Problem performansi računalne mreže Bufferbloat

Terihaj, Nikola

Sveučilište u Rijeci – Odjel za informatiku

Jednopredmetni preddiplomski studij informatike

Nikola Terihaj

Problem performansi računalne mreže Bufferbloat

Završni rad

Mentor: v. pred. dr. sc. Vedran Miletić

Rijeka, 22. rujna 2020.

Sažetak

Tema ovoga rada je problem bufferbloata koji se javlja na našoj internet vezi i metode smanjenja njegovog utjecaja, odnosno mogućnosti rješavanja tog problema. Bufferbloat je problem dugotrajnog vremena čekanja. Slanjem paketa od pošiljatelja prema primatelju zbog visokog vremena čekanja može se dogoditi da se paketi zagube. IP i TCP su najvažniji protokoli na našoj internet vezi, gdje protokol IP osigurava povezanost različitih mreža, a protokol TCP pouzdanu vezu s kraja na kraj. ECN je proširenje na protokol IP i protokol TCP te je njegova glavna uloga slanje obavijesti o zagušenju na mreži. Osim algoritma ECN, vrlo važni algoritmi su također AQM, RED i CoDel. Razvojem algoritma FQ_CoDel problem bufferbloata sveden je na minimum.

Ključne riječi

Bufferbloat, Latency, ECN, RED, CoDel, FQ_CoDel, Ethernet, Wi-Fi, router

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 24.02.2020.

Zadatak za završni rad

Pristupnik: Nikola Terihaj

Naziv završnog rada: Problem performansi računalne mreže Bufferbloat

Naziv završnog rada na eng. jeziku: Bufferbloat computer network performance problem

Sadržaj zadatka:

Porastom veličine međuspremnika usmjerivača na internetu raste prosječno vrijeme koje paketi čekaju u tim međuspremnicima na putu od izvora do odredišta. Posljednjih 10-ak godina taj se fenomen izučava pod nazivom Bufferbloat pa su vremenom razvijeni alati koji ga mogu mjeriti kao što je Flent i algoritam za aktivno upravljanje redovima čekanja paketa CoDel kojim se Bufferbloat želi izbjeći. Cilj rada je opisati kako dolazi do Bufferbloata, izmjeriti Bufferbloat u konkretnom slučaju po izboru studenta te analizirati način rada algoritma CoDel koji služi za njegovo izbjegavanje i doprinose projekta Make Wi-Fi Fast koji iskustvo stečeno kod rješavanja Bufferbloata prenosi na Wi-Fi.

Mentor

v. pred dr. sc. Vedran Miletić

_____

Voditelj za završne radove

doc. dr. sc. Miran Pobar

_____

Zadatak preuzet: 26.02.2020.

_____

(potpis pristupnika)

Sadržaj

1. Uvod

Ovaj rad opisuje fenomen bufferbloata.

Glavni problem koji razmatram je zapravo vrijeme čekanja te ću o njemu nešto više reći u sljedećem poglavlju. Osim toga govoriti ću i o dva vrlo važna protokola, IP i TCP kako oni funkcioniraju te njihove kvalitete. Svaki router ima svoj međuspremnik koji ima ograničenu memoriju. Slanjem velikog broja paketa može doći do zagušenja, što je rješivo TCP Tuning opcijom o čemu nešto više u TCP Window Scale poglavlju.

Ovaj rad također se bavi algoritmima poput ECN, RED, CoDel, FQ_CoDel i SQM bez kojih naša veza s internetom ne bi funkcionirala, odnosno uz pomoć AQM-a koji se nalazi u našim routerima, a koriste se navedeni algoritmi za što bolji rad naše internetske veze.

Govoriti ću o Wi-Fi vezama i kako se iz godinu u godinu razvija njena tehnologija pa dobiva sve bolje i bolje performanse. Poboljšanje Wi-Fi konekcije, njena stabilnost i rad na što većem dometu je od velike tehnološke važnosti.

Naposlijetku ću izvršiti mjerenja potkrijepljena grafovima koja će nam bolje predočiti problem bufferbloata.

2. Problem i motivacija

Pitamo se zašto je brzina internet veze toliko dugo od brzine svjetlosti [1]. Sama definicija bufferbloata nam govori kako je to zapravo problem visokog vremena čekanja, odnosno vremena čekanja da paket stigne od našeg računala do drugog i natrag [2]. Vrijeme čekanja je inače oko 50 ms, no recimo kod igranja internet video igara za stabilno igranje potrebno nam je vrijeme čekanja od kao što sam prije naveo, dakle 50 ms. Naravno to nije uvijek slučaj jer imamo mnogo faktora koji utječu na to pa kada vrijeme čekanja prijeđe preko 80 ms, to nas dovodi do neigrivosti. TCP Handshake, DNS Resolution, TCP Data Transfer i RTT su faktori koji utječu na nestabilnost vremena čekanja. Za sve njihove aktivnosti potrebno je previše vremena, kod npr. TCP Handshake-a potrebno mu je vrijeme za SYN-ACK, dakle vrijeme za poslati zahtjev prema pošiljatelju te čekanje njegovog odgovora (eng. *ACK*). Vrijeme čekanja se poveća kada dođe do zagušenja na mreži zbog previše slanja paketa nego što sam router može primiti što dovodi do pada kvalitete internet usluge gube se paketi ili dolazi do blokiranja novih konekcija. Svaki router ima svoj međuspremnik (eng. *buffer*) te može pohraniti određenu količinu podataka prije nego što ih proslijedi dalje.

Protokoli koji šalju pakete su organizirani od strane mrežnog planera (eng. *network scheduler*). On je od velike važnosti zato jer odlučuje u kojem redoslijedu će se paketi slati te za to koristiti algoritme poput AQM, ECN i sl. koji služe za kontrolu paketa te im je glavna uloga izbjeći zagušenje na mreži i povećanje vremena čekanja.

2.1. Protokoli IP/TCP

IP je mrežni protokol koji je zadužen za dostavljanje paketa od pošiljatelja prema primatelju. IP adresa ima tzv. IP header koji sadrži informacije poput odredišne IP adrese i sl. Protokol IP osigurava nepouzdanu mrežu jer se paketi šalju modelom koji se zove najbolji mogući (eng. *best-effort*) [3]. Taj model jako ovisi o samoj kvaliteti internet veze. Paketi se prilikom prijenosa mogu duplicirati, promijeniti redoslijed i izgubiti ukoliko dođe do zagušenja na mreži.

Ipak kako bi znali da neki paket nije uspješno stigao do odredišta imamo kontrolere koji pošalju poruke, ukoliko veza nije uspostavljena. Ipv4 sadrži protokol ICMP (eng. *Internet Control Message Protocol*) koji pošalje poruku o neuspjeloj konekciji [4].

Za razliku od protokola IP, imamo još protokol TCP koji osigurava siguran prijenos paketa, bez

gubljenja paketa i sl. E-mail, WWW jako ovise o TCP-u, jer TCP najprije uspostavi vezu između klijenta i servera pa kada je veza uspostavljena paketi se krenu slati. Kao što sam ranije spomenuo do zagušenja svejedno dolazi, jer TCP-ovo rukovanje (eng. *TCP Handshake*) puno vremena potroši na uspostavu konekcije.

2.2. TCP Window Scale

Propusnost u komunikaciji je limitirana s dva prozora, prozor zagušenja i prozor prijema [5].

Prozor zagušenja pokušava ne prekoračiti kapacitete interneta koje dozvoljava router [5], jer znamo još iz ranijih poglavlja kako svaki router ima određenu veličinu međuspremnika. Primatelj obavještava pošiljatelja o svojoj veličini prozora te pošiljatelj na osnovu te informacije šalje podatke. Kada je prozor primatelja pun, dok ne pošalje ACK pošiljatelju ostali paketi se ne šalju.

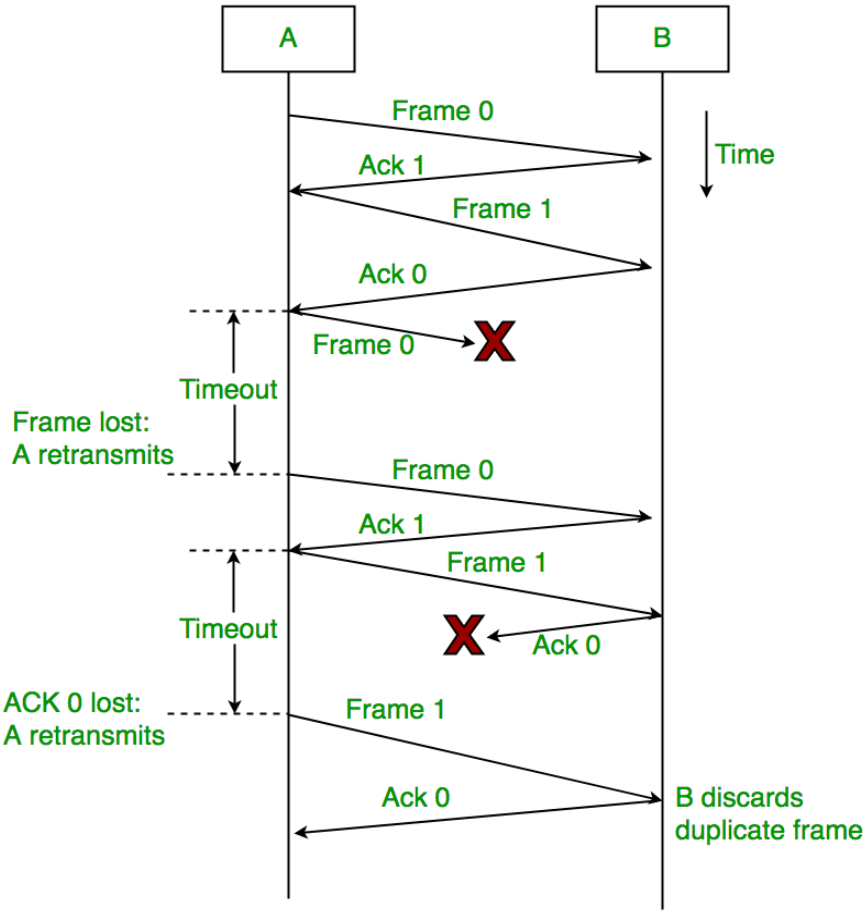
Ukoliko dođe do prekoračenja međuspremnika kod primatelja, imamo TCP opciju povećanja memorije prozora prijema. Memorija kod prozora prijema je 65.535 bita [5]. Povećanje memorije prozora još nazivamo TCP Tuning. Formula za računanje propusnosti ograničena veličinom prozora je oblika:

$$Throughput \leq \frac{RWIN}{RTT}$$

Pomoću ove formule možemo izračunati propusnost (eng. *throughput*) limitirana veličinom prozora. Propusnost se limitira, jer se može dogoditi da se paket nije izgubio, a TCP šalje svoje pakete dok ne popuni veličinu prozora primatelja sve dok ne primi ACK. RWIN nam označuje veličinu prozora primatelja te tu veličinu dijelimo s RTT-om. RTT je vrijeme koje se potroši na signal koji se pošalje plus još vrijeme koje je potrebno da se signal prijama primi.

Memorija prozora (eng. *window size*) nam služi da znamo koliko podataka možemo primiti od pošiljatelja bez njegove potvrdne poruke tzv. ACK. Npr. Ako pošiljatelj primi ACK veličine 4000 bita, tada je veličina prozora prijema postavljena na 10000 bita, pošiljatelj neće više slati paket veće od 14000 bita čak i ako ga prozor zagušenja može primiti [5].

Ukoliko se dogodi da pošiljatelj ne primi natrag potvrdnu poruku od primatelja, paket se onda pokušava ponovo poslati, jer se smatra da se paket izgubio ili je oštećen. Ponovno slanje paketa se još zove (eng. *retransmission*). Ova pojava se još zove ponašanje *stop-and-go*. Pošiljatelj uvijek u slučaju da nije primio ACK zadrži pakete u memoriji te ih onda može ponovo poslati.



Slika 1: Ponašanje stop-and-go

Ovime smo dokazali koliko je zapravo protokol TCP bolji od protokol IP, najviše iz razloga što imamo ACK poruke i ponovno slanje paketa.

2.3. Explicit Congestion Notification

Eksplicitna obavijest o zagušenju (eng. *Explicit Congestion Notification*, kraće ECN) je algoritam koji proširuje protokole IP i TCP [6]. ECN šalje obavijest o zagušenju mreže bez gubljenja paketa. Ukoliko je ECN uspješno uspostavljen, on postavlja oznaku u zaglavlju IP adrese kako ne bi izgubio pakete. Primatelj tako ima informaciju o zagušenju. Reducira se brzina prijenosa paketa kako ne bi došlo do zagušenja na mreži te gubljenja paketa.

ECN zahtjeva potporu internet sloja i transportnog sloja [6]. Najpoznatiji transportni sloj je TCP,

dok kod internet sloja imamo protokole IP, ICMP, ECN i sl.

Vrlo je važno za eksplicitnu obavijest o zagušenju da ima potporu oba sloja, jer routeru je potreban mrežni sloj, dok brzina prijenosa ovisi o krajnjim točkama transportnog sloja. Ukoliko nemamo eksplicitnu obavijest o zagušenju kod protokola IP/TCP, obavijest o zagušenju se šalje indirektno, prilikom gubitka paketa [6]. Kada je kao što sam prije naveo, eksplicitna obavijest o zagušenju uspješno uspostavljen, odnosno postavljen u zaglavlju IP adrese, IP postavlja u zaglavlje CE bit, a kod protokola TCP ukoliko je došlo do zagušenja u zaglavlje se stavlja ECE bit. Oba bita nam ukazuju na zagušenje na mreži.

Svaki router u sebi ima algoritam za aktivno upravljano redom (eng. A*ctive Queue Management*, kraće AQM) koji je zadužen za ispuštanje paketa unutar međuspremnika kako se međuspremnik ne bi popunio te time smanjilo vrijeme čekanja i izbjegnulo zagušenje [7]. Router je spojen s računalom pomoću Ethernet kabela ili Wi-Fi-ja. Računala imaju mrežne kartice koje se ugrađuju naknadno na matičnu ploču te se one mogu koristi za Wi-Fi ili Ethernet vezu. Osim toga sama matična ploča ima Ethernet vrata za spajanje Ethernet kabela. Pomoću svega ovoga AQM funkcionira, zatim upravljanje mrežnog sučelja (eng. *Network Interface Controller*, kraće NIC) ili mrežna kartica je važna stavka za izvršavanje njegove uloge.

AQM-om najveći problem je što ponekad koristi algoritme poput nasumične rane detekcije (eng. *Radnom Early Detection*, kraće RED), kontroliranog odstupanja (eng. *Controlled Delay*, kraće CoDel) i sl. umjesto da koriste eksplicitnu obavijest o zagušenju. Zašto je to problem? Najviše iz razloga, što recimo ukoliko je međuspremnik prazan, može se dogoditi da se ispuste paketi.

Kod algoritma RED ispuštanjem paketa može doći do TCP globalne sinkronizacije, što znači da ukoliko dođe do popunjena međuspremnika, pošiljatelj smanji brzinu prijenosa tada paketi koji se još nisu poslali su u stanju čekanja pa kada se međuspremnik kod primatelja oslobodi paketi se nastave slati. Ovo je vrlo dobra stvar jer se paketi nisu izgubili.

CoDel je algoritam koji je u prosjeku bolji od algoritma RED, jer postavlja limit čekanja za pakete koji se nisu poslali zbog punog međuspremnika. Vrlo je važno kakva je kvaliteta same konekcije, jer ukoliko ne dođe do zagušenja na mreži, vrijeme čekanja će bit kratko, no ukoliko je obrnuta siutacija, vrijeme čekanja će biti jako veliko jer međuspremnik ostaje pun.

Kao što sam i prije spomenuo, ako je ECN uspješno uspostavljen između primatelja i pošiljatelja, protokol IP za svoje pakete u zaglavlju adrese postavlja nulu ili jedincu na kraju bita, npr. 00 ili 01, što nam ukazuje na uspješan prijenos te pri zagušenju, AQM pomoću ECN algoritma obaviještava o zagušenju te u zaglavlju IP adrese imamo bit s oznakom CE, kako ne bi došlo do ispuštanja paketa,

nego sam primatelj smanji brzinu prijenosa.

TCP zaglavlje je puno drugačije od IP zaglavlja, jer imamo dva slučaja. Prvo je naznaka za zagušenje i drugo je reducirani prozor zagušenja (eng. *Congestion Window Reduced*, kraće CWR) te on sluzi za primanje obavijesti o zagušenju.

ECN je opcionalan kod TCP-a jer ovisi o uspješnosti uspostavljene konekcije na SYN-ACK segmentima. IP paketi koji sadrže u sebi TCP segmente, dakle oni paketi koji u IP zaglavlju imaju CE bit, TCP ih označuje s ECE bitom, što nam ukazuje da paket nije izgubljen.

Znamo da ECN šalje obavijesti o zagušenju na mreži, što je velika prednost. Ukoliko se pokaže da imamo ne usklađene TCP konekcije koje misle da su ECN kompatibline, odnosno da je ECN uspješno uspostavljen na konekciji, može se dogoditi da se ignoriraju ECN obavijesti te dolazi do ispuštanja paketa. Teško je garantirati da će TCP paketi s ECN bitovima u zaglavlju IP adrese paketa primiti obavijest o zagušenju.

Današnji windows serveri imaju odmah u sebi integriran ECN. Isto tako iOS i Linux operacijski sustavi.

2.3.1. Algoritam RED

Nasumična rana detekcija (eng. *Random Early Detection*, kraće RED) je mehanizam koji koristi svoje algoritme kako bi nasumično odredio koju konekciju obavijestiti o zagušenju [7].

Kada dođe do zagušenja na mreži, RED obavijesti vezu o zagušenju te ispušta pakete ili ih stavlja u stanje čekanja dok se međuspremnik primatelja ne oslobodi.

