Kratka priča o mrežama - preklopnici i usmjerivači

Autor : Hrvoje Horvat

Naslov : Kratka priča o mrežama - preklopnici i usmjerivači

Mjesto i godina izdavanja: Osijek, 2017

Verzija publikacije : 1.02

Nakalada : Vlastita naklada

ISBN: 978-953-59438-5-3 (HTML-online)

DokuWiki URL (HTML):

https://www.opensource-osijek.org/dokuwiki/wiki:knjige:kratka-prica-o-mrezama-preklopnici

ISBN: 978-953- 59438-6- 0 (PDF-online)

PDF URL : https://www.opensource-osijek.org/knjige/Kratka-priča-o-mrežama-preklopnici-i-usmjerivači.pdf

Ova knjiga je napisana unutar inicijative *Open Source Osijek*: https://www.opensource-osijek.org

Inicijativa *Open Source Osijek* je član udruge *Osijek Software City*: http://softwarecity.hr/

Sadržaj

Predgovor

U ovoj uvodnoj priči o mrežama, upoznati ćemo se s raznim osnovnim mrežnim pojmovima i tehnologijama te mrežnim protokolima. Pošto je namjena ove knjižice upoznavanje s mrežama i mrežnim tehnologijama, na većini tema ćemo samo malo zagrebati površinu priče. Nadalje, upoznati ćemo se s mrežnim uređajima koji se zovu preklopnici (Engl. *Switches*) te ćemo naučiti ponešto i o usmjerivačima (Engl. *Routers*). Vidjeti ćete s kojim problemima se susreću proizvođači mrežne opreme, od faze projektiranja i dizajna te u konačnici testiranja samih uređaja. Uvidom u ovu problematiku, dobiti ćete širu sliku o raznim problemima koje mogu prouzrokovati mrežni uređaji, njihov loš odabir ili nepravilna konfiguracija. Dakle ovo nije još jedna klasična školska priča o mrežama i mrežnim uređajima, već pogled na već stotine puta ispričanu prču, ali iz druge perspektive. Saznati ćete i razlike između teorije i prakse te vidjeti, postoje li odstupanja između (reklamiranih) specifikacija proizvođača i rada u praksi. I na kraju, pregledati ćemo i najnovija dostignuća te se upoznati s novim smjerovima razvoja najnaprednijih uređaja i tehnologija, koje će s vremenom postati standard i za široko dostupnu opremu te svakodnevnu upotrebu u daljoj budućnosti.

“ **Budi promjena koju želiš vidjeti u svijetu !.** ”

Mohandas Karamchand Gandhi (Mahatma Gandhi)

O Autoru

Hrvoje Horvat je rođen 1975 godine u Osijeku, gdje je završio Prvu tehničku školu, smjer elektronika i automatika. Od prvog druženja s računalom, sredinom 80-tih (ZX Spectrum) i prvih igara a potom prelaskom na Atari ST i kasnije na PC arhitekturu, odabir budućeg zanimanja mu je postao vrlo jasan. Prvo umrežavanje računala s kolegama, u naselju, tijekom 1994, doprinijelo je novim smjerovima osobnog rada i razvoja. Uslijedili su razni poslovi održavanja računala i mreže te se 2000. godine zapošljava u tvrtki ***Siemens d.o.o*** u kojoj i danas radi.

Prvih nekoliko godina radi kao sistem administrator, potom sistem i mrežni inženjer, na korporativnoj IT infrastrukturi tvrtke, te na nekoliko većih infrastrukturnih projekata. Pohađa razne specijalizacije, kako vanjske tako i unutar tvrtke. Zadnjih desetak godina radi kao razvojni inženjer, na projektima koji obuhvaćaju : visoko dostupne i redundantne sustave, AAA i PCRF servise, Virtualizaciju i NAS/SAN sustave te napredne mrežne servise i protokole.

Suosnivač je udruge za razvoj IT sustava *Udruga01*, unutar koje se pokreću deseci projekata razvoja i testiranja raznih mrežnih uređaja, protokola i tehnologija, u cilju učenja i stjecanja novih znanja. Suosnivač je inicijative „Open Source Osijek“, koja potiče i promovira razvoj i upotrebu sustava otvorenog koda, u kojoj i aktivno objavljuje stručne članke. Član je udruge *Osijek Software City* u kojoj djeluje kao aktivni član i predavač.

Preklopnici (switchevi) i usmjerivači (routeri)

Preklopnici (*switchevi*) i usmjerivači (*routeri*) uređaji su, koji se sastoje od sličnih dijelova kao i svako računalo. Dakle imaju matičnu ploču, CPU, RAM, neku vrstu diska (uglavnom *flash* memoriju), operacijski sustav, mrežne kartice i njihove pripadajuće upravljačke programe (*drivere*) i dodatni softver.

Doduše njihov operacijski sustav je malo drugačiji od onoga na koji smo naviknuli ali ne toliko koliko se čini.

Trebamo biti svjesni činjenice kako i obični "glupi" preklopnik u pozadini odrađuje nekoliko funkcionalnosti, odnosno pokreće nekoliko mrežnih protokola, te ima dodatne mrežne funkcionalnosti, od kojih neke i primjenjuje na svaki paket na mreži. Već na gigabitnim mrežama to znači milijune paketa u sekundi, koje je potrebno konstantno obrađivati.

Zapravo si možemo i trebamo postaviti mnoga pitanja a koja su si postavili i proizvođači kod razvoja mrežnih uređaja.

Što je važno kod odabira uređaja

Kod dizajna i konstrukcije uređaja, važno je znati, kako odabir operacijskog sustava uređaja ima vrlo veliku ulogu u njegovom radu. Tako si proizvođači uređaja prvo postavljaju pitanja:

- koji operacijski sustav je sigurniji
- koji od njih je brži
- koji je stabilniji i/ili brži
- koji ima bolju podršku
- što je s upravljačkim programima (*driverima*) za sav hardver
- za koji od njih podproizvođači hardvera češće objavljuju optimizacije i ispravke grešaka

Vezano za operacijski sustav, ako samo pogledamo operacijske sustave koji se koriste za stolna računala, možemo primijetiti trendove razvoja u kojima se sve manje pažnje obraća na optimizaciju općenito a poglavito na brzinu rada i zauzeće diskovnog prostora. Pogledajte zahtjeve za *hardverom*, primjerice, zadnjih nekoliko inačica operacijskog sustava koji koristite na vašem stolnom računalu. Kod mrežnih uređaja, barem ovo nije slučaj, najviše zbog toga što je hardver ovih uređaja vrlo ograničen, i to ako ništa drugo, njegova *RAM* memorija, *CPU* te diskovni prostor, stoga si nitko u razvoju ovih uređaja ne može priuštiti komfor razbacivanja sa stotinama ili tisućama nepotrebnih sistemskih biblioteka, raznoraznih nepotrebnih frameworka i funkcionalnosti koje nisu stvarno, izričito nužne. Naime ovdje govorimo o vrlo malo diskovnog prostora i *RAM* memorije, u odnosu na klasična računala. Dodatno, većina mrežnih uređaja učitava cijeli operacijski sustav u *RAM* memoriju, kako bih radio što brže i bez posredovanja diska (obično neke *flash* memorije).

Nadalje, važan je i dobar odabir programskog jezika u kojemu se razvijaju funkcionalnosti uređaja. Vezano za to, dolazimo i do pitanja, kako razvijati softver i tko ga razvija. Ovdje imamo još nekoliko važnih pitanja: Testiranje – je li uređaj stvarno (i kako) testiran

- Optimizacija – koliko znanja treba imati kako bi se radile optimizacije te jesu li moguće (i u kojoj mjeri)
- Podrška – što je s podrškom, koliko su ažurni
- Dokumentacija samog softvera i konačnog programskog rješenja (uređaja) te kolika je zajednica ljudi koja koristi uređaje određenog proizvođača, postoje li dostupna predavanja, knjige, kao i drugi materijali i koliko su dobro napisani i sl.

Neki od proizvođača, za određene funkcionalnosti koriste programske jezike u kojima je razvoj tražene funkcionalnosti ekstremno brz ali su performanse takvog rješenja katastrofalne ako se radi o funkcionalnostima koje se stalno koriste u radu. Tako neki korisnici koji povremeno trebaju ove funkcionalnosti hvale ovakvog proizvođača odnosno njegov konkretan uređaj, dok se drugi koji isti koriste u okruženju u kojemu je navedena funkcionalnost stalno u upotrebi, žale zbog lošeg odabira uređaja (i proizvođača).

Postoji i cijeli niz primjera u kojima imamo određene funkcionalnosti odnosno mrežne protokole koji ne rade pouzdano i stabilno, a pogotovo u određenim kombinacijama s drugim protokolima.

Ako recimo govorimo o uređajima koji imaju implementirane naprednije mrežne protokole poput *VRRP* protokola za redundanciju na OSI sloju 3 (*Layer 3*), dobar dio proizvođača preklopnika i usmjerivača ih ima implementirano. Od svih njih, neki imaju konstantnih problema upravo s njim a opet, nikako da se poprave (uozbilje).

Ovakvih i sličnih primjera ima vrlo mnogo, kao i protokola koje mnogi mrežni uređaji podržavaju (barem u teoriji).

I na kraju, važno je proučiti što više detalja o uređaju, kao i komentare naprednijih korisnika.

Mrežni uređaji i pojmovi

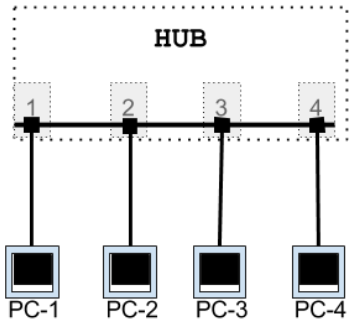
Prije nego krenemo dalje te se upoznamo s preklopnicima, moramo se samo malo vratiti u povijest razvoja mrežnih uređaja te se upoznati i s njihovim načinom rada i pokojim novim pojmom.

CSMA/CD i Kolizijska domena

CSMA/CD (*Carrier Sense Multiple Access With Collision Detection*).

Prije vremena preklopnika, koristili su se uređaji zvani *HUBovi*.

Računala i druga mrežna oprema se na njih spajala na isti način kao na preklopnike - prema topologiji : zvijezda [star] – sva računala i oprema, svaki se spajao na svoj *port* (utičnicu) na *HUBu*.



Pogledajmo i logičku shemu spajanja računala na *Hub*:

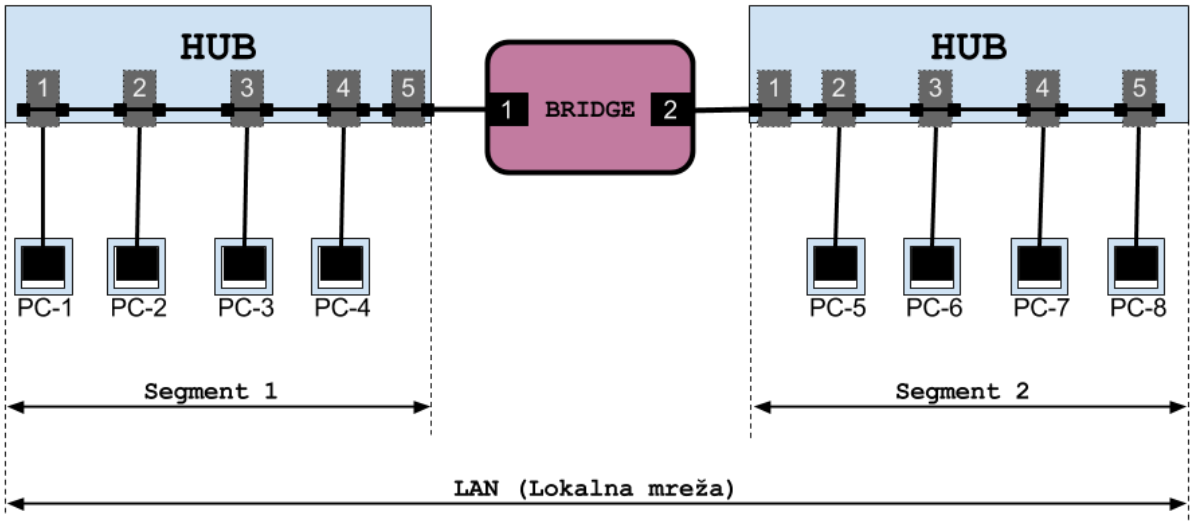
Na *HUBovima* se dijelila zajednička veza jer je on praktično radio kao pojačalo električnog signala s jednog *porta* (mrežnog sučelja) na drugi. Veza je prema tome bila jednosmjerna odnosno *Half Duplex*, što znači kako je moguće samo primanje ili slanje paketa na mrežu u svakoj jedinici vremena. Naime u ovakvoj komunikaciji samo jedno računalo u jednoj jedinici vremena smjelo je poslati pojedini mrežni paket na mrežu. Zbog ovakvog načina rada, stalno se događalo kako su dva ili više računala u jednom djeliću vremena krenula u slanje podataka na mrežu (tzv. mrežnih paketa).
Tu je uskakao *CSMA/CD* mehanizam, koji je tada detektirao ovaj slučaj, koji se zove **kolizija.** Dakle aktivirao se *Collision Detection* mehanizam, koji radi tako da tada svaki od pošiljatelja mora sačekati određeni nasumični broj od nekoliko milisekundi te probati slati podatke ponovno, i tako iz početka, svaki puta kada se detektira kolizija. Kod *HUBova* je kolizija bila konstantna pojava. Naime u normalnom radu mreže, svako malo, neko računalo želi slati neke podatke (pakete) na mrežu.

Ovakav način rada osim što je usporavao mrežu, zbog jednostrane komunikacije (svako računalo može ili slati ili primati podatke/pakete u jedinici vremena), usporavao je komunikaciju i stoga jer se dijelio mrežni medij (komunikacijski kabel) između svih računala na mreži. Dodatno i zbog *Collision Detection* mehanizma (bez kojeg ionako sve ne bi moglo raditi) odnosno vremena čekanja do ponovnog slanja paketa na mrežu. Na kraju, sve zajedno je bilo vrlo usporeno.

Bridge

Kako bi se problem s kolizijskom domenom ublažio, mreža na kojoj su računala spojena pomoću *HUBova*, morala se razdijeliti na segmente, u kojima je svaki od segmenata bio u svojoj kolizijskoj domeni. Da bi to bilo moguće, napravljeni su uređaji, koji su imali samo nekoliko *portova* odnosno mrežnih sučelja na koja se spajaju *Hubovi* te se mreža s *Hubovima* na taj način segmentirala. Ovi uređaj se zovu *Bridgevi*. *Bridgevi* svaki mrežni paket na svakom svom mrežnom sučelju promatraju na OSI sloju 2, odnosno gledaju mu *MAC* adresu s koje je paket došao (*Source MAC*) te provjeravaju na koju odredišnu *MAC* (*Destination MAC*) adresu mora biti prebačen (isporučen). Kako smo rekli, ovakvi uređaji su se zvali ***Bridge-evi*** odnosno mrežni mostovi, jer su povezivali više segmenata iste mreže, iste ako gledamo na razini IP adresa odnosno OSI sloja 3. To znači kako se lokalna mreža, koja može buti unutar istog opsega IP adresa, segmentirala (razdvojila) na više nezavisnih djelova, što se tiće kolizije. Ovi uređaji su preteča današnjih preklopnika (*switcheva*) ali su u to vrijeme imali samo nekoliko mrežnih sučelja, poput današnjih usmjerivača. Kako ne bi bilo zabune, ovi uređaji su prema svojoj funkcionalnosti preklopnici s malim brojem mrežnih sučelja, na kojima se također u radu, gradila tablica preklapanja i samo preklapanje, a koje je odrađivao centralni procesor (*CPU*) ovakvog uređaja. Standard koji definira rad **bridge** uređaja ali i **preklopnika** je 802.1D.

Pogledajmo i način spajanja **HUB** i **bridge** uređaja te segmentaciju lokalne mreže, pomoću njih.



I danas se pojam *Bridge* koristi unutar klasičnih operacijskih sustava, koji nam nude mogućnost povezivanja dvaju (ili više) mrežnih sučelja u "*Bridge*". Dakle u način rada, u kojemu se mrežni paketi (okviri) prebacuju iz jednog mrežnog sučelja u drugo, na osnovi tablice preklapanja. Naime u ovakvom načinu rada, vaše računalo postaje mali *bridge* odnosno preklopnik.

Kasnijim razvojem i ubrzavanjem hardvera, razvijali su se i preklopnici, koje možemo promatrati kao *bridge* uređaje s više mrežnih sučelja. Važnost ovakvog rada je činjenica kako se kolizijska domena smanjila, praktično na razinu svakog pojedinig mrežnog sučelja, u slučajevima ako se koristi dvosmjerni način komunikacije. Ovakav dvosmjerni način komunukacije svakog pojedinog mrežnog sučelja poznat je kao *Full Duplex* način rada.

Važno je znati i kako je zbog kompatibilnosti unatrag i dalje ostao jednosmjerni `Half Duplex` način rada svakog mrežnog sučelja, na koji treba paziti. Problem se može pojaviti jer se svako mrežno sučelje na preklopniku i krajnje računalo ili mrežna oprema mogu krivo "dogovoriti" te uspostaviti *Half Duplex* umjesto *Full Duplex* načina rada. Ovo se događa u slučaju upotrebe `Auto Negotiation` protokola, za koji uglavnom niste niti svjesni kako je već u upotrebi na vašem preklopniku. Ali o tome malo kasnije.

Full duplex je način rada u kojemu je moguće istovremeno slanje i primanje podataka i to samo ako je tako konfigurirano na strani preklopnika i druge strane (računalo, drugi preklopnik, poslužitelj , ….). Dakle obije strane moraju biti jednako konfigurirane !.

Duplex

Duplex je uz samu brzinu mreže (10Mbps/100Mbps/1000Mbps/…) drugi važan parametar vezan uz samu brzinu odnosno propusnost mreže. O čemu se radi ?.

Half Duplex

U starim mrežama (10 Mbps i 100 Mbps) koje su dijelile jedan medij (iz vremena korištenja *HUBova*), u svakom trenutku je bilo moguće ili primati ili slati podatke preko mreže (sjetimo se da je to jedan dijeljeni medij UTP kabel). Zbog problema ako je netko u takvoj mreži pokušao slati podatke istovremeno s nekim drugim je i uveden CSMA/CD protokol, koji je to riješio, tako što je natjerao sve koji su spojeni na mrežu da u jednom djeliću sekunde samo jedan može slati podatke a ostali samo slušati.

Ovakav način komunikacija nazivamo **`Half Duplex`**. Važno je razumjeti kako je **`Half Duplex`** način komunikacije u samo jednom smjeru : ili se šalje (ili prima) podatke, nikako istovremeno.

Full Duplex

Razvojem mreža i prelaskom s *HUBova* na preklopnike, više nije bilo potrebe za **`Half Duplex`** komunikacijom. Dakle upotrebom preklopnika, komunikacija više ne teče kroz jedan dijeljeni medij, između svih koji su spojeni na mrežu, već je svaka veza između računala (ili nekog drugog uređaja na mreži) i preklopnika, jedini dijeljeni medij pa je prema tome moguće istovremeno i primati i slati podatke. Naime prva komponenta između vas i susjednog računala je upravo preklopnik, prema kojemu svako računalo može imati dvosmjernu komunikaciju i na fizičkoj razini (*Layer 1*). Pošto preklopnici rade na OSI sloju 2 (*Layer 2*), mrežne paketa obrađuje preklopnik i preklapa ih iz jednog računala, preko sebe, do drugog računala i obrnuto, za svaki paket zasebno. S time je obostrana komunikacija zadržana. Ovu mogućnost dvostrane komunikacije (i primanje i slanje paketa mrežom u isto vrijeme) nazivamo **`Full Duplex`** način rada.

Postoji još jedna zanimljivost **`Full Duplex`** načina rada, a to je kumulativna brzina mreže. Ako recimo naša brzina mreže iznosi 1 000 Mbps (1 Gbps) to znači da tom brzinom možemo istovremeno i primati i slati podatke, što opet znači kako kumulativna brzina onda iznosi 2 000 Mbps (1 Gbps), što neki proizvođači i iznose u karakteristikama mrežnih uređaja ili komponenti.

Auto Negotiation

"Standardna" Brzina	Full Duplex Brzina
10 Mbps	20 Mbps
100 Mbps	200 Mbps
1000 Mbps (1 Gbps)	2000 Mbps (2 Gbps)
10 000 Mbps (10 Gbps)	20 000 Mbps (20 Gbps)

Kako su Ethernet mreže rasle s brzinom od 10 Mbps prema 100 Mbps i 1000 Mbps (1Gbps) ili dalje sa 10 000 Mbps (10 Gbps), morala je postojati i mogućnost korištenja smanjenja brzina odnosno kompatibilnosti unatrag. Kako bi se inače sa 1 000 Mbps (1 Gbps) mrežnom karticom mogli spojiti na mrežu koja radi na 100 Mbps ?. Osim same brzine rada ostao je i način komunikacije (*duplex*), koji je trebalo uvrstiti u varijablu.

Osim fizičke kompatibilnosti unazad te ručnog konfiguriranja brzine i *Duplex* načina rada, uveden je mrežni protokol imena : `Auto Negotiation`, koji omogućava dogovaranje parametara rada dvije strane mreže: pr. Računalo ⟷ Preklopnik. Ako sve zbrojimo, dva uređaja na mreži (i preklopnik kao posrednik) imaju nekoliko mogućih parametara, koje je potrebo uskladiti, a to su:

- **Brzinu rada** : 10 Mbps, 100 Mbps, 1 000 Mbps (1Gbps) ili 10 000 Mbps (10 Gbps)
- ***Duplex* način rada** : `Half` ili `Full`

`Auto Negotiation` radi tako da svaka strana šalje drugoj strani koje sve načine rada podržava - to izgleda otprilike ovako:

1. **Strana A → Strana B:** podržavam : 10 Mbps Half Duplex, 10 Mbps Full Duplex, 100 Mbps Half Duplex, 100 Mbps Full Duplex, 1 Gbps Full Duplex

2. **Strana B → Strana A:** podržavam : 10 Mbps Half Duplex, 10 Mbps Full Duplex, 100 Mbps Half Duplex, 100 Mbps Full Duplex, 1 Gbps Full Duplex

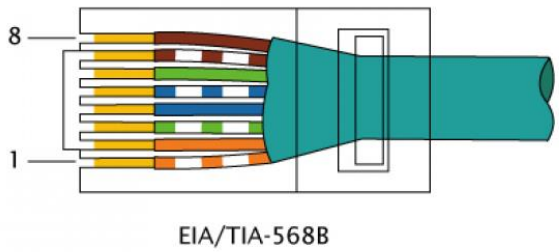
Nakon toga, svaka strana za sebe pronalazi najkompatibilniji način rada i postavlja svoje mrežno sučelje (*interface)* u taj način rada: pr. **Strana A**: 1 Gbps *Full Duplex*. Druga strana također postavlja svoje mrežno sučelje u nakompatibilnij način rada: pr. **Strana B**: također 1 Gbps *Full Duplex*. Dakle i brzina i *duplex* način rada se uvijek podešavaju od najveće brzine prema najmanjoj brzini i isto tako od `Full Duplex` do `Half Duplex` načina rada, prema parametrima primljenim od druge strane prema principu najvećeg zajedničkog nazivnika.

Problem je u tome što nakon ovog procesa obije strane ne provjeravaju za koji način rada se koja strana odlučila. Naime razni proizvođači mrežne opreme: od mrežnih kartica, preko preklopnika, usmjerivača i drugih mrežnih komponenti često se **ne** drže standarda ili ih ne implementiraju na najbolji način. Stoga se događa kako se dvije strane koje dogovaraju brzinu i *duplex*, krivo dogovore te tada počinju problemi i usporavanja u mreži.

Preporuka je kako se **`Auto Negotiation`** NE TREBA KORISTITI za međusobno spajanje preklopnika, usmjerivača, poslužitelja ili druge mrežne opreme.

Kabeli i konektori

Lokalnu (LAN) mrežnu opremu, povezujemo uglavnom s takozvanim *UTP* (Engl. *Unshielded twisted pair*) kablovima. Ovakva vrsta kabela sastoji se od četiri uvijene parice, što čini ukupno osam (8) vodova (žila).
Slika prikazuje RJ-45 utikač sa *zakrimpanim* (spojenim) UTP kabelom, spojenom prema ANSI/TIA/EIA 568B standardu:



Svaki vodič (žila kabela) je obojan posebnom bojom, pa tako unutar svake parice imamo slijedeće boje vodova:

- Narančasti i bijelo-narančasti (drugi par)
- Zeleni i bijelo-zeleni (treći par)
- Plavi i bijelo-plavi (prvi par)
- Smeđi i bijelo smeđi (četvrti par)

Kablovi se spajaju na posebne konektore s osam (8) nožica, znane kao **8P8C** (*8 position 8 contact*) naziva **RJ-45** (Engl *Registered Jack*) ili na utičnice odnosno na *patch panele* a koji na kraju opet imaju **RJ-45** utičnicu.

Tablica prikazuje shemu spajanja prema **ANSI/TIA/EIA 568 A** ili **B**:

Vodič	Pin prema 568 B	Pin prema 568 A
Bijelo-narančasta	1	3
Narančasta	2	6
Bijelo-zelena	3	1
Zelena	6	2
Plava	4	4
Bijelo-plava	5	5
Bijelo-smeđa	7	7
Smeđa	8	8

Što se boja tiče, razlika je samo u tome što su zamijenjene pozicije drugog (2) i trećeg (3) para/parice tj. narančaste i zelene.

Što se spajanja i upotrebe tiče (ako gledamo obije strane kabela/utikača) imamo slijedeće stanje.

Jedna strana kabela	Druga strana kabela	Opis
568 A	**568 A**	normalni kabel (znani kao *patch* ili *straight through* kabel)
568 B	**568 B**	normalni kabel (znani kao *patch* ili *straight through* kabel)
568 A	**568 B**	ukrižani odnosno ethernet *crossover* kabel

Dakle ako su obije strane kabela iste (A→A ili B→B) dobivamo klasični (obični) kabel. Ali ako izradimo kabel na kojemu su obije strane različite (A→B ili B→A) tada dobivamo ukrižani odnosno *crossover* kabel.

Ovakva vrsta kabel se obično koristi za povezivanje istih ili mrežnih uređaja - primjerice:

- *switch → switch*
- *router → router*
- *router → switch*
- *računalo 1 → računalo 2*

Kod ovoga kabela su direktno spojeni **TX** na **RX** pinove/signale. Konkretno :

- Pin 1 (**TX +**) → Pin 3 (**RX +**)
- Pin 2 (**TX -**) → Pin 6 (**RX -**)

To znači da je ono što se šalje (**TX**) s jedne strane, dolazi direktno na prijem (**RX**) s druge strane. Slična vrsta kabela postoji i kod serijskih kabela, koji se nazivaju i *null modem* kabeli.

