Preddiplomski studij
Računarstvo

Komunikacijske mreže

Upute za izvođenje 2. ciklusa laboratorijskih vježbi

Osnove funkcionalnosti mrežnog i transportnog sloja

Ak.g. 2007./2008.

Sadržaj

1 Uvod

U ovoj vježbi ćemo detaljnije razmotriti i ispitati funkcionalnost mrežnog i transportnog sloja u IP mrežama. U odjeljku 2.1 opisana je naredba `traceroute`. Ova naredba, uz naredbu `ping` obrađenu u prošloj vježbi, predstavlja temeljni alat za dijagnostiku IP mreže. U odjeljcima 2.2 i 2.3 opisani su protokoli usmjeravanja općenito te konkretan protokol usmjeravanja, RIP.

Drugi dio vježbe odnosi se na protokole transportnog sloja, i započinje opisom naredbe `netcat` koju koristimo kao generator prometa pri analizi rada protokola UDP i TCP opisanih u odjeljcima 3.2 i 3.3.

Materijal u ovoj skripta izložen je na način da su, u svakom odjeljku, prvo opisani primjeri na stvarnoj mreži, a nakon toga slijede zadaci vezani za rad u emulatoru/simulatoru IMUNES koje je potrebno riješiti u sklopu vježbe. Primjere na stvarnoj mreži nije potrebno izvoditi u laboratoriju, odnosno, zbog specifičnosti laboratorija nije ih niti moguće izvesti.

U nekim zadacima potrebno je u emulator/simulator IMUNES učitati gotove scenarije. Ovi scenariji nalaze se u direktoriju `/root/EXAMPLES/`.

2 Mrežni sloj

2.1 Naredba `traceroute`

Kako odrediti preko kojih je čvorova određeni IP paket prošao na svom putu do odredišta? U Internet mreži odgovor na ovo pitanje je, nažalost, nikako. IP mreže, od kojih se sastoji Internet, ne pružaju mogućnost ispitivanja putova koji se koriste u mreži za prijenos paketa. Nakon što je paket otišao s izvorišnog računala, ono ne može saznati kojim je putem paket prošao kroz mrežu do odredišta. Iako sâm IP protokol, doduše, omogućava zapisivanje skokova kroz koje je paket prošao na svom putu do odredišta, ova funkcionalnost povlači određene probleme te je u Internet mreži uglavnom administrativno onemogućena.

Međutim, vrlo domišljatim korištenjem nekih postojećih mehanizama i protokola, koji izvorno imaju potpuno drugu namjenu, Amerikanac Van Jacobson je predložio rješenje koje omogućava saznavanje čvorova kroz koje će paketi poslani prema nekom odredištu *najvjerojatnije* proći. Njegovo rješenje ugrađeno je u popularnu naredbu `traceroute` (u *Microsoft Windows* operacijskom sustavu koristi se naredba `tracert`).

Naredba `traceroute` uzima jedan argument i to je adresa ili ime odredišnog računala. Slijedi primjer njenog korištenja.

```
> traceroute www.google.com
Tracing route to www.google.akadns.net [216.239.59.99] over a maximum of 30 hops:

 1   12 ms    12 ms    11 ms  161.53.19.1
 2    1 ms     1 ms     2 ms  161.53.16.9
 3    1 ms     2 ms     2 ms  193.198.229.9
 4    4 ms     4 ms     5 ms  193.198.228.5
 5    2 ms     3 ms     2 ms  carnet.hr1.hr.geant.net [62.40.103.217]
 6    9 ms    10 ms    10 ms  hr.hu1.hu.geant.net [62.40.96.146]
 7   17 ms    17 ms    18 ms  hu.at1.at.geant.net [62.40.96.177]
 8   34 ms    36 ms    35 ms  at.ch1.ch.geant.net [62.40.96.2]
 9   44 ms    44 ms    43 ms  so-6-0-0.ar2.cdg2.gblx.net [208.48.23.161]
10   51 ms    50 ms    51 ms  so6-0-0-2488m.ar2.lon3.gblx.net [67.17.66.2]
11   51 ms    51 ms    51 ms  level-3public-peering.ge-5-0-0.ar2.lon3.gblx.net [208.51.239.162]
12   51 ms    54 ms    52 ms  ae-0-17.gar1.london1.level3.net [212.187.131.169]
13   52 ms    51 ms    51 ms  so-6-0.metro2-londencyh00.london1.level3.net [212.113.3.26]
14   50 ms    50 ms    50 ms  195.50.116.70
15   58 ms    49 ms    50 ms  216.239.46.173
16   77 ms    80 ms    78 ms  216.239.49.254
17   80 ms    79 ms    80 ms  216.239.49.121
18   78 ms    80 ms    79 ms  216.239.59.99

Trace complete.
```