Kada znamo četiri vrijednosti RED parametara, možemo izračunati koliko paketa će se ispustiti. Parametri su **min** i **max treshold**, **avg** nam označuje vrijednost reda (eng. *queue*) i parametar **p** koji označuje vjerojatnost ispuštenog paketa. Npr. ako se dogodi da je **avg < min treshold** tada se paketi neće ispustiti, ali zato ako je **avg > max treshold** tada će se ispustiti svi paketi koji stižu prema primatelju. Ukoliko su **min treshold <= avg <= max treshold** tada će paketi biti ispušteni s velikom vjerojatnošću **p**.

P vjerojatnost možemo izračunati pomoću tri formule:

Formula 1: **Pb=maxp(avg - min treshold) / (max treshold – min treshold)**

U prvoj formuli, vjerojatnost ispuštenih paketa određujemo prema zadanim parametrima koje sam ranije opisao te još u ovom slučaju moramo znati vrijednost parametra **maxp** koji označuje gornju granicu vjerojatnosti ispuštenih paketa.

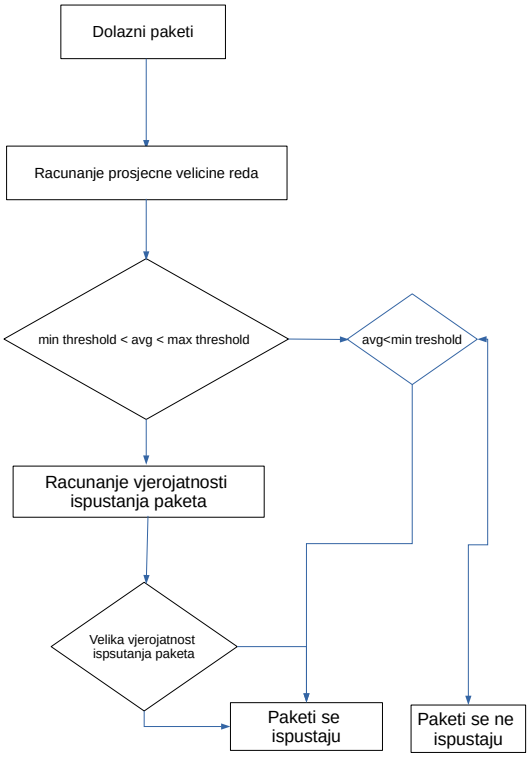
Formula 2: **Pa= Pb / (1-count * Pb)**

Druga formula nam omogućuje računanje vjerojatnosti ispuštenih paketa nakon posljednjeg ispuštanja paketa (**Pb**) te se **count** povećava nakon svakog sljedećeg ispuštanja paketa.

Formula 3: **Pb=Pb * PacketSize / MaximumPacketSize**

Treća formula nam omogućuje izračunavanje vjerojatnosti ispuštenih paketa, kada znamo vrijednost pristiglih paketa u bitovima (**PacketSize**) te najviše dopuštene veličine paketa u bitovima (**MaximumPacketSize**).

Algoritam RED zapravo koristi dvije vrste algoritama koje rade na kontroli prosječne veličine reda. Prva vrsta algoritma je važna za izračunavanje prosječne veličine reda (eng. *average queue size*). Prosječna veličina reda se zapravo računa kada je red prazan, zapravo računa se koliko paketa jako male veličine se može prosljediti dok je red prazan (eng. *idle*). Sljedeća vrsta je algoritam koji se koristiti kako bi se izbjegla TCP globalna sinkronizacija, tako da se označi paket kada je **avg** u središnjoj točki (eng. *midway point*) između **min** i **max treshold.** Kada se dogodi da je **avg** veci ili jednak **max treshold** tada je vjerojatnost ispuštenih paketa veća od paketa koji su ispušteni, konekcija se o tome obavijesti i brzina prijenosa paketa je manja. No ukoliko je obrnuta situacija, dakle vjerojatnost ispuštenih paketa manja, onda se paketi ne ispuštaju. Kompletna situacija je vidljiva na dijagramu ispod.



Oblik 1: RED diagram

Algoritam RED:

for each packet arrival
    calculate the new average queue size avg
     if **min threshold** <= **avg** < **max threshold**
            calculate probability **Pa**
             with probability Pa:
                    mark/drop the arriving packet
else if **max threshold** < **avg**
       drop the arriving packet

3. Rješenje problema

3.1. CoDel/FQ_CoDel

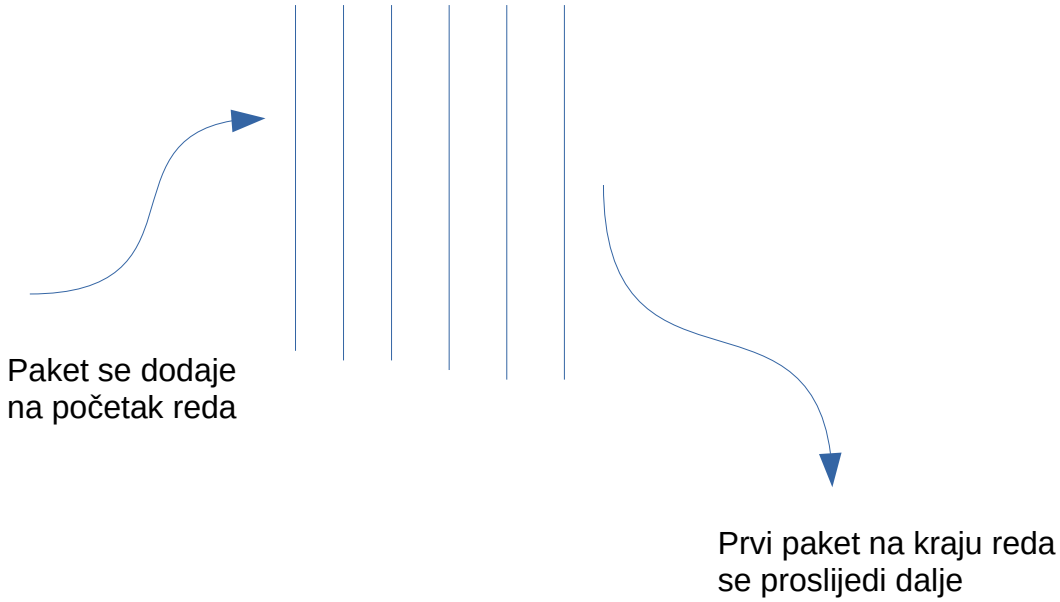
Znamo još iz prijašnjih poglavlja da ECN obavještava primatelja o zagušenju, dok algoritam RED smanji brzinu prijenosa ako se popuni međuspremnik kod primatelja kako ne bi došlo do ispuštanja paketa. Sada ću govoriti o algoritmu kontroliranog odstupanja (eng. *Controlled Delay,* kraće CoDel*)* te kasnije o pravednom redanju kontroliranog odstupanja (eng. *Fair Queuing Controlled Delay,* kraće FQ_CoDel). CoDel je algoritam koji kontrolira odstupanje odnosno (eng. *delay*) te je on kao što sam ranije spomenuo bolji od algoritma RED.

Glavni cilj CoDel-a je smanjiti duljinu reda, tako da paketi koji su u redu predugo, odnosno nisu proslijeđeni dalje se odbace. Dakle, ukoliko je paketu potrebno više od 5 ms da se pošalje dalje paket se odbaci. Svi paketi koji imaju vrijeme čekanja manje od 5 ms se proslijede dalje. Kako bi kontrola paketa bila bolja koristi se FQ_CoDel.

FQ_CoDel je disciplina koja kombinira pravedo redanje (eng. *Fair Queuing,* kraće FQ) s CoDel-om [8].

Pravedno redanje (eng. *Fair Queing*, kraće FQ) je algoritam za kontrolu toka podataka na internetu, odnosno algoritam dizajniran kako bi se postigla pravednost (eng. *fairness*) kod raspodjele resursa sistema koji su potrebni za pokretanje neke aplikacije. Npr. prilikom slanja paketa, može se desiti da se šalju paketi velikih veličina ili se procesiraju mali poslovi kojima je potrebno dosta vremena od strane središnje jedinice(eng. *Central Process Unit*, *kraće CPU*) vremena za razliku od nekih drugih procesa ili je potrebna veća propusnost kod toka podataka [9].

Pravedno redanje kontroliranog odstupanja (eng. *Fair Queuing Controlled Delay, kraće FQ_CoDel*) koristi stohastični model za kontrolu pristiglih paketa. Odnosno, cilj je pravilno rasporediti pakete u vise različitih tokova paketa, kako bi se mogao pravilno rasporediti mrežnu propusnost (eng. bandwith) za sve tokove paketa. Paketi se slažu u red gdje sada FQ_CoDel koristi algoritam CoDel koji sadrži metodu FIFO kontrole paketa [8].



Slika 2: Metoda FIFO

Na slici 3 vidimo kako zapravo funkcionira metoda FIFO. FIFO je akronim za prvi unutra (eng. *first in*), prvi van (eng. *first out*). Dakle, paket koji stigne se dodava na početak reda, a zatim paket koji je prvi na kraju reda se proslijedi dalje.

Važno je napomenuti, da u svemu ovome jest kvaliteta samog interneta. Jer ukoliko dođe do zagušenja, red se napuni i potrebno mu je puno vremena da proslijedi dalje paket što dovodi do ispuštanja paketa.