Vratimo se kabelima

Svaka uvijena parica se u komunikaciji koristi za svoju namjenu:

- Za brzine **do 100 Mbps**: jedna parica je za slanje (*Transmit* - **TX**), a druga za primanje (*Receive* - **RX**) podataka (signala). Za brzine do 100 Mbps se koriste samo dvije od četiri parice (iako sve moraju biti uredno spojene prema standardima **ANSI/TIA/EIA 568A** ili **ANSI/TIA/EIA 568 B**). Tako imamo:
  - Pin 1 (**TX +**) i Pin 2 (**TX -**)
  - Pin 3 (**RX +**) i Pin 6 (**RX -**)
- Za brzine **od 1000 Mbps (1 Gbps) i više**, svaka parica se koristi za bidirekcionalnu (dvosmjernu) komunikaciju. Ovdje se koriste sve četiri parice. Tako imamo :
  - Pin 1 (bidirekcionalni par **A +**), Pin 2 (bidirekcionalni par **A -**)
  - Pin 3 (bidirekcionalni par **B +**), Pin 6 (bidirekcionalni par **B -**)
  - Pin 4 (bidirekcionalni par **C +**), Pin 5 (bidirekcionalni par **C -**)
  - Pin 7 (bidirekcionalni par **D +**), Pin 8 (bidirekcionalni par **D -**)

Mrežne kabele (i utičnice) dijelimo prema:

- **Kategoriji** (Kategorije danas u upotrebi su 5,6 i 7, uz razne podkategorije):
  - **Kategorija 5 (Cat 5)**, koja je danas minimalna kategorija u upotrebi i to podkategorija 5 (E), znana kao **CAT 5E** (omogućava brzine do 1 Gbps)
  - **Kategorija 6 (Cat 6)** ova kategorija je u upotrebi za brzine 1Gbps (duljine do 100.m.) i za 10Gbps (duljine do 50.m.)
    - podkategorija 6A (Cat 6A) - dvostruko boljih karakteristika od Cat 6 - za brzine 1Gbps (duljine do 100.m.) i za 10Gbps (duljine do 100.m.)
  - **Kategorija 7 (Cat 7)** - za brzine do 10 Gbps (duljine do 100.m.) i više.
    - podkategorija 7A (Cat 7A) - za brzine od 10 Gbps (duljine do 100.m.), 40 Gbps (duljine do 50.m.) i 100 Gbps (duljine do 15.m.)
- **Tipu kabela**:
  - UTP - neoklopljen, uvijene, četiri (4) parice. Ova vrsta kabela je podložna na razne elektromagnetske smetnje koje mogu uzrokovati probleme u radu mreže.
  - FTP - oklopljene s folijom (F - *Foil*), uvijene, četiri (4) parice. Zaštićeno od smetnji zbog zaštitne folije.
  - STP - oklopljene s opletom žica (S - *Shield*), uvijene, četiri (4) parice. Zaštićeno od smetnji zbog zaštitnog opleta žica.
  - SFTP oklopljene s opletom žica i s folijom (S i F), uvijene, četiri (4) parice. Zaštićeno od smetnji zbog zaštitne folije i opleta žica.

Navedene maksimalne duljine kabela se odnose za ukupnu duljinu kabela, od računala do utičnice, preko *patch panela* do druge utičnice te od te utičnice do mrežnog uređaja ili drugog računala. Dakle : uređaj 1 → *patch kabel* → *patch panel* → *patch kabel* → uređaj 2.
Svi navedeni S ili F tipovi imaju oplet ili foliju oko svih parica zajedno, na vanjskom dijelu kabela a ispod vanjskog zaštitnog izolacijskog materijala. Dodatno, ovisno o kategoriji moguće su izvedbe u kojima su i pojedine parice zasebno obavijene folijom.
Kod svih varijanti S ili F tipa (koje imaju ili oplet ili foliju), potrebno je uzemljenje istih, preko konektora i utičnice i (patch) panela.
Konektor (RJ-45) i utičnice i *patch paneli* moraju biti identične i kategorije i tipa, kao i kabel, inače može doći do problema u radu mreže .

Kada govorimo o standardima, slijedeći standardi se odnose na *Ethernet* komunikaciju preko (U)TP kabela odnosno bakrenih uvijenih parica:

- za rad 10 Mbps (**10BASE-T**) je zadužen standard **802.3i**
- za rad 100 Mbps (**100BASE-TX**) je zadužen standard **802.3u**
- za rad 1000 Mbps (1 Gbps) (**1000BASE-T**) je zadužen standard : **802.3ab**
- za rad 10 000 Mbps (10 Gbps) (**10GBASE-T**) je zadužen standard : **802.3an**

MDI i MDI-X

MDI (*Medium Dependent Interface*) opisuje specifični mrežni *interface* (bakar ili optika) na fizičkom sloju komunikacije. Isto tako standardima je definirana i mogućnost *interfacea* zvana **MDI-X** (*medium dependent interface crossover*). Ova mogućnost/sposobnost određenog *interfacea* se odnosi na unutarnju shemu spajanja odnosno rasporeda pinova (nožica/vodova).

Tako su *interfacei* odnosno *portovi* na preklopniku označeni sa *MDI-X* zapravo spojeni kao da imaju *crossover* unutar sebe tj. zamijenjeni su im *pinovi* za primanje (RX) i slanje (TX). Tako da bi spajanjem normalnog odnosno *patch* kabela na njima, zapravo dobili *crossover* spoj (kao da smo koristili *crossover* kabel).

U pravilu mrežne kartice na računalima i usmjerivačima (*routerima*) interno koriste MDI (dakle normalan raspored vodova), dok su mrežne kartice/portovi na *HUBovima* i preklopnicima (*switchevima*) izvedene u MDI-X varijanti. Možemo to gledati i ovako (kako se spajaju uređaji):

- Računalo (**MDI**) → **običan kabel** → *Switch* (**MDI-X**)
- Računalo 1 (**MDI**) → *crossover* kabel → Računalo 2 (**MDI**)
- *Switch 1* (**MDI-X**) → *crossover* kabel → *Switch 2* (**MDI-X**)

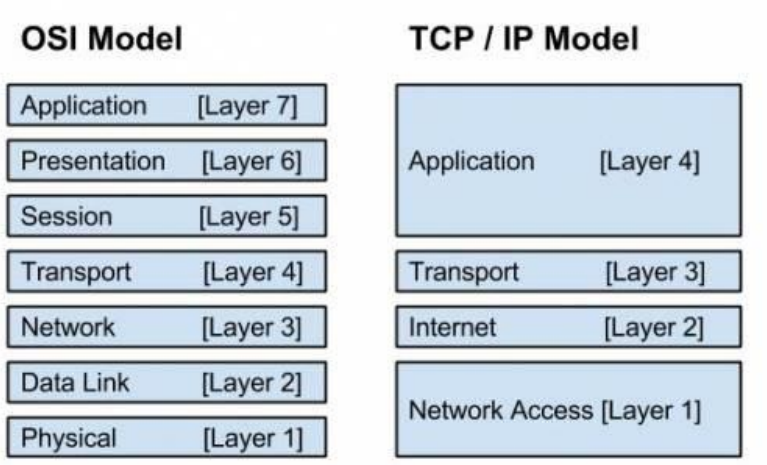
Auto MDI-X

Na svim novijim mrežnim uređajima ozbiljnijih proizvođača, koristi se i noviji standard koji se zove **Auto MDI-X**. Ovdje se radi o sposobnosti svakog *porta* na uređaju da prvo prepozna način spoja druge strane te se automatski prilagodi odnosno promjeni svoje interno kabliranje. Drugim riječima ovakvi uređaji nakon (ispravne) detekcije druge strane, sami sebe prebacuju iz *MDI* u *MDI-X* način unutarnjeg kabliranja. Ova detekcija je prilično brza (oko 500 ms) ali je u algoritam prepoznavanja dodan dodatni sigurnosni brojač, tako da se vrijeme detekcije i prebacivanja može povećati do 1.5 s. Naravno i ovo baš nisu svi proizvođači implementirali najbolje - pa stoga kod većine radi kako treba ali kod nekih baš i ne ili radi problematično.

OSI i TCP/IP modeli

Prije nego krenemo dalje, moramo se podsjetiti osnova mrežnih protokola. Dalje u tekstu ćemo se prisjetiti OSI i TCP/IP modela i protokola te načina komunikacije u mreži.

OSI model (Open Systems Interconnection model) je konceptualni slojeviti model mrežne komunikacije, koji opisuje funkcije i međusobnu komunikaciju između njegovih slojeva.Održava se od strane ISO organizacije (International Organization for Standardization), pod oznakom : **ISO/IEC 7498-1**.



Na osnovi **OSI** modela, razvijen je **TCP/IP** model mrežne komunikacije.

Naziv **TCP/IP** potječe od *Transmission Control Protocol* (TCP) i *Internet Protocol* (IP) kao prva dva protokola koja su bila razvijena u skupu TCP/IP protokola.

Današnji TCP/IP skup protokola sadrži cijeli niz raznih protokola (zbog toga govorimo o skupu ili grupi protokola naziva TCP/IP). Ovaj skup protokola često se naziva i **DoD** model zbog toga što je inicijalno razvijan od strane **DARPA** agencije (*Defense Advanced Research Projects Agency*) unutar Ministarstva obrane Sjedinjenih Američkih Država (*Department of Defense*), krajem 1960. i početkom 1970. godine. Tijekom narednih godina (od 1970) razvijale su se nove verzije : TCP v1, TCP v2, TCP v3 i IP v3, te TCP/IP v4 koji se koristi i danas (uz IP v6.).

TCP/IP je skup protokola nužan za komunikaciju računala. Današnji internet je baziran na TCP/IP skupu odnosno grupi protokola. On se sastoji od nekoliko dijelova odnosno slojeva od kojih svaki odrađuje određenu funkcionalnost te sve dalje predaje na obradu slijedećem sloju.
Na slici gore je vidljiva usporedba OSI i TCP/IP modela te može poslužiti kao primjer komunikacije TCP/IP protokola.

MAC adrese

MAC adresa (*Media Access Control*) je jedinstvena adresa, ugrađena u svaku mrežnu karticu, od strane proizvođača. Isto tako svaki port/utičnica na preklopniku (ili bilo kojem mrežnom uređaju) je isto mrežna kartica, koja također ima svoju MAC adresu.
MAC adresa je 48 bitna, označava se sa 12 heksadecimalnih brojeva, grupiranih po 2 i obično odvojenih sa **:** ili **-**. Prvih 6. hexadecimalnih brojeva označava proizvođača a ostalih 6 su redni brojevi.

Identifikator tvrtke proizvođaća			Redni broj proizvođača		
8 bita	8 bita	8 bita	8 bita	8 bita	8 bita
← 48 bita →					

Identifikator tvrtke proizvođaća			Redni broj proizvođaća		
00	60	2F	3A	FD	6C
Cisco			Redni broj		

Na osnovi MAC adrese možemo raspoznati proizvođaća same kartice (**Intel**, **Broadcom**, …).

Pogledajmo izgled MAC adrese: **`00-60-2F-3A-FD-6C`** (gore desno)

IP adrese

Prema TCP/IP protokolu, IP adresa je jedinstvena adresa, poput poštanske adrese. Prema tome, svako računalo na mreži, koje koristi TCP/IP protokol, mora imati svoju (jedinstvenu) IP adresu. U daljnjem tekstu, pričati ćemo samo o IPv4 protokolu.
Kako izgleda jedna IP (IPv4) adresa u dekadskom i binarnom sustavu:
Uzeti ćemo za primjer IP adresu : **172.17.100.18** , koja binarno izgleda ovako : **10101100.00010001.01100100.00010010**
Pogledajmo to u tablici, kako bi bilo malo jasnije:

Dekadski	172	17	100	18
Binarno	10101100	00010001	01100100	00010010
Broj bitova	8	8	8	8

Vidljivo je kako je IP adresa podijeljena u 4 segmenta od 8 bitova (4 x 8*Bytea*), odvojenih s točkom (**.**). Dakle IPv.4 adresa je 32 bitni broj (2^{32}), što omogućava adresiranje do: 4 294 967 296 IP adresa.

Klase IP adresa

IP adrese su podjeljene u točno definirane klase, od kojih svaka ima svoju (preporučenu) namjenu:

Klasa IP adresa	**Opseg (prvi oktet)**	**Namjena**
Klasa A	**0.0.0.0.** …do… **127.255.255.255**	Za ekstremno velike mreže
	127.	Rezervirano za loopback (lokalno računalo i sl.)
Klasa B	**128.0.0.0** …do… **191.255.255.255**	Za srednje do velikih mreža
Klasa C	**192.0.0.0** …do… **223.255.255.255**	Manje mreže
Klasa D	**224.0.0.0** …do… **239.255.255.255**	Multicast adrese
Klasa E	**240.0.0.0** …do… **255.255.255.255**	Rezervirana od IETF za istraživanje

Primjerice sve IP adrese koje počinju od 128.0.0.0 pa sve sve do 191.255.255.255 su adrese koje su unutar klase **B**.
Osim navedenih klasa, u novije vrijeme zatraženo je da se unutar klasa A,B i C rezerviraju određeni segmenti za privatne mreže (engl. *Private Networks*). prema RFC1918

Dodatne privatne adrese:

Klasa adresa	Opseg	Netmask	Max broj IP adresa
A	10.0.0.0 …do… 10.255.255.255	255.0.0.0	16.777.216
B	172.16.0.0 …do… 172.31.255.255	255.240.0.0	1.048.576
C	192.168.0.0 …do… 192.168.255.255	255.255.0.0	65.536

IP adrese unutar ovih okvira se smatraju privatnim IP adresama i standardno, za usmjerivače telekoma tj. *ISPova* (Engl. *Internet service provider*) nisu *rutabilne* odnosno javno dostupne. To znači kako primjerice, na kućnom usmjerivaču, kako bi izašli na "internet" morate koristiti neku od tehnika skrivanja odnosno preslikavanja, unutarnjih privatnih adresa, ma vanjsku(e) javnu IP adresu.

Pogledajte poglavlje **NAT**

Mreže i netmask

Korištenjem kombinacije IP adrese i ***Netmask*** adrese definiran je svaki mrežni element (računalo, poslužitelj, usmjerivač, preklopnik,…). Dakle svaka IP adresa, dolazi u paru s Tzv. *netmask* adresom koja se naziva maskom mreže.
Dodatno, svaka mreža ima svoj:

- **opseg IP adresa** (*ip range*) za upotrebu
- **IP adresu mreže** (*network IP*)
- **Broadcast IP adresu** (*broadcast IP*)
- **Netmask**

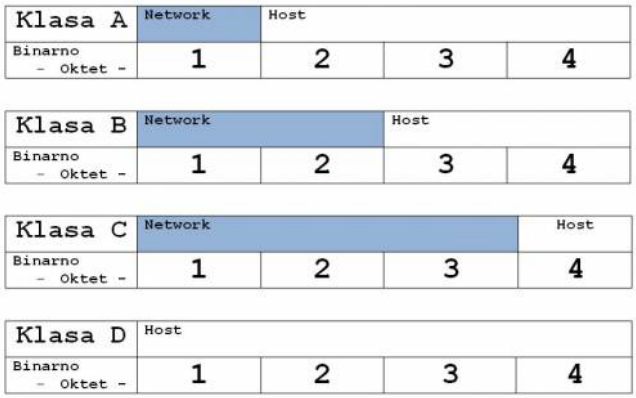
Netmask odnosno maska mreže je 32 bitni broj (2^{32}) koji uz pripadajuću IP adresu identificira svaki uređaj na mreži ali i samu mrežu. Maska mreže i njena pripadajuća IP adresa, govore nam na koji način će IP adresa biti interpretirana odnosno kojoj mreži svaka IP adresa pripada. Zbog potrebe podjela mreža na *podmreže* (engl. *Subnet*) i nadmreže (engl. *Supernets*) te samih klasa mreža, uz IP adresu se ***mora*** koristiti i maska mreže.
Na samom početku rada, kada naše računalo ili bilo koji uređaj na mreži, dobiju IP adresu i pripadajuću mu masku mreže, ono radi izračune, navedene dolje, kako bi izračunalo kojoj mreži pripada (prema IP adresi mreže) te koja je broadcast adresa te mreže. Ovo je važno u normalnoj komunikaciji unutar same lokalne (LAN) mreže, kao i u slučajevima komunikacije prema vanjskim mrežama. Naime kada recimo naše računalo želi komunicirati s drugim računalom u istoj mreži, a što je ustvrdilo pomoću izračunate IP adrese mreže, ono će odmah krenuti s komunikacijom. U slučaju kada mu IP adresa mreže govori kako sugovornik ne pripada njegovoj mreži, on paket šalje na IP adresu usmjerivača, koji će dalje znati kamo paket usmjeriti, nebi li došao do odredišta.

Dodatno, svaki element u mrežni (računalo, poslužitelj, usmjerivač, preklopnik, mrežni štampač i sl.) prihvaća samo one mrežne pakete koji su namijenjeni njegovoj mreži - ponovno na osnovi izračunate IP adrese mreže. Ovo je važno i stoga što neki mrežni protokoli u komunikaciji na IP sloju, koriste broadcast metodu komunikacije. U ovakvoj komunikaciji odredišna je IP adresa na koju se šalje paket, broadcast IP adresa mreže, koja se isto izračunava na osnovi maske mreže i IP adrese, a koju moraju primiti sva računala unutar određene IP mreže, koja znaju da je to njihova broadcast IP adresa mreže.
U slučaju kada mrežni paket dođe do usmjerivača, on za svaki paket, koji mu dolazi na mrežno sučelje, provjerava, kojoj IP adresi mreže pripada. Kako usmjerivač ima svoju tablicu usmjeravanja, u kojoj se nalaze i parovi IP adresa mreže i njen par - maska mreže, te mrežno sučelje kojemu pripadaju, on na osnovi IP adrese paketa koji mu je došao (odredišna IP), jasno zna gdje će svaki taj paket i usmjeriti.

Pogledajmo maske mreža, za različite klase mreža.

Klasa mreže	Maska mreže
Klasa **A**	255.0.0.0
Klasa **B**	255.255.0.0
Klasa **C**	255.255.255.0

Na slici dolje je vidljiva podjela na 4 okteta od kojih svaki ima 8 bitova (slično kao IP adrese). Bitovi s lijeve strane (plavo) označavaju mrežu (*Network*) a s desne strane identificiraju računalo (*Host*) na mreži.



Maska mreže se logički sastoji od dva dijela:

- mrežnog djela (*network* dio) - lijevo (plavo)
- djela za računala (*host* dio) - desni dio

U normalnoj upotrebi, maska mreže se često određuje tako da prvo moramo saznati kojoj klasi pripada naša IP adresa. U slučaju korištenja podmreža i nadmreža, potrebno je izračunati, koliko podmreža želimo dobiti od jedne mreže, te koliko velike podmreže uopće želimo.
Dalje u tekstu, govoriti ćemo o "standardnim" mrežama koje nisu podjeljene u podmreže, odnosno imati će standardnu masku mreže, prema njihovoj klasi mreže iz koje dolaze.

Mrežni dio maske mreže
Ovaj dio (prvi dio maske mreže, s lijeve strane), opisuje mrežu. Popunimo sve jedinice s lijeve strane, dok ne dođemo do naše mreže. Primjerice. maska mreže za klasu *C* je 255.255.255.0 tj. prva tri okteta su sve jedinice - ukupno 24 bita za ovaj primjer. Kod drugog čestog načina označavanja maske mreže se prebrojavaju bitovi (1) od lijevo na desno, pa se gore navedena maska označava i kao **/24.**
Pr. za našu IP adresa 198.150.11.1 možemo reći slijedeće :

- Ona je iz klase **C** - spada u opseg IP adresa, od 192.0.0.0 do 223.255.255.255
- Prema tablici gore za **C** klasu mreže vidimo da `host` dio u 4-tom oktetu čine nule - gledano s desna na lijevo.
- Njena maska mreže, će prema tome biti : 255.255.255.0 → Pogledajte mrežni dio na tablici gore.

Naime popunili smo sve jedinice binarno od lijeva na desno, prva tri okteta (8 bitova):
11111111.11111111.11111111.00000000 i dobili dekadski : 255.255.255.0.

Mrežni dio maske mreže se popunjava jedinicama od lijevo prema desno i to:

- prema definiciji klasa mreža ili
- prema potrebama naše mreže
- prema potrebama podmreže (*subnet*) ili nadmreže (*supernet*) - **samo ako ih koristimo**

Pogledajmo to ovako - mrežni dio maske mreže se kreće:

- od klase **A** mreže - od 255.0.0.0 : binarno : **11111111**.00000000.00000000.00000000
  - dakle prve podmreže kreću od : 11111111.**1**0000000.00000000.00000000
  - te ih možemo povećavati, dodavanjem `1` od lijevo prema desno, pa bi slijedeće podmreža bila : 11111111.**11**000000.00000000.00000000 i tako dalje sve dok ne dođemo do slijedeće klase.
- potom dolazimo do klase **B** koja kreće od 255.255.0.0 : binarno : **11111111**.**11111111**.00000000.00000000
  - a koja se proširuje na isti način s jedinicama, do klase **C**
- preko klase **C** mreže - od 255.255.255.0 : binarno : **11111111**.**11111111**.**11111111**.00000000
- i dalje do realno iskoristive zadnje maske : 255.255.255.248 : binarno: **11111111**.**11111111**.**11111111**.**11111**000

Host dio maske mreže

Nule na kraju, odnosno prebrojavanjem `0` od desno na lijevo (dok ne dođemo do jedinica) daje nam naš `Host` dio, unutar kojega možemo koristiti IP adrese. Dakle koliko bitova nula imamo u masci mreže, od desno na lijevo, toliko bitova za upotrebu, imamo na istoj poziciji, gledajući na našu IP adresu mreže.

S navedenom maskom mreže, za naš slučaj vidimo kako je cijeli zadnji oktet (8 puta po 0 tj. 2^8=256) slobodan za računala, što je 256 adresa (0 do 255 uključujući nulu).
Dakle : 11111111.11111111.11111111.**00000000** (sada gledamo nule za host dio)
Ne zaboravimo kako u svakoj mreži postoje i dvije dodatne IP adrese :

- Prva moguća IP adresa je rezervirana za mrežu (IP adresa mreže)
- Zadnja moguća IP adresa je rezervirana za *Broadcast* IP adresu

Nama zbog navedenog, u svakoj mreži ostaje ukupan broj IP adresa umanjen za ove dvije IP adrese.
U našem slučaju to su: 256 - 2 = 254 upotrebljive adrese.

Kako se izračunava IP adresa mreže ?

Podsjetimo se `AND` logičke operacije :

Ulaz A	Ulaz B	Rezultat
0	0	0
0	1	0
1	0	0
1	1	1

Uzmimo IP adresu: 198.150.11.1 i njenu masku mreže 255.255.255.0 11000110

Dekadski	198	150	11	1
Binarno	11000110	10010110	00001011	00000001

i njenu pripadajući Netmask adresu:

Dekadski	255	255	255	0
Binarno	11111111	11111111	11111111	00000000

Sada se radi logička **I** operacija (*AND*), sa IP adresom i pripadajućom maskom mreže:

IP adresa	11000110	10010110	00001011	00000001
Netmask	11111111	11111111	11111111	00000000
Rezultat AND operacije - binarno	11000110	10010110	00001011	00000000
Rezultat AND operacije - dekadski	198	150	11	0

Dobili smo kao rezultat : **198.150.11.0** , dakle ovo je IP adresa mreže, koja nije *subnetirana*. Rekli smo kako je IP adresa mreže, nulta (odnosno prva IP adresa koju ne smijemo i ne možemo koristiti), te kako, nakon nje slijedi niz upotrebljivih IP adresa, te je zadnja IP adresa unutar te mreže, *broadcast* IP adresa, koju isto ne smijemo koristiti.

Broadcast adresa mreže se izračunava, pomoću inverzne maske mreže, naše maske mreže te IP drese mreže, ali za sada toliko - kako ne bi izašli iz okvira “Kratke priče”.

Pogledajmo primjer.
Uzmimo jednu mrežu **C** klase, kao primjer:
Unutar velike mreže **C** klase, odabrati ćemo dvije manje mreže, s time da obije imaju standardni ***netmask*** *255.255.255.0*, za mrežu klase **C**:

```
Mreža 1: 198.150.11.*, koja se označava sa 198.150.11.0
Mreža 2: 198.150.12.*, koja se označava sa 198.150.12.0
```

Pogledajmo što te mreže sadrže:

Adresa mreže (Network Address)	**Opseg adresa za upotrebu**	**Broadcast adresa za (ovu) mrežu**
198.150.11.0	198.150.11.1 …do… 198.150.11.254	198.150.11.255
198.150.12.0	198.150.12.1 …do… 198.150.12.254	198.150.12.255

Dakle mreža **1**, ima IP adresu 198.150.11.0 (ova IP adresa označava cijelu ovu mrežu).
Unutar te mreže svim računalima, poslužiteljima i ostalim uređajima kojima ćemo dodijeliti neku IP adresu, IP adresu možemo dodijeliti iz opsega IP adresa od: 198.150.11.1 sve do 198.150.11.254. a na kraju našeg opsega IP adresa se nalazi **broadcast** IP adresa (koja je rezervirana i nju ne možemo koristiti), u ovom slučaju to je adresa : 198.150.11.255 (prethodno smo ju izračunali). Isto pravilo vrijedi i za mrežu **2** ali i druge mreže s istom mrežnom maskom.

Za više detalja o IP adresama, mrežama i netmasku, pogledajte (**64**)

TCP/IP portovi

Prema **TCP/IP** protokolu, definirani su i Tzv. *portovi*. Svaki port definira točno određenu vrstu komunikacijskog protokola odnosno standarda za komunikaciju. **TCP/IP** protokol, dopušta maksimalno 65535 *portova* [2^{16}] (16 bitni broj). Za svaki TCP/IP protokol je definiran i *port* preko kojega se koristi.

Dodatno, pomoću *portova*, svako računalo može (u svakom trenutku) paralelno komunicirati s više protokola, jer svaki od njih koristi drugačiji *port*. Neki od osnovnih *portova* odnosno protokola koji ih koriste, nabrojani su u tablici.

