaju pravokutnikom u koji se upisuje naziv objekta. Crte života su vezane uz oznake objekta jer se one spuštaju vertikalno od oznake objekta. One služe kako bi se prikazalo trajanje, odnosno život objekta. Na crti života se može nalaziti i oznaka „x“, ta oznaka označava dokinuće objekta, te se nakon te oznake objekt smatra „mrtvim“ i više ne sudjeluje u komunikaciji.

Težište nadzora je pravokutnik koji se nalazi na crti života objekta. Služi za označavanje vremena koje je objektu potrebno da procesira neki zahtjev te pošalje odgovor na upit koji je dobio. Upiti koje objekti dobivaju se nazivaju poruke, one se protežu vodoravno između crta života objekata.

Objekti ne moraju komunicirati po redu kako su posloženi, već je moguća komunikacija između bilo koja dva objekta. Ovaj dijagram ne prikazuje strukturu sustava već se orijentira na kronologiju poruka koje se šalju između objekata.

2.6.Dijagram klasa

Dijagrami klasa opisuju klase i njihove međusobne veze, odnosno vrste objekata nekog sustava i njihove međusobne odnose. Osobina dijagrama klasa je to što su statični s obzirom na vremensku komponentu.[8]

Svakoj klasi je potrebno definirati naziv, a dodatno je moguće odrediti atribute i operacije. Atributima je nužno dodijeliti naziv i tip podatka, dok je kod operacije nužno definirati ulazno-izlazne parametre te funkciju.

Klase su spojene vezama koje se nazivaju relacije, a postoji 3 vrste relacije koje su navedene niže, a to su:[9]

[7]Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram međudjelovanja, FPZ, studeni, 2018., str. 31.
[8] Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram klasa, FPZ, studeni, 2016, str. 2.

1. Relacija udruživanja (eng. *Association*) – opisuje statične odnose između dvije klase.
2. Relacija sastavljanja (eng. *Aggregation*) – ova vrsta veze prikazuje da određena klasa sadrži neku drugu klasu, postoji još i kompozicija, a glavna razlika je u tome da kod uništavanja objekta se uništavaju i svi pojedinci koji su vezani.
3. Relacija poopćavanja (eng. *Generalization*) – ova veza prikazuje podklase koje imaju ista svojstva kao i glavna klasa (roditelj).

[9]Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram klasa, FPZ, studeni, 2016., str. 5.

3. Opis značajki TCP protokola i različitih scenarija upotrebe

3.1. Općenito o TCP protokolu

Prva inačica TCP protokola predstavljena je 1974. godine. TCP protokol je jedan od najkorištenijih protokola za prijenos podataka od izvora do odredišta. To je protokol koji je spojno orijentiran, a to znači da prije slanja podataka na odredište, uspostavlja čvrstu konekciju između pošiljatelja i primatelja. Na taj način se osigurava sigurna veza i pouzdan prijenos podataka uz minimalno zagušenje.[10]

Uspostavlja se veza između pošiljateljeve i primateljeve strane, ta veza se naziva logička veza jer se određuje samo put kojim će se paketi kretati kroz mrežu, te ne postoji fizička veza, odnosno ne rezervira se određeni kapacitet kako bi se paketi poslali. Brigu o logičkoj vezi, odnosno putu kojim će se podaci kretati preuzima usmjernik (eng. *Router*).[11]

TCP protokol je protokol koji obavlja funkcije transportnog sloja OSI (eng. *Open Systems Interconnection*) referentnog modela te kao takav komunicira sa aplikacijskim slojem iznad njega i Internet slojem koji se nalazi ispod njega. Kako bi se definirao TCP protokol potrebno definirati OSI referentni model koji je u slijedećem poglavlju detaljnije opisan, a iz njega je vrlo lako iščitati TCP/IP model koji korelira sa OSI referentnim modelom.

3.2. OSI referentni model

OSI referentni model je podijeljen na slojeve (eng. *layer*) koji služe za lakše razumijevanje funkcioniranja pojedinih slojeva i pripadajućih protokola. OSI model ne specificira protokole nego služi da bi se opisala funkcija protokola pojedinih slojeva. Model je podijeljen na sedam slojeva[12]:

1. Aplikacijski sloj (eng. *Applicationlayer*),
2. Prezentacijski sloj (eng. *Presentationlayer*),

[10] https://www.fer.unizg.hr/_download/repository/INFMRE-2017_09.pdf
[11] https://personal.oss.unist.hr/~alen/Literatura-SKRIPTA!!!/slide6.pdf
[12] https://sysportal.carnet.hr/node/352

3. Sloj sesije (eng. *Sessionlayer*),
4. Transportni sloj (eng. *Transport layer*),
5. Mrežni sloj (eng. *Network layer*),
6. Podatkovni sloj (eng. *Data Link layer*),
7. Fizički sloj (eng. *Physicallayer*).

Navedeni slojevi komuniciraju samo sa svojim susjedom, odnosno sa slojem iznad i ispod sebe, a to je važno jer ukoliko bi svaki sloj komunicirao sa svim ostalim slojevima, nastao bi kaos u hijerarhiji, te sustav jednostavno više ne bi mogao funkcionirati.

Aplikacijski sloj omogućuje aplikacijama pristup mrežnim uslugama. On nije u vezi s programima koji zahtijevaju lokalne resurse, nego s programima koji koriste mrežne resurse tako da bi mogli komunicirati s aplikacijskim slojem. Ovaj sloj je najbliži korisniku te on dostavlja mrežne usluge isključivo aplikacijama krajnjeg korisnika, koje se nalaze izvan OSI modela[13].Protokoli koji obavljaju funkcije aplikacijskog sloja: HTTP (eng. *Hyper Tekst Transfer Protocol*) i FTP (eng. *File Transport Protocol*).

Prezentacijski sloj se brine o tome da podaci koje pošalje aplikacijski sloj budu čitljivi aplikacijskom sloju drugog sustava koji ih dobiva. Brine se o konverziji podataka ukoliko je potrebno.Standardi koje koristi prezentacijski sloj za prijenos grafičkih datoteka su PICT, TIFF, JPEG te za prijenos muzičkih datoteka su MIDI, MPEG[14].

Sloj sesije osigurava komunikaciju između dva računala u mreži, pa se onda ta komunikacija naziva sesija. Zadaća mu je da uspostavi, upravlja i prekine vezu između dva računala koja komuniciraju. Izvršava funkcije sigurnosne provjere da bi se utvrdilo imaju li računala dozvolu za uspostavu sesije.

Drugi zadatak sloja sesije je da prave informacije stignu na pravo mjesto. Primjerice ako korisnik pošalje zahtjev za otvaranjem web stranice preko preglednika tada se sloj sesije brine o tome da se taj zahtjev i izvrši, a ne da korisnik dobije e-mail poruku. Odnosno to znači da svaki paket nosi broj vrata (eng. *gates*) određene aplikacije, a sloj sesije preusmjerava pakete prema vratima na koja moraju stići tako da se ne bi dogodilo da neki paket bude preusmjeren prema krivim vratima, te da korisniku budu proslijeđene informacije koje uopće

[13]Mrvelj, Š. Predavanja iz predmeta Tehnologija telekomunikacijskog prometa 1: Slojevite arhitekture i norme umrežavanja otvorenih sustava, FPZ, studeni, 2013, str. 32
[14]Mrvelj, Š. Predavanja iz predmeta Tehnologija telekomunikacijskog prometa 1: Slojevite arhitekture i norme umrežavanja otvorenih sustava, FPZ, studeni, 2013, str. 31

nije tražio.Protokoli koji se koriste na ovom sloju su NFS (eng. *Network File System*), SQL (eng. *StructuredQueryLanguage*), ASP (eng. *AppleTalkSessionProtocol*)[15].

Transportni sloj omogućava krajnju komunikaciju između dva računala prilikom razmjene podataka. Zadaća transportnog sloja je segmentacija i spajanje podataka u jednu cjelinu. Segmentacija se obavljana obje strane, na strani pošiljatelja podaci se segmentiraju na manje dijelove koji se zovu segmenti te se kao takvi prosljeđuju dalje i šalju do odredišta. Na odredišnoj strani se ti segmenti slažu ponovno u cjeloviti tok podataka koji se dalje prosljeđuju višim slojevima. Zadatak zadnja četiri sloja uključujući i transportni sloj je da se brinu o prijenosu podataka, dok se prva tri, odnosno aplikacijski, prezentacijski i sloj sesije brinu o radu mrežnih aplikacija.Transportni sloj mora osigurati pouzdanost prilikom prijenosa podataka između dva računala koja komuniciraju, odnosno on brine o detekciji i otklanjanju pogrešaka koje se mogu pojaviti prilikom prijenosa podataka. Najčešće korišteni protokoli transportnog sloja su TCP i UDP (eng. *UserDatagramProtocol*) koji se koriste za prijenos podataka s jednog na drugo računalo u mreži[16].Ovaj sloj je detaljnije objašnjen jer se tema ovog rada fokusira na TCP protokol koji se nalazi na ovom sloju.

Mrežni sloj omogućava povezivost i određuje najbolji put za prijenos podataka između dva računala u mreži koja komuniciraju. Zadužen je za logičko adresiranje kao što je IP adresa. Nadalje mrežni sloj je zadužen za prevođenje logičkih adresa (IP) u fizičke adrese (MAC). Ukoliko je potrebno mrežni sloj može dodatno segmentirani IP pakete u manje segmente ako kapacitet mreže to zahtjeva[17].Protokoli koji obavljaju funkcije mrežnog sloja: IP protokol i ICMP (eng. *Internet ControlMessageProtocol*).

Podatkovni sloj je smješten između fizičkog i mrežnog sloja i njegove zadaće su stvaranje okvira, fizičko adresiranje odnosno dodavanje adrese prijemnika na odredišnoj strani, dodavanje pošiljateljeve adrese te upravljačke informacije. Nadalje njegova zadaća je i kontrola pogrešaka, kontrola pristupa i kontrola toka.

Fizički sloj je zadnji (prvi) sloj u OSI referentnom modelu, on prevodi niz bitova u odgovarajući signal te ga prenosi preko određenog medija do odredišta. Vrsta signala ovisi o prijenosnom mediju, te se u fizičkom sloju prema tome prevode bitovi.

[15]Kavran, Z. Predavanja iz predmeta Računalne mreže: OSI slojevi i mediji za prijenos podataka, FPZ, listopad, 2013. str. 39
[16]Mrvelj, Š. Predavanja iz predmeta Tehnologija telekomunikacijskog prometa 1: Slojevite arhitekture i norme umrežavanja otvorenih sustava, FPZ, studeni, 2013, str. 29
[17]Kavran, Z. Predavanja iz predmeta Računalne mreže: Mrežni sloj, FPZ, listopad, 2013. str. 20-33

3.3. Upotreba TCP protokola

Kako je već navedeno TCP protokol je jedan od najznačajnijih protokola u današnje vrijeme te se koristi u razne svrhe. Svaki protokol ima svoju primarnu funkciju, pa je tako funkcija TCP protokola siguran prijenos podataka. U tablici 1 su navedene neke od usluga koje koriste TCP protokol.[18]

Tablica 1. Prikaz najkorištenijih usluga koje koriste TCP protokol te kratak opis usluge.

Protokol aplikacijskog sloja	Transportni protokol	Opis usluge
FTP	TCP/UDP	Prijenos datoteka
SMTP	TCP	Pristup e-pošti
DNS	TCP/UDP	Pridruživanje informacija imenu domene
HTTP	TCP	Protokol za korištenje internetskih usluga
IMAP	TCP	Pristup e-pošti s udaljenog računala

Izvor:https://zir.nsk.hr/islandora/object/ffri:825/preview

Kako je vidljivo iz tablice 1 TCP protokol se primarno koristi kod usluga kao što su pristup e-pošti s udaljenog računala i Internet usluga gdje je potrebna sigurna isporuka paketa bez gubitaka. Ukoliko bi se u tu svrhu koristio UDPprotokol postoji mogućnost da e-mail poruka koja se šalje stigne korumpirana, odnosno u tom slučaju je moguće da nedostaju slova te da su slova u tekstu pomiješana.

U tablici 2 se nalaze podaci o uslugama te protokolima koje koriste, primjer je jedan od proizvođača računala i pametnih telefona – Apple Inc.

[18]Marchese, M.: QoS over Heterogeneus Networks, University of Genoa, Italy, 2007. str. 274

Tablica 2. Prikaz najkorištenijih usluga koje koriste TCP ili UDP protokol te kratak opis usluge.