FQ_CoDel se u praksi pokazao kao lijek za bufferbloat, što još uvijek nije te nam za sada svi ovi algoritmi poput, ECN-a, CoDel-a, RED-a pomažu u minimalnoj funkcionalnosti interneta, odnosno poboljšanje interneta.

4. Wi-Fi

Wi-Fi je tehnologija bezičnog umrežavanja. Bazira se na IEEE 802.11 standardima [10].

IEEE 802.11 standard sadrži LAN (eng. *Local Area Network)*, MAC (eng. *Media Access Control*) i PHY (eng. *Physical Layer*) protokole koji služe za implementaciju WLAN (eng. *Wireless Local Area Network)*.

Bežični (eng. *wireless*) prijemnici pomoću ranije navednog IEEE 802.11 standarda, omogućavaju povezivanje uređaja pomoću Etherneta ili Wi-Fi vezom.

Wi-Fi uglavnom koristi 2.4 Ghz UHF i 5 Ghz SHF ISM radio valove. Najveća kvaliteta Wi-Fi usluge je najbliža točka (eng.*hotspot*), odnosno što smo bliže internet routeru to nam je usluga bolja. Brzina interneta tada može doći do 1 Gbit/s. Do problema dolazi kada se udaljavamo od najbliže točke pa zidovi soba rade smetnje radio valovima te nam je internet usluga slabija. Potrebno je puno više vremena za učitati neku stranicu, video i sl.

Današnji internet prijemnici imaju uglavnom raspon Wi-Fi mreže od 20m, dok puno moderniji, odnosno s boljom tehnologijom od komercijalnih kućnih routera imaju domet od čak 150m, ali naravno govorimo kada nema ograničenja poput ranije spomenutih kućnih zidova.

Sve je počelo 1971. godine kada su na Sveučilištu na Havajima stvorili ALOHAnet ili ALOHA sustav. ALOHA sustav je koristio UHF radio frekvencije za bežično umrežavanje.

ALOHAnet i protokol ALOHA su preteča današnjeg Ethernet i protokola IEEE 802.11 [10].

Prvi protokol 802.11 izašao je 1997. godine i tada je imao brzinu od samo 2 Mbit/s, dok novija verzija iz 1999. je imala brzinu od 11 Mbit/s.

Danas kao što sam i ranije spomenuo brzine su znatno veće, npr. Wi-Fi 802.11ac, njegova brzina je u rasponu od 600 do 9608 Mbit/s te vidimo koliko se tehnologija čipova i svega ostalog razvija iz godine u godinu.

U ranim 2000-ima brojni gradovi diljem svijeta imaju u planu razviti mrežu kako bi se cijeli grad mogao spojiti na Wi-Fi mrežu pomoću svojih uređaja, ako su građani recimo izvan svog doma i sl. Grad Mysere u Indiji je 2004.godine postao prvi grad sa Wi-Fi mrežom, nakon njega brojni drugi gradovi diljem svijeta počinju imati Wi-Fi mrežu.

Wi-Fi za razliku od Ethrneta je puno nesigurniji, zato što svatko tko zna naziv interneta(SSID) i lozinku može pristupiti vašem internetu. Dakako, Wi-Fi mreža ima Wi-Fi zaštićen pristup (eng.

Protected Access, kraće WPA), algoritam koji odvaja internet od osobnog (eng. *personal*) i od interneta za poduzeća (eng. *enterprise*).

Slika 3: QR Code

Svi današnji pametni uređaji imaju mogućnost očitavanja QR koda, čime je pojednostavljeno spajanje na Wi-Fi mrežu tako da se samo očita kod sa slike.

Wi-Fi brzine su varijabilne, dakle što je više uređaja spojeno to nam je brzina manja te usluga lošija, odnosno da budem precizniji dolazi nam do zagušenja na mreži, javlja nam se tzv. bufferbloat.

O problemu bufferbloata sam već govorio te ću u sljedećem poglavlju, opisati da se bufferbloat osim Ethernet veze javlja i kod Wi-Fi konekcije te kako ga reducirati kako bi internet usluga bila bolja, odnosno internet veza brža.

4.1. Kako da Wi-Fi bude brži?

Wi-Fi veza kao i Ethernet veza se dakle, susreće sa istim problemom bufferbloata. Znamo da prijemnici imaju određenu veličinu međuspremnika te kada se međuspremnik popuni, dolazi nam do povećanja vremena čekanja, odnosno javlja nam se zagušenje na mreži [11].

Wi-Fi kada ima previše spojenih uređaja može se dogoditi problem bufferbloata.

Nažalost provedena su razna testiranja kako smanjiti bufferbloat i povećati kvalitetu Wi-Fi mreže. Najveći problem je što Wi-Fi mreža koristi radio valove, zato što radio valovi imaju smetnje poput kućnih zidova, drugih uređaja i sl.

Kućni prijemnici, dakle komercijalni prijemnici su najlakše pogođeni ovim problemom, jer imamo uređaje spojene Ethernet kabelom i mnogo drugih uređaja Wi-Fi-jem. Kao sto sam i ranije naveo, veliki broj uređaja na jednoj internet mreži spojeni Wi-Fi-jem dovodi do bufferbloata, jer imamo vise namjenski promet (eng. *multicast traffic*) te sve ovisi o jednoj internet mreži.

Ovakav problem je rješiv, tako da se višenamjenski promet rastereti, pomoću mnogo manjih stanica,

tzv. Wi-Fi pojačivača. Wi-Fi pojačivači povećavaju domet glavnog routera te najvažnije od svega brzina same internet veze je onda znatno veća, što dovodi do kvalitetnije internet usluge.

Znamo da Wi-Fi mreža u gradovima uglavnom slaba te u većini slučajeva se ne koristi zbog loše kvalitete internet usluge. Rješenje je moguće ako se smanji korištenje radio frekvencije od 2.4 GHz, a poboljšati radio odašiljač koji bi koristili Wi-Fi 802.11ac standard čija je brzina veze od 433 do 6933 Mbit/s [11].

5. Routeri

5.1. SQM

U ranijim poglavljima govorio sam o algoritmima za suzbijanje bufferbloata, poput ECN-a, CoDel-a, RED-a i sl. To su dakle, algoritmi koje kontrolira AQM te svaki router pomoću AQM koristi algoritme kako bi se koliko-toliko izbjegao bufferbloat.

No, baveći se dugo godina problemom bufferbloata današnji moderni routeri počeli su implementirati pametno menadžiranje reda (eng. *Smart Queue Management,* krace SQM) algoritme, poput ranije opisanog FQ_CoDel-a koji se dakle, pokazao najefikasniji za smanjenje bufferbloata.

SQM prednost naspram AQM-a je dakle, pokušati smanjiti odgodu kod slanja paketa prilikom velikog broja slanja paketa te smanjiti gubljenje paketa [12]. Kao kada recimo igramo neke internet igre ili gledamo filmove na „*Netflixu*" i sl. Dok recimo AQM algoritmi otpuste pakete kako se ne bi međuspremnik zakrcao i doveo do zagušenja. SQM je dakle, samo nadogradnja na AQM.

Ubiquiti ER-X i ER-4 su prijemnici koji imaju implementirane SQM algoritme te time daju znatno bolje performanse, odnosno internet usluge za razliku od običnih, komercijalnih routera neovisno dali je Wi-Fi ili Ethernet veza [13].

U sljedećim potpoglavljima govoriti ću što još možemo napraviti da smanjimo problem bufferbloata.

5.2. Ubiquiti ER-X i ER-4

Router Ubiquiti ER-4 je dovoljno snažan da podrži širokopojasnu vezu (eng. *broadband connections*) poput DSL-a te ne ovisi o mogućnostima središnje jedinice ili memorije na routeru [13].

ER-4 je dakle, router koji koriti algoritam FQ-CoDel te je vrlo popularan kod igrača koji igraju pute interneta zato što podržava ultra brze širokopojasne veze veće od 100 Mbps.

ER-X za razliku od ER-4 je znatno jeftiniji te podržava širokopojasnu vezu do 100 Mbps, no svejedno vrlo je dobar za internet igrače i sl.

Slika 4: Ubiquti EdgeRouter

Oba routera su vrlo jednostavna za postaviti te imaju jednostavne SQM postavke, gdje se mogu još poboljšati njegove performanse.