Broj porta	Naziv	Opis
20	FTP Data	File Transfer Protocol za podatke
21	FTP	File Transfer Protocol za kontrolu
22	SSH, SFTP i SCP	Secure Shell : udaljeni pristup shell-u i Secure File Transfer Protocol te Secure Copy Protocol
23	TELNET	Telnet udaljeni pristup
25	SMTP	Simple Mail Transport Protocol
53	DNS	Domain Naime Server protokol
67	BOOTP/DHCP	BOOTP i DHCP Protokoli
69	TFTP	Trivial FTP Protokol
80	HTTP	Hyper Text Transfer Protocol
123	NTP	Network Time Protocol
443	HTTPS	Hypertext Transfer Protocol preko zaštićenog TLS/SSL protokola/kanala

U **TCP/IP** mrežnoj komunikaciji, za svaki mrežni paket, potrebno je definirati izvorišni port (*Source port*) te odredišni port (*Destination port*). Pri tome izvorišni port (source) identificira komunikacijski kanal na računalu koje se spaja na mrežni servis poslužitelja a odredišni port (destination) definira servis na poslužitelju odnosno protokol za komunikaciju (Pr. FTP, HTTP, DNS, …).

Ako vas zanima malo veći popis poznatijih *portova* pogledajte na Wikipedia članak

Naime u transportnom sloju (OSI sloj 3 i 4) se osim drugih stvari dodaju i *portovi* i to *Source port* (izvorni) i *Destination port* (odredišni), koji definiraju protokol kojim će se odvijati komunikacija između aplikacija.

Osim portova važno je i kojim točno transportnim protokolom se koristi viši protokol. Od transportnih protokola unutar TCP/IP seta nalaze se **TCP** i **UDP** protokoli. Oni su zaduženi za sam prijenos paketa preko mreže.

Transportni protokoli

Transportni protokoli, zaduženi su za prijenos podataka preko mreže, prema TCP/IP protokolu. Postoje ih dvije vrste :

- Pouzdani, odnosno oni koji se brinu o tome da li je svaki poslani paket i primljen na odredištu
- i oni manje pouzdani, koji se brinu samo o tome da je svaki paket uredno poslan, bez informacije o tome da li je na odredištu i uredno zaprimljen.

UDP

UDP (User Datagram Protocol) transportni protokol koristi jednostavan ali vrlo brzi sustav za prijenos podataka bez provjere o uspješnom primanju ili integritetu podataka poslanih unutar mrežnih paketa. *UDP* uglavnom prepušta sve provjere na stranu same aplikacije koja šalje ili prima podatke.

Dakle UDP je nepouzdan ali vrlo brz. Želite li pouzdanost a morate ili želite koristiti UDP, sve zaštitne mehanizme morate ugraditi u vašu aplikaciju. Ako ipak želite samo brzinu onda je UDP pravi izbor. UDP se često koristi kada imamo komunikaciju koja je vremenski osjetljiva a nije nam previše bitno da li će se koji paket usput izgubiti u prijenosu.

Primjer ovakve upotrebe je Audio ili Video komunikacija, gdje nam nije bitno ako se izgubi koji dio slike na trenutak ili djelić glasa ali nam je bitno da nema vremenskog kašnjenja.

Osim ove primjene, koristi se i kod drugih tipova komunikacije kao što su razni protokoli za logiranje određenih poruka (spremanje sistemskih poruka sa raznih uređaja na centralnu lokaciju), kada nam nije najbitnije ako se jedna mala poruka izgubi i sl. Još jedan primjer su DNS (*Domain Name Service*) protokol ili NTP (*Network Time Protocol*) protokol, koji ako ne dobiju odgovor u nekom vremenu, poruku će poslati ponovno. Koriste ih i DHCP (*Dynamic Host Configuration Protocol*), TFTP (*Trivial FTP*) i mnogi drugi protokoli.

Zanimljivo je što unutar svakog paketa u UDP zaglavlju postoji polje unutar kojeg se izračunava i upisuje provjerni zbroj, koji se može koristiti za provjeru integriteta mrežnog paketa, točnije, samo UDP zaglavlja a ne i samih podataka. Problem je u tome što je to opcija, koja se obično ne koristi (ovisno o aplikacijskom protokolu).

TCP

TCP je tkzv. *connection-oriented* transportni protokol, protokol koji se brine za stanje veze, kao i za integritet podataka. To znači kako se kod komunikacije dva računala, otvara konekcija pomoću tkzv. metode *Three-way handshake* i TCP protokol se brine da je svaki segment podataka koji je poslan sa računala **A** na računalo **B** primljen u ispravnom (nepromijenjenom) obliku te pravilnim redosljedom.

Za svaki paket koji je s druge strane zaprimljen neispravan, zatraži se *retransmisija* i paket se šalje ponovno, sve dok nije potpuno ispravan.

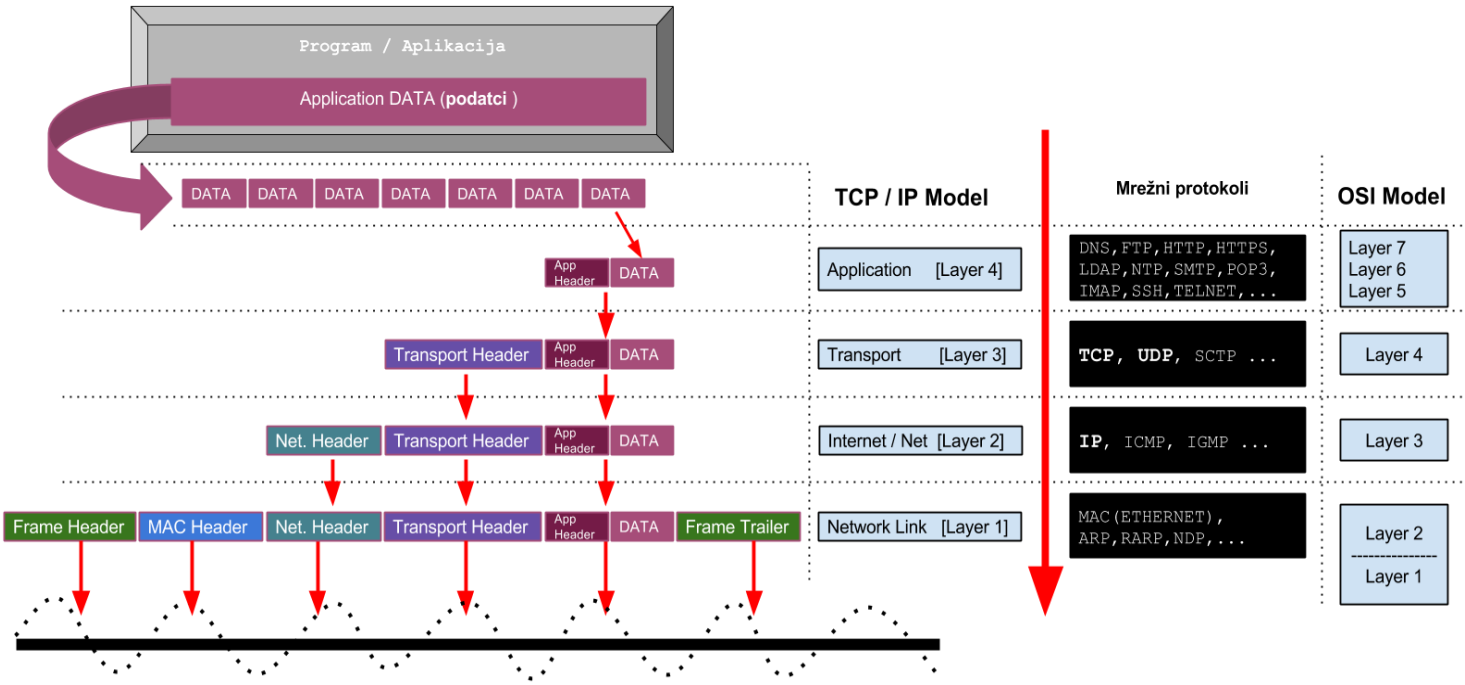
Kako bi se uopće mogla provjeriti ispravnost (integritet) podataka (RFC 793), unutar svakog paketa, u TCP zaglavlju postoji polje za koje se izračunava i upisuje provjerni broj. Dakle ako unutar TCP zaglavlja dođe do grešaka u prijenosu, to će biti detektirano.

Kod zatvaranja veze koristi se metoda dogovaranja oko zatvaranja, slična kao i kod otvaranja konekcije.

U prijenosu podataka kroz medij (kabel, optičko vlakno, zrak/vakuum) može doći i dolazi, do gubitka djela podataka.

Enkapsulacija

Sada kada smo se upoznali s **OSI** i **TCP/IP** modelom i njegovim slojevima, pogledajmo detaljnije što se događa kod **TCP/IP** komunikacije. Vidjeti ćemo kako se podaci iz naše aplikacije (programa) razdvajaju na manje cjeline te šalju pojedinim **TCP/IP** slojevima (od gore prema dolje). Biti će nam jasnije i s čime se bavi koji od slojeva te što sve nadodaje, sve do najnižeg sloja koji sve to (pakete) šalje na mrežu.
Slika prikazuje *enkapsulaciju* (ugniježđenje) prema TCP/IP protokolu, sa odredišnog računala. Ovdje će vam postati malo jasniji tok podataka, odnosno slojevita obrada podataka, kako prolaze kroz TCP/IP slojeve ali i veza između TCP/IP slojeva u odnosu prema OSI modelu i njegovim slojevima.



Zamislimo slijedeći primjer: Naš Web preglednik se želi spojiti na stranicu, na koju upisujemo korisničko ime i lozinku. Promatrati ćemo samo dio komunikacije u kojoj se obrađuju unešeno korisničko ime i lozinka.

Sada naš TCP/IP **protokol uskače u pomoć.**

Aplikacijski sloj uzima podatke iz same aplikacije - u našem slučaju naše korisničko ime i lozinku, te ih razdvaja na manje dijelove (*DATA*), a koji mogu stati u budući mrežni paket, i to u formatu koji je razumljiv višem sloju - konkretno aplikaciji. Ti naši podaci (ovdje su naše korisničko ime i lozinka koje smo upisali) su na slici prikazani kao [*DATA*] .

Transportni sloj - transportni sloj preuzima te podatake od aplikacijskog sloja te dodaje svoj dio [*Transport Header*] – ovdje se dodaju i *Source* i *Destination portovi* i paketi označavaju rednim brojevima (ako je u pitanju TCP) te se dodaju i ostale stvari : ovisno da li se za transport koriste **TCP** ili **UDP**. Između ostaloga, za TCP se izračunava i dodaje *TCP cheksum* odnosno provjerni zbroj. Na kraju se sve zajedno sa svime iz prethodnog sloja (Aplikacijskog) prosljeđuje slijedećem sloju. Prema OSI modelu ovo je **Layer 4**

Source port se nasumično odabire (1024 + Random [slučajni broj]) a **Destination port** označava protokol za komunikaciju (pogledajte tablicu sa portovima).

Internet sloj preuzima paket od *transportnog sloja* i dodaje svoj dio, koji sadrži i *Source IP* adresu (pošiljatelja), *destination IP* adresu (primatelja) ali i *IPv4 header checksum* odnosno jednostavni provjerni zbroj ali se izračunava SAMO za zaglavlje a ne i za podatke (NE za DATA dio) i to znatno jednostavnijim algoritmom nego u nižem sloju (dolje). Potom se dalje sve (što sadrži ono što su dodali prethodni slojevi i ovaj sloj), prosljeđuje slijedećem sloju.Prema OSI modelu ovo je *Layer 3*

Network Access/Link sloj preuzima paket od gornjeg sloja te dodaje i *Source* i *Destination MAC* adrese (dakle MAC adrese mrežne kartice ovog računala i računala na koje se šalje paket. Na ovom sloju se izračunava provjerni zbroj za provjeru integriteta *svih* podataka, pomoću *CRC* algoritma, te se zapisuje u svoje *FCS* polje. Prema OSI modelu ovo je *Layer 2*. Tada se sve preuzeto od prijašnjeg sloja i onoga što je ovaj sloj dodao, šalje na najniži sloj, koji dodaje svoj dio (svoja polja) te sve šalje na mrežnu karticu. Najniži sloj je OSI sloj 1.
Nadalje se sve pretvara u signale za slanje na mrežni medij :

- *bakar* preko **RJ-45** utičnice odnosno u električne signale ili
- *optiku* preko **LC**, **SC**, **ST** ili sličnih konektora odnosno u svjetlosne impulse.

Ovaj proces se na računalu koje prima podatke, odvija u suprotnom smjeru. Odnosno *Web* poslužitelj iz primjera, na koji smo se spajali, prima s mrežnog medija signale koji se pretvaraju u podatke, koje preuzima **Network Access/Link** (*OSI Layer 2*) sloj koji provjerava svoj dio i provjerava integritet podataka pomoću novog *CRC* izračuna i usporedbe s onim *CRC* zapisom koji se nalazi unutar mrežnog okvira (paketa).

Potom se skida ovaj sloj i prosljeđuje **Internet sloju** (*OSI Layer 3*) koji provjerava svoj dio, te radi svoj provjerni zbroj i ako je sve u redu, skida svoje dijelove i prosljeđuje sve **Transportnom sloju** (*OSI Layer 4*) koji provjerava svoj dio: ako je za transport bio korišten TCP, provjerava se ispravan redoslijed primljenih podataka i provjerni broj (*TCP checksum*).

Ako je sve u redu, skida se dio koji je od ovog sloja i prosljeđuje
Aplikacijskom sloju uz naznaku broja porta (jer on naznačava aplikacijski protokol i na kraju krajeva aplikaciju).
Aplikacijski sloj izvlači podatke i prosljeđuje ih našoj aplikaciji.

Ako je bilo više podataka nego li ih stane u jedan mrežni paket, obrađuju se i primaju svi ostali paketi iz kojih se izvlače podaci. Svi podaci iz više paketa se tada spajaju u konačan oblik podataka koji u ovom slučaju dolaze do Web poslužitelja koji ih onda prima - u našem slučaju su to korisničko ime i lozinka.

Cijelo vrijeme govorimo kojem **OSI** sloju pripada koja komunikacija, iako koristimo **TCP/IP** protokol. To je stoga što se sve uspoređuje prema **OSI** modelu, pa ćete tako i kod karakteristika raznih uređaja vidjeti upravo oznake za **OSI** slojeve (*Layer*).

Sada smo spremni upoznati se s mrežnim uređajima.

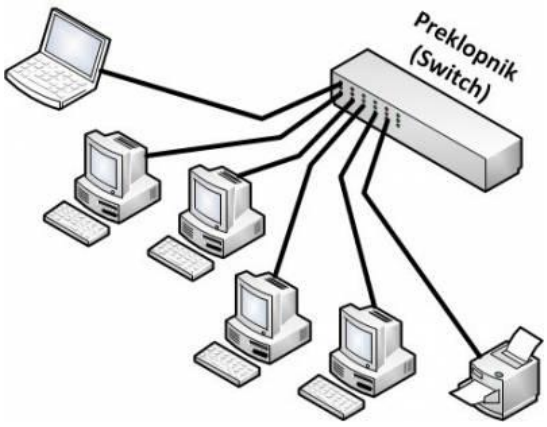
I konačno preklopnici

Preklopnik je mrežni uređaj koji spaja računala ili druge mrežne uređaje na mrežu. Dakle sva računala i mrežni uređaji, spajaju se na preklopnik kao na centralno mjesto u mreži.Ovakav spoj se prema topologiji naziva i *zvjezdastim* spojem, odnosno spojem u *zvijezdu*. Način spajanja je isti kao i na *HUBove* iako je sam rad preklopnika potpuno drugačiji.

Kao što smo spomenuli, mrežna komunikacija se odvija pomoću mrežnih paketa. Dakle podaci koje naša aplikacija šalje drugom računalu na mreži, odnosno njegovoj aplikaciji, razlamaju se na male dijelove, koji se, jedan po jedan, šalju TCP/IP slojevima unutar našeg operacijskog sustava.

Ovi TCP/IP slojevi, dodaju redom, svaki svoj dio, sve dok ne dođu do najnižeg sloja, koji sve to pretvara u električne signale, koje šalje na mrežu. Dakle naši podaci su razlomljeni u male dijelove, koji se šalju mrežom do odredišta, gdje se ponovno sastavljaju i šalju odredišnoj aplikaciji te se tamo formiraju kao prvobitno poslani podaci.

Računala i drugi mrežni uređaji, se na preklopnik spajaju u hijerarhijskom zvjezdastom spoju - dakle svi se spajaju na preklopnik kao centralnu točku, kao na slici:



Preklopnici (*switchevi*) su uređaji koji imaju više *portova* odnosno mrežnih utičnica (obično RJ-45), na koje se spajaju računala ili drugi mrežni uređaji (koje obično spajamo UTP kabelom).

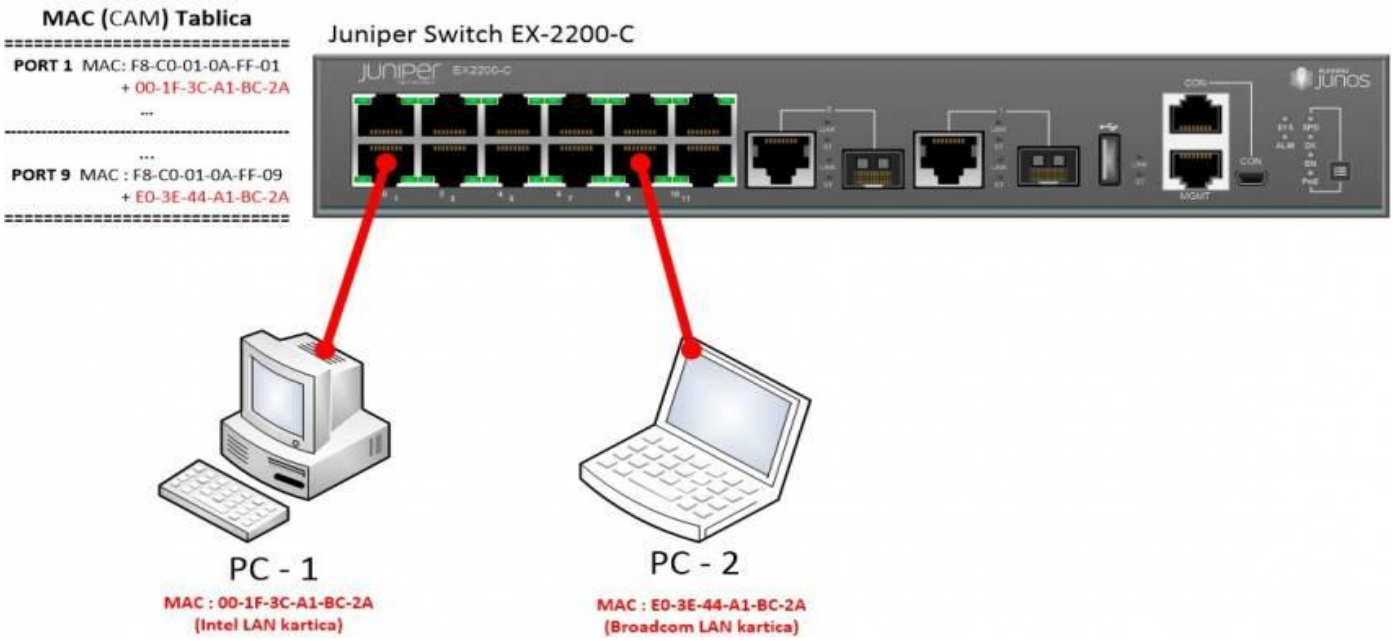
Oni rade tako što za svaki primljeni mrežni paket, koji dođe do njih (na svaku njihovu pojedinu utičnicu/port) provjeravaju MAC adrese, i to :

- izvorišnu (*source*) MAC adresu i
- odredišnu (*destination*) MAC adresu.

Dakle oni svaki mrežni TCP/IP paket gledaju samo do razine OSI sloja 2 (*Layer 2*).

Slika dolje prikazuje proces u kojemu su dva računala spojena na preklopnik (*switch*). Na slici je preklopnik tvrtke

Juniper, model **EX-2200-C**.



Svako računalo ima svoju mrežnu (*LAN*) karticu, a svaka LAN kartica ima svoju MAC adresu. Zamislimo ovakav scenarij, s dva računala spojena na preklopnik:

- prvo računalo (**PC-1**) ima mrežnu karticu tvrtke **Intel**, MAC adrese : `00-1F-3C-A1-BC-2A`
- drugo računalo (**PC-2**) ima mrežnu karticu tvrtke **Broadcom**, MAC adrese : `E0-3E-44-A1-BC-2A`

Isto tako i sam preklopnik ima po jednu MAC adresu, na svakom svom portu, pa tako ima :

- na portu 1, MAC : `F8-C0-01-0A-FF-01` te na
- portu 9, MAC : `F8-C0-01-0A-FF-09`

PC-1 je spojen na preklopnik na port 1 a PC-2 je spojen na port 9.

Kada se računala **PC-1** i **PC-2** uključe, preklopnik od njih prima prvi mrežni paket, iz kojega izvlači izvorišnu (*source*) MAC adresu te ju sprema u svoju *MAC* tablicu, koja se još često zove i **CAM** tablica.

Nakon primljenog prvog paketa s oba računala, preklopnik sada ima slijedeću tablicu:

Port na preklopniku	MAC adresa spojenog uređaja
1	00-1F-3C-A1-BC-2A
9	E0-3E-44-A1-BC-2A

Spajanjem svakog novog uređaja, pri slanju prvog mrežnog paketa, preklopnik gradi novu tablicu, te nakon nekog vremena ima izgrađenu cijelu tablicu, svih spojenih uređaja i *portova* na koje su spojeni.

Ova tablica se koristi kao tablica za preklapanje odnosno prosljeđivanje paketa. Naime ako računalo **PC-1** šalje jedan mrežni paket na računalo **PC-2** - izvorišna MAC adresa paketa je **00-1F-3C-A1-BC-2A** a odredišna MAC adresa je **E0-3E-44-A1-BC-2A** (što označava **PC-2**), tada preklopnik taj paket šalje sa svog **porta 1** na **port 9**. Ako je slijedeći paket, paket koji šalje računalo PC-2 , prema računalu PC-1. Ponovno se provjeravaju MAC adrese, te će taj paket biti proslijeđen (preklopljen) sa **porta 9** na **port 1**.

Ova operacija se događa za svaki pojedini paket koji dođe na bilo koji *port* na preklopniku. Nadalje preklopnik mora biti u stanju sve ih obraditi i to paralelno jer se često događa više istovremenih komunikacija, za razne pakete. Dakle vrlo vjerojatno istovremeno komunicira više računala, spojenih na različite portove i to u različitim smjerovima, a moguće i različitim brzinama,

Na gigabitnim brzinama (*1 Gbps*), (ovisno o veličini mrežnih paketa) ali za pakete od 64*byte*a, preklopnik bi MORAO biti u mogućnosti obraditi **1,488,095** (1.48 milijuna) paketa u sekundi, samo na jednom gigabitnom portu.

Dakle ako preklopnik ima 12 gigabitnih portova (12 x 1 Gbps), tada bi morao imati dovoljnu snagu obrađivati konstantno 17.76 milijuna paketa u sekundi, da ne bi došlo do zastoja u radu mreže. Važno je samo znati da je ovo hardverski i softverski vrlo zahtjevno. (o ovome malo kasnije)

Pogled na preklopnik iz druge perspektive

Preklopnici i usmjerivači su vrlo slični računalu kojemu je svako mrežno sučelje (port/mrežni ulaz/utičnica) praktično, jedna mrežna kartica.

A mrežnih kartica imamo svakakvih - Od običnih *desktop* (poput ove na prvoj slici dolje): Mrežna kartica (slika) je u vlasništvu tvrtke **Intel**

Do posebnih kategorija mrežnih kartica koje možemo nazvati *poslužiteljskim* poput ove na drugoj slici dolje. Mrežna kartica (slika) je u vlasništvu tvrtke **Intel**

Osim toga i *poslužiteljske* kartice nisu sve iste, kao niti njihovi upravljački programi, pa i neke od njih možda nisu loše ali imaju dosta loše upravljačke programe.

Neke odrađuju samo standardne stvari dok neke od snažnijih podržavaju cijeli niz dodatnih funkcionalnosti, kojima rasterećuju centralni procesor (CPU).

One snažnije *poslužiteljske*, danas standardno same odrađuju neke od sljedećih funkcionalnosti[o] ovome detaljnije nešto kasnije:

- *TCP Offload* (*Checksum/Large send*), UDP
- 802.1Q
- 802.1p (QoS)
- 802.3ad, *Fast Ether Channel* i *Gigabit EC*
- 802.3* (z, ab, u ,x) - *flow controll*
- Kriptiranje/Dekriptiranje
- I/O Virtualizaciju (u kombinaciji sa SR-IOV), …

U čemu su sve razlike :

- U cijeni
- Brzini
- Podržanim funkcionalnostima
- Upravljačkim programima (*driverima*) i njihovoj stabilnosti i brzini (loš driver = loša mreža)
- Podrški
- Dokumentaciji

Sada nam postaje jasno kako uopće nije nevažno koju mrežnu karticu treba odabrati ako želimo siguran, pouzdan i brz rad računala ili poslužitelja na mreži. Isto vrijedi i kod odabira proizvođača, koji izrađuju preklopnike. Razlike u praksi znaju biti drastične.
Ako govorimo o mrežnim karticama za računala, postoje razlike u cijeni od nekoliko desetaka ili stotina kuna a nekada i tisuća KN, te razlike u stabilnosti odnosno nestabilnosti te do brzine rada.
Vrlo se često događa, kako ste kupili sve mrežne komponente, kao i pasivni dio mreže (kablovi, utičnice, *patch paneli*), koje prema standardima podržavaju brzinu deklariranu kao **1Gbps** ali u praksi vam je brzina nešto malo veća od **100Mbps.**

Što mislite se zbog čega ?!.