Priključak	TCP ili UDP	Naziv servisa ili naziv protokola	Naziv servisa	Koristi ga
20	TCP	FTP (File Transport Protocol)	ftp-data	—
22	TCP	SSH (Secure Shell), SFTP (SSH File Transfer Protocol) i scp (Secure copy)	ssh	Xcode Server (hostirani i udaljeni Git + SSH; udaljeni SVN + SSH)
25	TCP	SMTP (Simple Mail Transfer Protocol)	smtp	Mail (slanje e-mailova); iCloud Mail (slanje e-mailova)
67	UDP	BootstrapProtocol Server (BootP, bootps)	bootps	NetBoot putem DHCP-a
68	UDP	BootstrapProtocolClient (bootpc)	bootpc	NetBoot putem DHCP-a
79	TCP	Finger	finger	—
500	UDP	Wi-Fi pozivi	IKEv2	Wi-Fi pozivi
687	TCP	administriranje poslužitelja	asipregistry	alati za administriranje poslužitelja u sustavu Mac OS X Server v10.6 ili starijima, uključujući AppleShare IP
985	TCP	statični priključak za NetInfo	—	—
993	TCP	Mail IMAP SSL	imaps	iCloud Mail (SSL IMAP)
2336	TCP	sinkronizacija mobilnog računa	appleugcontrol	sinkronizacija početnog direktorija
3004	TCP	iSync	csoftragent	—
3689	TCP	DAAP (Digital Audio Access	daap	dijeljenje iTunes glazbe, AirPlay

		Protocol)		
4488	TCP	Appleov servis za povezivanje širokog područja	awacs-ice	
4500	UDP	Wi-Fi pozivi	IKEv2	Wi-Fi pozivi
8171	TCP	HTTP (web-servis/mjesto)	—	Podcast Capture / podcast CLI
16384-16387	UDP	RTP (Real-Time Transport Protocol), RTCP (Real-Time ControlProtocol)	connected, –	FaceTime, Game Center

Izvor:https://support.apple.com/hr-hr/HT202944

Ovaj podatak je vrlo zanimljiv jer se pomoću njega može zaključiti da TCP protokol koriste usluge kojima je bitno da svaki paket u cijelosti stigne na odredište unutar određenog vremena. To je vrlo bitno jer ukoliko podaci ne bi stigli po redu i ukoliko bi bili oštećeni, pojavila bi se distorzija podataka na primateljevoj strani, te poslani podatak ne bi imao smisla i bio bi nečitljiv.

S druge strane UDPprotokol koriste servisi kojima nije bitna isporuka po redu i u cijelosti već je bitna velika brzina slanja paketa. Najbolji primjer za aplikacije kojima je bitna brzina prijenosa je RTP (eng. *Real Time Protokol*) protokol koji se koristi za usluge u stvarnom vremenu, tu je vrlo bitno da podaci što prije stignu kako se ne bi pojavilo kašnjenje, što bi uzrokovalo kašnjenje slike i zvuka. Kod slanja multimedijskih podataka nije toliko bitno da svaki paket stigne, ali je bitno da paketi koji se šalju, stignu brzo.

3.4.Osnovne funkcije TCP protokola

U prethodnom djelu rada je opisan TCP protokol, a u ovom podpoglavlju će biti navede osnovne funkcije TCP protokola[19]:

1. Osnovni prijenos podataka,

[19]http://www.etfos.unios.hr/~drago/predmeti/mip/predavanje10.pdf

2. Adresiranje i multipleksiranje – omogućava komunikaciju između više računala, odnosno putem jednog kanala se mogu prenijeti podaci od više uređaja,
3. Pouzdanost – TCP protokol ima veliku pouzdanost zbog korištenja mehanizama i algoritama za uklanjanje zagušenja,
4. Upravljanje tokom podataka – tok podataka je vrlo bitan jer ukoliko se tok podataka ne kontrolira dolazi do preplavljivanja primatelja, odnosno svi podaci koji su poslani stižu raznim putevima te u različitom vremenu, što znači da bi poruka na primateljevoj strani bila nečitljiva,
5. Upravljanje vezom – bitna stavka TCP protokola je što je to spojni protokol, odnosno prije nego se ostvari prijenos podataka, TCP protokol uspostavlja logičku vezu između pošiljatelja i primatelja,
6. Sigurnost – sigurnost kod TCP protokola se ostvaruje korištenjem svih navedenih značajki, a jedna od najbitnijih je uspostava logičke veze i upravljanje tokom podataka.

Navedene funkcije omogućavaju TCP protokolu njegovu karakterističnu pouzdanost i sigurnost kod prijenosa podataka kojima je pouzdanost vrlo bitna. Podaci kojima je bitna pouzdanost su mail poruka, prijenos multimedijskog sadržaja, prijenos raznih datoteka kao što je *word* dokument.

U slučaju da se kod prijenosa gore navedenih podataka koristi primjerice UDP protokol postoji mogućnost da poruke ne budu čitljive te na taj način korisnikovo iskustvo i zadovoljstvo budu narušeni.

4. Opis *3-way handshake* TCP mehanizma

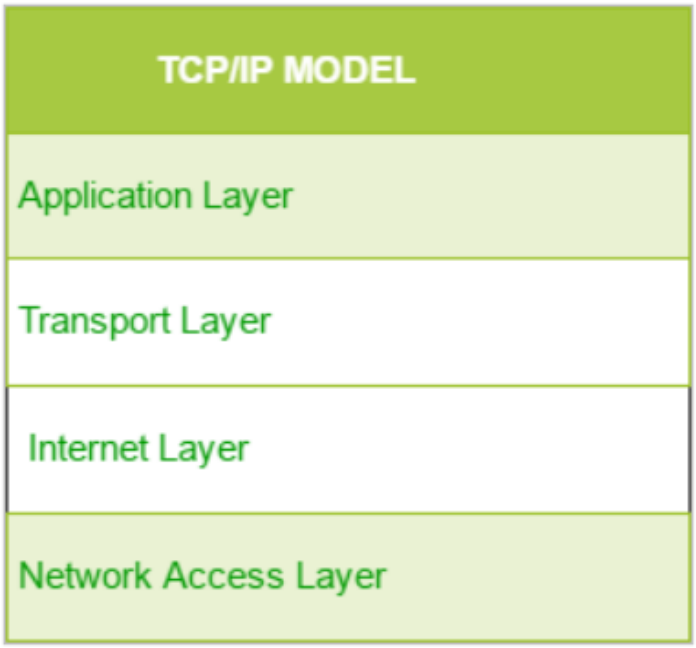
Mehanizam pod nazivom *3-way handshake* se koristi isključivo kod TCP protokola. Točnije koristi se kod uspostave konekcije između dvije mreže. Može se reći da uspostava konekcije kod TCP protokola podrazumijeva *3-way handshake*.

Kako je već navedeno, TCP protokol koristi sigurnu vezu, odnosno uspostavlja konekciju između dvije mreže koristeći PAR (eng. *PositiveAcknowledgementwith Re-transmission)*. TCP protokol kod svakog pojedinog paketa koji je poslao očekuje potvrdu da je on stigao na odredište, ukoliko pošiljateljeva strana ne dobije potvrdu, očekivano je da paket nije stigao ili je stigao sa oštećenjem. Primatelj taj paket zanemaruje te traži brzu retransmisiju. Taj proces se ponavlja sve dok pošiljatelj ne dobije potvrdu da je paket stigao, a tek nakon toga šalje idući paket.[20]

TCP ima još jednu funkcionalnost koja se zove kontrolna suma (eng. *Checksum*), ona funkcionira na način da primatelj računa vrijednost na temelju TCP zaglavlja i podataka koja mora odgovarati vrijednosti koju ima pošiljatelj. U slučaju da se izračunata vrijednost ne podudara, pošiljatelj zaključuje da paket koji se poslao nije cjelovit te da je došao oštećen. Ukoliko se to dogodi TCP mehanizam za brzu retransmisiju traži ponovno slanje tog paketa te se cijeli proces ponavlja.

Da bi se uspostavila konekcija između dvije mreže koristi se *3-way handshake*mehanizam. Proces komunikacije između dvije mreže je opisan TCP/IP (eng. *TransmissionControlProtocol* / *Internet Protocol*) modelom, gdje se nalaze četiri sloja, a slikom 1 su grafički i prikazani.

[20] https://www.geeksforgeeks.org/tcp-3-way-handshake-process/

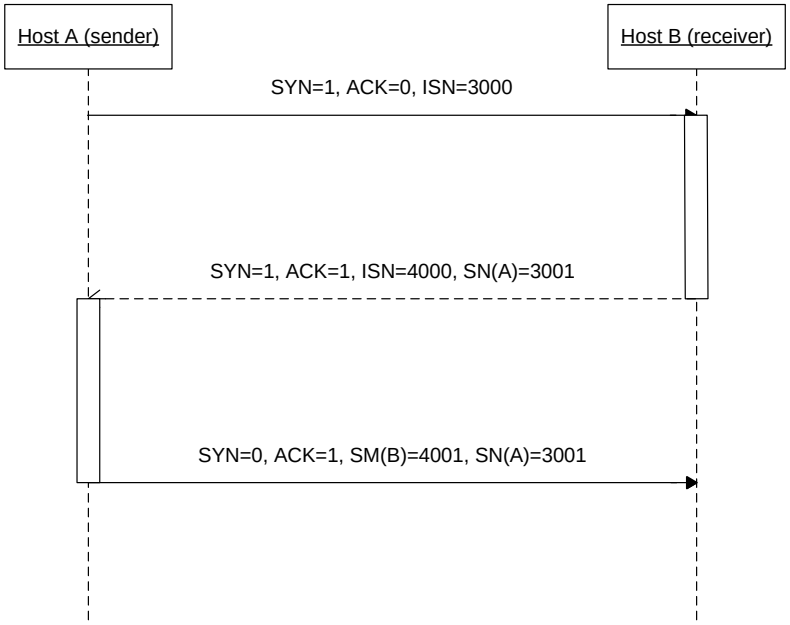


Slika 1. Grafički prikaz TCP/IP modela[21]

Na vrhu modela se nalazi aplikacijski sloj na kojem se uspostavlja veza između aplikacija i servera. Podaci za slanje se spuštaju na transportni sloj na kojem se nalazi TCP protokol koji uspostavlja vezu sa drugom mrežom. Taj proces uspostave se naziva *3-way handshake.*

Dijagramom je opisan proces uspostavljanja veze kod TCP protokola što je vidljivo sa slike 2.

[21]https://www.geeksforgeeks.org/tcp-ip-model/



Slika 2. Proces uspostavljanja veze korištenjem *3-way handshake* mehanizma

Kako je vidljivo na slici 2, proces započinje na strani pošiljatelja (Host A) koji šalje inicijalnu poruku SYN (eng. *Synchronize*) prema primatelju (Host B). Nakon što primatelj primi poruku SYN, odgovara primatelju sa porukom SYN + ACK (eng. *Acknowledgement*). Kada ta poruka stigne do primatelja on povratno odgovara sa porukom ACK kojom potvrđuje uspostavu konekcije. U tekstu ispod je po točkama opisana detaljna komunikacija sa nasumično odabranim brojevima.

Korak 1: Host A šalje TCP segment gdje je SYN=1, ACK=0, ISN (eng. *InitialSequenceNumber*)=3000

Korak 2: Host B prima SYN poruku te odgovara sa TCP segmentom gdje je SYN=1, ACK=1, ISN=4000, SN(A) (eng. *SequenceNumber*)=3001.

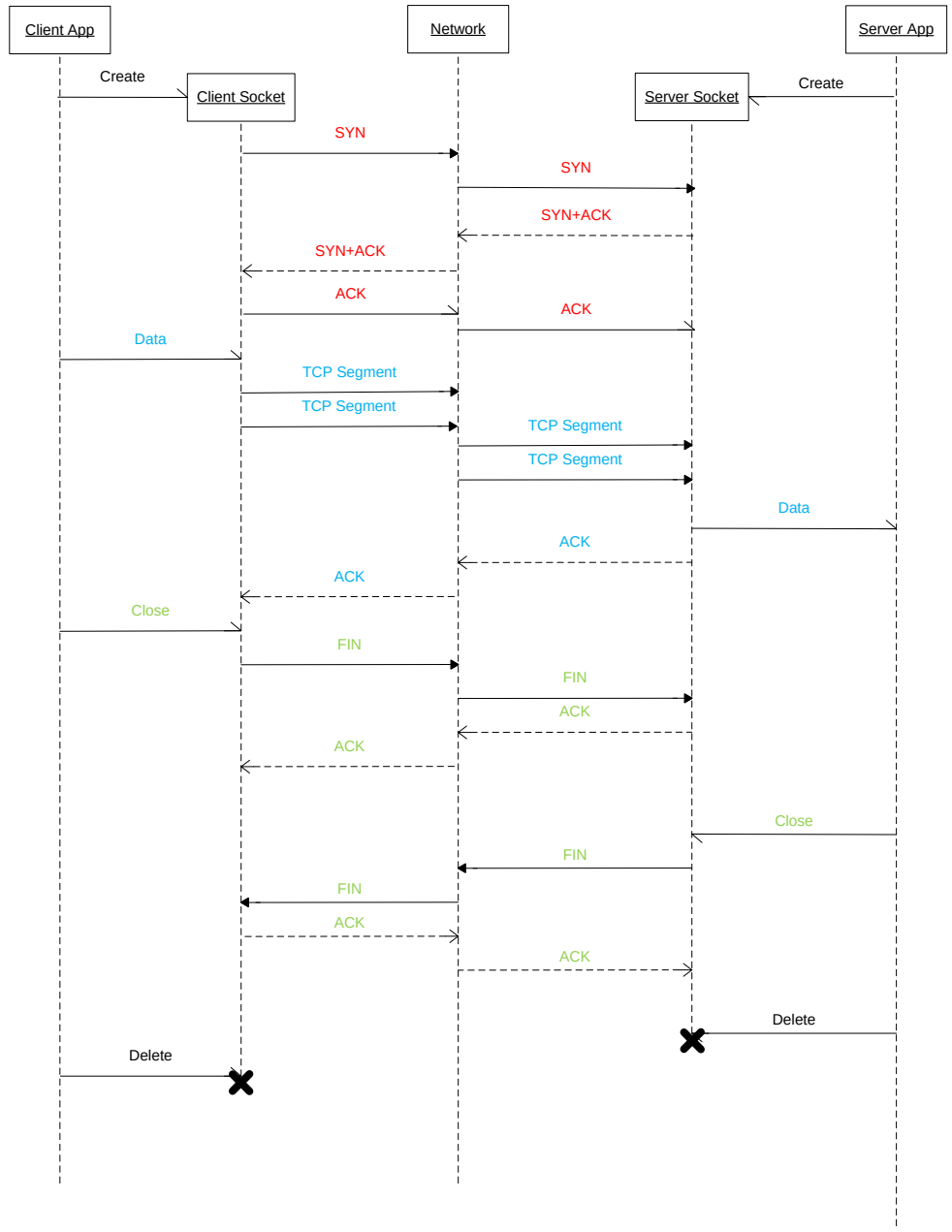
Korak 3: Host A prima SYN+ACK te odgovara sa TCP segmentom gdje je SYN=0, ACK=1, SN(B)=4001, SN(A)=3001.