5.3. Ugradbeni softver OpenWrt

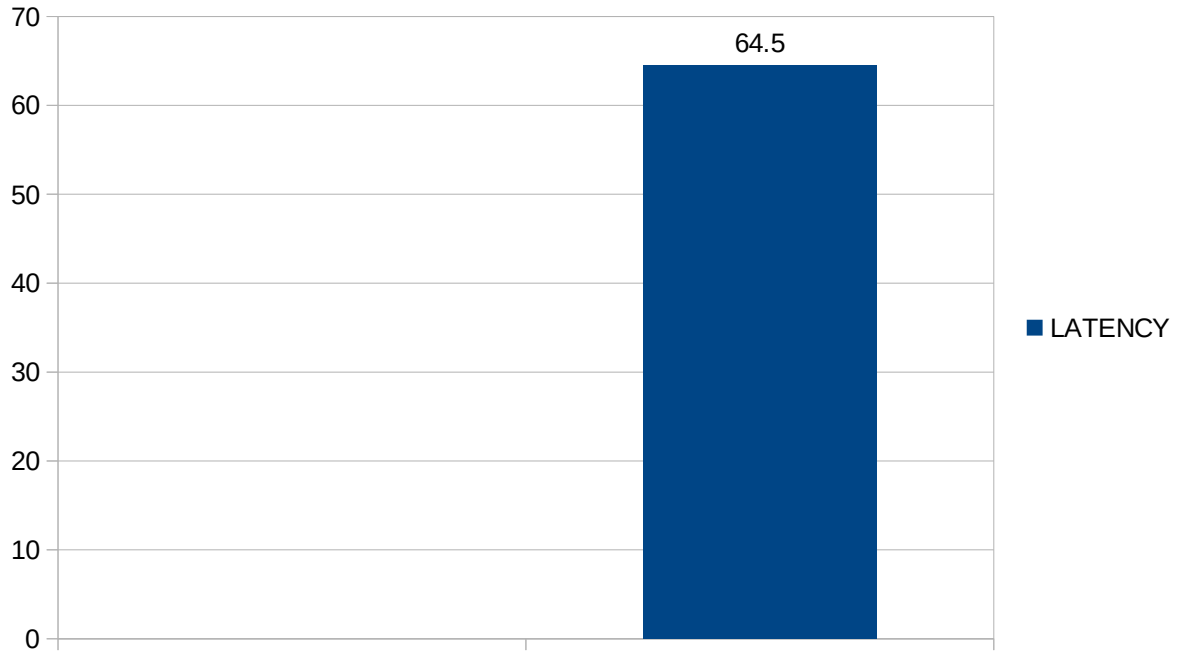
OpenWrt je projekt otvorenog koda koji se temelji na operativnom sustavu Linux i dodaje podršku za brojne internetske usluge. Osim toga, može se koristiti i na uređajima poput routera, pametnih telefona, prijenosnih računala i sl.

U prijašnjem poglavlju govorio sam o vrlo skupim prijemnicima koji daju odlične performanse pomoću algoritma SQM za sprječavanje bufferbloata. Zato postoji besplatna mogućnost promjene ugradbenog softvera na komercijalnim prijemnicima, tako da se instalira ugradbeni softver OpenWrt koji sadrži u sebi SQM algoritme.

Instalacija OpenWrta je donekle rizična, iz razloga što prvo treba provjeriti dali je router kompatibilan s OpenWrtom, što se naravno radi na vlastiti rizik. Isto tako ukoliko je instalacija uspješna performanse routera će biti znatno bolje, no ukoliko to nije tako potrebno je uložiti u bolji router.

6. Mjerenja

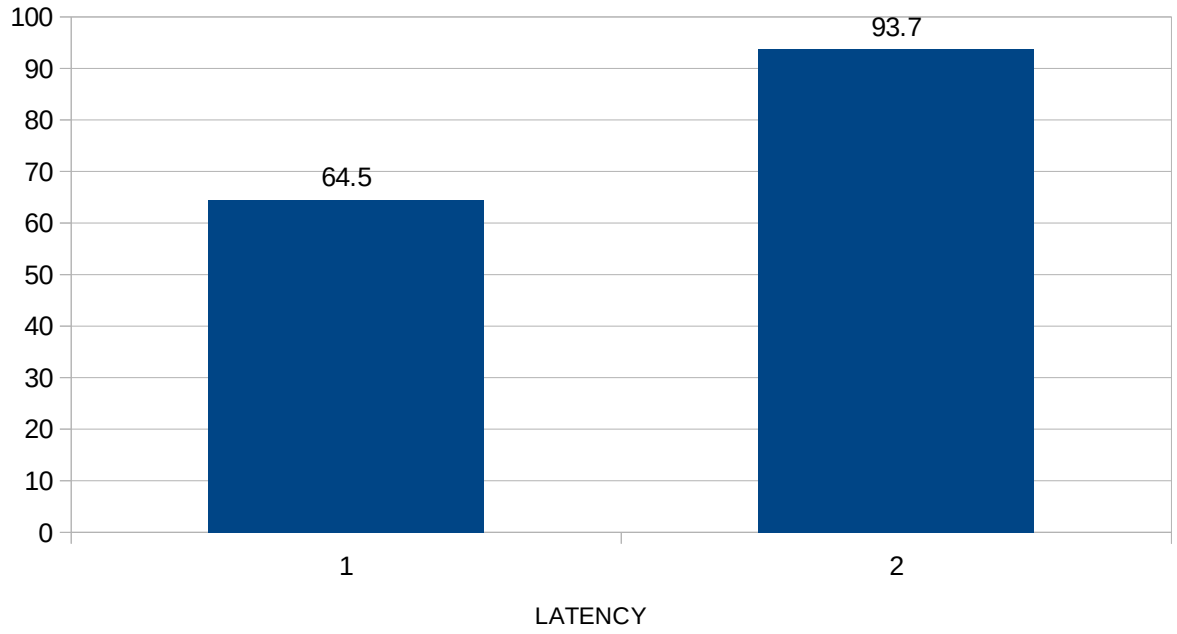
6.1. Ethernet veza



Graf 1: Mjerenje vremena čekanja

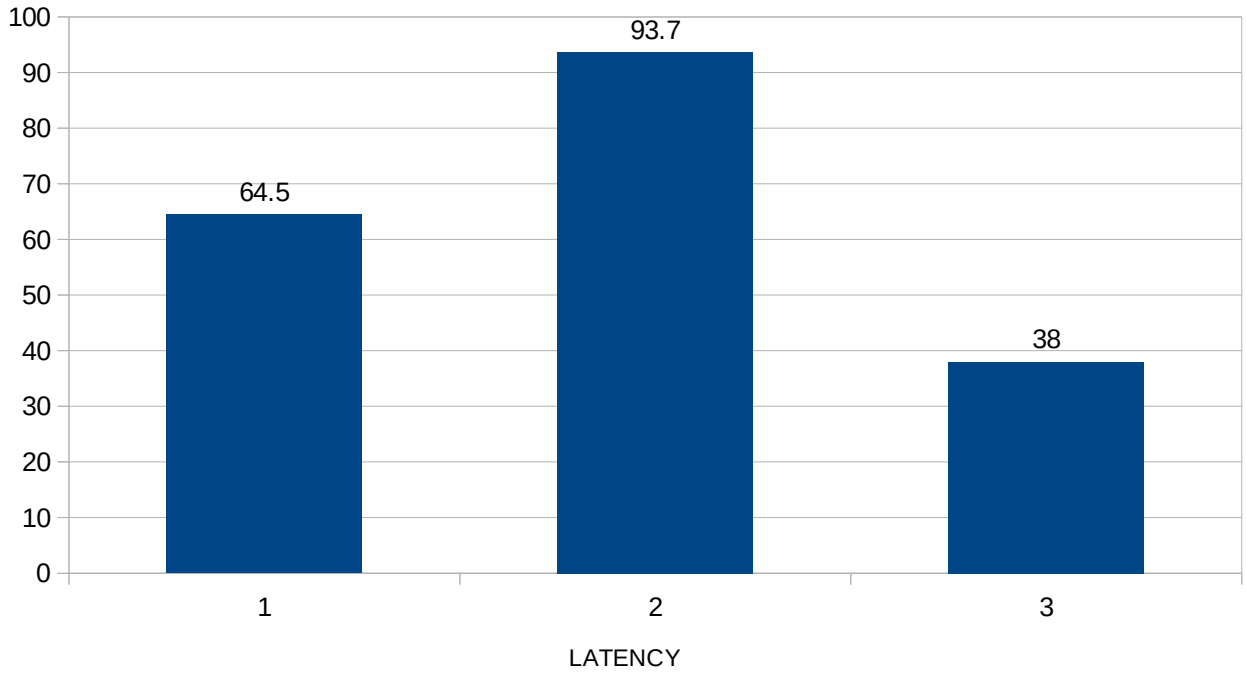
Na grafu broj 1 prikazao sam mjerenje zadržavanja između kućnog računala i YouTube-a kada je u pozadini pokrenut YouTube video veličine 4K. Uočio sam da je odmah vrijeme čekanja znatno veće te kao što sam u početku ove priče o bufferbloatu rekao, cilj je dakle, da vrijeme čekanja bude oko 50 ms.

Vrijeme čekanja od 64.5 ms nije značajan problem, ali već pomalo dovodi do sporije komunikacije putem interneta, odnosno potrebno je nešto više vremena da se učita cijeli video te ga nije moguće pogledati u kontinuitetu. Isto tako otežava otvaranje drugih web stranica.