Izlaz naredbe `traceroute` prikazuje niz skokova kroz koje paketi prolaze na putu do odredišta. U prethodnom primjeru, od računala na kojem je pokrenut `traceroute` do računala `www.google.com` postoji 17 međučvorova (zadnji čvor u nizu, 18., predstavlja odredište `www.google.com`). Prvi stupac predstavlja redni broj detektiranog čvora. Sljedeća tri stupca predstavljaju tri mjerenja, odnosno "ping" vrijeme. Zadnji stupac označava IP adresu čvora, a u slučaju da je ova adresa registrirana u DNS sustavu navedeno je i ime čvora.

Primijetimo još jednu interesantnu činjenicu vezanu uz navedeni primjer. Zašto je "ping" vrijeme do prvog čvora veće od ping vremena za čvorove 2, 3, 4, 5 i 6? Ukratko, kad usmjerivač dobije paket kojem je TTL vrijeme 1, a odredište tog paketa nije sâm usmjerivač, on mora paket ispustiti te pošiljatelju odgovoriti posebnom ICMP porukom. Ova obrada zahtijeva određene procesorske resurse u usmjerivaču i traje duže nego samo prosljeđivanje paketa. Ovisno o kapacitetu procesora u usmjerivaču, ovo vrijeme može biti duže ili kraće. Izgleda da je procesor u prvom usmjerivaču sporiji u odnosu na nekoliko ostalih na putu do računala `www.google.com`.

Zadatak 1. U simulatoru IMUNES, ispitajte način rada naredbe `traceroute` na mreži iz primjera `ping/ping.imn`. Potrebno je slušati promet na pojedinim sučeljima i utvrditi mehanizam na kojem se temelji naredba. Kako se koristi TTL polje?

Zadatak 2. Utvrdite neke od mogućih situacija u kojima naredba `traceroute` može proizvesti rezultat koji nije ispravan (*naputak*: pogledajte što piše u uputama za naredbu, `man traceroute`).

2.2 Protokoli za usmjeravanje

Tablica usmjeravanja u svakom IP čvoru, bio on usmjerivač ili "obično" računalo, pohranjuje informaciju o dostupnim odredištima te o "smjeru" kojim treba prosljeđivati datagrame do tih odredišta. Prosljeđivanje IP datagrama obavlja se isključivo na temelju informacija u IP tablici (usmjeravanja).

Informacije se u IP tablicu čvora mogu upisivati od strane administratora sustava ili pak od strane protokola za usmjeravanje. Protokoli za usmjeravanje u IP čvorovima rade na način da s ostalim čvorovima u mreži razmjenjuju informacije o odredištima, te na temelju tih informacija donose zaključke o usmjeravanju datagrama. Kad protokol za usmjeravanje donese odluku o najkraćem putu kojim treba prosljeđivati datagrame prema nekom odredištu, on u tablicu usmjeravanja upisuje odredište i sljedeći skok na tom najkraćem putu.

Bitno je napomenuti da se sama odluka na koje sučelje treba proslijediti IP datagram donosi isključivo na temelju onoga što piše u tablici usmjeravanja. Nadalje, usmjerivački protokol na usmjeravanje utječe isključivo na način da mijenja tablicu usmjeravanja.

Protokoli za usmjeravanje koriste IP protokol za komunikaciju među čvorovima koji obavljaju usmjeravanje.