Koliko je važan OS i softver odnosno “firmware” preklopnika

Ako gledamo s točke proizvođača ili osobe zadužene za razvoj novog preklopnika postoji još nešto što je vrlo važno. Naime softver kao i sam operacijski sustav te svi upravljački programi (za hardverske komponente preklopnika), odnosno njegov odabir i sam razvoj, drastično utiču na pouzdanost, izdržljivost i stabilnost.
Zapravo ovaj utjecaj je drastično veći nego li je to slučaj kod našeg osobnog računala ili nekog poslužitelja (o ovome detaljnije u kasnijim poglavljima).
Ovdje se radi o uređajima koji moraju raditi besprijekorno, bez obzira koliko ih puta palili ili gasili (zbog nestanaka struje, uslijed nadogradnje ili sl.), bez obzira koliko milijuna mrežnih paketa u sekundi morali obraditi te bez obzira koliko se zapravo desetaka mrežnih protokola u svim mogućim ili nemogućim kombinacijama “vrtili” na vašem preklopniku.
Dakle nije isto je li u određenoj konfiguraciji preklopnika, konfiguriran protokol A, B i C ili neki četvrti. Ili su u upotrebi samo protokoli A, C, i D.
Kako sve ne bi ostalo samo na teoriji, pogledajmo službene ***Release Note*** dokumente za OS (*Firmware*) preklopnika tri poznatija proizvođača. Izdvojiti ću samo po nekoliko bisernih grešaka.
Uglavnom ih proizvođači otklone – samo je pitanje vremena : kod nekih je to pitanje tjedana, kod drugih mjeseci a kod trećih se na otklanjanje nekih grešaka za koje proizvođači znaju i priznaju ih, čeka i godinama (vjerovali ili ne).
“Pa da ne bude nisam rekao – a neke greške se pojavljuju u novijim verzijama, nakon ispravljenih nekih starih grešaka”
Kako to izgleda u praksi:

Proizvođač A:

v3.10.012

(DI20130717000008)
8. DGS-3120-24SC will auto reboot if user uses SNMP to set "swL2PortCtrlNwayState" under DGS3120-24SC-L2MGMT-MIB tree. **(DRU20130723000003)**
9. The device did not send the traps when OAM detected error frame. **(DRU20130917000001)**
10. The CPU utilization may become high after power on/off device 30~200 times. It will cause ping loss and Web-UI response will be extremely slow or no response. **(DUSA20131003000001)**
11. User cannot set bandwidth control by TX/RX rate on stacking member through Web UI. **(DEUR20131007000004)**
12. When enables DHCP relay and STP at the same time, DGS-3120 will flood the DHCP ACK or NAK packets to all ports and causes DHCP broadcast storm

Greška Br.10 je primjerice super : nakon 30 do 200 restartanja uređaja, počinju se gubiti (ping) paketi.

Ili Br. 12 kada se uključi *DHCP Relay* i *Spanning Tree Protokol* (STP) u isto vrijeme, preklopnik preplavljuje (*Flood*a) mrežu uzrokujući *Broadcast* oluju i to na svim portovima na preklopniku (šalje **DHCP ACK** ili **DHCP NAK**).

Proizvođač B:

What's New in Version 3.0.0.43

New Software Features

- None

Resolved Issues

- **DNS:** DNS behavior was previously inconsistent and is now corrected. (CQ130199 ATI285)
- **IGMP ACL:** IGMP ACL was not applied properly and this behavior is now corrected. (CQ 123252 ATI256)
- **Forwarding Traffic:** After a certain time of operation, switch stops forwarding traffic on some ports – and does not provide link up/down indications for those ports. This issue was previously resolved in Version 3.0.0.41, but was not documented. (CQ 123741 ATI 249)

Greška pod “**Forwarding Traffic**” nakon nekog vremena preklopnik na neke portove jednostavno više ne šalje nikakav mrežni promet (pakete).

Proizvođač C :

Version PK.1.21

Jumbo Frames

CR_0000145820 When Jumbo Frames are transmitted over a trunked port, the software increments both the counters `Transmitted Packets without Errors` and `Transmitted Packets with Errors`.

LLDP

CR_0000142692 The switch transmits incorrect LLDP MED TLVs that contain an incorrect VLAN ID, thus causing devices to be assigned to an incorrect or non-existing VLAN. The proper behavior for the switch is to not transmit LLDP MED packets.

Port Access

CR_0000128904 Randomly, a port on the switch may hang. No MAC addresses will be registered on the port and both inbound and outbound traffic will no longer be forwarded on an affected port. The only recovery method is to reset the switch.

Kod ovog proizvođača/modela uređaja , najbolja mi je greška **CR_0000128904** : nasumično pojedini portovi na preklopniku ostanu u zablokiranom stanju te ne mogu niti primati niti slati pakete. Rješenje resetiranje preklopnika.

Testiranje, testiranje i testiranje i još malo testiranja i na kraju još pokoje testiranje

Zamislimo i dalje da smo u ulozi proizvođača - i došli smo do testiranja uređaja koji smo napravili.
Samo testiranje uređaja nije uopće trivijalno te prilično podiže cijenu finalnog proizvoda – ako je za sve faze razvoja bilo potrebno 1000 vremenskih jedinica, sigurno je da će faza testiranja zahtijevati minimalno još toliko (ovo je moj osobni stav).

Dobar dio odgovornijih tvrtki osim svog internog testiranja, proizvode šalju i specijaliziranim vanjskim tvrtkama na dodatna testiranja, a pogotovo testiranja sigurnosnog tipa. Ovdje se radi o testiranjima ranjivosti uređaja, na razne sigurnosne propuste ali i izvršavanje kontroliranih sigurnosnih napada na navedene uređaje, u cilju pronalazaka svih mogućih problema u radu. Naime znatno jeftinije je platiti dodatna testiranja nego milijunske odštete korisnicima, zbog problema na vašoj opremi a koji su uzrokovali ogromne štete vašim kupcima ili njihovim korisnicima.
Testiranje se svodi na provjeravanje svih mogućih kombinacija protokola ili funkcionalnosti koje određeni uređaj podržava i to u kombinacijama, barem svih važnijih mrežnih protokola koji moraju biti u upotrebi i koji moraju prolaziti kroz mrežni uređaj koji se testira.

To znači da ako govorimo o *ARP* protololu koji mora podržavati svaki preklopnik, a koji je samo jedan on niza protokola, testiranje mora uključiti ponašanje u radu u kojemu su na točno određenoj hardverskoj verziji preklopnika, na točno određenoj verziji softvera (operacijski sustav, upravljački programi i sve ostalo) spojena računala koja generiraju testni ali i što realniji promet, i to u raznim kombinacijam :

- na svim portovima
- samo na nekim portovima
- neka računala se moraju uključivati i isključivati ,….

Dakle i ovdje postoji gomila scenarija koji se moraju proći.
Promet koji se generira mora biti točno definiran te je potrebno testirati razne mrežne protokole u raznim kombinacijama. Prema TCP/IP setu protokola, definirano je **65536** mogućih mrežnih protokola od kojih je posebno važno testirati njih sigurno stotinjak, koji su najčešći u upotrebi.
U praksi, pitanje je, testira li se možda nekoliko desetaka protokola maksimalno, a pitanje je da li i toliko, pogotovo kod *entry level* proizvođača - u pravilu su to najjeftiniji uređaji.
Osim toga pošto se u ovom slučaju testira ARP protokol, potrebno je testirati sve scenarije ponašanja ARP protokola – i one dozvoljene (definirane standardima) kao i one nedozvoljene koji bi mogli utjecati na sigurnost i/ili stabilnost i to, pojedinog porta na preklopniku ili cijelog uređaja ili cijele mreže. Naime nije dovoljno testirati standardne ARP pakete već i one za koje je potrebno "ručno" kreirati ARP pakete koji namjerno sadrže vrijednosti ili parametre koje ne bi smjeli sadržavati. Naime takve neispravne pakete bi uređaj morao odbaciti.
Stoga je samo za testiranje ARP protokola, potrebno na desetke scenarija za testiranje.
Sljedeći test bi bilo uključivanje drugog protokola ili funkcionalnosti – primjerice logiranje poruka na vanjski ***syslog*** poslužitelj. Zvuči banalno ali sada je potrebno ponoviti sve prethodne scenarije testiranja uz dodatne scenarije testiranje specifične za *syslog* protokol. Također s dozvoljenim i "nedozvoljenim" opcijama ili parametrima ili oblicima mrežnih paketa za sve protokole koje testiramo.
Ako uključimo još jednu funkcionalnost – recimo često korišten *Spanning Tree* protokol, sada ponovno moramo sve testirati s dodatnim stvarima vezanim za *Spanning Tree* ali i u varijantama s *ARP* protokolom i sa ili bez *syslog* funkcionalnosti. Sada smo došli do sigurno stotina i stotina scenarija za testiranje.

Ne zaboravimo kako i "najslabiji" preklopnici podržavaju desetak protokola ili funkcionalnosti te činjenice kako samo za isti hardverski model uređaja, proizvođači imaju nekoliko inačica operacijskog sustava (*Firmware*a). Naravno, potrebno je testirati sve pojedinačne inačice zasebno, u svim scenarijima testiranja. I naravno za sve modele uređaja koji se prodaju.
Ako sve zbrojimo, mislim da smo došli do desetaka tisuća scenarija koje je potrebno testirati.
I sada se možemo pitati tko od proizvođača to sve radi ?
Možete se pitati da li je to sve potrebno – moj odgovor je DA.

U praksi sam susreo nebrojeno puta kako već samo određene kombinacije uključenih funkcionalnosti i mrežnih protokola nekada uzrokuju neke treće probleme a u nekim kombinacijama sve radi. Ili u jednoj verziji *firmware*a radi nešto što u drugoj (čak i novijoj) više ne radi ili pravi probleme, na čije otklanjanje nekada možete potrošiti dane i dane, kako bi na kraju shvatili da nije problem u ničemu drugom nego u hardveru odnosno problematičnom *firmware*u (OS-u).

Potrošeni dani x sati x broj ljudi = jeftinije je kupiti pošteni (provjereni) uređaj u početku

Kada kažem provjereni, mislim na konkretnog proizvođača, točan model i verziju softvera (*Firmware/OS*), provjeren u okruženju u kojemu se, po mogućnosti, koriste svi mrežni protokoli i postavke koje i vi želite koristiti.
U konačnici previše je mogućih kombinacija koje ne želite sami testirati na svojoj mreži za koju želite da radi kako treba, pa je uglavnom jeftinije uložiti nešto malo više u opremu pouzdanijih proizvođača, i to po mogućnosti točnog modela i verzije *Firmwarea* za koji znate da radi kako treba.

Ne treba se 100% pouzdati niti u najveće *Brand Name* proizvođače – i njima se događaju greške – itekako, ali za red veličine manje od nekih koje ću nazvati *bezimeni*.

Vratimo se ponovno preklopnicima

Za razliku od računala, preklopnici nemaju *klasične mrežne* kartice nego tkzv. *Switching chip*ove koji zapravo na jednom *chipu* imaju integrirano nekoliko mrežnih kartica, od 4, 8 ,12 , 24, 48 ili više portova.

Ovdje je stanje još drastičnije nego kod *običnih* mrežnih kartica. Naime postoji cijela gama *switching chipova* od raznih proizvođača s ogromnim razlikama:

- Različitih su performansi,
- Različitih funkcionalnosti koje podržavaju
- Njihovi upravljački programi se razvijaju samo za određeni OS ili za više njih, (pitanje je i stabilnosti i dr.)

Što se samih preklopnika ili usmjerivača tiče (kao gotovih proizvoda), neki proizvođači koriste Linux a drugi određene specijalizirane varijante UNIX operacijskih sustava. I odabir operacijskog sustava kao i pripadajućih upravljačkih programa utiče na pouzdanost, sigurnost i performanse konačnog rješenja.

Tako primjerice **Cisco Systems** za svoj **IOS** (operativni sustav) koristi **BSD** i djelove **FreeBSD** Unixa, dok za **IOS XR** koristi **QNX** Unix, odnosno za **NX-OS** koristi **MontaVista Linux**. S druge strane **Juniper Networks** koristi **FreeBSD** Unix za sve svoje mrežne uređaje. **Dell Networking** uređaji za pr. **N** seriju preklopnika koriste Linux (**DNOS** 6.x OS), dok za **S** i **Z** seriju (koju su naslijedili kupnjom tvrtke **Force 10**) koriste **FTOS** koji je baziran na **NetBSD** Unixu.

Nadalje preklopnici imaju i CPU i RAM memoriju poput običnih računala, koji također utječu na performanse.

Osim toga razlikujemo i dvije kategorije preklopnika:

Standardne preklopnike koji rade na OSI sloju 2, te prema tome odluku o preklapanju, za svaki mrežni paket, donose na osnovi MAC adresa (OSI sloj 2). Oni imaju sve ostale funkcionalnosti samo na OSI sloju 2. Druga kategorija su višeslojni odnosno **Multilayer** preklopnici, koji odluke o preklapanju svakog paketa mogu donositi na osnovi analize OSI slojeva 2 i 3 a imaju i druge funkcionalnosti na OSI slojevima 2, 3 i 4. Ovakvi preklopnici prema tome imaju dobar dio funkcionalnosti klasičnih usmjerivača ali i veliku brzinu koju imaju klasični preklopnici (ako koriste *ASIC*).

Layer 2 - Switching – klasična upotreba preklopnika

U današnjim mrežama za međusobno umrežavanja/spajanje računala, poslužitelja i ostale mrežne opreme koriste se preklopnici, koji standardno rade na **OSI** sloju 2 (**Layer 2**) – barataju sa MAC adresama.

Spajanjem bilo kojeg računala ili uređaja na neki port na preklopniku, sam preklopnik prvo mora saznati njegovu *MAC* adresu te ju spremiti u svoju internu tablicu koja sadrži par :

```
Source MAC adresa ↔ port (interface) na preklopniku
```

To je i sva logika koja je potrebna za uspješno preklapanje, Naime kada spojimo računala A i B na preklopnik, i to primjerice:

```
Računalo A (MAC 00:01:02:A1:11:11) ↔ port (interface) 1
```

```
Računalo B (MAC 00:01:02:B2:22:22) ↔ port (interface) 2
```

Preklopnik si nakon nekoliko trenutaka izgradi gore navedenu tablicu koju primjenjuje na svaki paket koji mu dođe.

Ako u nekom trenutku, preklopnik nema *MAC* adresu od odredišnog računala, pokušava ju saznati slanjem pripadajuće *ARP* poruke te ju (MAC adresu) zapamti i gradi si novu tablicu.

Ova MAC tablica na preklopniku se zove i **CAM tablica** (Content addressable memory) tablica.

Spustimo se na OSI slojeve 2 i 1

Kako bi bolje razumjeli pojmove s kojima ćemo se malo kasnije upoznati, moramo detaljnije razumjeti što se događa na OSI slojevima 2 i 1. Dalje u tekstu govoriti ćemo o *Ethernet* vrsti mrežnih okvira. U praksi postoji nekoliko vrsta Ethernet okvira:

- *Ethernet II* - znan i kao Ethernet v.2 ili DIX, koji je najčešći u upotrebi, i na kojemu ćemo se bazirati.
- *802.3 okviri*
- *802.2 LLC* (Logical Link Control) okviri
- *802.2 SNAP* (Subnetwork Access Protocol) okviri

Mrežni okvir je naziv za blok podataka, na OSI slojevima 1 i 2 a koji su uokvireni u ono što često pogrešno nazivamo mrežnim paketima. Možemo sve pojednostaviti i reći: mrežni (Ethernet) okviri su na OSI slojevima 1 i 2, a na višim slojevima ih nazivamo paketima.

Ethernet okviri za najmanje mrežne okvire (64 bytea)

Prvo pogledajmo kako izgleda mrežni **Ethernet** okvir na drugom sloju mreže:

DMAC	**SMAC**	**TYPE**	**DATA/PAYLOAD**	**FCS**
Destination MAC	Source MAC	Vrsta protokola s višeg sloja	DATA/PAYLOAD - sve s gornjih slojeva	CRC Checksum
6 Bytea	6 Bytea	2 Bytea	od 46 do 1500 Bytea	4 Bytea

Zanimljivo polje je ***FCS*** (*frame check sequence*), koje sadrži provjerni broj (izračunat pomoću ***CRC*** algoritma), za cijeli okvir, uključujući i zaglavlje s ovog sloja. Dakle za svaki mrežni okvir na ovom sloju se za sva polja računa provjerni broj, pomoću navedenog *CRC* algoritma, te se sprema u *FCS* polje. Kada mrežni okvir dođe na drugu stranu (na odredišno računalo), ono ponovno izračunava provjerni broj pomoću *CRC* algoritma, te ako je sve u redu (i novo izračunati *CRC* i onaj u paketu su isti), paket se dalje obrađuje, a ako nije u redu, paket se odbacuje.

A sada pogledajmo kako naš najmanji mrežni okvir, koji sadrži samo 46 Bytea podataka, izgleda na drugom i prvom sloju mreže.
Drugi sloj (Layer 2):

DMAC	**SMAC**	**TYPE**	**DATA/PAYLOAD**	**FCS**
Destination MAC	Source MAC	Vrsta protokola s višeg sloja	DATA/PAYLOAD - sve s gornjih slojeva	CRC Checksum odnosno provjerni zbroj
6 Bytea	6 Bytea	2 Bytea	46 Bytea	4 Bytea
← U k u p n o 64 B y t e a →				

I kako izgleda na prvom sloju (Layer 1):

Preambula	**SFD**	**Sve s gornjeg sloja**	**Interframe gap (razmak)**
Označava početak okvira	Nakon Start Frame Delimitera slijedi dio s višeg sloja	Sadrži sve s gornjeg sloja mreže (Layer 2)	Označava razmak između mrežnih paketa/okvira
7 Bytea	1 Byte	64 Bytea	12 Bytea
← U k u p n o 84 B y t e a →			

Dakle, za 46 Bytea podataka, na drugom sloju mreže se dodaje ukupno još 18 Bytea zaglavlja, tako da mrežni okvir raste na 64 Bytea. Takav okvir se prosljeđuje najnižem sloju (Layer 1) a koji nadodaje još svojih 20 Bytea zaglavlja. Tako se na kraju, na mrežu šalje ukupno 84 Bytea podataka, za svaki pojedini mrežni okvir, navedene inicijalne veličine.

VLANovi

Prema standardu **802.1Q**[(60)], definirana je podrška za **VLAN** (*Virtual LAN*) na ethernet mrežama. Ovdje se radi o podrški za tzv. virtualne, potpuno izolirane mreže. Standard *802.1Q* definira način označavanja *VLANova* unutar mrežnih okvira (Engl. *VLAN Tagging*) te naravno sposobnosti mrežnih uređaja da ih prepoznaju, znaju obrađivati i koristiti.

Za početak, pogledajmo kako izgleda mrežni okvir na OSI sloju 2, koji koristi podršku za *VLANove*:

DMAC	SMAC	802.1Q Tag	TYPE	DATA/PAYLOAD	FCS
Destination MAC	Source MAC	802.1Q oznaka	Vrsta protokola s višeg sloja	DATA/PAYLOAD - sve s gornjih slojeva	CRC Checksum
6 Bytea	6 Bytea	4 Bytea	2 Bytea	od 46 do 1500 Bytea	4 Bytea

Dakle mrežni okvir izgleda slično klasičnom osim što je dodano polje u koje se umeću informacije o VLAN mrežama (*VLAN Tag*), a koje je veličine 4 *bytea*. S time mrežni okvir u odnosu na klasični, raste za tih 4 *bytea*.

Kako sada izgledaju najmanji okviri, koji sadrže 46 *Bytea* podataka ?

Drugi sloj (Layer 2):

DMAC	SMAC	802.1Q Tag	TYPE	DATA/PAYLOAD	FCS
Destination MAC	Source MAC	802.1Q oznaka	Vrsta protokola s višeg sloja	DATA/PAYLOAD - sve s gornjih slojeva	CRC Checksum odnosno provjerni zbroj
6 Bytea	6 Bytea	4 Bytea	2 Bytea	46 Bytea	4 Bytea
← U k u p n o 68 B y t e a →					

Te sve zajedno kada se spušta na OSI sloj 1:

Preambula	SFD	Sve s gornjeg sloja	Interframe gap (razmak)
Označava početak okvira	Nakon Start Frame Delimitera slijedi dio s višeg sloja	Sadrži sve s gornjeg sloja mreže (Layer 2)	Označava razmak između mrežnih paketa/okvira
7 Bytea	1 Byte	68 Bytea	12 Bytea
← U k u p n o 88 B y t e a →			

Što se još događa kod upotrebe *VLANova*
Upotrebom *VLANova* razdvajamo mrežu na više potpuno odvojenih mreža, ako to naši mreži uređaji omogućuju. Kako je vidljivo, *VLANovi* se zapisuju unutar mrežnih okvira, na OSI sloju 2 , na kojemu rade i preklopnici, što je logično, jer su upravo preklopnici, uređaji koji se brinu o *VLAN* mrežama (iako i usmjerivači i drugi mrežni uređaji također prepoznaju VLAN mreže). Naime na preklopniku koji podržava *VLANove*, moguće je za svaki pojedini port (*interface*) na preklopniku, definirati njegovu pripadnost:

- pojedinom *VLANu* (*ACCESS mod*)
- SVIMA ili samo odabranom broju *VLANova* (*TRUNK mod*)

Ako pojedini port na preklopniku pripada točno definiranom *VLANu* (on se nalazi u Tzv. *ACCESS* načinu rada) - pr. VLAN 15. Tada, na tom portu sa svakog mrežnog okvira koji je namijenjen za *VLAN* mrežu br. 15, preklopnik MORA ukloniti *VLAN* polje (ono polje od 4 *Bytea*) te ponovno izračunati provjerni zbroj, pomoću CRC algoritma, za cijeli mrežni okvir, te provjerni zbroj upisati u FCS polje. Svi drugi mrežni okviri koji sadrže bilo koji drugi *VLAN* broj OSIM br. 15, neće nikada (niti smiju) biti proslijeđeni na port koji pripada *VLANu* br. 15. Dakle samo portovi koji pripadaju točno određenom *VLANu*, mogu proslijeđivati mrežne okvire među sobom.

Ako pak taj mrežni okvir izlazi na posebno konfiguriran port na preklopniku, koji je u mogućnosti propuštati sve mrežne okvire, sa svih *VLANova*, a to su portovi koji rade u Tzv. *Trunk* načinu rada - obično prema drugom preklopniku, tada preklopnik za svaki pojedini mrežni okvir mora dodati pripadajuće VLAN polje u mrežni okvir, te ponovno izračunati CRC provjerni zbroj te ga upisati u FCS polje mrežnog okvira.

U konačnici, *VLANovi* donose mnoge prednosti ali sa strane performansi mrežnih uređaja, a poglavito preklopnika, unose značajnu potrebu za snažnijim hardverom, koji cijelu priču s baratanjem VLAN oznaka i izračunavanjem CRC provjernog zbroja, mora moći odraditi bez usporavanja mreže. Dakle i ovdje se radi o milijunima ili desecima milijuna mrežnih okvira (ovisno o broju i brzinama portova na preklopniku) u sekundi, koje je potrebno dodatno procesirati (obraditi). Naravno to sve im svoju cijenu.

Što je što: Gbps i Mpps ?

Oznaka Gbps (gigabita u sekundi) označava brzinu pojedinog mrežnog *interfacea* (porta) ili ukupnu propusnost preklopnika, koju dijele SVI portovi. S druge stane, vrlo važno pitanje je i koliko stvarno mrežnih paketa u jedinici vremena (sekundi), može obraditi određeni preklopnik. Odgovor na ovo pitanje slijedi.

Prvo malo računanja

Pretvorimo brzinu od 1 Gbps u bitove u sekundi: `1Gbps = 1,000,000,000 bps`. Pretvorimo bitove u bajte odnosno bajte (byte) u sekundi (Bps): **`8 bitova = 1 Byte`**. Mi moramo sve pretvoriti u Bps : **`1,000,000,000 / 8 = 125,000,000 Bps`**
Dakle za brzinu od 1Gbps imamo protok od 125 milijuna *bytea* u sekundi. Prvo ćemo analizirati najmanje mrežne pakete tj. okvire, od 64 *byte*a. Za Ethernet okvire vrijedi:
`18 byte je Frame Header odnosno zaglavlje okvira - ovo dodaje mrežni sloj 2`

`20 byte je preambula i dio za početak okvira (start of frame delimeter SFD) te na kraju i standardni minimalni razmak između mrežnih okvira (Engl. Interframe Gap - IFG) - ovo dodaje mrežni sloj 1`
Prema **Ethernet** standardu[44], najmanji paket je od 46 *byte*a a uz njega ide i 18 *byte* ***Frame Header*** te još ukupno 20 *byte* ostalih dijelova, s čime konačni paket poslan na mrežu raste na 84 *byte*. Naime ovdje govorimo o mrežnim paketima odnosno okvirima na drugom sloju (Layer 2) mreže, odnosno o njihovoj veličini koja se odnosi na ovaj sloj.

I sada dobivamo : 125,000,000 Bps / (46+18+20) = 1,488,095 pps (paketa u sekundi) = 1.488 Mpps (Milijuna paketa u sekundi)

Dakle za brzinu mreže od 1Gbps, odnosno 1 x 1Gbps, za mrežne okvire veličine 64 bytea, potrebno je moći procesirati (obraditi) 1.488 milijuna mrežnih paketa u sekundi. Pogledajmo kako to izgleda za druge brzine, za ovu veličinu mrežnih paketa (okvira):

Brzina	Broj paketa u sekundi
1 Gbps	1.488 Mpps
10 Gbps	14.88 Mpps
40 Gbps	59.2 Mpps
100 Gbps	148.8 Mpps

Za veće mrežne okvire (1500 byte)
Dakle i ovdje kod 1500 *byte*nih mrežnih okvira mislimo na 1500 *byte Payload*. Dakle korisni dio okvira koji sadrži nama korisne podatke je veličine 1500 *byte*.
Računica je: `1500 byte paket + 18 byte frame header + 20 byte ostali već objašnjeni dio = 1538 byte`

To znači kako se za 1500 *byte*a podataka, šalje mrežni okvir veličine 1538 *byte*

Sada dobivamo: `1538 byte x 8 = 12 304 bitova po paketu`
Za 1Gbps brzinu, pretvorimo bitove u bajte odnosno bajte u sekundi (Bps): `8 bitova = 1 Byte, a mi moramo sve pretvoriti u Bps = 1,000,000,000 / 8 = 125,000,000 Bps`

I sada dobivamo : 125,000,000 Bps / (1500+18+20) = 81 274 pps (paketa u sekundi) = 81.274 kpps (tisuća paketa u sekundi)

Dakle za brzinu mreže od 1Gbps, odnosno 1 x 1Gbps, za mrežne okvire veličine 1500 bytea, potrebno je moći procesirati (obraditi) 81 tisuću mrežnih paketa u sekundi. Pogledajmo kako to izgleda za druge brzine, za ovu veličinu mrežnih paketa (okvira):

Brzina	Broj paketa u sekundi
1 Gbps	81 kpps
10 Gbps	812 kpps
40 Gbps	3.25 Mpps
100 Gbps	8.12 Mpps

Zaključak:
Kao što je i bilo za očekivati, najzahtjevnije je procesirati male mrežne pakete (64 bytea) - za 1 x 1Gbps potrebno ih je obrađivati 1.48 milijuna u sekundi. S druge strane za najveće mrežne pakete (1500 bytea), za 1 x 1Gbps, potrebno ih je moći obrađivati samo 81 tisuću u sekundi.

cut-through i store and forward metode preklapanja

Iako preklopnici gledaju samo OSI sloj 2, svakog mrežnog paketa i na osnovi pročitane odredišne MAC adrese, odlučuju na koji *port* će proslijediti svaki pojedini mrežni paket, postoje dvija načina na osnovu kojih se to može odraditi.