Inicijalni redoslijedni brojevi (ISN) i redoslijedni broj (SN) su proizvoljno uzeti brojevi te ih Host A i Host B također nasumično biraju prilikom inicijalne poruke.

4.1.Dijagram međudjelovanja za *3-way hadshake* mehanizam

U ovom podpoglavlju je opisan *3-way hadshake* mehanizam kroz dijagram međudjelovanja. Kako već spomenuto dijagram međudjelovanja opisuje tijek komunikacije objekata unutar promatranog sustava.

Slikom 3 je opisan proces uspostave konekcije, slanja podataka i zatvaranje konekcije. Uspostava i zatvaranje konekcije se izvodi prilikom svakog slanja podataka, ali će u ovom radu to biti prikazano samo za *3-way handshake* mehanizam, dok će kod ostalih mehanizama biti prikazan samo dio koji se odnosni na pojedini mehanizam.



Slika 3.: Dijagram međudjelovanja za *3-way handshake* mehanizam

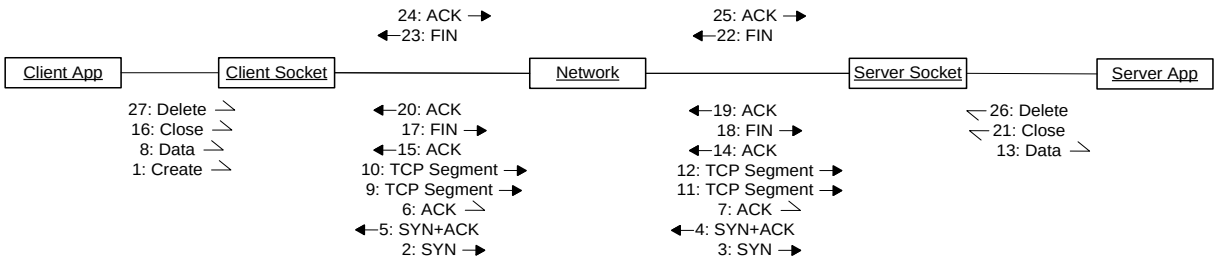
Na slici su pojedini segmenti označeni određenim bojama pa je tako crvenom bojom označen dio koji se odnosni na *3-way handshake*. Prije svega klijentska aplikacija (eng. *Client App*) kreira novi objekt pod nazivom klijentska utičnica (eng. *ClientSocket*). Nakon toga klijent šalje SYN prema mreži (eng. *Network*) koja taj SYN prosljeđuje prema poslužiteljskoj utičnici (eng. *Server Socket*) koja na taj SYN odgovara sa SYN+ACK te potvrđuje proces uspostave konekcije. Mreža taj podataka prosljeđuje nazad prema klijentu te klijent potvrđuje da je primio SYN+ACK. Klijent šalje ACK prema poslužitelju čime se ostvarila konekcija te se *3-way handshake* mehanizam proveo u potpunosti.

Nakon što je konekcija uspostavljena kreće slanje podataka koje je označeno plavom bojom. Klijent šalje pakete, a poslužitelj odgovara sa ACK ukoliko ih je primio. Nakon što su se svi paketi poslali kreće zatvaranje konekcije sa obje strane, a taj proces je označen zelenom bojom.

Zadnje naredbe sa strane klijenta i poslužitelja su obriši (eng. *Delete*) kojom se objekti „*server socket*“ i „*clientsocket*“ brišu te se njihova linija života dalje ne nastavlja dok se linija života za „*server app*“ i „*clientapp*“ i dalje nastavlja.

4.2.Dijagram suradnje za *3-way hadshake* mehanizam

Dijagram suradnje je vrlo sličan dijagramu međudjelovanja, međutim ovdje se fokus stavlja na prikaz komunikacije između objekata. Ovaj dijagram nema vremensku komponentu već samo prikazuje slijed poruka koje objekti razmjenjuju. Slikom 4 prikazana je komunikacija između objekata kod *3-way hadshake* mehanizama.



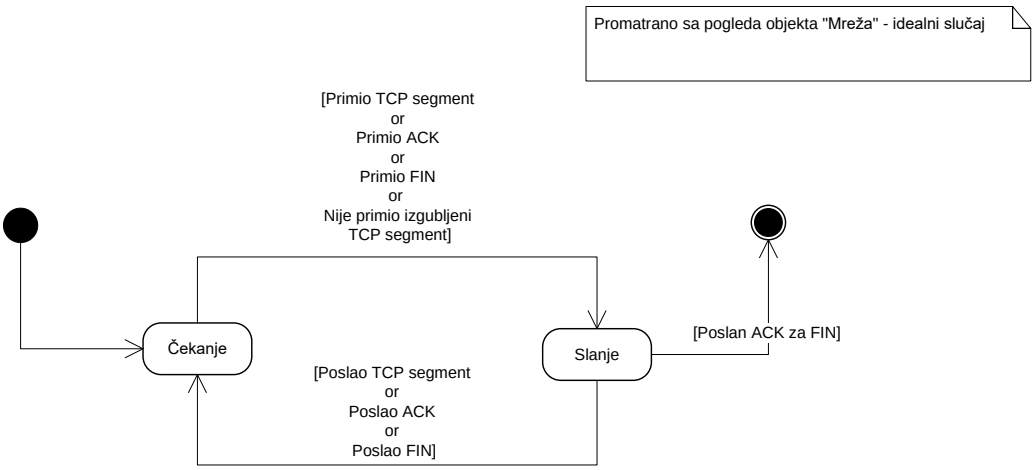
Slika 4.:Dijagram suradnje za *3-way handshake* mehanizam

U ovom dijagramu objekti su isti kao i kod dijagrama međudjelovanja, te su same poruke također iste.

4.3.Dijagram stanja i prijelaza za *3-way hadshake* mehanizam

Ovaj dijagram prikazuje stanja u kojima se pojedini objekt nalazi kroz cijeli proces koji se opisuje. S obzirom da kod ovog primjera postoji pet objekata, obrađen će biti samo jedan od njih, a to je objekt „mreža“, ovaj objekt je u sredini između klijenta i poslužitelja, te većina komunikacije prolazi kroz mrežu pa je tako taj objekt vrlo interesantan za promatranje.

Slikom 5 prikazana su stanja u kojim se nalazi objekt „mreža“ te su navedeni uvjeti koji se moraju ispuniti kako bi objekt promijenio stanje u kojem se nalazi.



Slika 5.:Dijagram stanja i prijelaza za *3-way handshake* mehanizam

Kako je vidljivo na slici ovdje se opisuje uspostava konekcije, prijenos paketa i zatvaranje konekcije. Vrlo je važno napomenuti da ovaj dijagram ne opisuje kako se procesi izvode te ne opisuje redoslijed izvođenja procesa. Primarna zadaća ovog dijagrama je opisati stanja u kojima se objekt nalazi te što je potrebno zadovoljiti da se objekt pomakne iz jednog u drugo stanje.

5. *Slow-start* mehanizam i njegove značajke

Vrijeme koje je potrebno da paket stigne iz jedne u drugu mrežu ovisi o puno parametara. Najvažniji parametar je brzina Internet konekcije, odnosno *upload*i *download*. Brzina prijenosa podataka se često naziva i kapacitet, odnosno količina podataka koju je moguće prenijeti između dvije mreže u jedinici vremena[22].

Zagušenje se stvara kada pošiljatelj u vrlo kratkom vremenu pokuša poslati veliku količinu podataka. U trenutku kada dođe do zagušenja komunikacija je otežana, odnosno vrlo spora, a moguće je i da se dio podataka, odnosno paketa izgubi.

TCP protokol koristi i vremensku komponentu kao jedan od mehanizama sigurne isporuke paketa, to vrijeme se naziva RTT (eng. *RoundTrip Time*). Funkcionira na način da TCP protokol mjeri vrijeme kada je poslao pojedini paket te mu sukladno odredištu gdje treba biti dostavljen, dodjeljuje određeno vrijeme života. Nakon što to vrijeme istekne, paket se smatra neisporučenim te se radi ponovna retransmisija tog paketa.

RTT se računa prema formuli (1)[23]:

$$RTT\ (t) \leq (1-\ \alpha) * RTT\ (t-1) + \alpha * IzRTT \qquad (1)$$

gdje je:

- RTT(t): nova procijenjena vrijednost vremena povratnog puta,
- RTT(t-1) : prethodna procijenjena vrijednost vremena povratnog puta,
- IzRTT: izmjerena vrijednost vremena povratnog puta,
- α: koeficijent iz intervala [0<α<1].

Vrlo je bitno ograničiti brzinu slanja podataka kako ne bi došlo do zagušenja, a tu ulogu ima *slow-start* mehanizam. Zagušenje se može dogoditi na pošiljateljevoj, primateljevoj strani, ali isto tako je moguće da do zagušenja dođe u nekom čvoru u mreži kroz koju se paketi šalju.

[22] https://blog.stackpath.com/tcp-slow-start/
[23] https://zir.nsk.hr/islandora/object/ffri:825/preview

Ovaj mehanizam regulira i ograničava brzinu slanja podataka baš iz razloga da ne dođe do zagušenja u mreži. Kako i sam naziv mehanizma govori „spori početak“, on u početku transmisije vrlo sporo šalje podatke (pakete), a kako komunikacija i prijenos odmiču, povećava se i brzina prijenosa sve do onog trenutka kada se popuni cijeli kapacitet određenog linka.

Slow-start mehanizam se koristi u dva slučaja, prvi slučaj je kada TCP uspostavi konekciju te krene sa slanjem paketa, pa mehanizam u malim koracima propušta sve veći broj podataka kroz mrežu. Drugi slučaj je kada se neki od poslanih paketa izgubi te se obavlja retransmisija, pa *slow-start* mehanizam ponovno ispituje mrežu i njezin maksimalni kapacitet kako nebi došlo do zagušenja.

Formula (2) koja se koristi kod izračuna početne veličine podataka koji se mogu slatikod nestandardnih TCP ekstenzija glasi[24]:

$$IW = min\,(4 * SMSS, max\,(2 * SMSS, 4380\ bytes)) \quad (2)$$

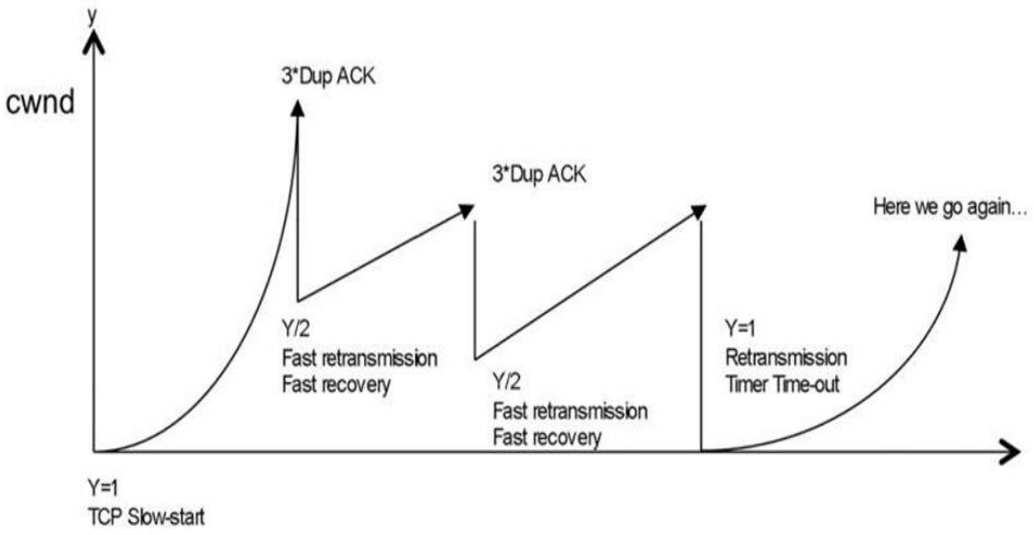
gdje je:

- IW – *Initial Window* – veličina pošiljateljevog prozora zagušenja nakon *3-way handshake* uspostave konekcije
- SMSS – *SenderMaximum Segment Size* – maksimalna veličina segmenta koju pošiljatelj može poslati temeljena na MTU (eng. *MaximumTransmissionUnit*) mreže kojom se podaci šalju.

Kod standardnih TCP ekstenzija maksimalna početna vrijednost ne smije biti veća od 2*SMSS-a, te ne smije imati više od dva segmenta.

Na slici 6prikazana je komunikacija TCP protokola uz korištenje *slow-start* mehanizma.

[24] M, Marchese: QoS over Heterogeneus Networks, University of Genoa, Italy, 2007. str. 276



Slika 6.: Prikaz funkcioniranja *slow-start* mehanizma[25]

Na slici 6 prikazan je graf koji prikazuje početak slanja paketa između izvorišta i odredišta. Na osi „y“ se nalazi cwnd (eng. *Congestion Window*), a na osi „x“ se nalazi RTT. Prijenos kreće sa manjom veličinom prozora, kako vrijeme odmiče prozor se povećava sve do trenutka kada pošiljatelj zaprimi tri uzastopna ACK smatra se da je paket izgubljen te se veličina prozora smanjuje. Prozor se smanjuje na točno pola iznosa maksimalne veličine prozora, a u međuvremenu se radi brza retransmisija izgubljenih paketa.Nakon toga ponovno se povećava brzina prijenosa te kad predajnik primi tri uzastopna ACK se ponovno veličina prozora smanjuje za pola te se obavlja proces brze retransmisije. *Slow-start* mehanizam ponovno povećava veličinu prozora sve do trenutka kada dobije tri uzastopna ACK, ali ovog puta je RTT stigao na nulu te cijeli proces kreće od početka.