Protokole za usmjeravanje u Internetu dijelimo u sljedeće dvije kategorije: ''unutarnji'' i ''vanjski''. Unutarnji protokoli koriste se unutar tzv. *autonomnih sustava*. Autonomni sustav (AS) je IP mreža koja se nalazi pod administrativnom nadležnošću jednog tijela. Protokol koji se koristi unutar AS-a nije vidljiv izvan tog AS-a. Unutar autonomnih sustava nadležno tijelo može koristiti bilo koji protokol za unutarnje usmjeravanje, u čemu se, između ostalog, i sastoji njegova autonomija. Autonomni sustav zadužen je za ostvarivanje povezanosti među čvorovima unutar sebe. Primjeri protokola za unutarnje usmjeravanje su *Routing Information Protocol* (RIP), protokol *Open Shortest Path First* (OSPF) i protokol *Intermediate System - Intermediate System* (IS-IS).

Za razmjenu usmjerivačkih informacija između autonomnih sustava koriste se vanjski usmjerivački protokoli. S obzirom da svi autonomni sustavi moraju moći razmjenjivati informaciju s ostalim sustavima, izbor protokola za vanjsko usmjeravanje u Internetu nije proizvoljan, već se mora koristiti protokol *Border Gateway Protocol* verzije 4 (BGP4).

2.3 Protokol RIP

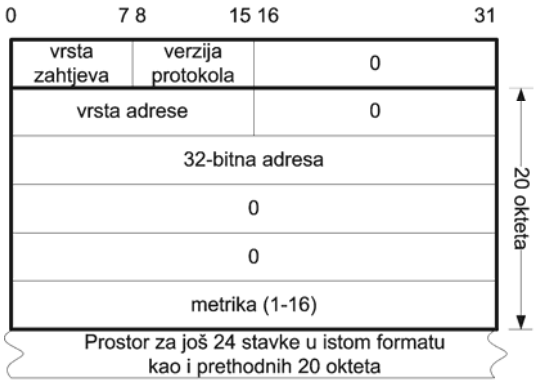
Jedan od najstarijih i najjednostavnijih protokola za unutarnje usmjeravanje jest protokol RIP. Ovaj protokol spada u kategoriju tzv. protokola temeljenih na vektorima udaljenosti. Vektor udaljenosti je lista odredišta i pripadajućih udaljenosti do tih odredišta. Razmjenom ovakvih vektora udaljenosti, čvorovi mogu jednostavno pronaći najkraće putove do svih ostalih odredišta.

Način na koji se interpretiraju vektori udaljenosti u čvorovima je sljedeći. Neka, na primjer, čvor R1 primi vektor udaljenosti od susjednog čvora R2, kojim R2 kaže da se do odredišta 161.53.10.0 (D) može doći preko njega u 4 skoka. Ukoliko R1 prije primitka tog vektora nije znao da uopće postoji odredište D (tj., nije ga imao u tablici usmjeravanja), on u svoju tablicu usmjeravanja jednostavno dodaje redak koji izgleda ovako:

Odredište	Sljedeći skok	Udaljenost
161.53.10.0	R2	5

Međutim, ako je R1 u svojoj tablici usmjeravanja već imao odredište 161.53.10.0, te ako mu je udaljenost do tog odredišta bila, na primjer, 3, onda se postojeći redak u tablici ne mijenja. To je zato što R1 već zna ''doći'' do navedenog odredišta, i to u manje skokova nego preko R2. Dakle, redak u tablici se ažurira samo ako je novi put do odredišta kraći od onoga koji se već koristi (koji je već zapisan).

Zaglavlje protokola RIP prikazano je na slici 1.