Graf 2: Mjerenje vremena čekanja

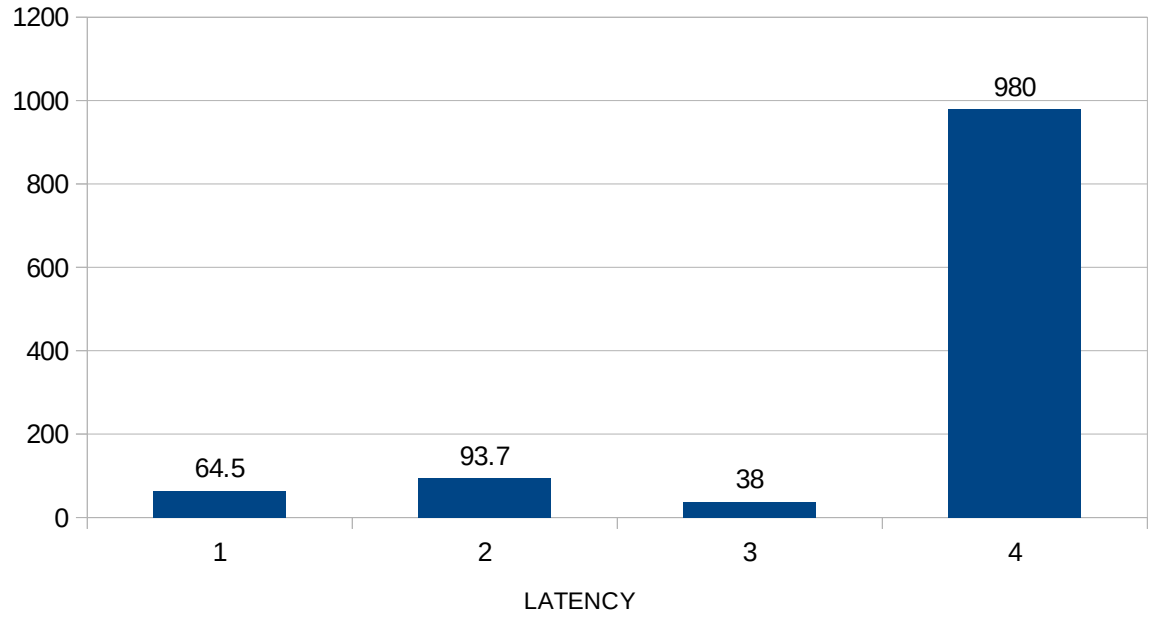
Na grafu broj 2 prikazao sam mjerenje s dva pokrenuta videa veličine 4K te se vidi kako je vrijeme čekanja znatno veće u odnosu na mjerenja u 1. grafu. Konkretno, stupac s brojem 2 prikazuje vrijeme čekanja od 93.7 ms s dva pokrenuta videa, a stupac s brojem 1 vrijeme čekanja od 64.5 ms s jednim pokrenutim videom u pozadini.



Graf 3: Mjerenje vremena čekanja

Na grafu broj 3 prikazao sam mjerenje bez YouTube videa i ostalih web stranica te sam odmah

uočio kako je vrijeme čekanja odmah ispod 50 ms, dakle 38 ms, što nam dozvoljava optimalan rad veze prema internetu, bez zagušenja. Konkretno, stupac s brojem 1 prikazuje vrijeme čekanja od 64.5 ms sa samo jednim pokrenutim videom, stupac s brojem 2 prikazuje vrijeme čekanja od 93.7 ms s dva pokrenuta videa, a stupac s brojem 3 prikazuje vrijeme čekanja od 38 ms bez pokrenutog videa u pozadini te je tada kao što sam ranije opisao vrijeme čekanja najbolje.



Graf 4: Mjerenje vremena čekanja

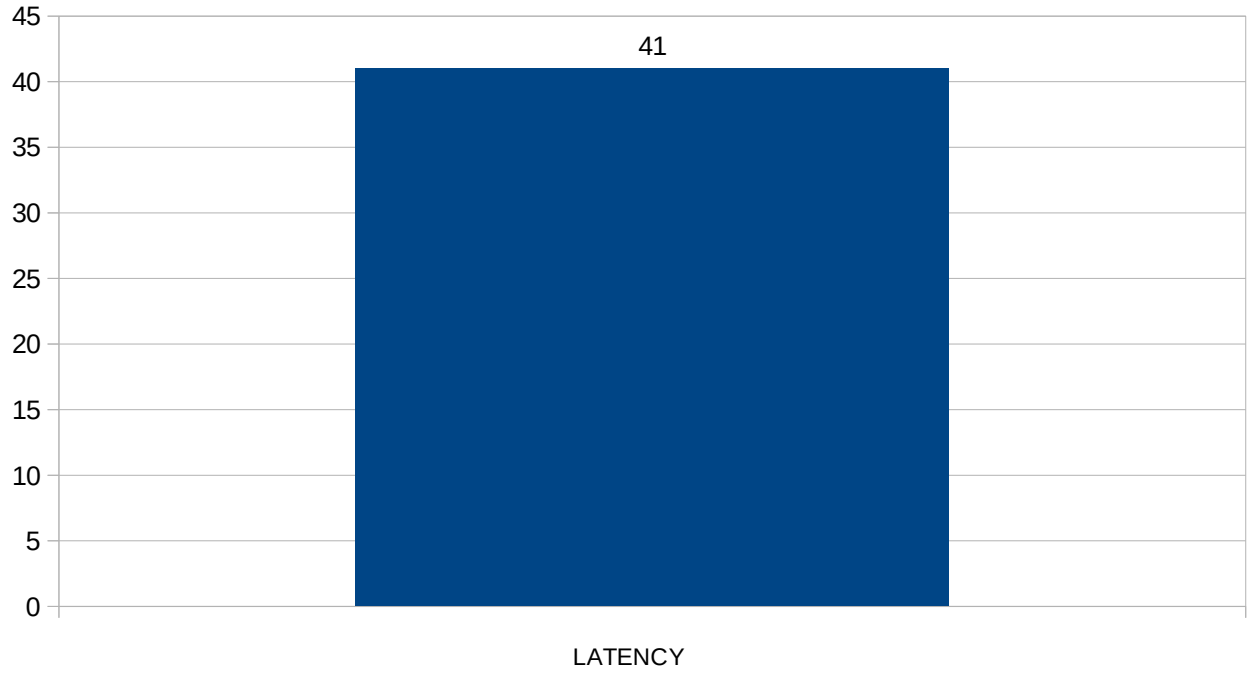
Na grafu broj 4 prikazao sam mjerenje s pokrenutom internet igrom „*Call of Duty Warzone"* te igrajući nekoliko minuta vrijeme čekanja je naraslo do čak 980 ms. Konkretno, stupac s brojem 1 prikazuje vrijeme čekanja od 64.5 ms sa samo jednim pokrenutim videom, stupac s brojem 2 vrijeme čekanja od 93.7 ms s dva pokrenuta videa, a stupac s brojem 3 nam prikazuje vrijeme čekanja od 38 ms bez pokrenutih videa, dok stupac s brojem 4 prikazuje vrijeme čekanja od 980 ms s pokrenutom video igrom u pozadini.

Warzone je igra za više igrača (eng. *multiplayer*) koja osim što daje mogućnost od igranja protiv 200 drugih igrača iz cijelog svijeta te ima i ogromnu mapu s jako puno detalja koje je potrebno učitati. Naravno to stvara problem jer se sve ne stvara odmah, nego kako se krećemo prikazuju nam se detalji npr. na kući ili u prirodi te još nailazimo na nekoliko nasumičnih razbacanih kutija (eng. *loot box*) s oružjem, municijom i sl.

Vrlo važno napomenuti kako je na ovom primjeru uočljivo kako komercijalni routeri dovode do velikog vremena čekanja ukoliko dođe do opterećenja veze, ali isto tako važno je napomenuti kako je bio otvoren i Mozilla Firefox web preglednik za testiranje brzine veze, što je isto tako jedan od

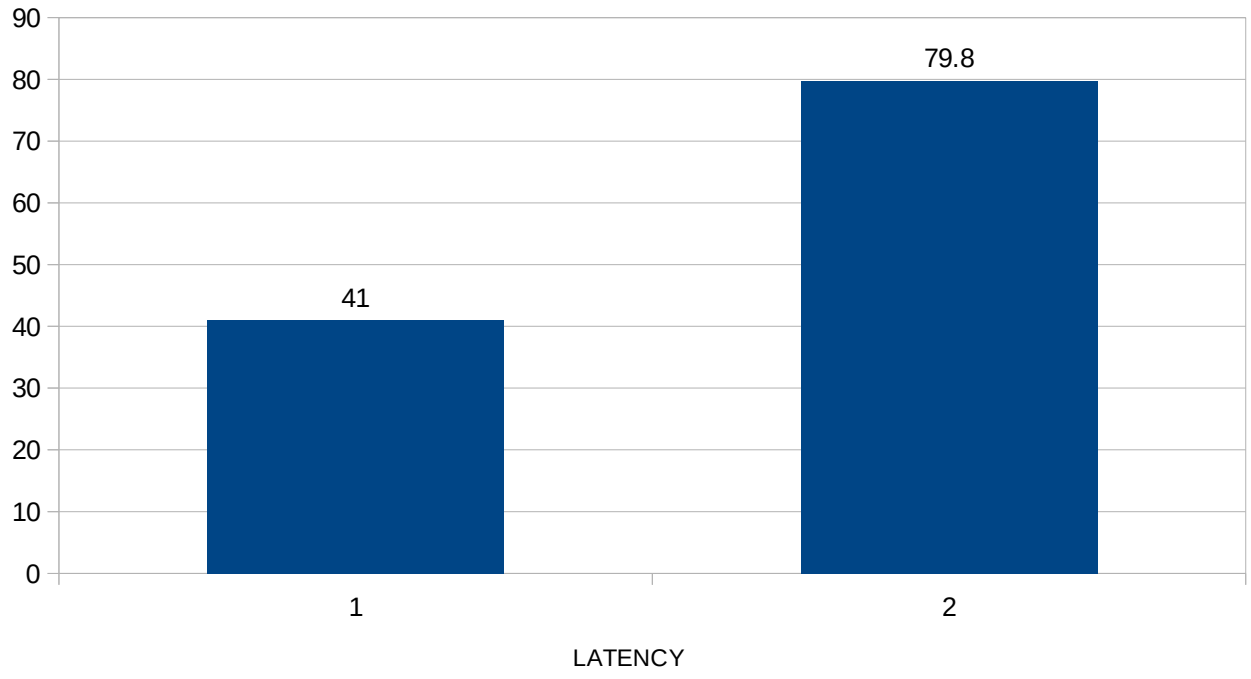
uzroka velikog vremena čekanja.