Prednosti TCP *slow-start* mehanizma:

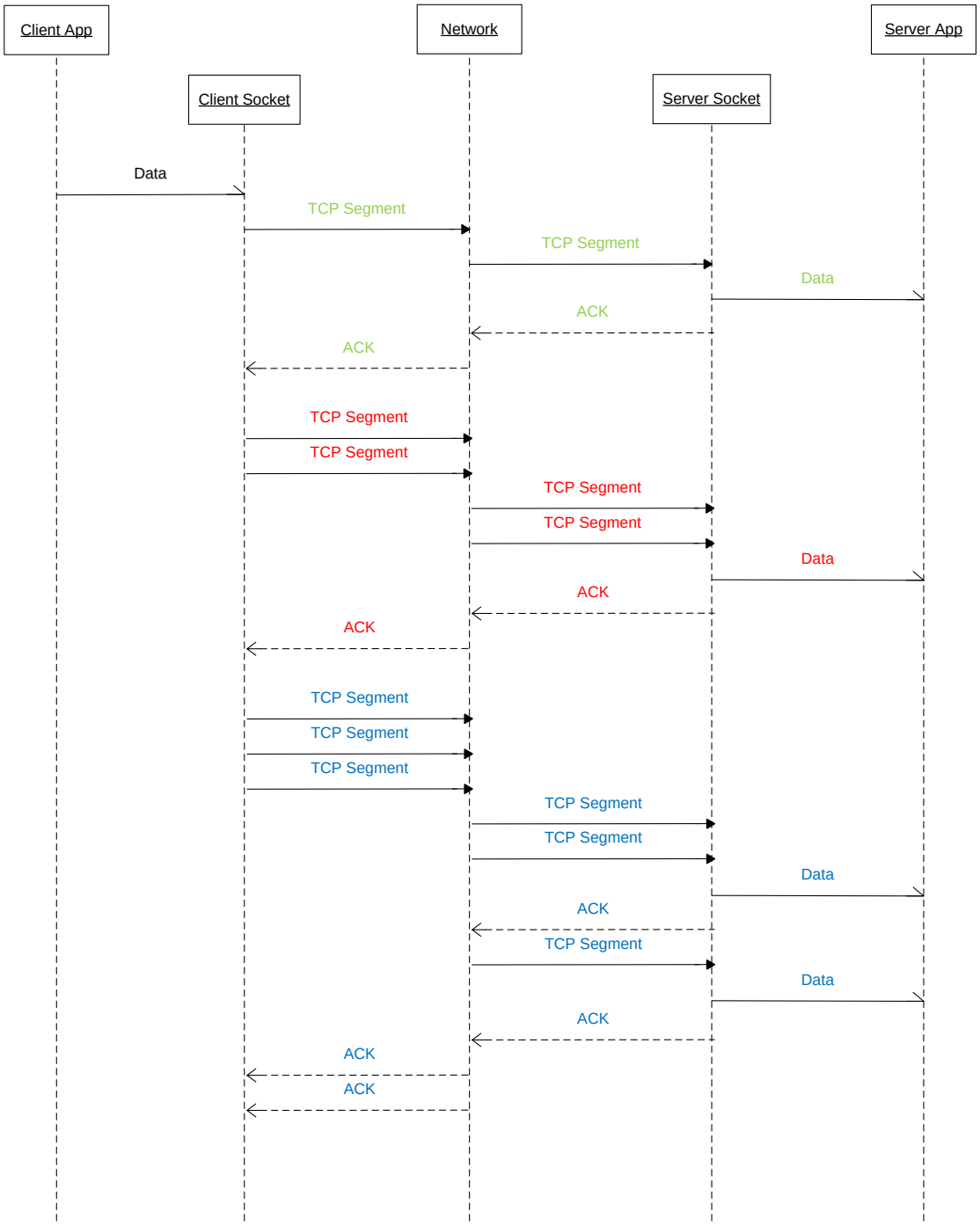
1. PovećanjeQoS-a (eng. *Qualityof Service*) – povećanje kvalitete usluge jer ne dolazi do zagušenja pa sukladno tome usluga radi bez prekida i na onaj način na koji je zamišljena,
2. Povećanje brzine prijenosa – realizira se na način da ne dolazi do zagušenja i ponovnog slanja jer se ograničava maksimalna moguća brzina koja može varirati o mreži iz koje se paket šalje,

[25]http://what-when-how.com/qos-enabled-networks/traffic-types-qos-enabled-networks-part-2/

3. Nema zagušenja – ovo se direktno odnosi na sam *slow-start* mehanizam koji regulacijom brzine prijenosa paketa onemogućuje stvaranje zagušenja.

5.1. Dijagram međudjelovanja za *slow-start* mehanizam

Slikom 7 prikazan je dijagram međudjelovanja za *slow-start* mehanizam. Sam mehanizam je opisan u tekstu iznad, a ovdje je to prikazano i putem dijagrama.



Slika 7.: Dijagram međudjelovanja za *slow-start* mehanizam

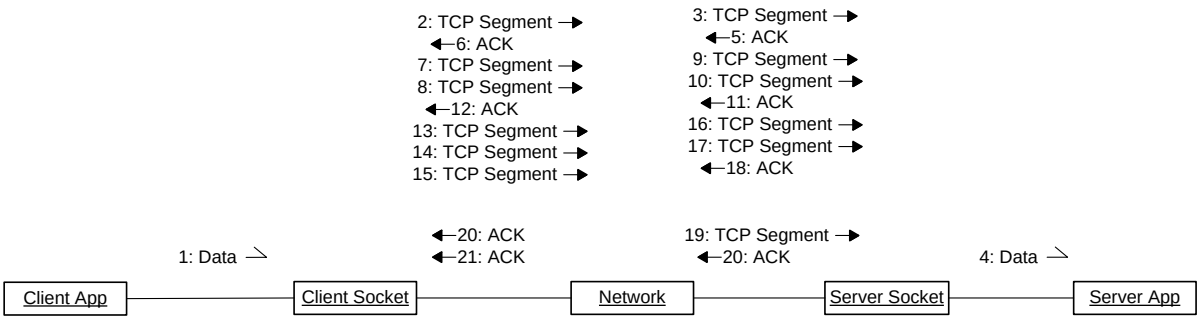
U dijagramu je prikazan samo dio koji se odnosi na *slow-start*, dok su početak (uspostava konekcije) i završetak (zatvaranje konekcije) prikazani dijagramima u poglavlju četiri.

Zelenom bojom je označena prva iteracija, odnosno prvi TCP segment koji se šalje, kada ga mreža proslijedi te on stigne do primatelja, on odgovara sa ACK da je dobio poslani paket. Druga iteracija je označena crvenom bojom, ovdje se šalju dva TCP segmenta jer se povećava prozor odnosno povećava se brzina slanja paketa. Kada primatelju stignu oba poslana paketa, on odgovara sa ACK. Nakon toga ide treća iteracija koja je označena plavom bojom, ovdje se šalju tri TCP segmenta, ali se potvrda o primitku TCP segmenta šalje nakon dva primljena TCP segmenta, a onda se posebno šalje ACK za treći TCP segment.

Na dijagramu je opisan sustav do treće iteracije, a tih iteracija može biti puno više, ovisno o sustavu, međutim logika kojom se TCP segmenti šalju i potvrđuju od strane primatelja je ista.

5.2.Dijagram suradnje za *slow-start* mehanizam

Dijagram suradnje prikazuje *slow-start* mehanizam, odnosno prikazuje komunikaciju objekata unutar sustava, a to se nalazi na slici 8.



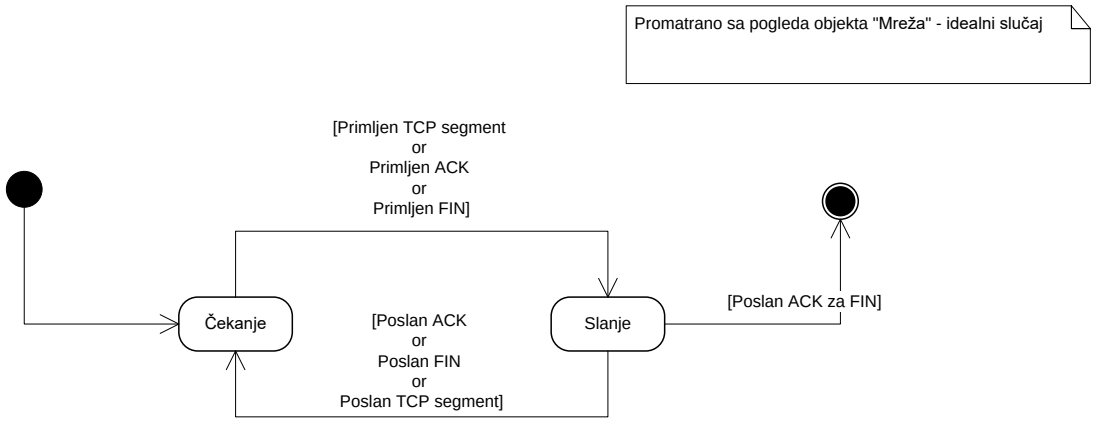
Slika 8.: Dijagram suradnje za *slow-start* mehanizam

Dijagram kreće od poslanog TCP segmenta, koji se preko mreže dostavlja do poslužitelja, nakon toga poslužitelj potvrđuje primitak TCP segmenta porukom „ACK“. Kada potvrda stigne do pošiljatelja, kreće povećanje prozora, odnosno šalju se dva TCP segmenta za koje se također čeka potvrda da su stigli.

Prema ovom modelu je vidljivo da za svaka dva poslana TCP segmenta se očekuje ACK da su stigli, a tek nakon toga pošiljatelj povećava broj TCP segmenta koje uzastopno šalje bez da čeka potvrdu da su stigli, a baš iz tog razloga brzina slanja paketa nije ista, nego se ona mijenja, odnosno povećava se sve do maksimalnog kapaciteta koji je moguće zauzeti.

5.3.Dijagram stanja i prijelaza za *slow-start* mehanizam

Slika 9 prikazuje dijagram stanja i prijelaza za *slow-start* mehanizam. Ovaj dijagram ne može pratiti sve objekte, pa su ovdje prikazana stanja objekta „mreža“ s obzirom da taj objekt sudjeluje u svim aktivnostima i komunikaciji između objekata.



Slika 9.: Dijagram stanja i prijelaza za *slow-start* mehanizam

Kako je vidljivo sa slike, prvo stanje u kojem se objekt nalazi je „Čeka TCP segment“, to stanje je od trenutka kada se uspostavi konekcija (3-way handshake), kada pristigne prvi TCP segment, objekt se nalazi u stanju da šalje taj TCP segment prema poslužitelju, te nakon toga odmah prelazi u stanje da čeka potvrdu primitka tog segmenta. Kada stigne potvrda (ACK) prelazi u stanje da taj ACK prosljeđuje prema klijentu. Nakon toga je ciklus završen i dolazi do kraja procesa. Postoji još jedan smjer u kojem se odmah dolazi do kraja, a to je onda kada je objekt u stanju „čeka TCP segment“, ako niti jedan TCP segment ne stigne, proces je završen jer ovaj objekt tada više nema nikakvu ulogu u procesu.

6. Algoritmi za izbjegavanje zagušenja

Algoritmi za izbjegavanje zagušenja i *slow-start* mehanizam su dva odvojena mehanizma te djeluju neovisno jedan o drugome, međutim u praksi se oni implementiraju zajedno jer time TCP protokol postaje učinkovitiji te se time osigurava određena razina QoS-a.

Nakon što se dogodi zagušenje u mreži TCP usporava slanje paketa u mrežu, pa se tako mora ponovno uključiti *slow-start* mehanizam kako bi ponovno pokrenuo slanje paketa optimalnom brzinom.

Algoritmi koje TCP protokol koristi za izbjegavanje zagušenja su[26]:RENO,BIC (eng. *BinaryIncreaseCongestion*),CTCP (eng. *CodedTransmissionControlProtocol*),CUBIC,HSTCP(eng. *HighSpeedTransmissionControlProtocol*),ILLINOIS, YEAH,SCTP(eng. *StreamControlTransmissionProtocol*),VEGAS,VENO,WESTWOOD.

6.1.RENO

Reno algoritam je jedan od algoritama koje koristi TCP protokol u upravljanju, odnosno izbjegavanju zagušenja. TCP Reno je modificirana verzija TCP Tahoe algoritma, a funkcionira na način da kombinira *slow-start* mehanizam i mehanizam brze retransmisije.

Reno zahtjeva potvrdu primitka svakog paketa, te ukoliko tu potvrdu ne dobije, pokreće se brza retransmisija tog paketa, a tu ponovno dolazi i *slow-start* mehanizam koji ponovno mora „testirati“ kapacitet mreže i maksimalnu količinu paketa koja se može poslati bez da dođe do zagušenja.

Ukoliko se pak dogodi da pošiljatelj dobije više potvrda primitka paketa, logika je da su paketi stigli krivim redoslijedom. Također ovdje se uzima u obzir i vrijeme koje je potrebno da paket stigne do odredišta, a ukoliko je u tom vremenu stiglo više potvrda da su paketi stigli, TCP Tahoe zaključuje da su paketi stigli krivim redoslijedom ili je neki paket izgubljen.[27]

[26] Yang, P., Shao, J., Luo, W., Xu, L., Deogun, J.: TCP Congestion Avoidance Algorithm Identification, University of Nabraska - Lincoln, 2014. str. 1313.
[27] http://intronetworks.cs.luc.edu/current/html/reno.html

Vrijeme koje je potrebno paketu da stigne od izvora do odredišta obično je vrlo slično RTT-u, te ako se paket zadrži u mreži dulje vrijeme ili se izgubi, on se odbacuje te se zahtjeva brza retransmisija.