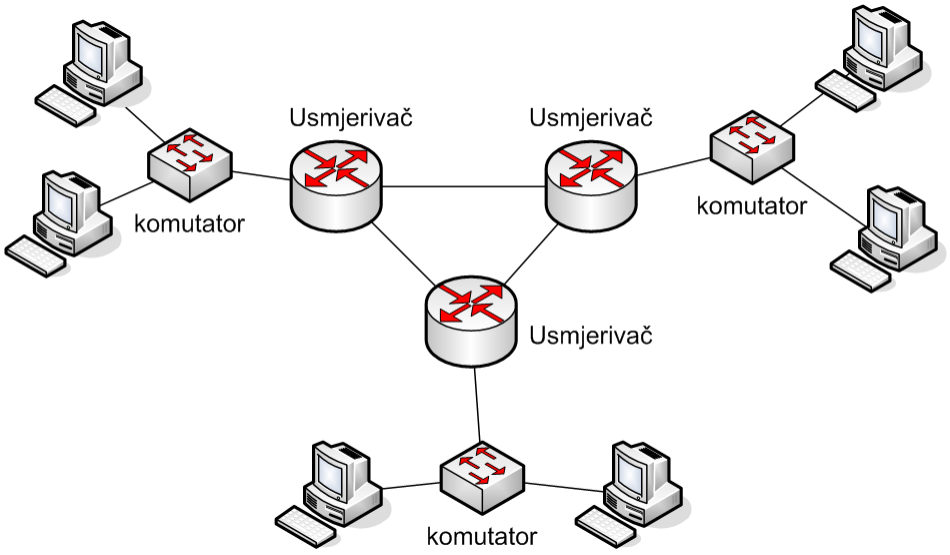


Slika 1: Zaglavlje protokola RIP

Polje *vrsta zahtjeva* označava predstavlja li paket zahtjev za informacijom (vrijednost 1) ili odgovor na zahtjev (vrijednost 2). Polje *verzija protokola* označava verziju protokola RIP koja se koristi. Polje *vrsta adrese* najčešće ima vrijednost 2, što označava da se radi o 32-bitnim IP adresama. Nakon vrste adrese idu sama adresa i pripadajuća metrika, čiji je smisao broj skokova do odredišta koje adresa predstavlja.

Zadatak 3. U simulatoru IMUNES, pomoću alata Ethereal, uhvatite paket koji pripada protokolu RIP, te proučite njegov sadržaj.

Zadatak 4. U simulatoru IMUNES konstruirajte mrežu koja sadrži 3 podmreže povezane usmjerivačima, kao što je prikazano na slici 2. Konfigurirajte IP adrese na svim čvorovima u mreži tako da mreža bude povezana.



Slika 2: Arhitektura mreže u *zadatku 4*

Zadatak 5. U mreži iz prethodnog zadatka konfigurirajte usmjeravanje tako da dođe do petlje u usmjeravanju (bez korištenja protokola za usmjeravanje). Pomoću Ethereala utvrdite što se tada događa s paketima koji ''uđu'' u petlju.

Zadatak 6. Proučite primjere `RIP/RIP.imn` i `RIP/RIP1.imn` u skladu s uputama navedenim u datoteci `RIP/README`.

Zadatak 7. Ustanovite na koji je način CARNet povezan s ostatkom Internet mreže.

3 Transportni sloj

Protokoli koji su na raspolaganju za obavljanje funkcije transportnog sloja u Internetu su TCP i UDP. TCP protokol omogućava pouzdan i slijedni prijenos niza okteta između udaljenih procesa. Sa stajališta procesa koji razmjenjuju niz okteta, TCP predstavlja pouzdan ''tunel'' kroz koji je oktete moguće prenijeti bez gubitaka i u očuvanom redoslijedu. Da bi omogućio ovakvu vrstu usluge procesima koji ga koriste, TCP interno koristi velik broj složenih mehanizama. Iako su ovi mehanizmi u potpunosti transparentni za procese koji koriste TCP usluge, korisno ih je poznavati jer oni uvelike utječu na brzinu i kašnjenje pri prijenosu podataka između komunicirajućih procesa. Tako je, na primjer, zbog svojstava ponavljanja slanja izgubljenih paketa i oscilacije kašnjenja u isporuci paketa, nepovoljno koristiti TCP za prijenos podataka osjetljivih na kašnjenje. To su, primjerice, višemedijske aplikacije poput prijenosa govora i videa.