6.2. Wi-Fi konekcija



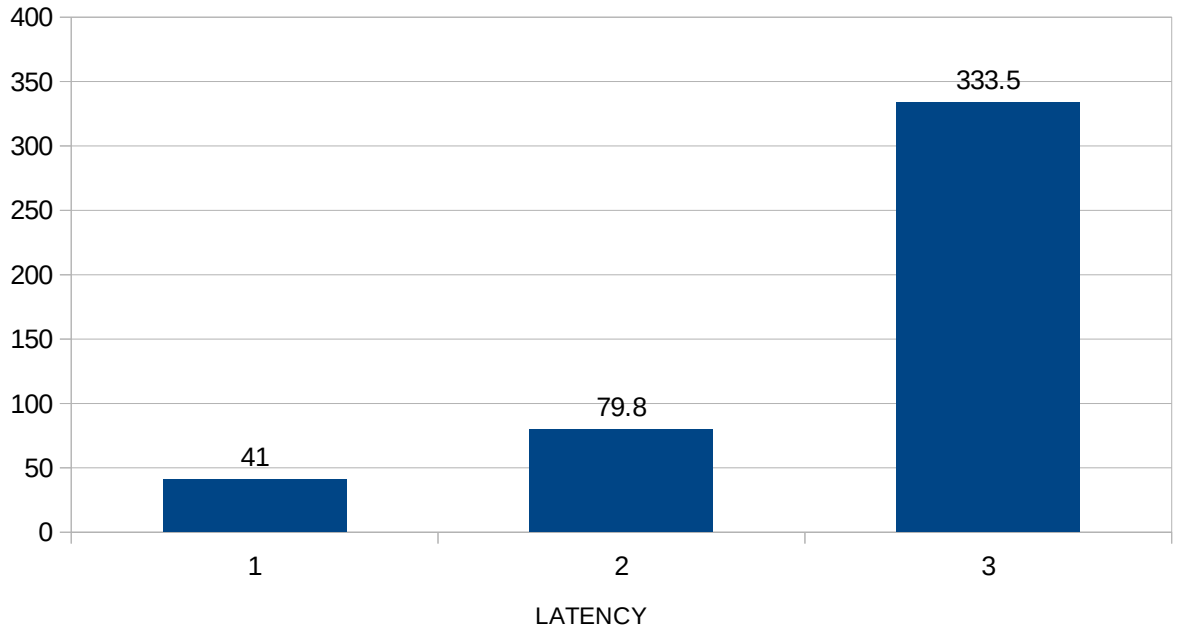
Graf 5: Mjerenje vremena čekanja

Na grafu broj 5 prikazao sam mjerenje Wi-Fi konekcijom gdje je u internet pregledniku bila otvorena samo stranica za mjerenje brzine interneta. Uočljivo je kako je vrijeme čekanja nešto manje od 50 ms, što nam naravno odgovara te nam je kvaliteta internet usluge vrlo dobra.



Graf 6: Mjerenje zadržavanja

Na grafu broj 6 prikazao sam mjerenje s pokrenutim videom u veličini 4K na YouTube kanalu te je odmah vidljivo koliko je vrijeme čekanja veće od prijašnjeg mjerenja, što nam naravno dovodi do slabije internet usluge. Konkretno, stupac s brojem 1 prikazuje vrijeme čekanja od 41 ms bez pokrenutih videa, a stupac s brojem 2 prikazuje vrijeme čekanja od 79.8 ms s pokrenutim videom u pozadini.



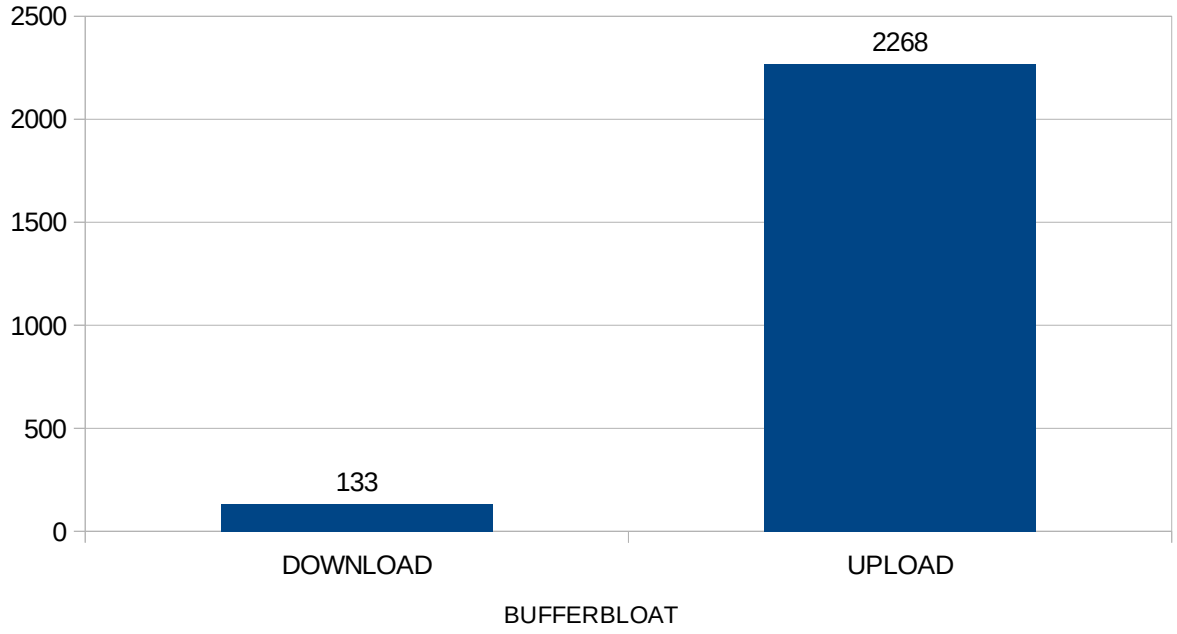
Graf 7: Mjerenje zadržavanja

Na grafu broj 7 prikazao sam mjerenje tako da sam u internet pregledniku otvorio dva videa u veličini 4K na YouTube kanalu te sam uočio kako je vrijeme čekanja znatno veće u odnosu na prijašnja mjerenja te je i znatno veće od istog mjerenja Ethernet vezom kod grafa broj 2. Konkretno, stupac s brojem 1 prikazuje vrijeme čekanja od 41 ms bez pokrenutih videa u pozadini, stupac s brojem 2 prikazuje vrijeme čekanja od 79.8 ms s jednim pokrenutim videom, a stupac s brojem 3 vrijeme čekanja od 335 ms s dva pokrenuta videa u pozadini.

Ovim mjerenjima zaključujem kako Wi-Fi konekcija nije najbolja solucija za npr. gledanje videa u visokoj kvaliteti, igranje internet igara i sl. O čemu sam nešto više rekao u poglavlju o Wi-Fi-ju.