Problem kod Reno algoritma je taj da on ne može detektirati veliki broj izgubljenih paketa jer ne postoji mehanizam kojim bi „saznao“ koji paket je izgubljen, tako da je ovaj algoritam dobar samo onda kada se radi o jednom izgubljenom paketu kojeg vrlo brzo može otkriti te napraviti brzu retransmisiju.[28]

6.2.BIC

TCP BIC je napredniji algoritam kojeg koristi TCP protokol za izbjegavanje zagušenja. Sustav javlja pošiljatelju je li trenutna brzina slanja paketa unutar granica kapaciteta linka, jer ukoliko se brzina slanja paketa poveća iznad te granice, dolazi do zagušenja jer neki paketi neće biti poslani.[29]

Kapacitet u kojem se paketi šalju se naziva prozor, to je bitno jer ovaj algoritam radi na način da postavlja brzinu slanja paketa na sredinu između maksimalnog i minimalnog iznosa prozora, maksimalni iznos je onaj kada se paketi šalju ali ne dolazi do gubitaka, odnosno popunjava se cijeli kapacitet linka.

Kada transmisija krene ovaj protokol agresivno povećava brzinu slanja paketa, sve do srednje vrijednosti između minimalnog i maksimalnog iznosa prozora. Ukoliko u tom periodu dođe do gubitka paketa, postavlja se novi maksimalni iznos prozora koji je manji u odnosu na inicijalnu vrijednost te se ponovno testira te na taj način se dobiva optimalna brzina bez gubitaka paketa.

6.3.CTCP

CTCP algoritam je nadogradnja TCP protokola u uvjetima kad je veza nestabilna ili su prisutne interferencije. CTCP se u praksi često koristi kod bežičnih (eng. *Wireless*) mreža jer se može implementirati u svu mrežu opremu bez dodatnih hardverskih izmjena.

[28] https://web.archive.org/web/20160103040648/http://www.isoc.org/inet2000/cdproceedings/2d/2d_2.htm
[29] Xu, L., Harfoush, K., Rhee, I.: Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks, North Carolina State University, Raleingh, 2004., str. 347.

Kod *wireless* mreža se koristi protokol 802.11 koji je namijenjen slanju podataka, međutim u takvim mrežama TCP protokol ima velikih problema zbog interferencije koja dovodi do gubitka paketa pa se iz tog razloga razvijaju razni algoritmi koji bi anulirali anomalije u sustavu.

CTCP algoritam na pošiljateljevoj strani pakira segmente podataka u manje blokove podataka koji su nepromjenjive veličine koji se onda šalju u mrežu. Svi paketi moraju biti iste veličine, a ukoliko su neki paketi manji dodaju se nule te se na taj način dolazi do dogovorene veličine paketa, dok se njihov sadržaj ne mijenja.[30]

Veličina paketa se može mijenjati ali samo kod inicijalnog povezivanja pošiljatelja i primatelja u fazi stvaranja logičke veze.

Nakon što se blokovi podataka pošalju, od primatelja se čeka potvrda da su oni stigli, ukoliko potvrde nema, smatra se da je paket izgubljen te se radi brza retransmisija.

6.4.CUBIC

CUBIC algoritam je nasljednik BIC algoritma, međutim razlikuje se u tome što je CUBIC manje agresivan u povećanju brzine slanja paketa. Povećanje brzine slanja paketa kod BIC algoritma definira se kod uspostave konekcije te nije bitno da li se u toku prijenosa dogodio gubitak paketa, dok CUBIC kod svakog gubitka paketa ponovno evaluira brzinu slanja te ju prilagođava u toku prijenosa.

To znači da kod prekida u prijenosu (gubitka paketa) CUBIC algoritam ponovno računa maksimalnu brzinu prijenosa koju je moguće ostvariti s obzirom na kapacitet linka te pogreškama koje su se dogodile tokom prijenosa.

Velika razlika između standardnog TCP protokola i CUBIC algoritma je u tome što se standardni TCP za računanje veličine prozora oslanja na izračun RTT-a, dok se CUBIC za računanje veličine prozora oslanja na zadnji događaj koji se dogodio, pritom se misli na gubitak paketa u prijenosu podataka.[31]

CUBIC povećava veličinu prozora za vrijeme prijenosa podataka, a to povećanje se događa tek nakon što o primatelja dobije ACK (eng. *Acknowledgement*) da je paket isporučen na vrijeme.

[30]Fall, K., Richard Stevens, W.: TCP/IP Illustrated Volume 1, Addison-Wesley, New York, 2012., str 781.
[31]Fall, K., Richard Stevens, W.: TCP/IP Illustrated Volume 1, Addison-Wesley, New York, 2012., str 776.

6.5.HSTCP

HSTCP je algoritam koji je implementiran u TCP protokol, a koristi se u mrežama velike brzine. Internet je danas jedan od osnovnih alata za rad, stoga nije čudno da se razvija velikom brzinom, a ponajviše u zadnjih desetak godina.

Razvijene su mreže velike brzine koje se ponajprije temelje na novim tehnologijama kao što je optičko vlakno (eng. *Fiber*), a koje se komercijalno implementira stoga je bilo potrebno razviti algoritam koji će funkcionirati u takvim mrežama velike brzine, a s druge strane ponuditi odgovarajuće rješenje i osigurati QoS.

HSTCP funkcionira na sličan način kao i osnovni TCP protokol, međutim koristi AIMD (eng. *AdditionalIncreaseMultiplicativeDecrease*) koji mu omogućava bržu prilagodbu kod prijenosa velikih brzina. AIMD omogućava agresivnije i brže povećanje prozora kada se realizira konekcija između izvora i odredišta. Također kada se detektira gubitak paketa agresivnije se smanjuje prozor i to mu omogućava pouzdan prijenos podataka u mrežama velike brzine.[32]

6.6.ILLINOIS i YEAH

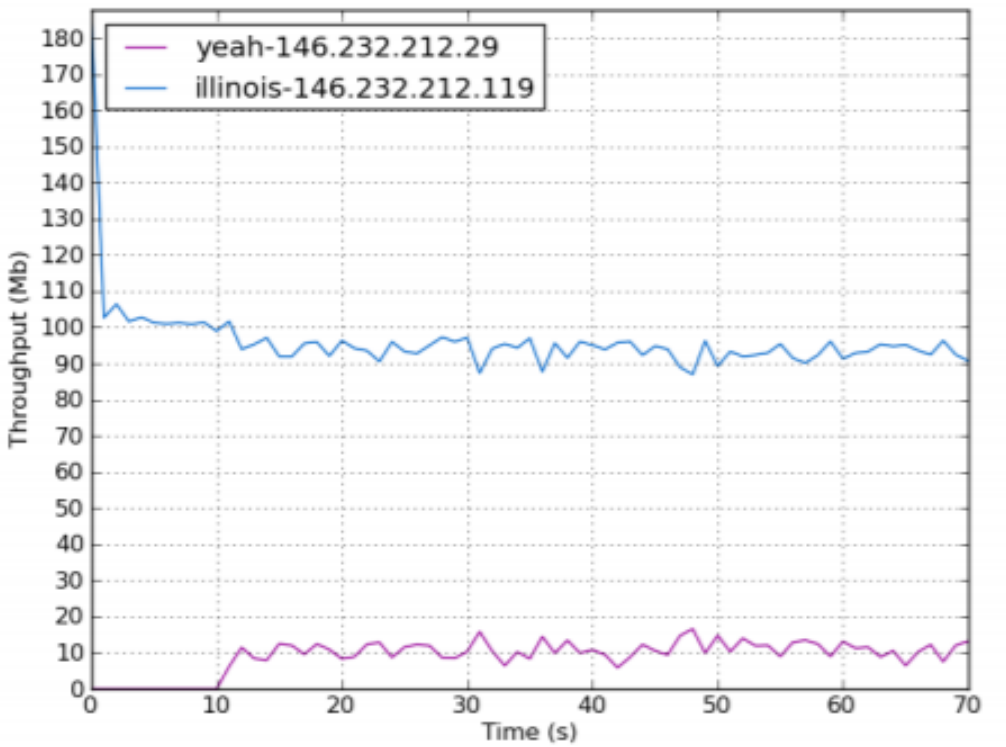
Ilinois je algoritam kojeg koristi TCP protokol na pošiljateljevoj strani, a najčešće se koristi u mrežama jako velike brzine. Kada se govori o jako velikim brzinama misli se na brzine iznad 1Gb/s. Takve mreže nisu jako česte jer su dosta skupe, pa ih primarno koriste korisnici kojima je od iznimne važnosti brzina Internet veze.

Illinois algoritam funkcionira na sličan način kao i Reno te koristi AIMD algoritam za regulaciju veličine prozora. Veličina prozora se agresivno mijenja s obzirom na gubitak paketa, a izračun veličine prozora se računa prema informaciji o kašnjenju paketa.

Razlika u načinu funkcioniranja između Reno i Ilinois algoritma je u tome da Illinois veličinu prozora drži na iznosu trenutno RTT-a, dok je kod Reno algoritma veličina prozora konstantna.

[32] Huang, X., Lin, C., Ren, F., Yin, H.: HighSpeed TCP Modeling and Analysis, Tsinghua University, Bejing, 2004., str. 18.

Na slici 10 je prikazana usporedba Illinois i Yeah algoritma, gdje je vidljivo da Illinois ima daleko veću brzinu prijenosa paketa.[33]

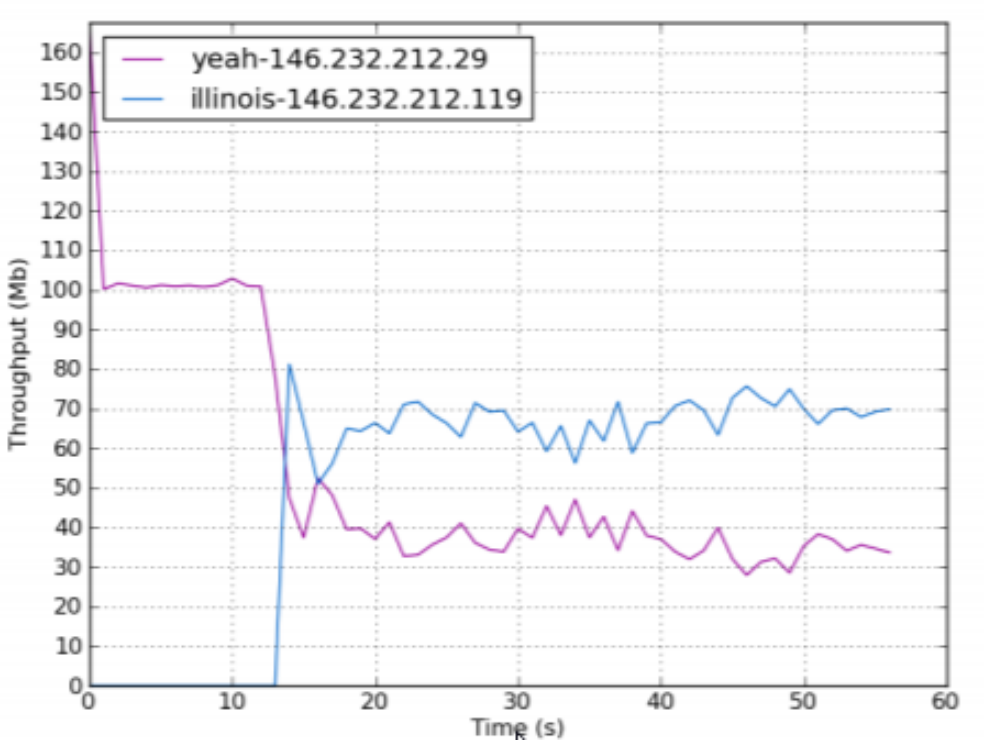


Slika 10.: Usporedba Yeah i Illinois algoritama[34]

U ovom primjeru Illinois algoritam je krenuo deset sekundi prije Yeah algoritma, te je Illinosi zadržao jako veliku propusnost.

Na slici 11 je prikazan primjer gdje je Yeah algoritam krenuo prvi sa slanjem paketa.

[33]Esterhuizen, A.,Krzesinski, A.E.: TCP Congestion Control Comparison, University of Stellenbosch, 2012., str 4.
[34]Esterhuizen, A.,Krzesinski, A.E.: TCP Congestion Control Comparison, University of Stellenbosch, 2012., str 4.



Slika 11.: Usporedba Yeah i Illinois algoritama[35]

Ovdje je Yeah algoritam krenuo prvi sa prijenosom paketa, ali čim se uključio algoritam Illinois, brzina slanja mu je pala, to se događa zato jer je Illinois algoritam jako agresivan i ne može se koristiti zajedno sa nekim drugim algoritmom.

6.7.SCTP

SCTP je algoritam kojim se služi TCP kod prijenosa podataka strujanjem (eng. *Stream*). Takav prijenos podataka se koristi za prijenos multimedijskog sadržaja ili telefonija ostvarena preko interneta (eng. *VoIP – Voiceover Internet Protocol*) u realnom vremenu. Najčešći primjer su nogometne utakmice ili razgovor između dva korisnika koji se prenosi preko interneta u realnom vremenu.[36]

SCTP prenosi više struja podataka u isto vrijeme od izvora do odredišta, pa ovaj algoritam nazivaju *nextgeneration*TCPprotokol.