UDP protokol također omogućava prijenos paketa između procesa koji se odvijaju na različitim računalima. Karakteristika UDP protokola je da on ni na koji način *ne garantira* dostavu paketa na odredište. Također, UDP ne garantira niti da će redoslijed kojim su paketi primljeni odgovarati redoslijedu kojim su poslani. Umjesto toga, UDP jednostavno šalje sve što se od njega zatraži, ne očekujući potvrde niti ispravljajući redoslijed isporuke paketa. S

obzirom na ova svojstva, UDP je pogodniji za aplikacije poput prijenosa višemedijskog sadržaja (gledanje filma, slušanje radija i slično), jer kod ovih aplikacija postoji izvjesna tolerancija na ispuštene pakete i pakete isporučene van slijeda.

U nastavku su, kroz niz primjera i uz pomoć Ethereal analizatora mrežnog prometa, opisana svojstva dvaju navedenih protokola. Ali prije toga, opisat ćemo naredbu `netcat` koju ćemo koristiti kao jednostavan generator TCP i UDP prometa.

3.1 Naredba `netcat`

Naredba `netcat` implementira funkcije klijenta i poslužitelja u Internetu te omogućava prijenos proizvoljnih podataka između ta dva entiteta. Detaljan opis korištenja naredbe `netcat` dobiva se pokretanjem `man nc`[1].

Naredba `netcat` može koristiti TCP ili UDP protokol za prijenos podataka između klijenta i poslužitelja.

Zadatak 8. Učitajte u simulator mrežu `ping.imn`. Iskoristite naredbu `netcat` kako bi uspostavili TCP vezu između dva proizvoljna računala u mreži[2].

Zadatak 9. Utvrdite što se događa nakon uspostavljanja veze između klijenta i poslužitelja[3].

Naredba `netcat` očekuje podatke na svom *standardnom ulazu*, i onda te podatke šalje po uspostavljenoj vezi. Nadalje, sve što naredba `netcat` primi putem uspostavljene veze, ispisuje na *standardni izlaz*. Standardni ulaz i izlaz se obično poistovjećuju s tipkovnicom i zaslonom, te se čitanje sa standardnog ulaza svodi na upisivanje podataka putem tipkovnice, dok se ispisivanje na standardni izlaz svodi na ispisivanje na zaslon. Međutim, korištenjem operatora "<" i ">" moguće je ulaz i izlaz preusmjeriti iz datoteke, odnosno u datoteku. Na primjer,

```
nc www.tel.fer.hr 80 < datoteka.txt
```

će se spojiti na računalo `www.tel.fer.hr` na TCP port 80 te sadržaj datoteke `datoteka.txt` poslati kao da je upisan s tipkovnice. Nadalje, izvođenjem

```
nc -l 4000 > datoteka.txt
```

sve što klijent pošalje, umjesto na zaslon, upisat će se u datoteku `datoteka.txt`.

Zadatak 10. Pokušajte pomoću naredbe `netcat` kopirati proizvoljnu datoteku s jednog računala na drugo.

---

[1] Naredba `netcat` uobičajeno se pokreće s `nc`.

[2] *Uputa:* na jednom računalu prvo ćete pokrenuti `netcat` u tzv. *listen* načinu rada, uz navođenje odgovarajućeg porta na kojem `nc` "sluša" zahtjeve za uspostavom veze; na drugom računalu pokrećete naredbu `netcat` i od nje zahtijevate uspostavu veze prema prvom računalu i portu na kojem `nc` sluša.

[3] *Uputa:* pokušajte nešto upisati putem tipkovnice i pritisnuti *"Return"*.

Zadatak 11. Utvrdite mogu li se na jednom računalu pokrenuti dva procesa koji slušaju na istom portu[4].

Zadatak 12. Ukoliko naredbi `netcat` ne specificirate protokol koji će koristiti, ona podrazumijeva protokol TCP. Ponovite pokus iz prethodnog zadatka uz korištenje UDP protokola[5].