Prvi način je ***cut-through*** [45], pri kojemu preklopnik gleda samo odredišnu MAC adresu svakog mrežnog paketa te odmah kreće s prosljeđivanjem (preklapanjem) tog paketa. Ova metoda je brza i ne zahtjeva puno resursa, sa strane preklopnika (dakle jeftinija je što se tiče hardvera preklopnika). Mana joj je u tome, što se ne provjerava *FCS* (*frame check sequence*) dio paketa, koji sadrži provjerni zbroj (izračunat pomoću *CRC* algoritma).

Zbog toga se događa, kako se neispravni paketi, kojima se o ovoj metodi ne provjerava integritet, uredno prosljeđuju (preklapaju) prema odredištu. U tom slučaju se mrežom šire neispravni paketi, koje može generirati neispravna mrežna kartica ili neke druge greške sa izvorišnog računala. Ova metoda je nastala u trenutku razvoja *ethernet* preklopnika, u vremenima kada su mreže radile na 10Mbps brzinama a kasnije i 100Mbps, te je većina mrežnih uređaja radila u *Half Duplex* načinu rada, u kojemu je dolazilo do kolizije.

Stoga je u teoriji, kada dođe do kolizije paketa na mreži, moguće da su neki paketi (tj. okviri na mreži) oštećeni te da su manji od 64 *bytea*. Ovakvi paketi se nazivaju ***runt*** paketi odnosno okviri. Dakle jedino što ovakvi preklopnici dodatno provjeravaju je minimalna veličina mrežnog okvira (paketa) na OSI sloju dva, točnije svi mrežni okviri koji su manji od 64 *bytea* se odbacuju.

Drugi način je ***store and forward*** [46], kod kojeg se za svaki primljeni mrežni okvir (paket) ponovno preračunava *FCS* provjerni zbroj, pomoću istog *CRC* algoritma. Potom se uspoređuje s onim koji se nalazi u mrežnom okviru (paketu). Ako preračunati *CRC* i onaj koji se nalazi u paketu, nisu isti, paket se odbacuje. Jasno je kako je za ovo potrebna dodatna snaga i brzina odnosno snažniji hardver preklopnika. Prednost je u tome što se neispravni mrežni paketi ne prosljeđuju dalje te ne zagušuju mrežu. Mana je dodatno vrijeme obrade odnosno kašnjenje, koje uvelike ovisi o brzini hardvera preklopnika.

Bandwidth odnosno širina komunikacijskog pojasa

Bandwidth odnosno širina komunikacijskog pojasa, se definira kao količina informacija koje mogu prolaziti određenim mrežnim medijem u jedinici vremena. Mrežni mediji mogu biti žičani kabeli (primjerice parični FTP/STP/UTP), optički kabeli [single/multi mode] ili eter za bežične mreže (zrak, vakuum)…

Za svaki medij postoje ograničenja širine komunikacijskog pojasa. Uvijek postoje razlike između *bandwidtha* i *throughputa* (propusnosti), i to zbog raznih gubitaka, što u samom mediju (dolazi do slabljenja ili miješanja signala, …), zbog sporosti mrežnih uređaja, a što zbog dodataka na "korisne" podatke, koje dodaju svi mrežni protokoli.

Jedinica za širinu komunikacijskog pojasa se označava sa **bps** (bits per second) [bitova u sekundi]:

Jedinica	Oznaka	Vrijednost
Bitova u sekundi	**bps**	1bps = Osnovna jedinica
Kilobita u sekundi	**kbps**	1kbps = 1 000 bps
Megabita u sekundi	**Mbps**	1Mbps = 1 000 000 bps
Gigabita u sekundi	**Gbps**	1Gbps = 1 000 000 000 bps
Terabita u sekundi	**Tbps**	1Tbps = 1 000 000 000 000 bps

Throughput odnosno propusnost

Throughput odnosno propusnost označava uspješno prenesenu količinu informacija u jedinici vremena. Radi se o tome kako u prijenosu podatak kroz mrežu dolazi do gubitaka i usporavanja, uvjetovanih mrežnom opremom ili kvalitetom veza.

Jedinica za propusnost je također `bps` (bitova u sekundi) ali se češće koristi `Bps` (*bytea* u sekundi) - uočite kako je slovo `B` veliko.

Maksimalnu teoretsku propusnost mreže izraženu u `Bps`, možemo izračunati: tako da maksimalnu teoretsku propusnost podijelimo s 8, jer bitove u sekundi želimo pretvoriti u *byte* u sekudi (zbog lakšeg razumjevanja).

Za Ethernet mreže možemo općenito koristiti slijedeću tablicu:

Bandwidth	Maksimalna teoretska propusnost
1 Mbps	128 kBps
10 Mbps	1.28 MBps
100 Mbps	12.8 MBps
1 000 Mbps (1Gbps)	128 MBps
10 000 Mbps (10 Gbps)	1280 MBps

Postoji i još jedan pojam a to je stvarna odnosno **efektivna propusnost** mreže, koja varira ovisno o mnogim čimbenicima:

- kvaliteti veza
- mrežnoj opremi
- izvorišnoj i odredišnoj točki komunikacije
- kontrolnim podacima, koje unose mrežni protokoli

Klasični primjer može biti naša lokalna mreža. Zamislimo lokalnu (LAN) mrežu u kojoj imamo računala s običnim mrežnim karticama, koje imaju propusnost od 1Gbps te mrežni preklopnik koji također na svim portovima ima propusnost od 1Gbps. U teoriji bi komunikacija između dva računala na ovoj mreži, spojena preko preklopnika, imala također propusnost od 1Gbps.
Dakle maksimalnu teoretsku brzinu prijenosa podataka od 128 MB/s. Kod običnih preklopnika, koji su deklarirani kao **Gbps**, to u praksi uopće nije tako. Ako smo odabrali prosječan preklopnik i ako računala koja komuniciraju preko njega uopće mogu podnijeti te brzine, dobiti ćemo prosječnu efektivnu propusnost od 200Mbps do 500Mbps (od 25MB/s do 63 MB/s) ali se nećemo niti približiti punoj propusnosti gigabitne veze. Često se pojam brzine brka s propusnošću mreže. Tako primjerice 1Gbps ne označava brzinu već propusnost.

Dodatno postoji i još jedan mali detalj, ako sve promatramo s razine aplikacije, recimo našeg programa preko kojega primjerice kopiramo podatke s jednog na drugo računalo, a to je efikasnost mrežnih protokola. Ako govorimo o TCP/IP protokolu, svaki sloj mrežnog protokola dodaje i svoje "kontrolne" podatke, odnosno zaglavlja.
Ako govorimo o mrežnim okvirima na OSI sloju 2, tada imamo, za mrežne okvire od 1500 *byte payload*:

1. OSI sloj 2 : ukupno 18 *bytea* zaglavlja
2. OSI sloj 1 : ukupno 20 *bytea* zaglavlja

Dakle za 1500 *bytea* payload (podaci) imamo još ukupno 38 *bytea* zaglavlja s OSI slojeva 1 i 2.
To znači kako je maksimalna efikasnost protokola zaovu veličinu mrežnih okvira, sa OSI sloja 2 jednaka:
Efikasnost okvira na OSI sloju 2 = 1500/1538 = 0.9753 odnosno **97.53%** . Drugim riječima **2.47 %** se gubi na mrežni protokol.
Ako govorimo o višim protokolima, tj onima koji koriste više OSI slojeve (dakle prema TCP/IP : IP, i TCP) dobivamo još malo više zaglavlja. Te ako se podignemo na aplikaciju razinu - pr. ako se radilo o FTP komunikaciji, tada imamo dodatno zaglavlje odnosno dijelova koji se dodaju, tako da se ukupna efikasnost još malo smanjuje.
I na kraju neki aplikacijski protokoli za razmjenu datoteka su učinkovitiji od drugih što zbog svog zaglavlja koje dodaju na svaki mrežni paket a što zbog svoje potrebe za dodatnim procesiranjem.

Latencija odnosno kašnjenje

Latencija odnosno kašnjenje se odnosi na vrijeme potrebno informaciji odnosno bitu podataka kako bi prošlo put između točke A i točke B (izvor-odredište). Dakle ovo je vrijeme koje je potrebno kako bi podaci preko mreže došli od jednog do drugog računala koja međusobno komuniciraju.

U praksi, kašnjenje se događa zbog:

- prolaska kroz medij (kabel/optika/eter)
- vremena potrebnog mrežnim uređajima kako bi zaprimili, obradili i proslijedili/preklopili mrežni okvir ili paket
- vremena potrebnog odredišnoj točki (računalo) da prihvati i obradi mrežni paket

Stvarna, izmjerena vemena kašnjenja su :

- kašnjenje koje unosi preklopnik - preklapanje unutar nekoliko mikro sekundi (us), za brzine do 1Gbps, te stotinjak nano sekundi (ns) za brzinu 10Gbps [58][59]
- kašnjenje koje unosi *multilayer* preklopnik s *ASIC chipom* - usmjeravanje unutar nekoliko mikro sekundi (us), za brzine do 1Gbps, te stotinjak nano sekundi (ns) za brzinu 10Gbps [58][59]
- kašnjenje koje unosi usmjerivač - usmjeravanje unutar mili sekundi (ms)
- u lokalnim mrežama - između računala unutar lokalne mrže : unutar mili sekunde (ms) ili do maksimalno par mili sekundi (ms)
- u većim lokalnim mrežama s nekoliko usmjerivača - do nekoliko mili sekundi (ms)
- za WAN mreže - unutar jednog grada - ispod 10 ms
- za WAN mreže - između gradova (pr. 200-300km) - oko 20-30 ms
- za WAN mreže između dvije države (do 1 000 km) - oko 30-80 ms
- za WAN mreže između kontinenata [57]:
    - Amsterdam → Washington : 92 ms
    - Amsterdam → Toronto : 105 ms
    - Amsterdam → Tokyo : 252 ms

Veza između latencije, propusnosti i TCP Window size parametra TCP veze

Iako ovi pojmovi naizgled nisu povezani, veza između njih je vrlo opipljiva. Naime na kašnjenje odnosno latenciju ne možemo puno utjecati zbog zakona fizike. Jedino na što bi eventualno mogli utjecati je kašnjenje uzrokovano sporijom mrežnom opremom, od strane telekoma ali u praksi su razlike prilično male. U svakom slučaju važno je znati kako postoje određena ograničenja koja nisu toliko očita.

Pogledajmo o čemu se radi.

Upotrebom TCP/IP protokola, konkretnije TCP protokola, kao transportnog protokola, svi današnji operacijski sustavi, koriste TCP Window funkcionalnost, za povećanje propusnosti. Standardna veličina ovog TCP "prozora" je obično 64kB.

64kB = 65535 bytea = 524 280 bitova

1Gbps = 1 000 000 000 bitova u sekundi

$$\textbf{MAKSIMALNA DOZVOLJENA LATENCIJA} = \frac{\textbf{TCP WINDOW SIZE}}{\textbf{ŠIRINA POJASA MREŽE}} \quad (63)$$

Prema tome : MAKSIMALNA DOZVOLJENA LATENCIJA = 524 280 / 1 000 000 000 = 0.00052428 sekundi = **524 mikro sekunde (us)**

To znači, kako za mrežu koja radi na **1Gbps**, te se u komunikaciji oslanja na TCP protokol, koji koristi 64kB veličinu TCP prozora, kako bi se postigla puna propusnost mreže, kašnjenje između dvaju strana u komunikaciji ne smije biti veće od 524 mikro sekunde (us) odnosno 0.524 mili sekunde (ms). Kod **10Gbps** mreža vrijeme je deset puta manje, dakle **52.4 mikro sekunde (us)** odnosno 0.0524 mili sekunde (ms).

Sve ovo isto tako znači i kako se povećanjem kašnjenja (latencije) uz istu veličinu *TCP prozora*, smanjuje i efektivna propusnost mreže. Dakle vrlo duge međuveze, između gradova ili država i kontinenata, zbog fizike medija i rasprostiranja signala kroz njega, kao i mrežne opreme, unose kašnjenja signala, koja utječu na efektivnu propusnost mreže. Jedino na što možemo utjecati je veličina *TCP prozora*.

Routing odnosno usmjeravanje (Layer 3)

Usmjeravanje (engl. *Routing*) je proces odabira, najboljeg puta za prolaz mrežnih paketa prema odredištu. Uređaji koji odrađuju usmjeravanje se zovu usmjerivači (engl. *Routers*).
Za potrebe usmjeravanja možemo koristiti :

- Statičke rute
- protokole za usmjeravanje (*routing* protokole)

Statičke *rute* se dodaju, kao što i ime kaže, statički odnosno ručno, za svaku novu mrežu, što postaje malo nezgodno kada imamo više mreža ili kada se mreže mijenjaju. U slučajevima kada imamo veliki broj mreža statičke *rute* postaju problematične tj. vrlo teško ih je održavati ručnim i stalnim dodavanjem i brisanjem, stoga se za sve imalo veće mreže koriste protokoli za usmjeravanje, koji dinamički, automatski dodaju i brišu odnosno mijenjaju *rute*. Većina usmjerivača na internetu, koristi neki od protokola za usmjeravanje.
Statički odnosno ručno se dodavaje i Tzv. **Default route**, koja je osnovna *ruta*, koja prolazi preko primarnog usmjerivača, koji se zove i *Default gateway*. U manjim mrežama, to je jedini usmjerivač na mreži. Drugi slučaj upotrebe statičkih *ruta* je ako imamo prilično statične (nepromjenjive) ili male mreže, u kojima nam je praktično, ručno dodavati, mijenjati ili brisati *rute*.
Default ruta s *Default gateway* usmjerivačem obično ima oblik :

IP Adresa	Netmask	Default Gateway IP
0.0.0.0	0.0.0.0	192.168.1.1

Default ruta koju vidite znači: *za sve* IP adrese odnosno sve mreže, za koje ne postoji izričito definirana *ruta*. Dakle u slučaju kada usmjerivač ili računalo na mreži, ne zna gdje poslati neki

paket tj. kada nema definiranu *rutu* (putanju) za njega, paket se šalje na IP adresu *Default Gateway* usmjerivača.

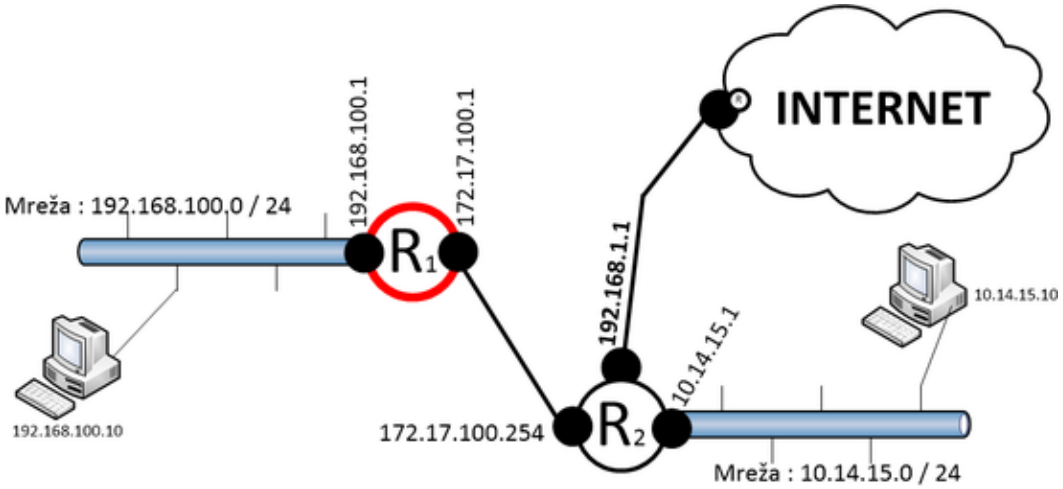
Što je uopće tablica usmjeravanja odnosno *rute* ?

Route nam pokazuju tko je od usmjerivača odgovoran za koju mrežu, tako da znamo gdje poslati koji mrežni paket koji izlazi iz naše mreže, prema drugoj mreži.
Možemo promatrati *rutu* kao putokaz za mrežne pakete, poput autoceste: dali skrenuti lijevom ili desnom cestom, prema odredištu.
*Primjer tablice usmjeravanja, na usmjerivaču (**R1**), može opisno izgledati ovako :*

```
Za mrežu 192.168.100.0 / 255.255.255.0 (/24) koristi mrežno sučelje eth0
Za mrežu 10.14.15.0 / 255.255.255.0 (/24) koristi usmjerivač - IP : 172.17.100.254

Default route:
Za mrežu 0.0.0.0 / 0.0.0.0 koristi IP : 172.17.100.254
```

U konkretnom primjeru to znači kako će komunikacija unutar mreže 192.168.100.0, biti bez posredovanja usmjerivača **R1** jer mu je to lokalna mreža, dostupna preko njegovog mrežnog sučelja **eth0**, dok će svi paketi koji trebaju završiti na mreži : 10.14.15.0, biti usmjereni na usmjerivač : 172.17.100.**254**. I na kraju *default ruta* kaže, kako za sve ostale udaljene mreže, treba slati pakete na *Default gateway* usmjerivač, s IP adresom : 172.17.100.**254** .
Pogledajte kako ovakva mreža izgleda, topološki, ako se navedene *rute* nalaze na usmjerivaču **R1**:



Protokoli za usmjeravanje, prema logici rada mogu koristiti dvije vrste algoritama. Prema tome, protokole za usmjeravanje dijelimo prema vrsti algoritma koji koristi, a to su :

- ***Distance vector*** protokoli (RIP, RIP v.2, IGRP, …)
- ***Link-state*** protokoli (OSPF, IS-IS, …)
- **Hibridni protokoli**, koji su kombinacija gornja dva (BGP, EIGRP, …)

Malo detaljnije o osnovnom usmjeravanju

U slučaju usmjeravanja (Engl. Routing) koji se događa na **OSI** sloju 3 (IP adrese), uređaji odluku za usmjeravanje paketa donose na osnovu IP adresa.
Tablica na osnovu koje se odrađuje usmjeravanje se zove *Routing tablica* odnosno tablica usmjeravanja. Dakle slična priča kao za preklopnike, ali na drugom OSI sloju, i još malo kompleksnija zbog *routing* protokola i mogućnosti dinamičkih promjena *routing* tablica pomoću *routing* protokola (RIP, OSPF, BGP, …).
Uređaji koji se bave usmjeravanjem, zovemo usmjerivači (*Routeri*) ali postoje i preklopnici (*switchevi*) koji mogu odrađivati i taj dio posla. Ovakve preklopnike zovemo **Multilayer switchevi** odnosno preklopnici koji rade na više OSI slojeva (2,3 i 4). Ovakvi preklopnici vrlo su važni u današnjim mrežama, jer nam daju mogućnost da uredno segmentiramo lokalnu mrežu a da to ne unese velika usporavanja, koja bi uveli klasični *routeri*.
Kako smo rekli, kod usmjeravanja, *usmjerivač* ili *Multilayer* preklopnik, zaprimaju svaki, pojedini mrežni paket te gledaju samo OSI sloj 2 odnosno sloj na kojemu je prema TCP/IP modelu, IP protokol. IP protokol na svom sloju između ostalih polja ima i polja u kojima se nalaze:

- izvorišna IP adresa (*Source IP*)
- odredišna IP adresa (*Destination IP*)
- provjerni zbroj (*Checksum*)
- ukupna duljina paketa (i IP zaglavlja i podataka)
- TTL (*Time to Live*) odnosno vrijeme dozvoljenog života paketa
- deseci drugih opcija i zastavica

Usmjerivač za svaki primljeni mrežni paket, pristigao na bilo kojem njegovom portu (sučelju), prvo provjerava *IP* zaglavlje i to dio s izvorišnom i odredišnom IP adresom. On potom uspoređuje odredišnu IP adresu sa svojom tablicom usmjeravanja. Na osnovi ove tablice, koja se može i dinamički osvježavati pomoću *routing* protokola, usmjerivač usmjerava paket preko onog sučelja preko kojega će paket moći doći do svog odredišta, ili preko drugih usmjerivača ili direktno, sve ovisno o topologiji mreže.

TTL (Time to Live)

Pošto svaki mrežni paket sadrži i *TTL* polje, svaki usmjerivač, mora pri preuzimanju svakog paketa, ovo polje (koje je zapravo brojač vremena), smanjiti. Prema osnovnom standardu u ovo polje se inicijalno upisuje maksimalan broj sekundi koje paket smije biti "na životu". Svaki usmjerivač kod prihvaćanja paketa, mora ovaj brojač smanjiti minimalno za jedan (1). Maksimalna vrijednost je 255, ali se obično (ovisno o operacijskom sustavu koristi neka vrijednost između 64 i 128). U praksi TTL polje se smanjuje za jedan (1) prolaskom kroz svaki usmjerivač.
Ovaj mehanizam osigurava da paketi koji ne bi mogli stići do odredišta, ne bi vječno lutali mrežom, već se nakon isteka postavljenog brojača, odnosno prolaska kroz određeni maksimalni broj usmjerivača paket odbacuje, na zadnjem usmjerivaču. S obzirom kako svaki usmjerivač smanjuje ovaj broj, te se paket zbog toga promijenio, usmjerivač stoga također mora ponovno izračunati novi provjerni zbroj za *IP* zaglavlje paketa, te ga zapisati u novi paket. Dakle ova operacija ponovnog preračunavanja i zapisivanja provjernog zbroja, mora se raditi za svaki pojedini paket.
Ova operacija je naravno procesorski zahtjevna.

MTU

Dodatno svaki usmjerivač, na svakoj svojoj mrežnoj kartici (sučelju) ima postavljenu i Tzv. **MTU**[62] (*Maximum Transmission Unit*) veličinu. *MTU* definira maksimalnu veličinu podataka (bez zaglavlja) koji mogu stati u mrežni paket, na mrežnom sloju (*Network Layer* tj. OSI sloj 3 - upravo ovaj sloj). MTU veličina za lokalne ethernet mreže je obično 1500 *bytea* ali za neke druge mreže može biti drugačija (pr. A/S/HDSL, ADSL2+, ATM, …) te pogotovo za neke vrste dodatnih enkapsulacija, poput raznih VPN tunela. Pogledajmo neke od MTU vrijednosti[61] :

MTU (byte)	Tip konekcije
1501- 9198 (ili više)	Ethernet v.2 - standardni ethernet uz upotrebu "Jumbo frame" okvira
1500	Ethernet v.2 - standardni ethernet
1480	PPPoE
1460	L2TP
1454 - najčešće	PPPoE-over-DSL
1372	PPTP

Zbog čega je važan MTU ?

Usmjerivač može na različitim mrežnim sučeljima, koje mogu biti spojene na različite vrste mreža, imati različito postavljen *MTU*. Kada mrežni paket (*datagram*) dođe na jedno mrežno sučelje usmjerivača, on prvo provjeri svoj MTU, postavljen na tom sučelju, te na osnovi tablice usmjeravanja odluči na koje mrežno sučelje će usmjeriti taj *datagram*, te provjerava *MTU* tog mrežnog sučelja. Ako postoji razlika u *MTU* vrijednostima između mrežnog sučelja na kojem je *datagram* primljen i onoga na koje se preusmjerava, usmjerivač mora raditi Tzv. **IP fragmentaciju** odnosno rastavljanje na manje *fragmente*.

Ako je primjerice naš paket/datagram došao s mrežnog sučelja koje je na našoj *LAN* mreži , te ima *MTU* 1500 *bytea* a odredište mu je na mreži na kojoj je mrežno sučelje za *VDSL*, koji koristi *PPPoE*, a koji ima *MTU* 1454 *bytea* očito da svi podaci od izvornog paketa/datagrama neće stati u novi datagram. Stoga usmjerivač mora razlomiti izvorni datagram na dva nova, od kojih će prvi sadržavati 1454 *bytea payload*, a drugi samo 45 *bytea payload* dijela.

Ova operacija se mora odrađivati za svaki pojedini mrežni paket/datagram, kod kojega se mijenja MTU vanjeskog mrežnog sučelja usmjerivača. S druge strane, na odredištu se radi suprotan proces. I ovaj proces je također procesorski i memorijski zahtjevan za usmjerivač.

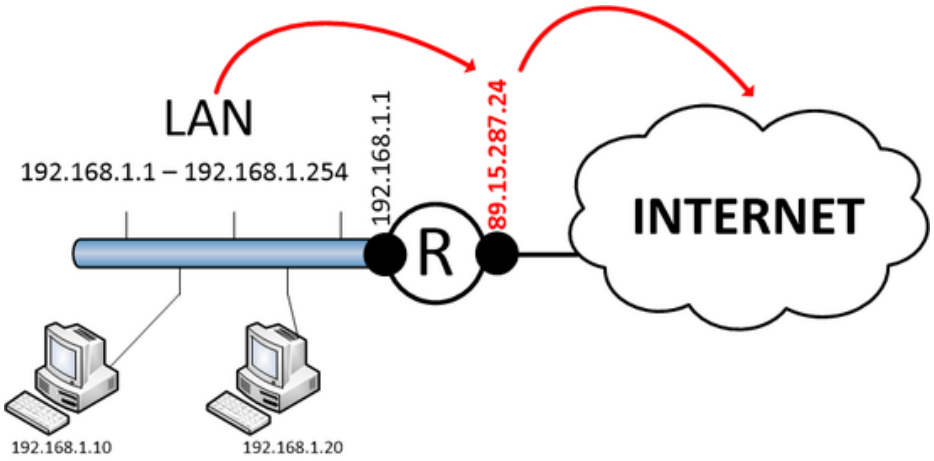
IP fragmentacija je definirana u RFC 791, dok je suprotan proces de fragmentacije definiran u RFC 815.

NAT (Network address translation)

Network address translation (NAT) se odnosi na metodu preslikavanja jednog IP adresnog prostora (opsega IP adresa) u drugi. Dalje u tekstu, govoriti ćemo o samo jednoj od metoda, koja je najčešće u upotrebi. Dakle radi se o preslikavanju IP adresa s jedne mreže u kojoj imamo jedan IP adresni prostor na drugu mrežu s drugim IP adresnim prostorom.
Ovu operaciju odrađuje usmjerivač, tako što na razni IP protokola, unutar njegovog zaglavlja, mijenja IP adrese, i naravno ponovno izračunava provjerni zbroj, na razini *IP* protokola (OSI sloj 3), za svaki takav paket (datagram), koji mora izaći na vanjsku mrežu (pr. internet).

Pojednostavljeno, ova metoda se koristi za skrivanje i zamjenu cijelog, unutarnjeg IP adresnog prostora u lokalnoj (LAN) mreži s javnom IP adresom (vanjskom) adresom našeg usmjerivača.