6.3. Bufferbloat mjerenja

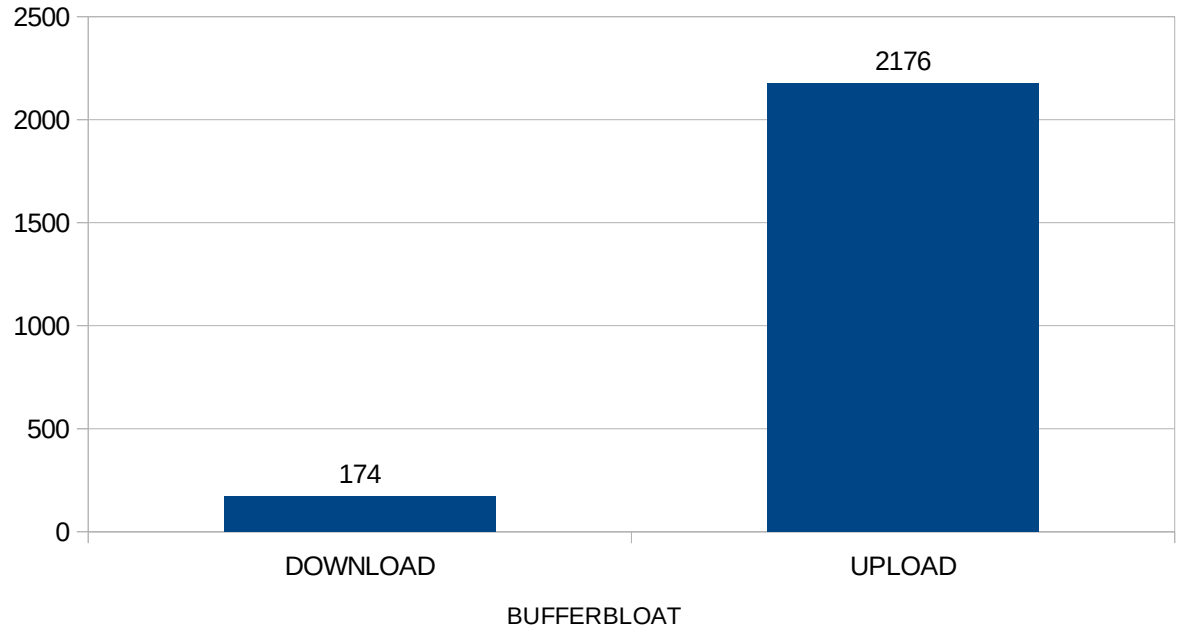
6.3.1. Ethernet mjerenja



Graf 8: Bufferbloat mjerenja

Na grafu broj 8 prikazao sam bufferbloat mjerenja, ali sam koristio dva routera. Uočio sam kako oba routera imaju problema s bufferbloatom. Prilikom mjerenja, brzina preuzimanja datoteke (eng. d*ownload speed*) je u prosjeku oko 35 Mbit/s te je time i bufferbloat u prosjeku od 133 ms, što je vrlo loše, nebi smijelo biti preko 50 ms. No prilikom slanja podataka (eng. u*pload*) gdje je njegova brzina oko 1.5Mbit/s, ali ona stalno varira ovisno o tome što se šalje s mojeg računala prema drugom npr. video igre, video pozivi i sl. Bufferbloat je tu u prosjeku oko 2268 ms te nam to ukazuje na velika zagušenja prilikom slanja podataka, vrijeme čekanja je naravno onda vrlo visoko. Konkretno, stupac preuzimanje datoteka (eng.*download*) prikazuje bufferbloat od 133 ms, a stupac

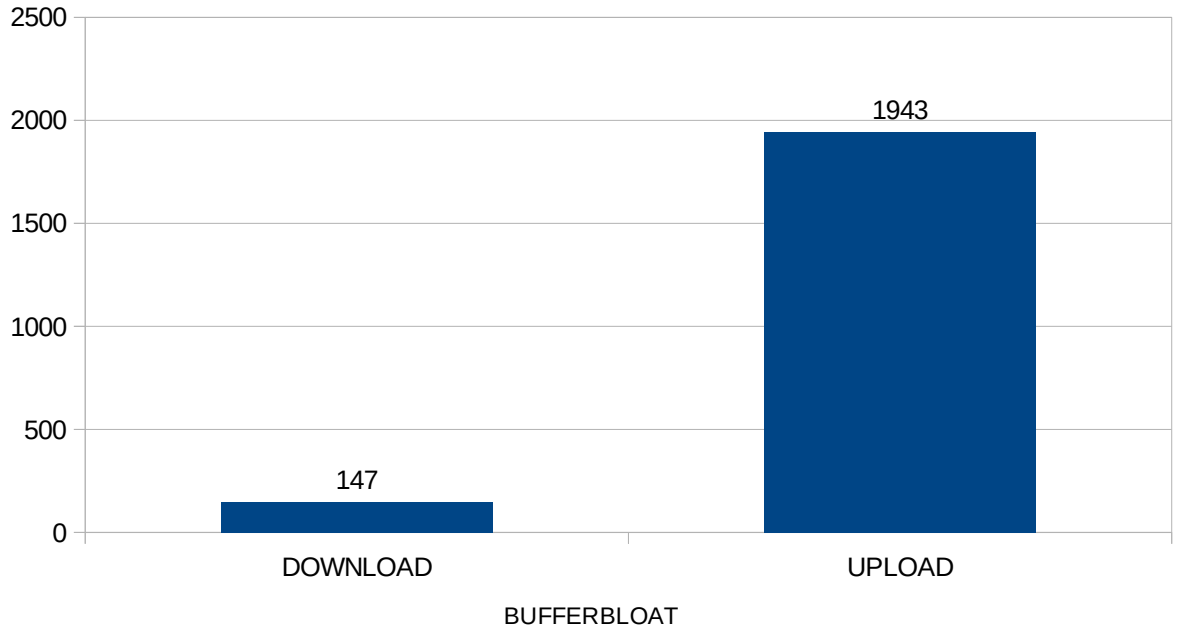
slanja podataka (eng. *upload*) bufferbloat od 2268 ms.



Graf 9: Bufferbloat mjerenja

Na grafu broj 9 prikazao sam bufferbloat mjerenja sa samo jednim routerom. Uočio sam kako je prilikom preuzimanja datoteka brzina skoro ista, ali je bufferbloat zato oko 174 ms, dakle 40 ms veći u odnosu na prošlo mjerenje u grafu broj jedan. Prilikom slanja podataka njegova brzina je veća pa je zato bufferbloat oko 2176 ms, dakle oko 100 ms manji u odnosu na prošlo mjerenje. Konkretno, stupac preuzimanje datoteka (eng.*download*) prikazuje bufferbloat od 174 ms, a stupac slanja podataka (eng. *upload*) bufferbloat od 2176 ms.

6.3.2. Wi-Fi mjerenja



Graf 10: Bufferbloat mjerenja

Na grafu broj 10 prikazao sam Wi-Fi bufferbloat mjerenja te sam uočio da je bufferbloat prilikom slanja podataka oko 1943 ms. Uspoređujući ovo mjerenje sa Ethernet mjerenjima, vidimo da je bufferbloat prilikom slanja podataka kod Wi-Fi-ja puno manji. Konkretno, stupac preuzimanje datoteka (eng.*download*) prikazuje bufferbloat od 147 ms, a stupac slanja podataka (eng. *upload*) bufferbloat od 1943 ms.

Naravno treba napomenuti da ukoliko opteretimo Wi-Fi s više uređaja povezanih na mrežu rezultati bi bili mnogo gori.

7. Zaključak

Bufferbloat je zapravo problem koji će biti još dugo prisutan. Razvojem algoritma FQ_CoDel omogućio je problem svesti na minimum, odnosno pokazao se kao najbolje rješenje. Stoga najnoviji routeri, pogotovo malo skuplji imaju već u sebi algoritam FQ_CoDel, što nije slučaj kod komercijalnih koje dobijemo od pružatelja internetskih usluga. Tu se referiram na Ubiquiti EdgeRoutere koje sam ranije opisao, jer su se pokazali kao jedni od najboljih routera za stabilnu i brzu vezu prema internetu bez ograničenja to sve zahvaljujući dobro implementiranim algoritmima.

Rješenja dakle, za kompletno suzbijanje bufferbloata još nema, nego se može ublažiti kupovinom skupih routera poput Ubiquiti ili instalacijom OpenWrta uz provjeru dali ga router podržava te se to radi na vlastitu odgovornost.

Literatura

[1] I. N. Bozkurt *et al.*, “Why Is the Internet so Slow?!,” in *Passive and Active Measurement*, vol. 10176, M. A. Kaafar, S. Uhlig, and J. Amann, Eds. Cham: Springer International Publishing, 2017, pp. 173–187.

[2] “What Is Latency? | How to Fix Latency,” *Cloudflare*. https://www.cloudflare.com/learning/performance/glossary/what-is-latency/ (accessed Sep. 03, 2020).

[3] “TCP/IP Tutorial and Technical Overview,” p. 1004.

[4] “2002IntroductiontoTCPIPProtocolSuite.pdf.” .

[5] M. Fisk and W. Feng, “Dynamic Adjustment of TCP Window Sizes,” p. 12.

[6] “hjp: doc: RFC 3168: The Addition of Explicit Congestion Notification (ECN) to IP.” https://www.hjp.at/doc/rfc/rfc3168.html (accessed Sep. 03, 2020).

[7] S. Floyd, “Random Early Detection (RED) gateways,” p. 16.

[8] “tc-fq_codel(8) - Linux manual page.” https://man7.org/linux/man-pages/man8/tc-fq_codel.8.html (accessed Sep. 03, 2020).

[9] “Fair queuing,” *Wikipedia*. Jun. 24, 2020, Accessed: Sep. 03, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Fair_queuing&oldid=964239124.

[10] “Wi-Fi,” *Wikipedia*. Aug. 31, 2020, Accessed: Sep. 03, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wi-Fi&oldid=976019231.

[11] “MakeWiFiFast,” *Google Docs*. https://docs.google.com/document/d/1Se36svYE1Uzpppe1HWnEyat_sAGghB3kE285LElJBW4/edit?usp=embed_facebook (accessed Sep. 03, 2020).

[12] “Smart Queue Management - Bufferbloat.net.” https://www.bufferbloat.net/projects/cerowrt/wiki/Smart_Queue_Management/ (accessed Sep. 03, 2020).

[13] “Bufferbloat: Why it is Harming Your Broadband and How to Easily Fix It | Increase Broadband Speed.” https://www.increasebroadbandspeed.co.uk/bufferbloat-broadband-fix (accessed Sep. 03, 2020).