3.2 UDP protokol

Još u prethodnom ciklusu vježbi utvrđeno je da protokol IP omogućava razmjenu paketa između računala u mreži. Ukoliko neko računalo A želi poslati podatke na drugo računalo B, formira se IP paket, kao izvorišna adresa paketa upisuje se adresa računala A, za odredišnu adresu stavlja se adresa računala B i paket se prosljeđuje prvom čvoru na putu prema B.

Međutim, s obzirom da se na jednom računalu može odvijati više međusobno neovisnih procesa (na primjer, različite korisničke aplikacije) koji imaju potrebu komunicirati s odgovarajućim procesima na drugom računalu, sama IP adresa u paketima koji pristižu na računalo nije dovoljna da bi se odredilo kojem procesu je potrebno dostaviti sadržaj paketa. IP adresa je paket "dovela" do računala, ali unutar računala ima više potencijalnih primatelja, odnosno, procesa.

Stoga je, osim IP adrese, u komunikaciju između procesa potrebno uvesti dodatnu oznaku koja određuje proces unutar računala kojem je potrebno dostavljati sadržaj IP paketa. Ova oznaka naziva se **vrata** (*port*).

Zadatak 13. Prikupite i analizirajte promet koji `nc` generira prilikom prijenosa podataka s jednog računala na drugo, u slučaju kad se koristi UDP protokol. Što možete zaključiti o portovima i adresama; što se mijenja, a što ostaje isto?

3.3 TCP protokol

Za razliku od UDP protokola, TCP pruža bitno složeniju uslugu. Prije svega, TCP osigurava pouzdan prijenos niza okteta čuvajući njihov redoslijed. Pojam *pouzdan prijenos* znači da će, u slučaju gubitka paketa u mreži, TCP ponavljati slanje izgubljenih paketa sve dok se svi ne prenesu na odredište. Pojam *prijenos niza okteta* odnosi se na činjenicu da TCP procesima isporučuje jedan po jedan oktet, ne pružajući informaciju o načinu na koji su skupine okteta prilikom prijenosa kroz mrežu grupirane u IP pakete. Nadalje, TCP osigurava isporuku okteta u istom redoslijedu kojim su i poslani. Bez obzira na činjenicu da se u IP mrežama relativno često događa da paketi na odredište stižu različitim redoslijedom od onoga u kojem su poslani, TCP "ispravlja" redoslijed IP paketa i procesu isporučuje "izvorni" slijed okteta.

Da bi osigurao pouzdan prijenos okteta, TCP koristi mehanizam potvrđivanja. Ispravan primitak svakog poslanog okteta mora biti potvrđen od strane primatelja. Ukoliko pošiljatelj ne dobije potvrdu o primitku okteta koje je već poslao, ponavlja se slanje istih okteta sve dok se ne dobije potvrda da ih je odredište primilo.

---

[4] *Uputa:* pokušajte dvaput pokrenuti `nc` (istovremeno), iz dvije konzole istog računala.

[5] *Uputa:* proučite `man nc` kako bi ustanovili na koji se način koristi UDP umjesto TCP-a.

Prije slanja okteta između udaljenih procesa, TCP mora uspostaviti vezu. Da bi razlikovao procese kojima mora dostavljati pakete, koriste se portovi. Kao i kod UDP-a, TCP veza je jednoznačno određena četvorkom {*izvorišna IP adresa, izvorišna vrata, odredišna IP adresa, odredišna vrata*}.

Zadatak 14. Pomoću analizatora Ethereal uhvatite proizvoljan TCP promet koji pripada jednoj vezi te odredite datagrame koji se razmjenjuju u fazama uspostave veze i raskida veze (uputa: za generiranje TCP prometa iskoristite `netcat`).

Zadatak 15. Za uhvaćeni promet odredite koje se adrese i vrata koriste. Imaju li svi segmenti istu četvorku {*izvorišna IP adresa, izvorišna vrata, odredišna IP adresa, odredišna vrata*}?

Da bi osigurao pouzdan prijenos okteta između procesa, protokol TCP koristi mehanizam potvrđivanja. Za svaki poslani oktet primatelj mora potvrditi da ga je primio. U TCP zaglavlju za to se koristi posebno polje ACK (detaljnije u udžbeniku [1], str. 251).