Obično u lokalnoj mreži imamo privatni opseg IP adresa (prisjetite se poglavlja **Klase IP adresa**) poput:: 192.168.1.1 - 192.168.1.254, dok je javna IP adresa ona koju nam dodjeljuje telekom (*ISP*) i ona se nalazi na vanjskom (*WAN*) mrežnom sučelju usmjerivača, poput primjera na slijedećoj slici.



Radi se o tome, kako usmjerivač za svaki naš mrežni paket na lokalnoj mreži, mijenja izvorišnu IP adresu (*Source IP*) sa svojom vanjskom (javnom) IP adresom, skrivajući našu stvarnu IP adresu. U konkretnom slučaju bi se IP adresa prvog računala (192.168.1.10) zamaskirala kao vanjska IP adresa usmjerivača (89.15.287.24) isto kao što bi se i IP adresa drugog i bilo kojeg računala na lokalnoj mreži, zamaskirala kao vanjska IP adresa usmjerivača (89.15.287.24).

Ova metoda je potrebna jer inače mrežni paketi nikada ne bi našli put nazad do našeg usmjerivača, pošto bi im izvorišna IP adresa bila naša privatna IP adresa, poput 192.168.1.x, a koja nije javno poznata niti je dostupna njena "lokacija".
U toku ovog postupka, usmjerivač, mora zapamtiti s koje je stvarne (naše) unutarnje IP adrese izvorno došao paket, da bi ga znao vratiti nazad.

Kako se za svaki paket koji izlazi (pr. prema internetu) mijenja izvorišna IP adresa, usmjerivač mora pratiti svaku pojedinu konekciju, kako bi, kada se paket vrati nazad, znao kojem računalu (IP adresi) u lokalnoj mreži pripada.
Potom usmjerivač, za svaki mrežni paket, mora zamijeniti svoju vanjsku IP adresu, sa stvarnom unutarnjom IP adresom računala u lokalnoj mreži, kojemu je paket i bio namijenjen.
Detalji ove metode *NATiranja* su opisani u (RFC 2663).

I ova metoda, naravno troši resurse usmjerivača.

Navedena *NAT* metoda se često naziva i :

- **One-to-many NAT**
- **port address translation (PAT)**
- **IP masquerading**
- **NAT overload** ili
- **Many-to-one NAT**

Dizajn

U konačnom dizajnu preklopnika, u pravilu postoje dva pristupa:

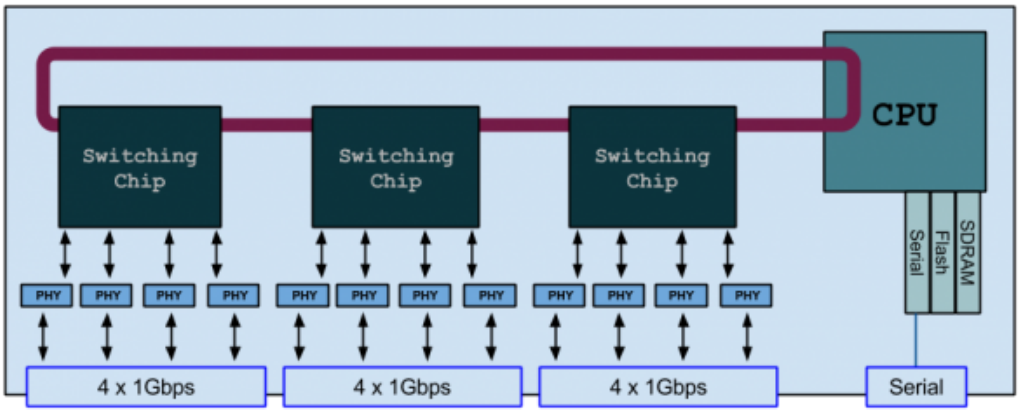
1. *CPU+Switching chip* ili
2. *CPU+Switching chip + ASIC*

Već kod standardnih preklopnika (koji odluke o preklapanju donose na OSI sloju 2), funkcionalnosti koje su implementirane u *Switching chip*ove nisu dostatne za sve operacije koje jedan preklopnik treba podržavati, već su uglavnom implementirane najrudimentarnije stvari pa čak i ovdje, kontrolu za dobar dio mrežnih protokola mora odraditi *CPU* samog preklopnika.

Ovdje je važnost u odabiru dobrog *Switching chip*a krucijalna jer dobar dio njih već u nižoj srednjoj klasi nudi dobar dio funkcionalnosti implementiran u sam **Chip** te se samim time rasterećuje *CPU* koji ionako nije sposoban za obavljanje tih zadataka na gigabitnim brzinama.

I ovdje imamo varijacije na temu

Prva varijanta : više manjih i slabijih *Switching chipova*, povezanih preko zajedničke sabirnice



Ovakav dizajn koriste mnogi jer drastično snižava cijenu.

Problemi su uglavnom sporost sabirnice te samim time komunikacija između *switching chipova* – u ovom slučaju sve ide donekle brzo u komunikaciji između *portova* 1- 4 te u komunikaciji između *portova* 5-8 te između *portova* 9 -12. U slučaju da komunikacija treba ići između *portova* koji su na različitim *switching chipovima* – nastupaju usporavanja, jer cijela komunikacija prolazi kroz zajedničku sabirnicu.

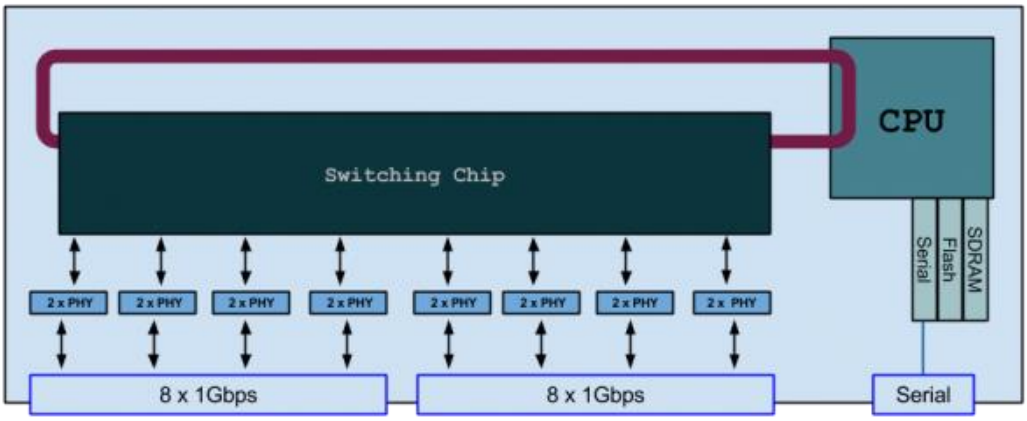
Dodatna mana je u tome što se u ovoj (najgoroj varijanti) ugrađuju *switching chipovi* koji nemaju gotovo nikakve funkcionalnosti osim onih najosnovnijih. Ispravno bi bilo da ih proizvođači niti ne omogućavaju ali u praksi nailazimo na uređaje koji podržavaju cijeli niz funkcionalnosti i protokola ali koriste ovakav dizajn u kojemu na kraju većinu toga mora odraditi *CPU* te se u slučajevima kada ste uključili određene funkcionalnosti (mrežne protokole i sl.) uređaj počinje ponašati užasno usporeno.

Druga varijanta je upotreba naprednijih switching chipova ili switching chipova s više portova

Druga varijanta je upotreba naprednijih *switching chipova*, koji podržavaju više funkcionalnosti te i neke mrežne protokole. Većina njih ima i veći broj *portova*, pa komunikacija ne mora ići preko još uvijek spore sabirnice. Druga podvarijanta je samo upotreba *switching chipova* s više portova.

Obije varijante dizajna su vidljive na slici:

Problem i ovog modela je vrlo čest odabir (od strane većine "slabijih" proizvođača), prilično slabih *switching chipova* te prebacivanje previše toga na CPU (naravno preko iste sabirnice).



Drugi potencijalno problematični dizajn je upotreba *switching chipova* s više portova s direktnom vezom s *CPUom*. Točnije u izvedbi u kojoj je CPU integriran unutar *switching chipa*.
Problematični dio ovog dizajna je samo u sporosti jedne i/ili druge kompomente ili njihove međuveze (interne sabirnice). Najčešće je odabir *CPUa* za preklopnike iz područja **ARM** ili **MIPS** arhitekture procesora, koji za vrlo nisku cijenu i potrošnju energije nude solidne karakteristike.

Nažalost u ovom svijetu u kojemu su potrebne analize i obrade svakog pojedinog mrežnog paketa – dakle *switchinga* (Layer 2) ili *routinga* (Layer 3 i 4) , pogotovo na *gigabitnim* brzina, ovi procesori ne nude zadovoljavajuće brzine obrade paketa. Stoga mnogi odabiru procesore sa sve više i više jezgri.
Dodatni problem je i u tome što recimo specifikacija koja govori da se koristi *ARM* procesor na 1GHz ne znači zapravo ništa.
Da malo razjasnimo – *ARM* nije model ili tvrtka koja u konačnici proizvodi procesore. Tvrtka koju mnogi zovu **ARM** (*Advanced RISC Machine*) zapravo dizajnira procesore a zove se **ARM Holdings plc**. Dakle bilo koji proizvođač može kupiti licencu za *ARM* dizajn i arhitekturu procesora te ga i izraditi (ili dati nekom drugom da ga izradi za njega - ovo je najčešći model).
Nadalje i *ARM* licenci (dizajna) postoji cijelo čudo:

- Cortex-A (A72, A57, A17, A15, A53,A35 A7, …, pa postoje arhitekture ARM v6, v7, v8, …)
- Cortex-R (R4,R5,R7,..)

- Cortex-M (…)

Zapravo samo nekoliko je velikih proizvođača koji izrađuju (ARM) procesore a postoji na desetke onih za koje ih izrađuju, prema ARM licenci i prema željama onih koji ih naručuju (uključite ovo, isključite ono, dodajte ovu funkcionalnost, izbacite onu, ….).
U tome zapravo leži dio problema. Ima ih svakakvih, odnosno neki imaju mnoge dobre funkcionalnosti ali su malo skuplji a neki imaju samo najosnovnije ali su jeftiniji… ne moram dalje pričati.

Pogledajte "proizvođače" ili proizvođače *ARM* procesora za pametne telefone – lista je dugačka.
Gruba računica govori kako ARM procesori koji obično završe u nekoj od *budget* varijanti mrežnih uređaja, kategorije *ARM Cortex A* procesora (pr. v6), takta 1GHz po moći obrade mrežnih paketa jednak *Pentium 3* ili u krajnjem slučaju *Pentium 4* procesoru radnog takta 300MHz. Novije *ARM* arhitekture (v7, v8) postižu veće performanse ali se tek u novije vrijeme počinju češće upotrebljavati.

Slična priča je i sa **MIPS** arhitekturom procesora ali i raznim drugim (kombinacijama) arhitektura. Pogledajte primjerice MikroTik referentni dizajn, Routera : **CCR1072-1G-8S+**, koji koristi **Tilera** CPU sa 72 jezgre (*Tile-Gx*), koje rade na 1GHz. [47],[48],[49],[50],[51],[52],[53]

Analiza problema

Preporuka za *routing*

Na osnovi iskustva te izvora [47],[48],[49],[50],[51],[52],[53], ako pogledamo *Multilayer* funkcionalnosti (što vrijedi za *Routere* ali i *Multilayer Switcheve*), neke generalne preporuke za snagu *CPUa* bi bile (na osnovi samo dva *interfacea* odnosno dvije mrežne kartice : WAN i LAN) :
CPU : Pentum 4 klasa, 2GHz : 100-500Mbps (ovisno što se sve treba procesirati), dobivamo sljedeći zaključak:

```
6 x ARM CPU na 1 GHz = 1 x Pentium 3 CPU na 2 GHz (budimo optimistični pa recimo da je Pentium 3 = Pentium 4).
```

Dakle 6 x ARM 1Ghz je dovoljno za max 2 mrežna *interfacea* za brzine procesiranja između 100Mbps i 500Mbps.
Za 1Gbps u najboljem slučaju to znači 12 x ARM 1GHz CPU za 1Gbps propusnosti između dva mrežna *interfacea*.
Pošto 1Gbps omogucava *Full duplex* način rada, to znači kako nam je za popunjavanje pune propusnosti 2 mrežna *interfacea* koja oba rade u *Full duplex* načinu rada , potrebna propusnost od 4Gbps. To tada znači 12 x ARM 1GHz x 4 = 48 Core ARM CPU na 1GHz
Ako uzmemo u obzir kako su preklopnici, koji se najčešće koriste, obično sa 24 x 1Gbps porta, to bi značilo kako bi morali imati 576 jezgreni *ARM* procesor na 1GHz.
Ovo je računica u slučaju kada *Switching chip* ne odrađuje veći dio funkcionalnosti – što je i slučaj kod niže do niže srednje kategorije ovih *chipova* ali i kada se ne radi o samo čistom prosljeđivanju paketa
Ako budemo optimistični ili ako se radi o novijim *ARM* arhitekturama (*v7* ili *v8*) ili ako *Switching chip* odrađuje nekih 50% potrebnih funkcionalnosti, tada nam je za sve navedeno potrebno "samo" 288 jezgri *ARM CPUa* na 1GHz.

CPU - olakotna okolnost

U slučajevima kada se mrežni paketi samo moraju slati (ili primati) na mrežu, generalno pravilo govori:
`1 Hz CPUa (1 takt) je potreban kako bi se poslao ili primio 1 bit podataka, upotrebom TCP/IP protokola` **(38)** i **(39)**

To znači kako je za 1Gbps i to *Full Duplex*, potrebna jedna *CPU* jezgra na 2GHz. Prema tome 2 x 2GHz za 2 x 1Gbps *Full Duplex*, primanje ili slanje (bez *ACL* lista, ili dodatnih obrada - dakle čisto teoretski za našu primjenu). Tada bi za 24 x 1Gbps *Full Duplex*, bilo potrebno 24 *CPU* jezgre na 2GHz (trenutno zaboravimo na brzinu sabirnice, RAM memorije i njene sabirnice te druge komponente koje realno utječu na rad). Stvarni pokazatelji su vjerojatno negdje između prve i druge pretpostavke (**CPU olakotne okolnosti**). Dodatno imamo i priču s drugim arhitekturama poput arhitekture tvrtke **Tilera** (sada u vlasništvu tvrtke **Mellanox**). Konkretno za *CPU Tile-Gx*, a što je vidljivo u referentnom dizajnu **MikroTik** usmjerivača : CCR1072-1G-8S+ i njegovim specifikacijama (dolje u tablicama).

Dakle ovdje se radi o 72 *CPU* jezgre na 1GHz, gdje ovakav dizajn osigurava maksimalnu propusnost od **119 kpps**, za 64 *byte* pakete (prema njihovim specifikacijama samog uređaja).

Slično je i s njihovim preklopnikom sa 24 x 1 Gbs CRS226-24G-2S+RM a koji koristi *switching chip* **QCA8519**, tvrtke **Qualcomm Atheros**.
Rezultat ovakvog dizajna je propusnost vidljiva u tablici (na dnu njihove službene stranice - iz linka gore). Ovdje je propusnost za 64 *byte* pakete, **65 kpps**

`kpps` označava broj tisuća poslanih ili primljenih mrežnih paketa u sekundi, pa 119 *kpps* znači 119 000 paketa u sekundi.
`100 kpps = 0.1 Mpps.`

Upravljački programi, drugi softver te optimizacije OS-a

Još jedan element u ovoj priči su upravljački programi kao i druge softverske komponente niže razine. One su u pravilu proporcionalno loše ili dobre ovisno o proizvođaču, namjeni i na kraju modelu komponente ili samog *chipa*. Iz mog iskustva pouzdaniji igrači imaju bolje upravljačke programe (*drivere*) koje redovito održavaju i optimiziraju, dok oni s manje "resursa" ili naglaska na kvalitetu uporno ostaju loši. Ovdje na vidjelo izlaze stvarno izvrsni programeri i njihove vještine. Naime svima je jasno kako se bilo koji problem, programski može riješiti na gotovo beskonačan broj načina, ali je samo nekoliko onih koji su posebni, brzi i pouzdani ali i čitljivi drugima. Kako je vrijeme razvoja i testiranja hardvera vrlo važno, jednako važno je i vrijeme razvoja i testiranja, prvenstveno upravljačkih ali i drugih programa niže razine/namjene.

Kao očiti primjer uzeti ćemo razvoj programa koji mora raditi drastično brzo, pouzdano i sa što manjim zauzećem RAM memorije. Dakle radi se o programu `haproxy`, koji je softverski *load balancer* za *http*, *https* i *TCP* protokole. Ovaj primjer je prilično blizak razvoju upravljačkih i drugih programa niže razine, koji na kraju barataju s mrežnim paketima i konekcijama.

U razvoju ovog programa, a koji je trajao godinama, glavni programer se susreo s mnogim problemima - od onih kada se upotrebom postojećih biblioteka (engl. *library*) za **C** jezik nisu dobile dovoljno dobre performanse, do desetaka i desetaka slučajeva u kojima se moralo paziti na svaki kB RAM memorije, za svaki pojedini takt procesora, za svaku novu konekciju, smanjivanje opterećenja *CPUa*, kao i pristupa RAM memoriji. U konačnici, nakon godina i godina razvoja, konstantnih optimizacija i testiranja, ovaj program je postao najbrži i najpouzdaniji, široko poznati softverski *load balancer*.

Za više detalja o osnovnim optimizacijama ovog programa, pogledajte http://www.haproxy.org/#perf.

Druge primjere optimizacije upravljačkih programa ili načina pristupa mrežnim karticama ćete upoznati malo kasnije u tekstu koji slijedi, a odnosi se na tehnologije odnosno *frameworke*: **DPDK, netmap i PF RING** [32], [33], [34], [35], [36] i [37].

Polling mehanizam

Mrežne kartice, kao i drugi uređaji (pr. disk kontroleri) kada im je potrebna pažnja *CPUa*, odnosno kada od *CPUa* traže procesorsko vrijeme za obradu njihovih zahtjeva, generiraju signal prekida (Engl *interrupt*). U slučaju mrežne kartice, kod svakog novog mrežnog paketa koji je zaprimljen, mrežna kartica kreira signal prekida. *CPU* tada zaustavlja trenutni proces s obradom, tj, pauzira ga, privremeno pohranjuje međurezultate, prazni memoriju i registre, te preuzima na obradu zadatak od mrežne kartice (preuzima na obradu taj, konkretni mrežni paket).
Kako smo vidjeli, za gigabitne brzine (1Gbps) to znači 1.48 milijuna paketa u sekundi, koje mrežna kartica može primiti ili poslati.
Jasno je kako se broj signala prekida kod ovakvog rada, također mora popeti na isti broj (1 paket = 1 *interrupt*).
Mehanizmi implementirani u *Linux*, *FreeBSD* i druge moderne operacijske sustave, u slučajevima, kada broj mrežnih paketa i samim time *interrupta* postaje prevelik, da bi zagušio cijeli operacijski sustav i sve programe, prelazi u drugačiji način rada a koji ovisi o tome da li ga uopće podržava upravljački program mrežne kartice kao i sam hardver kartice. Dakle u tom trenutku se prelazi na rad u kojem se više ne generiraju *interrupti* za svaki paket, već se sustav počinje brinuti o tome, kako bi svako malo, s posebnim mehanizmima, sam dohvaćao pakete od mrežne kartice (Engl. *Polling* mehanizam).
Na ovaj način sam sustav se počinje brinuti i o tome kako će opsluživati ostale procese (programe) na najnižoj razini. Kod boljih implementacija ovdje se pazi na Tzv. context switching na najbolji mogući način i time se dodatno dobiva na performansama jer se *CPU* prebacuje između procesa na način koji je optimalan.

U konačnici:

- izbjeglo se zagušenje cijelog sustava zbog generiranja prevelikog broja *interrupta*
- paketi koji se povlače (*poll*) s mrežne kartice na obradu, mogu se dohvaćati u nizu
- kod najgorih slučajeva, kada se tolika količina paketa ne može obraditi niti ovom metodom, sustav ih odmah odbacuje na samoj mrežnoj kartici, bez nepotrebnog dodatnog procesiranja od strane *network stack* dijela kernela operacijskog sustava [40], [41] i [42].

Ponoviti ćemo: sve ovo vrijedi samo za one mrežne kartice i njihove upravljačke programe koji podržavaju *polling* način rada, pod uvjetom kada ga podržava i operacijski sustav i kada je ta opcija opće uključena.

Interrupt moderation

Novije i snažnije mrežne kartice danas imaju i ***Interrupt coalescing*** znan i kao ***interrupt moderation*** mehanizam. U slučaju upotrebe ovog mehanizma (uz pripadajući upravljački program), mrežna kartica ne generira signal prekida za svaki pojedini mrežni paket, već tek kada joj se međuspremnici napune većim brojem paketa. Dakle signal prekida se generira za više paketa a ne za svaki pojedinačno. Ova metoda značajno smanjuje broj signala prekida, te može ubrzati rad mreže [(43)].

Pod operacijskim sustavom **Linux**, obije metode (***Pooling*** i ***Interrupt moderation***) implementirane su u novi mrežni *API* (*application programming interface*), koji se zove **NAPI**. Osim toga implementiran je i zaštitni mehanizam, koji se aktivira u slučajevima potpunog zagušenja sustava, poznatog pod nazivom **Packet throttling**. Dakle kada je mrežna kartica toliko zagušena, ona nove mrežne pakete neće niti pokušavati prosljeđivati dalje prema procesoru, uzrokujući dodatno zagušenje sustava, već će svi novopristigli mrežni paketi tada biti odbačeni, direktno od strane mrežne kartice, točnije od strane njenog upravljačkog programa, koji je napravljen korištenjem *NAPI*ja. [(54)]

Optimizacije operacijskog sustava

Osim odabira provjerenog hardvera, upravljačkih i drugih programa, te nužnih novih tehnologija koje je potrebno implementirati na razini operacijskog sustava, uvijek su potrebne stotine i stotine optimizacija, na raznim područjima operacijskog sustava, krenuvši od signala prekida (*interrupta*) te *polling* mehanizama, preko vezanja specifičnih procesa za određene *CPU* jezgre [(26)], optimizacije u slučajevima kada se koristi NUMA arhitektura.

Dakle kod *NUMA* arhitekture (računala s dva ili više fizičkih procesora) poseban oprez je potreban kod povezivanje mrežne kartice, njenog signala prekida (*interrupta*), ovisno o pripadnosti fizičkom procesoru i njegovoj lokalnoj *RAM* memoriji, kao i lokalnoj sabirnici na koju je spojena mrežna kartica.

Potrebne su i razne modifikacije na razini upravljačkih programa, pristupa i rada s DMA i slično [(29)] i [(30)].

I na kraju dolaze optimizacije iz područja mreža, iz samog operacijskog sustava:

- optimizacija parametara drivera
- raznih međuspremnika (pr. *RX, TX* i drugi *bufferi*)
- raznih *timera* i *timeouta*
- redova za obradu (*queues*)
- drugih stotina vrlo specifičnih optimizacija, na svim TCP/IP slojevima.

Navesti ćemo i jedan primjer, iz kojega će postati jasno koliko optimizacije na razini samog operacijskog sustava, mogu drastično ubrzati obradu mrežnih paketa. Inicijalni rad iz 2000.g. na Click Router projektu, u kojemu su napravljene mnoge optimizacije postojećeg *Linuxa*, korištenjem *Click* arhitekture, pokazao je kako su dobivene **četiri puta** veće performanse na području obrade mrežnih paketa. [(35)]

Sabirnica

Pri dosadašnjim analizama nismo spominjali brzinu sabirnice između *Switching chipa* i *CPUa*, koja bi morala moći podnijeti sve ove brzine, kao i brzine memorijske sabirnice i u konačnici RAM memorije koja bi to morala također moći izdržati.

Probajmo kratko analizirati i to.

Dakle preklopnik sa 24 x 1Gbps *porta*, mora imati minimalnu unutarnju propusnost od 48Gbps, zbog *full duplex* načina rada.
Brzina od 1Gbps = znači maksimalno 125MB/s
Full duplex 1Gbps je maksimalno 250 MB/s
U ovom slučaju 48Gbps = 125 MB/s * 48 = 6.000 MB/s = 5.8 GB/s

Sjetimo se samo kako **Intel Skylake** arhitektura sa pripadajućim Z170 *chipsetom* ima maksimalnu propusnost sabirnice između *CPUa* i *Southbridge chipseta* (DMI v.3.0) od nekih 4 GB/s.

Dakle u ovom slučaju niti **Intelova** 6 generacija Pentium 4 procesora (**i5-6xxx** ili **i7-6xxx**) nema dovoljnu propusnost za tako nešto. Memorijska sabirnica u ovom slučaju ima propusnost od 34 GB/s pa bi ona zadovoljila ali nedostaje brzine prema *Southbridge* dijelu na koji je spojena mreža.
Za više informacija o arhitekturi računala pogledajte knjigu **Uvod u Linux u Linux napredno** - poglavlje "Matična ploča, CPU, "chipset" i sabirnica(e)"

Osim toga koliko god procesor bio brz, baratanje s mrežnim paketima će se i dalje događati unutar granica milisekundi.
Pozitivno je kako ipak određeni postotak obrade paketa uspije odraditi *Switching chip* ali na kraju sve ovisi o proizvođaču – koji je točno model *switching chipa* odabrao (odnosno koliko je bio spreman platiti – mada su često razlike u cijeni smiješne – ali razlika od 1 U). Reći ću samo:

postoje proizvođači i “proizvođači”. Pa čak ovisi i o seriji odnosno modelu uređaja gdje proizvođač za relativno malu razliku od nekoliko stotina KN ili malo vise od toga (u KN), nudi poprilično više ili manje.

Sabirnica za mrežne (i druge) kartice
Sada pogledajmo brzine sabirnica, na koje spajamo mrežne (i druge) kartice.
Starije vrste sabirnica (**PCI** i **PCI-X**)

Naziv	Brzina rada (MHz)	Širina sabirnice	Propusnost Mb/s
PCI	33 MHz	32 bitna	1,064 Mb/sec
PCI	33 MHz	64 bitna	2,128 Mb/sec
PCI	66 MHz	32 bitna	2,128 Mb/sec
PCI	66 MHz	64 bitna	4,256 Mb/sec
PCI-X	100 MHz	64 bitna	6,400 Mb/sec
PCI-X	133 MHz	64 bitna	8,192 Mb/sec

Novije sabirnice (**PCIe** - PCI Express), kod kojih se komunikacija odvija prema Tzv. komunikacijskim kanalima ili trakama (Engl. Lane), svaki kanal je dvosmjeran, i zapravo se sastoji od dva kanala (para) jedan za primanje i jedan za slanje podatka.