[35]Esterhuizen, A.,Krzesinski, A.E.: TCP Congestion Control Comparison, University of Stellenbosch, 2012., str 4.
[36]https://tools.ietf.org/html/rfc4960#page-10

Velika razlika između TCP i SCTP je to što SCTP odredištu pruža listu transportnih adresa koje se sastoje od jedne ili više IP adresa. Transportne adrese identificiraju adrese na koje se šalju i primaju SCTP paketi.

6.8.VEGAS

Vegas algoritam u TCP protokolu je nasljednik Reno algoritma sa dodatnim poboljšanjima u vidu izbjegavanja zagušenja.

Reno algoritam funkcionira tako što smanjuje prozor tek onda kada se zagušenje dogodi, odnosno kada se paket ne isporuči, to je problem koji je riješen sa Vegas algoritmom.

Vegas algoritam radi na način da kontrolirano povećava prozor na temelju RTT iznosa te na taj način prevenira da do zagušenja niti ne dolazi. Povećanjem RTT-a Vegas zaključuje da negdje u mreži dolazi do zagušenja, ali paketi i dalje prolaze, čim se to dogodi smanjuje se veličina prozora uz neprekidno provjeravanje RTT iznosa. Ako se RTT smanjuje tada se prozor povećava i pušta se veća količina paketa.[37]

Vegas radi u suradnji sa *slow-start* mehanizmom, a kod inicijalne konekcije veličina prozora je duplo manja nego kod Reno algoritma. Međutim Vegas kod svakog primljenog ACK-a povećava prozor te na taj način je daleko napredniji od Reno algoritma.[38]

6.9.VENO

Veno algoritam se koristi kao pomoć TCP protokolu i to u bežičnim (eng. *Wireless*) mrežama kao što su Wi-FI (eng. *Wireless Fidelity*) i mobilne (eng. *Cellular*) mreže. Ovaj algoritam koristi značajke Vegas algoritma. Te značajke se prvenstveno odnose na prepoznavanje zagušenja prema RTT vremenu te na temelju iznosa RTT-a se prozor povećava ili smanjuje. Druga vrlo bitna značajka je ta da Veno prepoznaje o kakvom se zagušenju, odnosno gubitku paketa radi.[39]

Generalno postoji gubitak paketa uzrokovan zagušenjem (eng. *CongestionLoss*) ili nasumičan gubitak paketa (eng. *Random Loss*). Na temelju informacije da je paket izgubljen se prozor smanjuje, ali samo u slučaju da se radi o gubitku paketa zbog zagušenja, a taj

[37] https://lafibre.info/images/doc/201207_TCP_Congestion_Control_Comparison.pdf
[38] https://web.archive.org/web/20160103040648/http://www.isoc.org/inet2000/cdproceedings/2d/2d_2.htm
[39] https://www.ie.cuhk.edu.hk/fileadmin/staff_upload/soung/Journal/J3.pdf

gubitak se prepoznaje ako se RTT poveća. Ako se RTT ne poveća, Veno algoritam zaključuje da se radi o slučajnom gubitku paketa te se veličina prozora ne smanjuje.

6.10. WESTWOOD

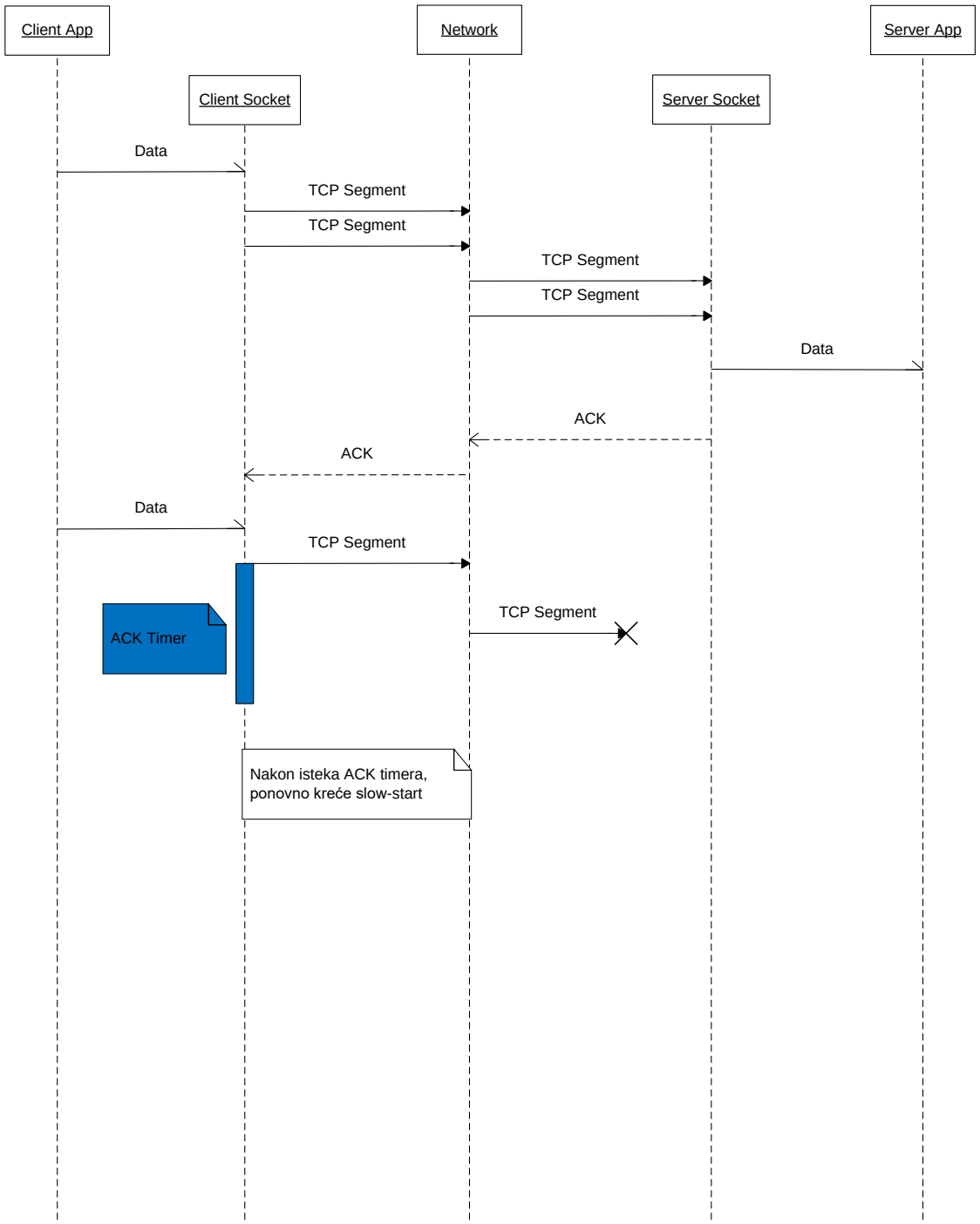
Westwood je algoritam kojeg koristi TCP protokol na pošiljateljevoj strani, a on je poboljšana verzija algoritma Reno te se koristi u žičanim i bežičnim mrežama velikih brzina. Najveća prednost Westwood algoritma je kod bežičnih mreža gdje standardni TCP protokol ima problema. Problemi nastaju onda kada TCP protokol ne može utvrditi razlog gubitka paketa, odnosno ne može odraditi da li se radi o nasumičnom gubitku ili se radi o zagušenju.[40]

Prednost ovog algoritma se temelji na konstantnoj provjeri ACK potvrda te na temelju njih se veličina prozora povećava ili smanjuje te u suradnji sa *slow-start*mehanizmom se ponovno uspostavlja veza i kreće prijenos. Kada dođe do gubitka paketa Reno smanjuje prozor na pola veličine koja je bila prije gubitka paketa, dok Westwood prati najnižu granicu brzine *slow-start*mehanizma te na tu vrijednost postavlja veličinu prozora.[41]

6.11. Dijagram međudjelovanja za algoritme za izbjegavanje zagušenja

Dijagramom međudjelovanja na slici 12 je prikazan proces komunikacije između objekata u procesu izbjegavanja zagušenja te što se događa kada do zagušenja dođe.

[40] https://ieeexplore.ieee.org/document/965869
[41] https://lafibre.info/images/doc/201207_TCP_Congestion_Control_Comparison.pdf



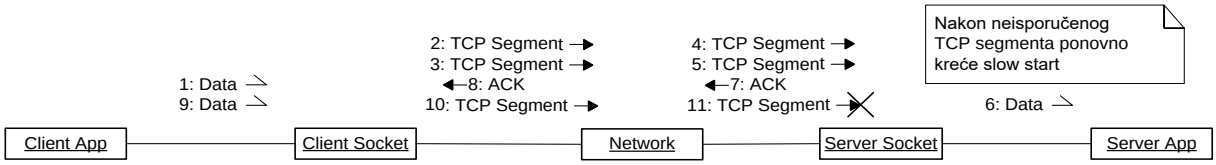
Slika 12.:Dijagram međudjelovanja za algoritme za izbjegavanje zagušenja

Dijagram prikazuje slanje TCP segmenta za koje se čeka potvrda (ACK) te se nakon pristigle potvrde ponovno šalje TCP segment, međutim ovog puta TCP segment nije stigao na odredište. Mreža je dobila TCP segment koji je proslijedila do poslužitelja, ali poslužitelj nikad nije poslao potvrdu da je TCP segment stigao.

Svi algoritmi djeluju na pošiljateljevoj strani, pa je tako i sa ovim algoritmom, pa svaki poslani TCP segment ima svoj iznos RTT-a, odnosno vrijeme koje je potrebno da se za poslani TCP primi ACK da je stigao. Nakon što to vrijeme prođe pošiljateljeva strana smatra da poslani TCP segment nije stigao. Ono što se nakon toga događa je da se algoritam vraća na *slow-start* te se kod početka ponovi izgubljeni (zadnji) TCP segment za koji pošiljatelj nije dobio potvrdu primitka.

6.12. Dijagram suradnje za algoritme za izbjegavanje zagušenja

Na slici 13 prikazan je dijagram suradnje koji kako je već navedeno prikazuje komunikaciju između objekata te se usko veže uz dijagram međudjelovanja.

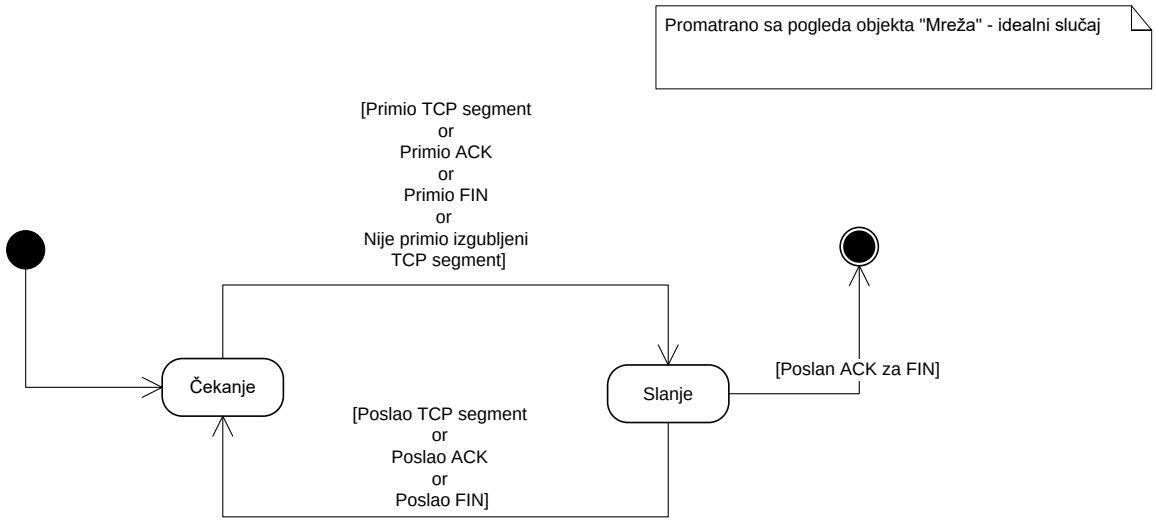


Slika 13.:Dijagram suradnje za algoritme za izbjegavanje zagušenja

Slika prikazuje komunikaciju između objekata, gdje je vidljivo da se u koraku jedanaest šalje TCP segment, međutim taj TCP segment nije stigao do odredišta. Nakon toga slijedi ponovno pokretanje *slow-start*mehanizma kako bi se ponovno poslao izgubljeni TCP segment, te se smanjio prozor da ne dođe do ponovnog gubitka segmenta.

6.13. Dijagram stanja i prijelaza za algoritme za izbjegavanje zagušenja

Dijagram stanja i prijelaza za algoritme za izbjegavanje zagušenja se nalazi na slici 14 gdje su navedena sva stanja u kojima se objekt „mreža“ nalazi te u kojim stanjima će se objekt naći tokom cijelog procesa.



Slika 14.:Dijagram stanja i prijelaza za algoritme za izbjegavanje zagušenja

Prvo stanje u kojem se objekt nalazi je „čeka TCP segment“, iz tog stanja može ići u stanje „šalje TCP segment“ ali je uvjet da je prethodno dobio TCP segment od klijenta. Iz tog stanja može ići na kraj procesa ukoliko više nema TCP segmenata koje može poslati prema poslužitelju.

U stanje „šalje ACK“ može doći onda kada šalje uzastopne potvrde (ACK) prema klijentu kako bi pošiljatelj znao da je TCP segment izgubljen te kako bi se ponovilo slanje, a drugi način da dođe do stanja „šalje ACK“ može doći onda kada prosljeđuje ACK da je TCP segment stigao.