Zadatak 16. U prethodno uhvaćenom prometu, utvrdite na koji se način koriste potvrde.

Jedna od bitnih funkcija koju obavlja TCP je i tzv. *kontrola toka*. Radi se o sljedećem. Pretpostavimo da jedan proces šalje određenu količinu podataka nekom drugom procesu, te da se proces koji šalje podatke (pošiljatelj) nalazi na računalu koje je u mrežu spojeno vezom kapaciteta 100 Mbit/s, dok se proces koji prima podatke (primatelj) nalazi na računalu koje je spojeno vezom kapaciteta 10 Mbit/s. S obzirom da je pošiljatelj spojen vezom od 100 Mbit/s, on bi mogao slati podatke tom brzinom, ali ti podaci ne bi mogli biti isporučeni primatelju (istom brzinom), jer je pristup do primatelja brzine 10 Mbit/s. Kako pošiljatelj zna kojom brzinom smije slati podatke, a da ih primatelj stigne obraditi? Kontrola toka koju obavlja TCP upravo rješava ovaj problem.

Protokol TCP koristi mehanizam tzv. klizećeg prozora (*sliding window*) za kontrolu toka. Primatelj je zadužen za kontinuirano obavještavanje pošiljatelja o količini podataka koju je on trenutno u stanju obraditi. Pošiljatelj nikad neće poslati više podataka od ove količine, bez da mu primatelj potvrdi primitak podataka. Na primjer, pri uspostavi TCP veze obje strane objavljuju jedna drugoj koliko su podataka spremne primiti u danom trenutku. Ova veličina naziva se *prozor*. Svaka strana smije odjednom poslati samo toliko podataka koliko je druga strana objavila u prozoru. Nakon što je poslala tu količinu podataka, strana pošiljatelja mora čekati potvrdu da je barem jedan dio podataka primljen na odredištu. Kad dobije potvrdu, pošiljatelj može poslati dodatnu količinu podataka, ali samo onoliko novih koliko je okteta potvrđeno. Na taj se način u mreži, u svakom trenutku, nalazi najviše onoliko nepotvrđenih podataka kolika je veličina prozora. S obzirom da pošiljatelj ne smije slati nove podatke dok primatelj ne potvrdi primitak starih, primatelj diktira brzinu kojom mu (brži) pošiljatelj šalje podatke.

Zadatak 17. Uhvatite proizvoljan TCP promet (od jedne veze) i utvrdite veličine prozora. Mijenja li se veličina prozora često u toku trajanja TCP veze?

Osim pouzdanog i slijednog prijenosa, TCP protokol vodi računa i o kontroli zagušenja u mreži izazvanog TCP prometom te o ravnopravnosti podjele mrežnih kapaciteta između konkurentnih TCP veza. Tematika kontrole zagušenja je vrlo složena, te neće biti razmatrana u okviru ovih vježbi.

Zadatak 18. Primijetite da, iako su bitno različiti po svojstvima, UDP i TCP po funkcionalnosti spadaju u transportni sloj referentnog modela OSI. Objasnite zašto.

Zadatak 19. Pokušajte prouzročiti pojavu gubitka TCP segmenata. Na koji način možete utvrditi da je došlo do gubitaka? Možete li izazvati gubitke bez podešavanja simulatora na način da namjerno ispušta pakete?

Zadatak 20. Pokušajte identificirati promet koji pripada jednoj TCP vezi za vrijeme u kojem dolazi do gubitka paketa. Utvrdite što se tada događa s potvrdama i veličinom prozora.

Zadatak 21. Proučite mogućnosti koje Ethereal pruža u svrhu analize TCP prometa.

Zadatak 22. Utvrdite kako propusnost TCP prijenosa ovisi o kašnjenju s kraja na kraj. Koliku bi najveću propusnost mogao ostvariti TCP na linku kapaciteta 1 Gbit/s, na kojem nema gubitaka, a kojem propagacijsko kašnjenje iznosi 10 sekundi?

Reference

[1] Bažant et al., Osnovne arhitekture mreža, Element, 2003.