Jedna traka - (*per lane*)	Propusnost Gb/s (zaokruženo) - u svakom pojedinom smjeru	Dvosmjerno Gb/s	Upotreba
PCIe - 1x v.1	2 Gbps	4 Gbps	do 2 x 1Gbps
PCIe - 1x v.2	4 Gbps	8 Gbps	do 4 x 1Gbps
PCIe - 1x v.3	8 Gbps	16 Gbps	do 8 x 1Gbps ili 1 x 10 Gbps
PCIe - 1x v.4	16 Gbps	32 Gbps	do 16 x 1Gbps ili do 2 x 10 Gbps

Maksimalne brzine su prema tome (sa 16 traka):

Prema trakama (*per lane*)	Propusnost Gb/s (zaokruženo) - u svakom pojedinom smjeru	Dvosmjerno Gb/s	Upotreba
PCIe - 16x v.2	64 Gbps	126 Gbps	do 1+ x 40Gbps
PCIe - 16x v.3	126 Gbps	252 Gbps	do 1+ x 100 Gbps
PCIe - 16x v.4	252 Gbps	504 Gbps	do 2+ x 100 Gbps

Za ostale brzine/trake/*lanes*: 2x, 4x i 8x, izračunajte si sami. U svakom slučaju, važno je kako sabirnica mora biti dovoljno brza kako bi podnijela brzine svih mrežnih kartica koje ugrađujete u računalo, poslužitelj ili mrežni uređaj (obično *router* ili *firewall*) [(28)] .

Vrijeme obrade svakog mrežnog paketa

Zanimljivo bi bilo i vidjeti koliko je vremena potrebno kako bi se procesirao svaki pojedini mrežni paket. Vidjeti ćemo unutar kojih vremenskih okvira se svaki mrežni paket mora uspjeti pripremiti i obraditi te poslati na mrežu, kako ne bi došlo do gubitka paketa.
Za pakete veličine 64 *byte* (*Payload*), ako govorimo o brzinama **1 Gbps**, tada imamo 1.488.095 paketa u sekundi. Izračunajmo koliko vremena imamo, kako bi se paket obradio i poslao na mrežu:

```
1 / 1.488.095 x 1.000.000.000 (ns) = 672 ns
```

Ako s druge strane govorimo o većim brzinama - primjerice **10 Gbps** (isto za pakete veličine 64 *byte*) - tada imamo manje vremena:

```
1 / 14.880.952 x 1.000.000.000 (ns) = 67,2 ns
```

To znači, kako procesoru (*CPU*), koji radi na 2GHz, jedan takt (*clock*) traje **0.5ns**. Prema tome ovisno o brzini mreže imamo slihedeći izračun:
Tablica za pakete veličine 64 *byte* (*Payload*) i 2GHz CPU

Brzina mreže	Maksimalno vrijeme obrade	Broj taktova/ciklusa CPU unutar koji se mora odraditi svaki paket
1 Gbps	672 ns	1340 ciklusa
10 Gbps	67.2 ns	134 ciklusa
40 Gbps	16.8 ns	33.5 ciklusa
100 Gbps	6.7 ns	13.4 ciklusa

Dakle za brzinu od 1 Gbps, procesor ima vremena 672ns kako bi obradio svaki mrežni paket i poslao ga na mrežu. Povećanjem brzine mreže na 10 Gbps to se vrijeme smanjuje na samo 67,2ns unutar kojih se svaki mrežni paket mora obraditi. Dakle za 10Gbps za *CPU* koji radi na 2GHz, svaki mrežni paket mora obraditi unutar 134 takta procesora, što je vremenski okvir od 67.2ns. Za još veće brzine (40 Gbps i 100 Gbps) problemi postaju sve veći. [(31)]. Tablica gore, pokazuje vremena, za najmanje mrežne pakete veličine 64 *byte*, jer je najzahtjevnije raditi s velikim brojem malih paketa.

Sada ćemo pogledati kako to sve izgleda, kada imamo znatno veći paket veličine od 1500 *byte*.
Tablica za pakete veličine 1500 *byte* (*Payload*) i 2GHz CPU

Brzina mreže	Maksimalno vrijeme obrade	Broj taktova/ciklusa CPU unutar koji se mora odraditi svaki paket
1 Gbps	12300 ns	24600 ciklusa
10 Gbps	1230 ns	2460 ciklusa
40 Gbps	300 ns	614 ciklusa
100 Gbps	123 ns	246 ciklusa

Payload se odnosi na koristan dio mrežnog paketa - pogledajte poglavlje Izračun

Sada vidimo kako je lakše obrađivati manji broj većih mrežnih paketa nego veći broj manjih, što je i logično.
Zbog razumijevanja ove problematike, moramo znati kako se unutar vremenskih okvira u kojima se svaki paket mora obraditi događaju i druge stvari, poput :

- dohvaćanja podataka iz priručne (*cache*) memorije procesora:
  - *L1 cache*: oko 5ns
  - *L2 cache*: oko 5-7ns
  - *L3 cache*: oko 7-9ns
- U slučaju kada podaci nisu u priručnoj memoriji procesora (*CPU*), dolazi do promašaja, te se moraju ponovno dohvatiti: ova vremena mogu biti između 10ns i 40ns (i više). Primjerice za **Intel Xeon E5-2650**, ovo vrijeme je 32ns.
- Dodatno, za prebacivanje na određeni proces, događa se *lock* odnosno *mutex* operacija kojoj treba od 8ns do 17ns (trebamo dvije ovakve operacije za svaki paket) - recimo da je to 17ns.
- I na kraju (zapravo se ovo događa na početku) sistem mora nekako odreagirati kod primanja svakog mrežnog paketa. Ovdje se radi o vremenu potrebnom kako bi se uopće odradio određeni sistemski poziv i pokrenula cijela procedura (sve točke od gore), a nerijetko je ovo vrijeme u granicama između 40ns i 90ns.

Navedena vremena obrade su za *Intel Xeon E5 26xx* poslužiteljske procesore, a za slabije i *desktop* varijante su ova vremena još lošija (veća).
Za Gigabitne brzine je još sve u granicama upotrebljivosti, mada i mnogi proizvođači 1Gbps opreme također imaju velikih problema s performansama. Prelaskom na brzine od 10 Gbps, definitivno nastupaju problemi, te se ovdje potrebno posebno potruditi (što vrijedi poglavito za proizvođače *hardvera*, *drivera* i drugih sistemskih komponenti= kako bi sve stvarno i radilo dovoljnom brzinom, bez usporavanja ili gubitka mrežnih paketa. Kako je vidljivo iz tablice, granične vrijednosti kreću već od 10Gbps brzina, stoga su optimizacije prijeko potrebne, kako bi se što više smanjila vremena odziva i obrade.

Iako izgleda kao teorija, stvarno su moguće i dostupne razne tehnike i tehnologije za tu svrhu a jedan od primjera je upotreba sistemskih biblioteka, koje su visoko optimizirane za brzu obradu mrežnih paketa - **DPDK** [32], koji unutar **80 CPU** ciklusa, modernih procesora, može primiti ili poslati pojedini mrežni paket.

Drugi primjer je **netmap** *framework* [33], koji je također razvijen za sličnu namjenu, a koji omogućava programima mogućnost procesiranja i prosljeđivanja mrežnih paketa, vrlo velikim brzinama (unutar **90 CPU** ciklusa) [30], korištenjem *netmap API*. **netmap** *Framework* se danas standardno koristi na FreeBSD Unixu (još od inačice **9.1**) ali sve više i na Linuxu, na kojemu je integriran i sa KVM/QEMU za virtualizaciju [34].

Slijedeći primjer je **PF RING** koji je razvijen za potrebe brzog dohvaćanja, filtriranja i analize mrežnih paketa, najviše za potrebe programa za analizu mrežnog prometa, poput programa ntop odnosno ntopng. [37]

U sva tri slučaja, zbog dobivanja drastično boljih performansi, zaobilazi se *kernel* ili njegov *network stack* i omogućava pristup fizičkoj (ili virtualnoj) mrežnoj kartici, preko *API* poziva koje nam daje neki od navedenih *frameworka*. Drugi primjer je razvoj visoko optimiziranih upravljačkih programa, koje smo već prije spomenuli.

Hyper-Threading
Neki izvori [15] i [24] govore i kako upotreba tehnologije poput *Hyper-Threadinga* odnosno "uključivanja" logičkih CPU jezgri na *Intel* procesorima (isto vrijedi i za *AMD SMT* ekvivalent), donosi dodatne probleme kod obrade mrežnih paketa . Točnije u zadaćama u kojim su potrebne što manje latencije uzrokovane slučajevima promašaja čitanja iz priručne memorije *CPUa* Tzv. *cpu cache misses*, preporuća se isključiti ovu tehnologiju (obično u *BIOSu*).

Napredak tehnologija

S vremenom se razvijaju i noviji i brži procesori ali i operacijski sustavi kao i brži upravljački programi i pripadajuće mrežne kartice te razne druge tehnologije. Govorili smo o *routingu* i preporukama za odabir hardvera, jer je to bio najočitiji i najdohvatljiviji primjer, kako bi opisali problematiku ali i zahtjevnost ovih operacija. Priča o *routingu* opisuje i *multilayer switching*, koji objedinjuje i klasični *switching* ali i *routing*.
Još jednom ću naglasiti koliko samo pisanje boljeg programskog koda, očekivano može poboljšati performanse. Najnoviji primjer je NASA i njihov program poboljšanja programskog koda u programu za simuliranje dinamike fluida FUN3D. Oni su raznim metodama analize programskog koda zaključili, kako bi poboljšanje performansi, upotrebom bolje optimiziranog koda, bilo od 10 do 1000 puta ?!. Ovdje se radi o programu pisanom u programskom jeziku FORTRAN.

Drugi noviji primjer, koji je pak iz svijeta *routinga*, govori kako su optimizacijama na razini operacijskog sustava FreeBSD, pametnim odabirom mrežnih kartica i njihovih optimiziranih *drivera* ali i optimizacijama programskog koda zaduženog za obradu mrežnih paketa, dobivene

nevjerojatne performanse, korištenjem samo jedne jezgre *CPUa*, na 3.2GHz te dvije 10Gbps mrežne kartice. Osim navedenih optimizacija, koristili su i *Fast Data IO* framework [56], koji objedinjuje cijeli niz projekata.

Unutar ovog projekta su objedinjene razne biblioteke kojima je za cilj ubrzanje obrade mrežnih paketa, a koji se naslanja na DPDK[32] i druge tehnologije/projekte, poput *Vector Packet Processing (VPP)* biblioteka[koje je u *open source* donirala tvrtka Cisco Systems]. Samo ću napomenuti kako je cilj projekta *Fast Data IO*, u svakoj novoj inačici napraviti ubrzanja između 10% i 19%, koji su ponovno uspjeli ostvariti i prelaskom s v.17.01 na najnoviju v.17.04. Dakle upotrebom ovih tehnologija, dobivena su značajna ubrzanja a pogotovo za **IPSec** :

- 4x veće performanse kod korištenja hardverske akceleracije **AES-NI** unutar novijih procesora
- 14x veće perofrmanse ako se koristi **Intel QuickAssist** tehnologija implementirana u nove generacije mrežnih kartica.

Ono što je u konačnici postignuto s *FreeBSD* operacijskim sustavom, unutar ***pfSense*** plarforme, je propusnost između dvije 10Gbps mrežne kartice, od **14.6 Mpps**, korištenjem 64 *byte*nih paketa. Naime dobivena propusnost je 98% iskorištenosti 10Gbps veze (100% bi bilo 14.88 Mpps). [21] i [31] . Za veće brzine od 40Gbps, postignuta su također poboljšanja, kod *Layer 3* funkcionalnosti (usmjeravanje), te je dobivena brzina od **42.6 Mpps** za 64 *byte*ne pakete.

Što je još novo ?

Ako samo pogledamo najnovije generacije poslužiteljskih **Intel** mrežnih kartica, koje osim što su u stanju odrađivati sve više funkcionalnosti bez posredovanja procesora (CPU), opremljene su i raznim dodatnim naprednim tehnologijama, poput:

- ***Intel® Data Direct I/O*** - pomoću ove tehnologije, moguća je direktna komunikacija mrežne kartice s *cache* memorijom procesora (CPU), bez prolaska kroz sistemsku memoriju. Ovom tehnologijom se dodatno smanjuje latencija i povećava propusnost (i naravno potrošnja el. energije).
- ***TCP/IP checksum offload*** tehnologija je s kojom mrežna kartica sama izračunava TCP/IP provjerni zbroj (*checksum*), za svaki TCP/IP mrežni paket te tako rasterećuje *CPU*
- ***Large send offload (LSO) i TCP segmentation offload (TSO)*** tehnologije. U normalnom radu *CPU* podatke, koji se trebaju poslati na mrežu tj. prvo prema mrežnoj kartici, razlama na male segmente, veličine koje je definirana kao **MTU** (*Maximum Transmission Unit*) a koji se onda pretvaraju u mrežne pakete i šalju na mrežu. Ovaj proces se zove segmentacija i odrađuje ga *CPU* na *TCP* sloju (ako je TCP odabrani transportni protokol). U slučaju kada se prema mreži mora poslati 64KB podataka, *CPU* ih prvo mora razlomiti na manje dijelove - pr. veličine 1500 *byte*. Dakle *CPU* mora prema mrežnoj kartici slati : 64 KB = 65.536 *byte* / 1500 *byte* = 44 segmenta, jedan po jedan, te ih mrežna kartica tako i šalje na mrežu (kako su došli). Upotrebom *LSO* odnosno *TSO*, *CPU* ne mora sam segmentirati (razlamati) podatke u male dijelove, već može odjednom poslati svih 64KB prema mrežnoj kartici, koja ih sama razlama (segmentira) te šalje na mrežu. [23]
- ***Large Receive Offload (LRO)*** tehnologija radi slično poput *LSO* ali u drum smjeru. Dakle niz paketa koji je primljen na mrežnu karticu, se pohranjuje u međumemoriji na mrežnoj kartici te se svi ti segmenti odjednom šalju višim slojevima *TCP/IP*a, na obradu - prema *CPU*, a ne jedan po jedan, što bi bio slučaj bez upotrebe ove tehnologije.
- ***TCP Offload Engine (TOE)*** koji nudi dodatna rasterećenja *CPU*a jer praktično mrežna kartica preuzima gotovo svu obradu nad mrežnim segmentima/paketima a komunikacija prema *CPU*, kao i njegovo opterećenje su minimalni [24] i [25].
- ***Receive Side Scaling (RSS), Receive Packet Steering (RPS)*** tehnologija koja distribuira dolazne pakete na više CPU jezgri i to na način da svi paketi koji dolaze s jedne logičke konekcije (pr. pojedina TCP konekcija : klijent-poslužitelj) uvijek završe na istoj CPU jezgi. Svaki niz paketa u konačnici završava u jedom nizu za obradu (*receive queue*) a svakom od ovih nizova se dodjeljuje zaseban signal prekida (*interrupt*), kojim upravlja druga CPU jezgra, povećavajući performanse cijelokupne obrade. Ova funkcionalnost se odrađuje izračunavanjem *hasha* izvorišne i odredišne IP adrese a na jačim mrežnim karticama i ovu zadaću može odrađivati mrežna kartica. *Intel* je ovu tehnologiju patentirao i nazvao *Hashing packet contents to determine a processor* (i neki drugi proizvođači ju također imaju ali ju nazivaju drugačije). Dakle ovdje se odrađuje i redistribucija signala prekida i raspodjela poslova obrade, ovisno o svakoj konekciji.

- ***Receive Flow Steering (RFS)*** - **RFS** tehnologija s druge strane ne koristi samo *hash* za preraspodjelu primljenih mrežnih paketa, prema CPU jezgrama (Tzv. pseudo raspodjelu), već puno bolji način raspodjele, pomoću dodatne tablice a koja kao rezultat im bolju raspodjelu paketa na CPU jezgre na kojima i trebaju završiti a ovisno o aplikacijama koje ih koriste. Ovu tehnologiju *Intel* naziva *Ethernet Flow Director* (Podržanu na mrežnim karticama X520, X540, XL710 ili novijim) - Video. U svakom slučaju navedene tehnologije donose znatna ubrzanja - za RPS za FreeBSD pogledajte [17] te za RFS [18] i [22]
- ***Transmit Packet Steering (XPS)*** tehnologija je slična *RFS* tehnologiji ali djeluje u suprotnom smjeru - ne za primanje mrežnih paketa, već za njihovo slanje. Dakle svaki zasebni tok mrežnih podataka se šalje s druge CPU jezgre. [20]
- ***Quick Assist***[55], integrirana i u neke modele CPUa (55.1)] tehnologija koja hardverski ubrzava operacije poput:
    - kriptiranja i dekriptiranja (AES, DES, 3DES, ARC4)
    - *hash* operacija (SHA-1, MD5, SHA-2, SHA-224, SHA-256, SHA-384, SHA-512)
    - generiranja slučajnih brojeva (*random number generation*)
    - komprimiranja i dekomprimiranja (DEFLATE, LZV)
    - ostale kriptografske funkcije: RSA, Diffie-Hellman, ECDSA i ECDH, …
- ***Virtual Machine Device Queues (VMDq)*** u kombinaciji sa *Single-Root I/O Virtualization (SR-IOV)* - koji ubrzava rad u virtualizaciji, poglavito u slučajevima kada koristite *router* ili *firewall* kao virtualno računalo.
- ***SERDES Interface*** - zadužen za serijalizaciju i deserijalizaciju podataka od i prema mrežnoj kartici - isto može značajno poboljšati performanse
- ***On-chip QoS and Traffic Management*** - dakle sve vezano za *QoS* (*Quality of Service*) i kontrolu prometa
- … i mnoge druge tehnologije

I naravno, kao i u raznim primjerima do sada, mnoge navedene ali i još više drugih nenavedenih naprednih tehnologija, baš i ne radi kako treba, u svim kombinacijama hardvera i softvera. Je li pitanje loše hardverske ili softverske implementacije odnosno *drivera* ili podrške unutar operacijkog sustava, na kraju nije niti važno, važan je oprez, kod odabira kartice i *drivera* uz pripadajuću inačicu operacijskog sustava, kao i uključivanja funkcionalnosti koje postoje na određenim karticama.

Nadalje, nije dovoljno samo poukljućivati sve dostupne funkcionalnosti, bez jako dobrog i detaljnog poznavanja, što one uopće rade i čemu stvarno služe. Naime trivijalizacijom i brzanjem, vrlo često se dolazi do značajnih komplikacija, koje se obično očituju u kasnijem radu, tek nakon nekog vremena upotrebe. Stoga optimizacijama treba pristupiti inženjerski, uz intenzivno i dugotrajno testiranje, jer uključivanje jedne funkcionalnosti, može biti povezano s nekom desetom problematikom, koje se postaje svjestan tek nakon intenzivnog proučavanja i testiranja ili s druge strane u nedostatku istog, nakon što smo sve stavili u produkcijsku upotrebu i prouzročili probleme ili neku katastrofu manjih ili većih razmjera.

Ponoviti ću : Testiranje, testiranje i još testiranja, prije konačne odluke.

Postoji li rješenje za preklopnike

Zbog svih navedenih ograničenja, većina "jačih" proizvođača, počela je tražiti odgovor koji stvarno može riješiti ove probleme vrlo loše propusnosti preklopnika ali i jačih verzija usmjerivača.

Implementacija ASIC chipova.

Ovo rješenje zapravo nije ništa toliko novo – u širokoj upotrebi je sigurno već desetak i više godina, samo što je u današnje vrijeme postalo široko dostupno (i relativno jeftino).

ASIC (*Application Specific Integrated Circuit*) *chipovi* su specijalizirani *chipovi* koji imaju hardverski implementirane sve potrebne funkcionalnosti koje bi inače morao obrađivati *CPU*. S druge strane *ASIC* chipovi su povezani sa *switching chipovima* ili posebnom vrlo brzom sabirnicom ili se sam *switching chip* nalazi integriran unutar njih. Na ovaj način su se otklonili i drugi problemi loše propusnosti, s kojima smo se upoznali.

U praksi se pokazalo kako se upotrebom *ASIC chipova* u odnosu na klasični *CPU* i *switching chip*, dobivaju poboljšanja performansi **minimalno 500** puta, a vrlo često čak i do **1000** puta.

I ovdje postoje razlike. Naime postoje *ASIC* rješenja koja zadovoljavaju samo osnovne potrebe jednostavnih preklopnika ali i ona sa podrškom za sve moguće i nemoguće protokole.

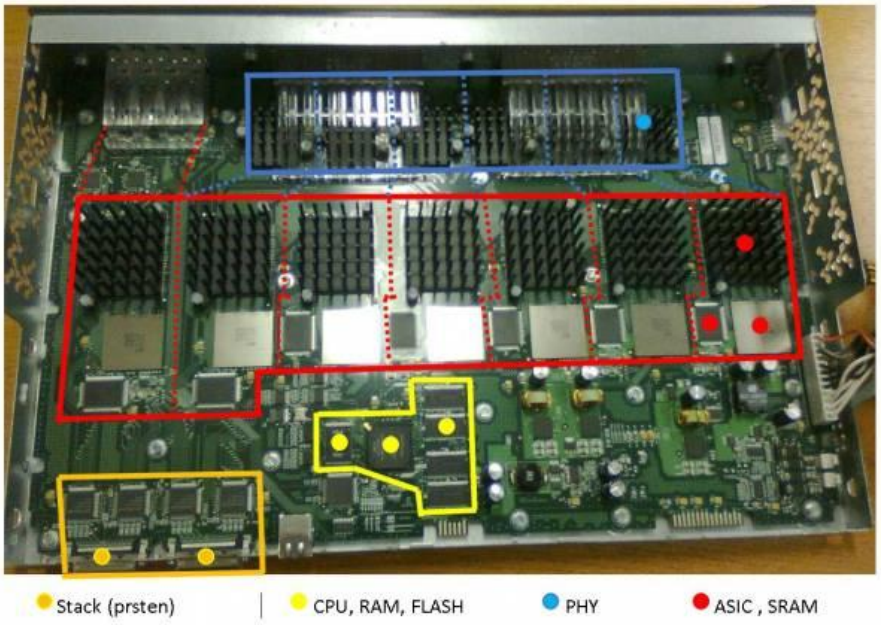
Vezano za rad ASICa, možemo povući paralelu s hardverskim dekodiranjem video sadržaja koji danas s lakoćom obavlja bilo koja grafička kartica, za što bi bez ove hardverske akceleracije, bio potreban čak četverojezgreni *CPU* koji radi na taktu od 3 GHz (za H265 codec, 4K video).

Dakle napredniji *ASIC chipovi* rade kompletan *switching* ili *routing*:

- na osnovi *MAC* adresa (Layer 2)
- na osnovi *IP* adresa (Layer 3)
- na osnovi *portova* (Layer 4)
- *Traffic forwarding* (obrada i prosljeđivanje paketa)
- *QoS* (*Quality of Service*)
- *ACL lookup* (*Access* Liste)
- *Route Processing* (Obrada *Routing* funkcionalnosti)
- *STP* (*Spanning Tree* protokol)
- …

Osim toga snažnije verzije *ASIC chipova* imaju implementirane gotovo sve mrežne protokole koje susrećemo u preklopnicima, kao i većinu mrežnih protokola koje susrećemo u klasičnim *routerima*. U slučaju upotrebe kod **Multilayer** preklopnika implementirani su gotovo svi protokoli koje uređaj podržava.

Pogledajmo fotografiju jednog starijeg Cisco 3750G preklopnika:



Na slici je **Cisco 3750G** (preklopnik sa 24 x 1Gbps + 4 x 1Gbps SFP) – vidljivo je da jedan *ASIC+SRAM* (set od tri *chipa*) odrađuje samo 4 x 1Gbps te je na njega spojen jedan **PHY** (pripadajući *chip* gore) a koji je zadužen za ta četiri 1Gbps porta. Konekcije s *PHY chipa* završavaju na RJ-45 konektoru (u ovom slučaju). Pogledajte desnu stranu – i sve tako do 6-tog **ASIC**,**SRAM** i **PHY**.

Sedmi **ASIC+SRAM** nema svoj **PHY** (krajnje lijeva strana) jer sve završava na *SFP portovima* u koje se uključuju *SFP* moduli koji imaju svoj *PHY* (jer mogu biti optički ili električni sa **RJ-45** konektorom).

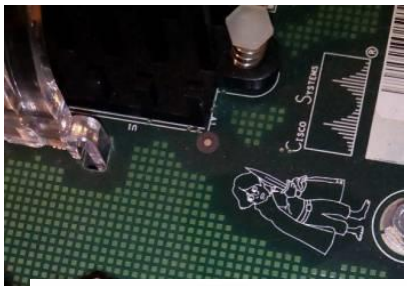
SFP moduli se često nazivaju i *Transceiveri* .

Lijeva slika prikazuje optički **SFP+** modul (sa LC optičkim konektorom) tvrtke **HP** (model J9150A) :

Desna slika prikazuje **Cisco 1000Base-T** (1Gbps sa RJ-45):

Malo zanimljivosti

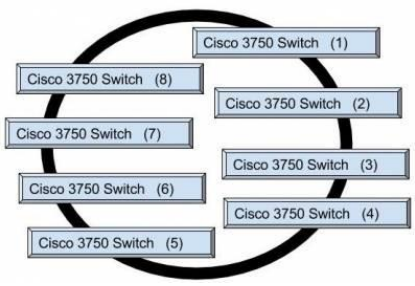
Molim vas pogledajte malo bolje gornji desni dio gornje velike slike **Cisco preklopnika Catalyst 3750G** (uvećana slika – dolje).



Radi se o tome kako **Cisco** za svaku generaciju uređaja ima kodno ime a pošto je ovo bio prvi model koji je koristio **Stackwise** tehnologiju koja omogućava povezivanje do osam preklopnika u prsten – prsten koji ih povezuje… kodno ime mu je bilo *Lord of the Rings*

"… One Ring to rule them all, One Ring to find them, One Ring to bring them all …"

Kada već govorimo o ovoj mogućnosti povezivanja do osam preklopnika serije 3750 u prsten (*Stack*) – to logički izgleda ovako:



Kod ovakvog načina spajanja, preko posebnih *Stack* konekcija visoke propusnosti (32Gbps ili 64Gbps – ovisno o modelu) preklopnici spojeni u prsten se počinju ponašati kao da se radi o jednom velikom preklopniku – sa strane konfiguracije i svih funkcionalnosti.