Kako je ovdje moguće puno alternativnih puteva, odnosno potencijalnih događaja, obrađen je samo idealni slučaj kada TCP segment stigne ili se izgubi na putu prema pošiljatelju, a sukladno tome postoje dva alternativna puta za svaki od stanja u kojem se objekt može naći.

7. Mehanizmi brze retransmisije

TCP protokol koristi mehanizam brze retransmisije, kako bi poslao pakete koji nisu stigli na odredište. Prije svega TCP protokol koristi mehanizme i algoritme za izbjegavanje zagušenja te smanjenje gubitka paketa. Ukoliko ipak dođe do gubitka paketa, vrlo je bitno da se ti neisporučeni paketi što prije ponovno pošalju i dostave na odredište.

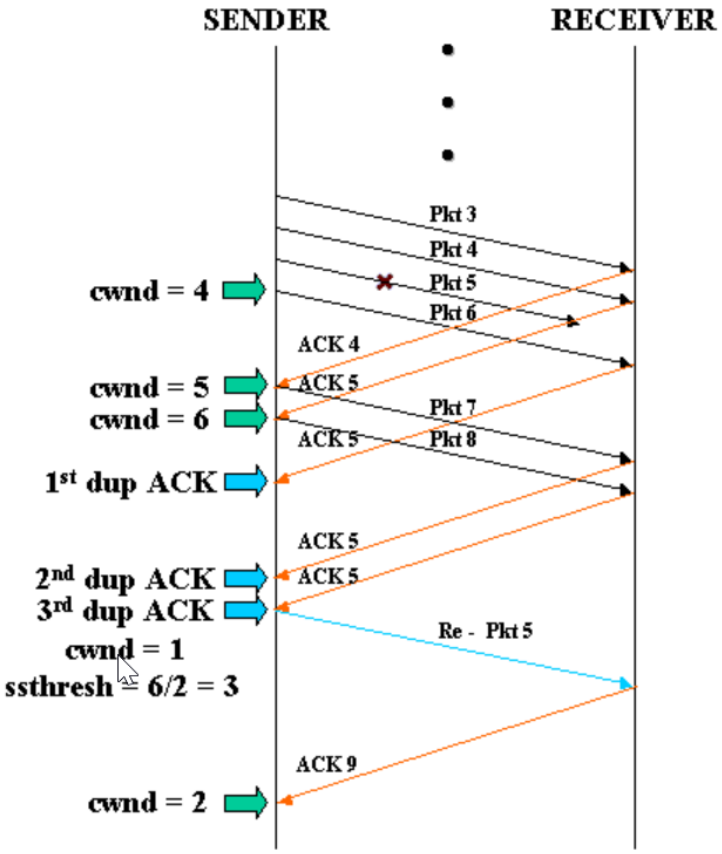
Nakon što istekne RTO (eng. *RetransmissionTimeout*) radi se ponovno slanje paketa, međutim to vrijeme daleko duže od potvrda koje pošiljatelj prima za svaki paket. RTO mora biti veći jer paket može putovati raznim putevima do odredišta, te se može dogoditi da se negdje u mreži dogodi zagušenje tako da paketu treba više vremena da dođe do odredišta. Međutim problem je ako zagušenja nema, a paket stigne na odredište te onda postoji neko vrijeme koje je potrebno da potvrda stigne do pošiljatelja.[42]

Nakon što stigne dupla potvrda (eng. *DoubleAcknowledgement*) TCP i dalje ne može pouzdano zaključitiradi li se o gubitku paketa ili su paketi stigli pogrešnim redoslijedom. Nakon toga slijedi još jedan ACK jer nakon tri uzastopno primljena ACK-a predajnik zaključuje da se radi o gubitku paketa.

Vrijeme koje je potrebno da predajnik primi tri ACK-a je daleko manje od RTO-a pa se zato ovaj mehanizam naziva brza retransmisija. Pomoću ovog mehanizma, TCP protokol ostvaruje bolji QoS.

Na slici 15 je prikazan dijagram koji opisuje potvrdu koju odredište šalje, te proces brze retransmisije.

[42] https://tools.ietf.org/html/rfc2001



Slika 15.: Proces brze retransmisije[43]

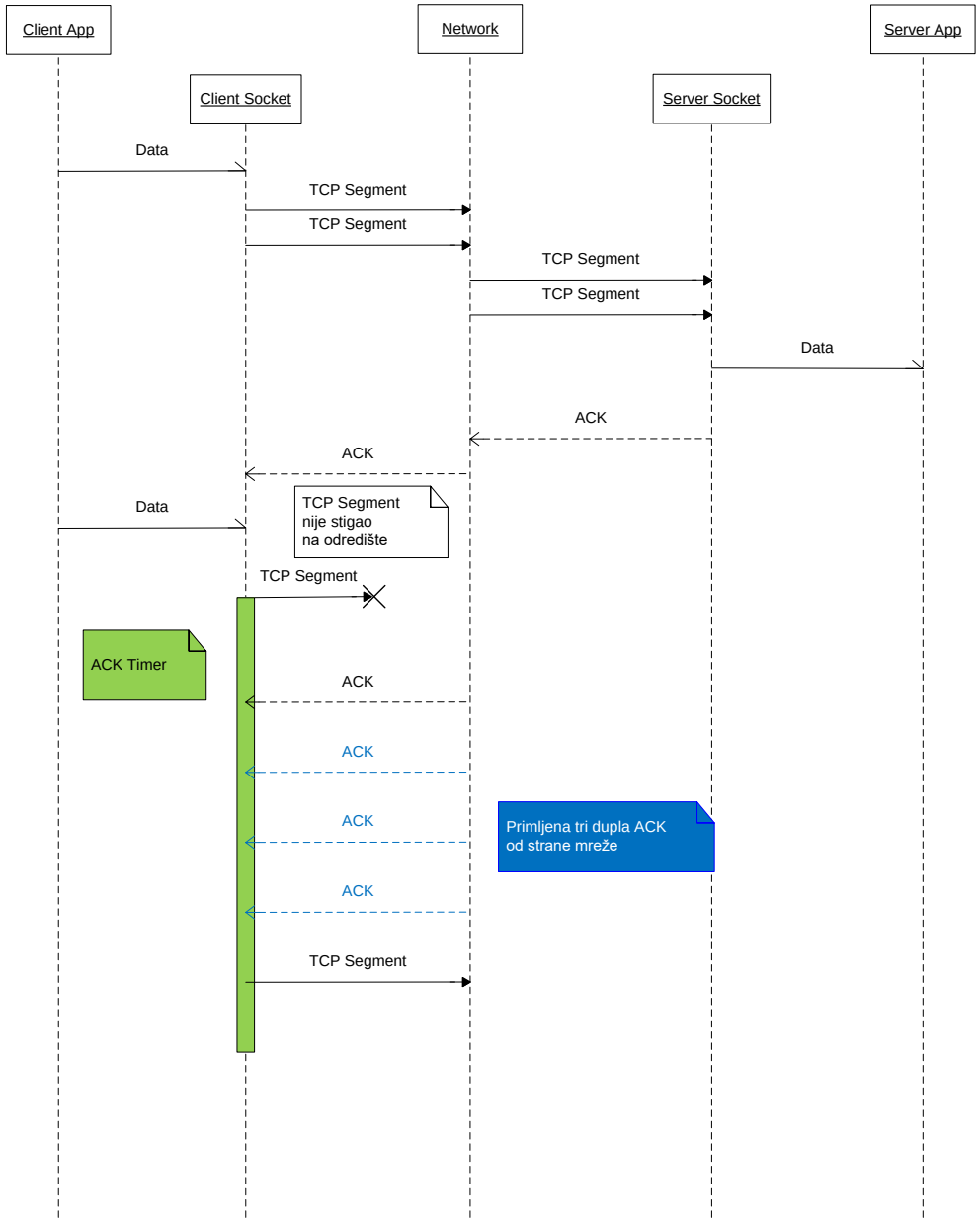
Ova slika se nadovezuje na sliku 8 jer se i sami mehanizmi brze retransmisije i *slow-start* koriste zajedno te jedan bez drugoga ne bi imali svrhu. Proces kreće sa povećavanjem prozora i slanjem paketa, paket broj pet nije stigao na odredište, a pošiljatelj je dobio tri uzastopna ACK, međutim do tada je već trebao imati i potvrdu da je paket broj pet stigao. Odmah se radi brza retransmisija te se prozor smanjuje za pola, kako je u ovom primjeru prozor prije gubitka paketa bio na šest, sad se smanjio na tri. Nakon što je izgubljeni paket ponovno poslan, čeka se potvrda od primatelja da je paket uspješno stigao na odredište.

Ovo je puno brža metoda od standardnog pristupa gdje se čeka da istekne RTT te da se tek onda pošalje izgubljeni paket. Kako je vidljivo i sa ove slike vrijeme koje je potrebno da izgubljeni paket stigne do odredišta je duplo manje u odnosu na standardni način retransmisije.

[43] https://www.isi.edu/nsnam/DIRECTED_RESEARCH/DR_WANIDA/DR/JavisInActionFastRetransmitFrame.html

7.1.Dijagram međudjelovanja za mehanizme brze retransmisije

Slikom 16 prikazana je komunikacija između objekata sa vremenskom crtom izvršavanja. Dijagram međudjelovanja prikazuje sve poruke koje se razmjenjuju unutar procesa brze retransmisije kada dođe do gubitka paketa.



Slika 16.:Dijagram međudjelovanja za mehanizam brze retransmisije

Sam proces kreće sa uspostavom konekcije te slanjem paketa što je opisano u dijagramu na slici 9 u četvrtom poglavlju. Nakon što se inicijalni dio izvrši kreće slanje TCP segmenata, a ovim dijagramom je opisano što se događa kada se TCP segment ne isporuči, odnosno ne stigne do objekta „mreža“.

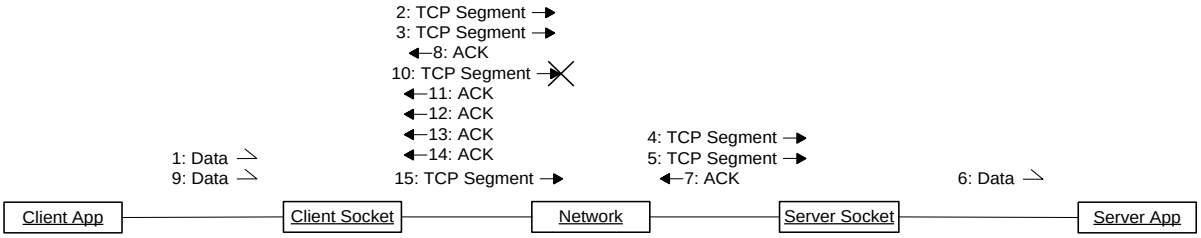
Prije svega kada pošiljatelj pošalje TCP segment, za njega kreće odbrojavanje, odnosno pali se *ACK timer*te nakon isteka tog vremena, proces se vraća na *slow-start.*

Kada TCP segment ne stigne do mreže, mreža dostavlja ACK te tri dodatna uzastopna ACK prema pošiljatelju. Kada pošiljatelj primi dupli ACK, ne zna radi li se o pogrešnom ACK ili je TCP segment izgubljen stoga čeka tri duplicirana ACK kako bi poslano izgubljeni TCP segment.

Vidljivo je sa slike da je izgubljeni TCP segment poslan prije isteka *ACK timera*, čime se ubrzava slanje izgubljenih TCP segmenata u slučaju da se izgube i ne stignu na odredište. Baš iz tog razloga se algoritam zove *fastretransmit* odnosno mehanizam brze retransmisije jer ne čeka istek *ACK timera*, nego čim dobije tri duplicirana ACK, ponavlja slanje izgubljenog TCP segmenta.

7.2.Dijagram suradnje za mehanizme brze retransmisije

Dijagram suradnje kako je već objašnjeno, prikazuje komunikaciju između objekata, a slikom 17 je prikazana komunikacija objekata kod mehanizma brze retrasnsmisije.

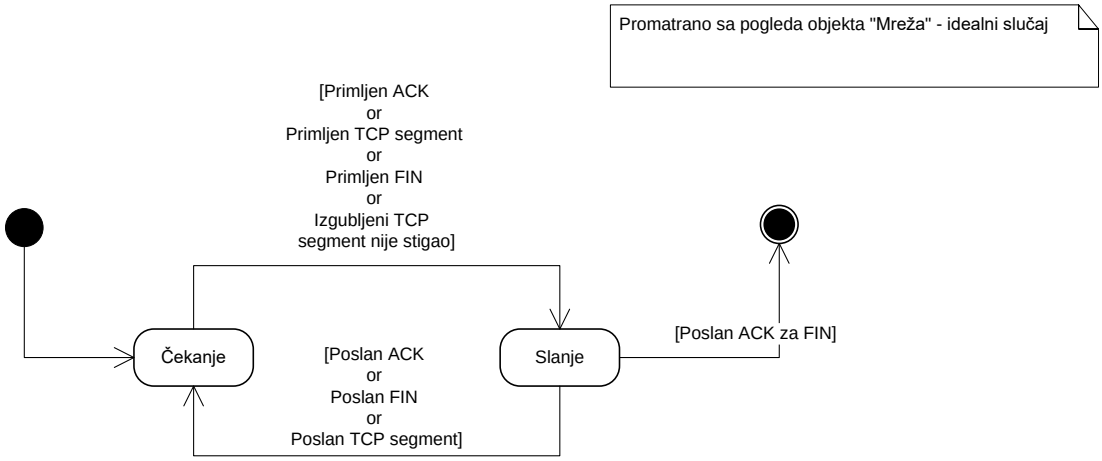


Slika 17.:Dijagram suradnje za mehanizam brze retransmisije

Sa slike je vidljivo da se primarna komunikacija odvija preko objekta „mreža“, te u koraku deset, prilikom slanja TCP segmenta, isti nije stigao do mreže. Od tog trenutka se pokreće ACK timer, odnosno vrijeme da se od mreže dobije ACK za isporučen TCP segment. U međuvremenu klijent (pošiljatelj) dobiva tri duplicirana ACK što znači da je segment izgubljen te se automatski radi brza retransmisija izgubljenog paketa. Ta retransmisija se odvija prije isteka ACK timera, te se na taj način ubrzava sam proces ponovnog slanja izgubljenog segmenta.

7.3.Dijagram stanja i prijelaza za mehanizme brze retransmisije

Dijagram stanja i prijelaza za mehanizme brze retransmisije je opisan slikom 18. Ovaj dijagram opisuje stanja u kojem se nalazi pojedini objekt, a u ovom primjeru se radi o objektu „mreža“ jer nije moguće opisati stanja svih objekata koji sudjeluju u procesu.



Slika 18.:Dijagram stanja i prijelaza za mehanizam brze retransmisije

Na slici je vidljivo kako se objekt „mreža“ nalazi u stanju čeka TCP segment, to stanje zadržava sve dok mu ne stigne TCP segment, tada ide u stanje šalje TCP segment, a ukoliko TCP segment ne stigne proces je gotov te ide u završetak. Ukoliko TCP segment nije stigao (izgubio se), objekt prelazi u stanje šalje ACK te ponovno prelazi u stanje čeka TCP segment ukoliko isti nije stigao. Taj proces se ponavlja dok pošiljatelj ne pošalje izgubljeni TCP segment, a onda nastavlja drugim tokom prema stanju šalje TCP segment. Iz stanja šalje TCP segment prelazi u stanje čeka ACK, a ukoliko nema više TCP segmenata, završava se proces i ide u završetak.

8. Zaključak

Ovaj rad je kroz sedam poglavlja opisao rad TCP protokola, njegovih mehanizama i algoritama koji se koriste ovisno o mrežu u kojoj se koristi. Od samih početaka razvoja mreža bili su predstavljeni protokoli, međutim kako su se mreže razvijale, morali su se razviti i novi protokoli prema kojima bi se prijenos podataka i ostvario. Tako je razvijen i TCP protokol, a ideja za njegov razvoj je bio siguran i pouzdan prijenos podataka. Nekoć je to bilo dovoljno da se osigura siguran prijenos, međutim sa razvojem mreža razvijao se i TCP protokol, pa su njegovim osnovnim funkcionalnostima nadodane funkcije koje mu omogućuju siguran prijenos podataka.

Prije svega potrebno je „skicirati“ sustav kako bi se saznalo što je sve potrebno za sam rad sustava, ali i njegovih protokola koji su vrlo važna stavka, pa se iz tog razloga koristi UML jezik. On omogućava detaljan pregled sustava iz više različitih kutova, pa je tako moguće „vidjeti“ kako sustav radi, koji elementi su uključeni u pojedini proces, a također je moguće vidjeti što i kada pojedini element u sustavu radi.

Nakon što se dobije detaljna slika o sustavu, moguće je odrediti koje funkcionalnosti je potrebno dodati TCP protokolu kako bi on efikasno radio. Tu je također bitno saznati na koji način se odvija sam prijenos, a taj pogled osigurava OSI i TCP/IP model koji detaljno navode protokole koji se koriste te način na koji se odvija komunikacija između slojeva.

TCP se nalazi na transportnom sloju pa je transportni sloj bio i tema rada te je detaljnije objašnjen. Na temelju tih podataka vidljivo je da se TCP protokol koristi onda kada je potrebno osigurati siguran prijenos podataka, kao što je mail, Internet servisi i svi senzorski podaci koji se trebaju prenijeti jer ukoliko bi došlo do gubitka paketa, na primateljevoj strani bi došlo do pogreške jer primatelj ne bi mogao razumjeti i pročitati dobivene podatke.

Da bi takav prijenos bio moguć koriste se razni mehanizmi kao što su 3-*way handshake*i*slow-start*. Kod uspostave konekcije, TCP protokol zahtjeva spojnu vezu, što znači da uspostavlja logičku vezu između primatelja i pošiljatelje, a tu do izražaja dolazi 3-*way handshake* mehanizam koji tu vezu uspostavlja. Na temelju funkcionalnosti PAR-a putem kojeg zaključuje da je veza uspostavljena te da je sve spremno za prijenos podataka.

Kada se ta veza uspostavi počinje djelovati *slow-start* mehanizam koji omogućava siguran i brz prijenos bez prekida, odnosno gubitka paketa. To radi na način da kad se veza uspostavi povećava brzinu slanja paketa sve do one granice kada se popuni kapacitet linka.

Također tu je bitan i mehanizam brze retransmisije koji kada dođe do neželjenog gubitka paketa, ponovno šalje paket prema odredištu, to je vrlo bitno jer da nema ovog mehanizma, ne bi bilo moguće osigurati siguran i pouzdan prijenos podataka. TCP koristi kontrolnu sumu kako bi izvorište i odredište bili sigurni da poslani paket sadrži točan podatak.

Vrlo je bitno spomenuti algoritme za izbjegavanje zagušenja koji ima jako puno, a u radu su navedeni oni koji se najčešće koriste te su objašnjene njihove pojedinačne funkcionalnosti. Vrlo je bitno da su ti algoritmi razvijeni jer sam TCP protokol nema dovoljno algoritama da obuhvati sve mreže u kojima se može koristiti, pa su tako neki algoritmi razvijeni za manje mreže gdje se šalje manja količina podataka, a također postoje algoritmi za bežične mreže i za mreže velike brzine koje imaju drugačije značajke.

Kod bežičnih mreža treba uzeti u obzir i smetnje koje se mogu dogoditi prilikom prijenosa podataka, a kojih nema u prijenosu putem optičkog vlakna. Također kod mreža velike brzine se šalje jako puno paketa u vrlo kratkom periodu jer se mogu koristiti linkovi čija je brzina veća od 10Gbps.

Literatura

1. Esterhuizen, A.,Krzesinski, A.E.: TCP CongestionControlComparison, University ofStellenbosch, 2012.
2. Fall, K., Richard Stevens, W.: TCP/IP IllustratedVolume 1, Addison-Wesley, New York, 2012.
3. Huang, X., Lin, C., Ren, F., Yin, H.: HighSpeed TCP ModelingandAnalysis, Tsinghua University, Bejing, 2004.
4. Kavran, Z. Predavanja iz predmeta Računalne mreže: Mrežni sloj, FPZ, listopad, 2013. str. 20-33
5. Kavran, Z. Predavanja iz predmeta Računalne mreže: OSI slojevi i mediji za prijenos podataka, FPZ, listopad, 2013.
6. Marchese, M.: QoSoverHeterogeneusNetworks, University ofGenoa, Italy, 2007.
7. Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram aktivnosti, FPZ, svibanj, 2018.
8. Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram slučaja uporabe, FPZ, svibanj, 2015.
9. Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram stanja i prijelaza, FPZ, svibanj, 2015.
10. Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram suradnje, FPZ, studeni, 2015.
11. Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram međudjelovanja, FPZ, studeni, 2018.
12. Mrvelj, Š. Predavanja iz predmeta Analiza i modeliranje prometnih sustava: Dijagram klasa, FPZ, studeni, 2016.
13. Xu, L., Harfoush, K., Rhee, I.: BinaryIncreaseCongestionControl (BIC) for FastLong-Distance Networks, North Carolina State University, Raleingh, 2004.
14. Yang, P., Shao, J., Luo, W., Xu, L., Deogun, J.: TCP CongestionAvoidanceAlgorithmIdentification, University ofNabraska, Lincoln, 2014.
15. http://what-when-how.com/qos-enabled-networks/traffic-types-qos-enabled-networks-part-2/ (22.06.2020.)
16. http://www.etfos.unios.hr/~drago/predmeti/mip/predavanje10.pdf (22.06.2020.)
17. https://ieeexplore.ieee.org/document/965869 (22.06.2020.)

18. https://lafibre.info/images/doc/201207_TCP_Congestion_Control_Comparison.pdf (22.06.2020.)
19. https://personal.oss.unist.hr/~alen/Literatura-SKRIPTA!!!/slide6.pdf(22.06.2020.)
20. https://tools.ietf.org/html/rfc2001 (22.06.2020.)
21. https://tools.ietf.org/html/rfc4960#page-10 (22.06.2020.)
22. https://web.archive.org/web/20160103040648/http://www.isoc.org/inet2000/cdproceedings/2d/2d_2.htm (22.06.2020.)
23. https://www.fer.unizg.hr/_download/repository/INFMRE-2017_09.pdf (22.06.2020.)
24. https://www.fpz.unizg.hr/hgold/aimps20092010/predavanja/ANiMOPS-20092010.pdf (22.06.2020.)
25. https://www.ie.cuhk.edu.hk/fileadmin/staff_upload/soung/Journal/J3.pdf (22.06.2020.)
26. https://www.isi.edu/nsnam/DIRECTED_RESEARCH/DR_WANIDA/DR/JavisInActionFastRetransmitFrame.html(22.06.2020.)
27. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/ (22.06.2020.)

Popis kratica

ACK (eng. *Acknowledgment*) – potvrda

AIMD (eng. *AdditionalIncreaseMultiplicativeDecrease*) – algoritam za određivanje veličine prozora

ASP (eng. *AppleTalkSessionProtocol*) – protokol za sesiju razgovora koju koristi Apple

BIC (eng. *BinaryIncreaseCongestion*) – protokol koji koristi binarno kodiranje za određivanje veličine prozora

CTCP (eng. *CodedTransmissionControlProtocol*) – protokol za kontrolu prijenosa koji koristi kodiranje

DNS (eng. *Domain Name System*) – sustav za pripisivanje domenskih imena

FIN (eng. *Finalize*) – zatvaranje TCP konekcije

FTP (eng. *File Transport Protocol*) – protokol za prijenos datoteka

HSTCP (eng. *HighSpeedTransmissionControlProtocol*) – protokol za kontrolu prijenosa u mrežama velike brzine

HTTP (eng. *Hyper Tekst Transfer Protocol*) – protokol za prijenos hiperteksta

ICMP (eng. *Internet ControlMessageProtocol*) – protokol kontrole internet poruka

IMAP (eng. *Internet Message Access Protocol*) – protokol za pristup elektroničkoj pošti sa udeljenog računala

IP (eng. *Internet Protokol*) – Internet protokol

IW (eng. *Initial Window)* – veličina pošiljateljevog prozora zagušenja nakon *3-way handshake* uspostave konekcije

JPEG – oznaka formata slike

MAC (eng. M*edia Access Control*) – medijska kontrola pristupa

MIDI – oznaka formata slike

MPEG – oznaka formata slike

MTU (eng. *MaximumTransmissionUnit*) – maksimalna veličina paketa

NFS (eng. *Network File System*) – mrežni sustav datoteka

OSI (eng. *Open Systems Interconnection*) – referentni model za opis mrežnog sustava i protokola

PAR (eng. *PositiveAcknowledgementwith Re-transmission)* – retransmisija na temelju pozitivnog odgovora

PICT – oznaka formata slike

QoS (eng. *Qualityof Service*) – razina kvalitete usluge

RTO (eng. *RetransmissionTimeout*) – vrijeme koje je potrebno da potvrda stigne do pošiljatelja

RTP (eng. *Real Time Protocol*) – protokol za prijenos podataka u realnom vremenu

RTT (eng. *RoundTrip Time*) – vrijeme koje je potrebno paketu da stigne od izvorišta do odredišta

SCTP (eng. *StreamControlTransmissionProtocol*) – protokol za kontrolu prijenosa strujanjem

SMSS (eng. *SenderMaximum Segment Size)* – maksimalna veličina segmenta koju pošiljatelj može poslati

SMTP (eng. *Simple Mail TrasferProtocol*) – jednostavni protokol za prijenos elektroničke pošte

SQL (eng. *StructuredQueryLanguage*) – strukturirani upitni jezik

TCP (eng. *TransmissionControlProtocol*) – protokol za kontrolu prijenosa

TCP/IP(eng. *TransmissionControlProtocol/Internet Protocol*) – protokol za kontrolu prijenosa / internet protokol

TIFF – oznaka formata slike

UDP (eng. *UserDatagramProtocol*) – protokol za prijenos korisničkih datagrama

UML (eng. *UnifiedModelingLanguage*) – računalni jezik za izradu dijagrama koji opisuju mrežni sustav

VoIP (eng. Voiceover Internet Protocol) – protokol za prijenos govora putem Internet protokola

Wi-Fi (eng. *Wireless-Fidelity*) – bežićni prijenos podataka

Sveučilište u Zagrebu
Fakultet prometnih
znanosti
10000 Zagreb
Vukelićeva 4

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOST

Izjavljujem i svojim potpisom potvrđujem kako jeovaj diplomskirad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu što pokazuju korištene bilješke i bibliografija.

Izjavljujem kako nijedan dio rada nije napisan na nedozvoljen način, niti je prepisan iz necitiranog rada, te nijedan dio rada ne krši bilo čija autorska prava.

Izjavljujem također, kako nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu diplomskograda pod naslovom KOMUNIKACIJSKI MODELI ZA OPIS RADA TCP PROTOKOLAI NJEGOVIHMEHANIZAMA na internetskim stranicama i repozitoriju Fakulteta prometnih znanosti, Digitalnom akademskom repozitoriju (DAR) pri Nacionalnoj i sveučilišnoj knjižnici u Zagrebu.

U Zagrebu, 1.9.2020

Studentica:

(potpis)