Dakle spajanjem do osam **Cisco 3750** preklopnika u *stack*, odjednom imamo veliki preklopnik sa puno dostupnih *portova*. Ova veza je direktna poveznica između *ASIC* chipova odnosno njihove sabirnice, a možemo reći da je to i direktna veza između *Matičnih ploča* preklopnika međusobno spojenih u prsten (*stack*). Stoga i dalje rade svi protokoli koji inače ne bi mogli raditi da se radi o nekom kvazi rješenju spajanja preklopnika u prsten u kojemu je svaki preklopnik na najnižoj razini vidljiv kao zaseban preklopnik.

Jedan od takvih protokola je i protokol za Agregaciju *portova* odnosno **EtherChannel** (prema *Cisco* terminologiji) a koji u svojoj komunikaciji, sadrži i podatak o fizičkom uređaju na koji je spojen.

Upravo zbog činjenice kako *EtherChannel* odnosno njegov ekvivalent prema IEEE, koji se zove **LACP** (*Link Aggregation Control Protocol*), u svojim mrežnim paketima nosi i informaciju o samom uređaju s kojeg paket dolazi, u nekim "jeftinim" pokušajima povezivanja preklopnika u

prsten, ovaj protokol uopće ne radi, ako je jedan dio konekcija unutar jedne *LACP* agregacije spojen na jedan preklopnik a drugi dio na drugi preklopnik u prstenu.

Iz fotografije prije (**Cisco 3750**), vidljivo je kako ova (starija) generacija **Cisco 3750G** preklopnika ima *ASIC* i *ASIC RAM* (CAM i TCAM) uz pripadajući *PHY* i *Switching chip* – samo za 4 x 1Gbps, dok novije generacije koriste mnogo snažnije *ASIC chipove* koji su u stanju "odraditi" do 24 x 1Gbps na *chipu* (nažalost nemam fotografiju nove generacije poput **Cisco 3750E** ali o njoj ubrzo).

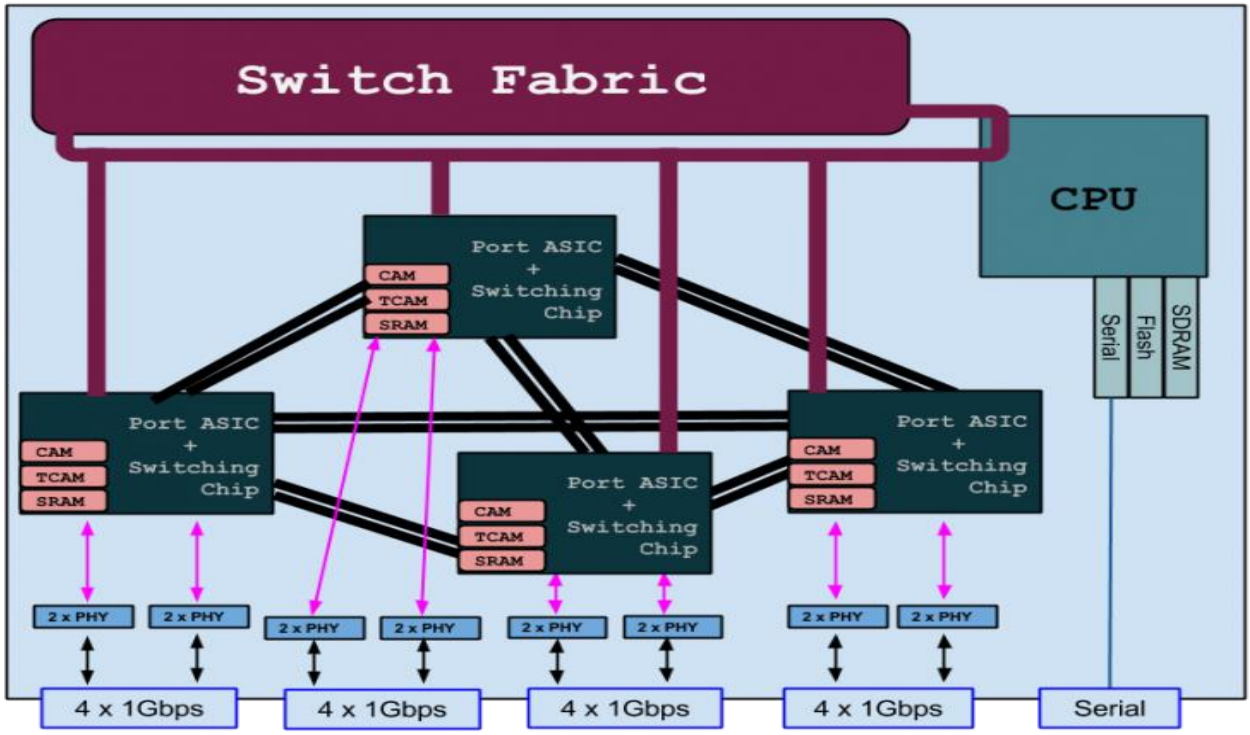
Jedina potencijalna slaba točka ovog dizajna je međuveza između *ASIC chipova* odnosno propusnost sabirnice koja ih povezuje. Ali o ovom djelu dizajna pogledajte tekst koji slijedi.

Mesh i druge topologije ASIC chipova

Nagađa se da **Cisco** i (drugi) veliki igraći zapravo koriste *Mesh* topologiju između A*SIC chipova*, preko koje su *ASIC chipovi* povezani sabirnicom velike propusnosti, te se s time ubrzala komunikacija između njih i komunikacija prema centralnoj sabirnici. Mana ovog dizajna je u tome što se sada za recimo ovakav preklopnik u primjeru koristi 10 *portni* (zamišljeni) *ASIC* koji koristi 6 *portova* za međusobno spajanje a samo 4 porta za spajanje prema *PHY* i na kraju fizičkim *interfaceima* (*portovima*).

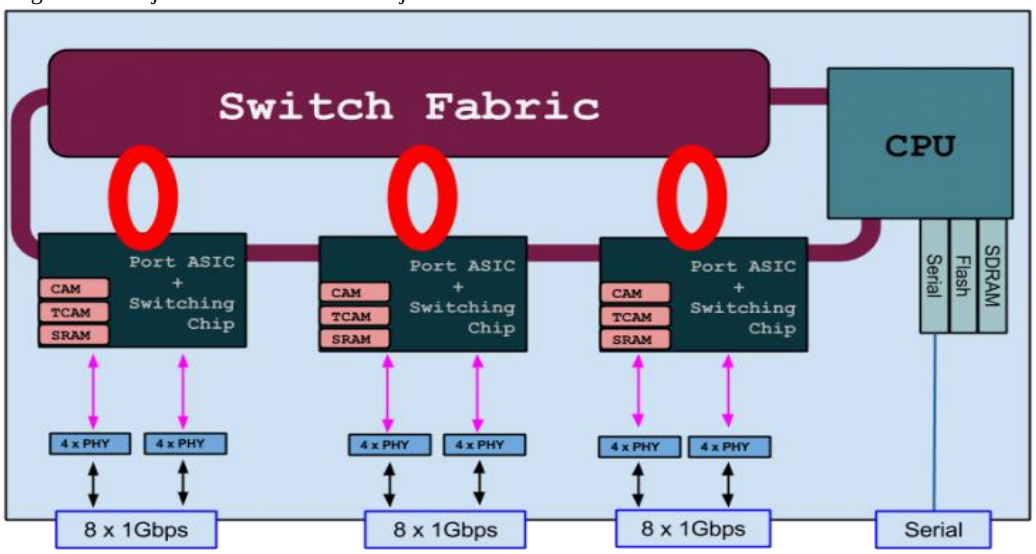
Dakle u konačnici u dizajnu na slici imamo 4 *ASIC chipa* sa 10 x 1Gbps *portova* (svaki) što je ukupno 40 x 1Gbps od kojih je upotrebljivo za spajanje (na vanjske *portove*) samo 16 x 1Gbps. Prema svemu sudeći međuveze između svakog *ASICa* bi u ovoj konfiguraciji trebale biti još veće (u ovom slučaju dvostruko ili još više).

Pogledajmo i *Mesh* dizajn (slika dolje).



I još jedna topologija

Druga vrsta dizajna bi bila ova na slici dolje:



Dakle kod ovog dizajna svi *ASIC chipovi* su spojeni direktno na *Switch fabric* međuvezom vrlo velike propusnosti, a sam *Switch fabric* interno je u stanju odrađivati sve što je potrebno, ekstremnom brzinom i propusnošću. Ovim dizajnom se postigla jednostavnost a zadržala brzina ali uz potrebu za znatno bržim *Switch fabricom*.

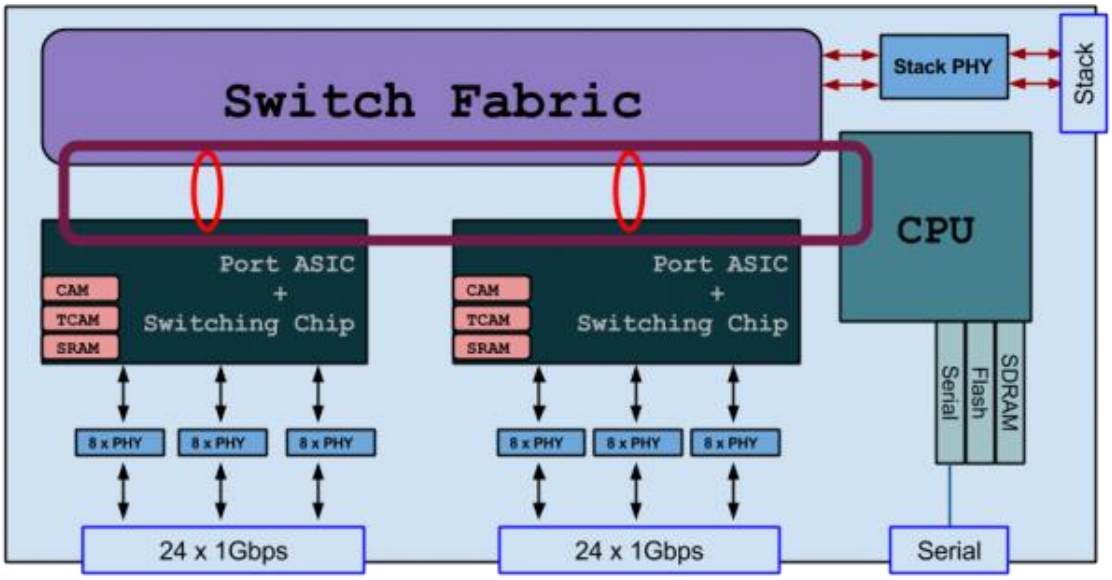
U svakom slučaju bilo koji dizajn koji uključuje *ASIC chipove*, omogućava postizanje vrlo velikih brzina procesiranja mrežnih paketa i to brzinama koje su neizvedive za standardne centralne procesore (CPU).

Dakle preklopnici koji imaju implementirane *ASIC chipove*, sve operacije i protokole (koje ASIC podržava) odrađuju 1000 puta brže od bilo kojeg *CPUa*.

U praksi to znači da će sve operacije koje *ASIC* obradi – poput *Switchinga* ili *Routinga* biti obrađene unutar granica nekoliko mikrosekundi za razliku od *Routinga* koji se inače na "običnim uređajima" u najboljem slučaju uspije obraditi unutar nekoliko milisekundi - dakle točno tisuću **(1000) puta sporije**.

Ponoviti ću – Ovo je dokazano u praksi – !!!! Razlika je tisuću puta brža obrada paketa !!!!

A sada pogledajmo i referentnu arhitekturu jedne novije generacije **Cisco 3750E** *Multilayer* preklopnika, koji upravo koristi gore navedeni dizajn:



Vidljivo je kako *ASIC* ima **CAM** i **TCAM** dio kao i vrlo brzu **SRAM** memoriju za širu upotrebu unutar *chipa* te kako je povezan sa **Switch Fabricom**. **Switch Fabric** je također posebna vrsta **ASICa** na čiju sabirnicu se spajaju svi *port ASIC chipovi* i *CPU*, kao i *Stack portovi* preko kojih se ova serija *svitcheva* može povezivati u prsten (*Stack*). Ovakva sabirnica je ekstremno velike brzine i propusnosti. Konkretni *Switch fabric* ima propusnost od 128 Gbps. Dodatno, u ovom dizajnu konkretno, postoje dvije vrste sabirnica:

- između **Switch Fabrica** i svakog pojedinog *port* **ASIC** *chipa* - kod ovog modela se radi o prstenastoj sabirnici propusnosti 32 Gbps prema svakom *ASIC chipu*
- između **Switch Fabrica**, svakog pojedinog *port* **ASIC** *chipa* i *CPUa* - ovdje se isto radi o prstenastoj sabirnici.

Ako pogledamo dolje, svaki od ovakvih *ASIC port chipova* podržava spajanje do 24x1Gbps **PHY** (*Physical Layer*) *chipova* koji su zaduženi za sam mrežni medij (*interface*) - obično bakar - RJ-45.
PHY mogu biti za bakar (RJ-45 konektor), optiku (bilo koji tip konektora za optiku) ili *SFP* port u koji se uključuje *SFP* adapter koji može biti ili za "bakar" ili za optiku.

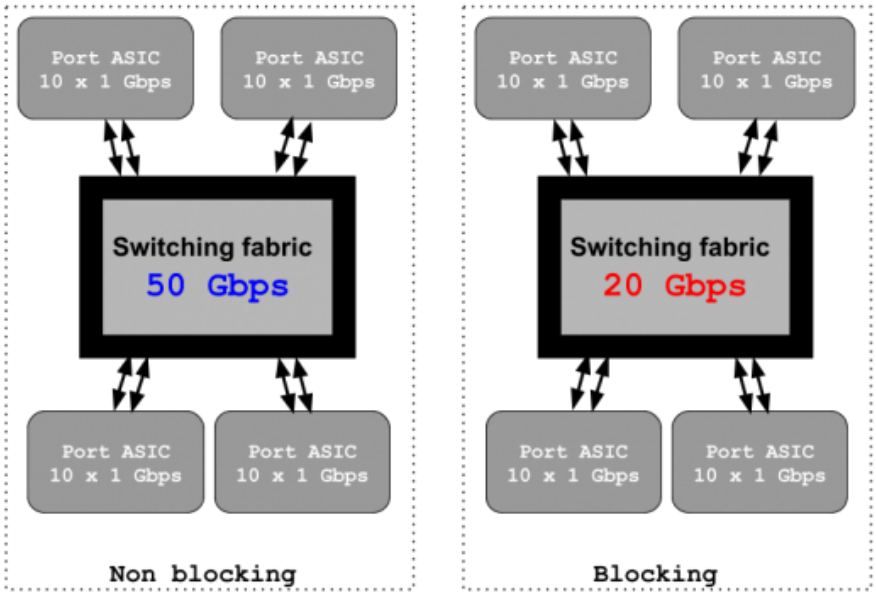
Dakle **PHY** chip je u konačnici *chip* koji sve pakete pretvara u električne signale ili optičke impulse za slanje na mrežu odnosno sam mrežni medij.

CAM dio (*Content Addressable Memory*) je zapravo tablica u kojoj se zapisuju i pretražuju mapiranjia MAC adresa – *port* na preklopniku, na osnovu koje se radi preklapanje na OSI sloju 2.
TCAM dio (*Ternary Content Addressable Memory*) je zapravo tablica sa mapiranjima na višim OSI slojevima (OSI 3) dakle ovdje se spremaju *routing* tablice i sl.
CAM i **TCAM** se nalaze u posebnoj superbrzoj memoriji *ASIC chipa* ili u starijoj generaciji kao zaseban *chip*, a koja omogućava nevjerojatno brzi pristup i pretraživanje navedenih tablica i prema tome ekstremno brze funkcionalnosti na OSI slojevima 2 (*switching*) i 3 (*routing*).
Naime osim velike brzine i propusnosti *ASIC chipova* i *Switch Fabrica* važno je znati kako je vrlo važna i brzina RAM memorije, stoga se i koriste **CAM** i **TCAM** memorije velike propusnosti i brzine rada. Bez obzira da li su ove vrste RAM memorije integrirane u sam *ASIC* ili su izvedene kao vanjske komponente, one se ekstremno većih brzina od svih današnjih RAM memorija, poput *DDR3* ili *DDR4*.
Tako je primjerice propusnost DDR4 memorija maksimalno do 200 Gbps. Kod vrlo velikih brzina mreža (s *portovima* koji rade na 10Gbps, 40Gbps ili 100Gbps) ovo postaje veliki problem. Neki proizvođači poput tvrtke **Juniper Networks** za svoje najjače serije preklopnika, uz razvoj vrlo snažnih *ASCI chipova* razvili su posebne vrste super brzih/propusnih RAM memorija - u zasebnim *chipovima*. **Juniper Networks** naziva ove *chipove* **Hybrid Memory Cube**.
Ovakva kombinacija **Juniper** *Q5 ASIC* i vanjske RAM memorije (**Hybrid Memory Cube**) postiže se propusnost od i prema ovoj memoriji od 1 Tbps (1.000 Gbps), između svakog *Q5 ASICa* i njegove **Hybrid Memory Cube** RAM memorije.
S time se sumarizirano dobiva ogromna propusnost jer se svaki par *Q5 ASICa* + **Hybrid Memory Cube** obično spaja na **Switch Fabric** vrlo visoke propusnosti.

Za više detalja o ovom **Juniper Networks** dizajnu, pogledajte izvore informacija (**13**)
Više detalja od **Cisco 3750E** seriji preklopnika možete dobiti iz Cisco dokumentacije - pogledajte izvore informacija (**10**) i (**11**).

Non-Blocking i Blocking dizajn

Za ono što bi bio **Non blocking** dizajn minimalno isto toliko *portova* koliko ih svaki *ASIC* propušta van (na mrežne *portove*) bi morala biti i veza prema svakom susjednom *ASICu* i *Switch fabricu* (ako se koristi).



Dakle *Non blocking* se odnosi na propusnost preklopnika, odnosno da li je on u mogućnosti osigurati nesmetan promet, bez blokiranja, usporavanja ili zaustavljanja, ako apsolutno svi *portovi/interafcei* rade na maksimalnoj mogućoj brzini.

To znači ako imamo preklopnik sa 24 x 1Gbps *porta* i na njega su spojena 24 računala s brzinom 1Gbps svaki, kako bi preklopnik morao moći obrađivati mrežne pakete ukupnom propusnošću 24 x 1Gbps `Full Duplex` ,bez ikakvog usporavanja ili blokiranja (o ovome malo kasnije).

Pogledajmo ova dva pojma na slici lijevo:

Pogled na Cisco preklopnik iz Cisco IOS operacijskog sustava

Pogledajmo i što nam kaže *Cisco IOS* (Cisco operacijski sustav), na nekoliko *Catalyst* serija preklopnika

Sada ćemo sve o čemu smo pričali - vezano za *ASIC chipove*, pogledati na stvarnim *Cisco* preklopnicima i to od najslabijih prema snažnijim kategorijama preklopnika :

- Catalyst 2960 [Layer 2, 10/100] (“najslabija” serija)
- Catalyst 3560G [Multilayer, 10/100/1000] (Layer2,3,4) – prvi *Multilayer* preklopnik
- Catalyst 3750G [Multilayer, 10/100/1000] (Layer2,3,4) – sličan kao 3560 samo što ima mogućnost *stackiranja* do 8 preklopnika preko specijalne međuveze vrlo visoke brzine (32+Gbps)
- Catalyst 3850 [Multilayer 10G] (Layer2,3,4) – s novim cisco *ASICom* (*Cisco Unified Access Data Plane* (**UADP**))

Koristiti ćemo naredbu koja će prikazati vezu između *ASIC chipa* i *interfacea* na preklopniku odnosno koji *interface* je spojen na koji *ASIC chip*.

Naredba je `sh platform pm interface-numbers` ili `sh platform pm if-numbers` ovisno o platformi. Izlistanje je malo duže tako da ću izvući samo važne detalje.

WS-C2960-24TC-L

```
Portovi         ASIC
-----------------------------------------------------
Fa 0/1 - 24     ASIC 0    (portovi 10/100 Mbps)
Gi 0/1 - 2      ASIC 0    (portovi 10/100/1000 Mbps)
```

WS-C3560G-24TS (24 x 1Gbps portova + 4 SFP porta (25-28))

```
Portovi              ASIC
--------------------------------
Gi 0/1 - 4       ASIC 1
Gi 0/5 - 8       ASIC 0
Gi 0/9 - 12      ASIC 3
Gi 0/13 - 16     ASIC 2
Gi 0/17 - 20     ASIC 6
Gi 0/21 - 24     ASIC 5
Gi 0/25 - 28     ASIC 4
```

WS-C3750G-48TS (48 x 1Gbps portova + 4 SFP porta)

```
Portovi              ASIC
-------------------------------------------------
Gi1/0/1 - 4          ASIC 6
Gi1/0/5 - 8          ASIC 5
Gi1/0/9 - 12         ASIC 8
Gi1/0/13 - 16        ASIC 7
Gi1/0/17 - 20        ASIC 4
Gi1/0/21 - 24        ASIC 3
Gi1/0/25 - 28        ASIC 10
Gi1/0/29 - 32        ASIC 9
Gi1/0/33 - 36        ASIC 2
Gi1/0/37 - 40        ASIC 1
Gi1/0/41 - 44        ASIC 12
Gi1/0/45 - 48        ASIC 11
Gi1/0/49 - 52 (SFP portovi)        ASIC 0
```

WS-C3850-12S-S (12 x 10Gbps portova) (ovdje je bilo malo teže povezati ASIC → port)

```
   (10Gbps portovi)         ASIC
-------------------------------------
Te1/0/1  -  12                1
```

Rezimirajmo

Što se događa na kojem sloju i kojom brzinom – za preklopnike koji koriste *ASIC*

1. **Layer 2 preklopnik** : sve se odrađuje na osnovi *MAC* adresa. Svi *Layer 2* protokoli koji su podržani u *ASICu* odrađuju se brzinom hardvera (tkzv. *Wire Speed*) – dakle 1000 puta brže nego na "običnim" preklopnicima
2. **Layer 3 preklopnik** : sve operacije preklapanja tj, ovdje govorimo o *routingu* (usmjeravanju) se odrađuju na osnovi IP adresa., sve se odrađuje brzinom hardvera , kao i svi mrežni protokoli koji su podržani od strane *ASICa*.
3. **Layer 4** : Dodaje se mogućnost rada na Transportnom sloju (TCP/UDP portovi) – možemo reći da je ovaj sloj *Application aware* – odnosno svjestan aplikacija.

Stvarna mjerenja

Zbog potrebe da se uvede red u stvarne pokazatelje odnosno stvarnu propusnost preklopnika, postoji nekoliko pokazatelja na koje treba obratiti pažnju.

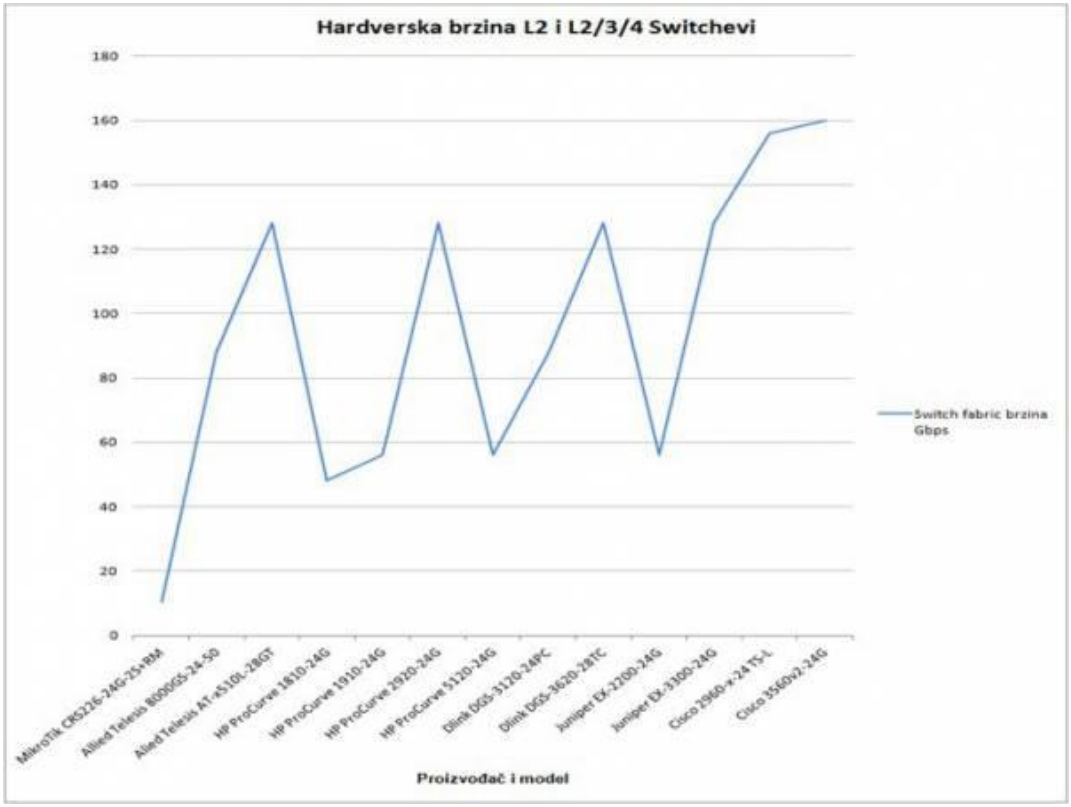
Što je što (Gbps i Mpps) - drugi dio

Oznaka Gbps (Gigabita u sekundi) označava ukupnu propusnost preklopnika koju dijele SVI portovi. Ovo je tkzv *Fabric* ili *Bus* brzina. Minimalna brzina koju uređaj mora imati je jednaka zbroju brzina svih *interface*a uređaja.

Dakle ako imamo preklopnik sa 24 x 1Gbps, pošto uređaj na gigabitnoj brzini mora moći raditi u **Full Duplex** načinu rada to znači da 24 x 1Gbps mora moći podnijeti 24 x 2Gbps = 48Gbps. To znači da uređaj mora moći obrađivati pakete propusnošću od 48Gbps.

Napomena : mnogi proizvođači samo pozbrajaju i pomnože sve navedeno bez stvarnih mjerenja – da bi rezultati bili bolji.

Pogledajmo usporednu tablicu nekoliko modela i proizvođača preklopnika, za 24 x 1Gbps preklopnike, kako za *Layer2* , tako i za *Layer2/3/4* preklopnike (donja granica bi morala biti 48Gbps):



Oznaka **Mpps** = (Milijuna paketa u sekundi) označava broj paketa koji se mogu obraditi u jednoj sekundi a ovise o veličini paketa. Većina ozbiljnih proizvođača navodi najmanje pakete (64 *byte*a), koji su i najzahtjevniji za obradu.

Oznaka **kpps** = (tisuća paketa u sekundi) je za red veličine manja jedinca od *Mpps*.

Dakle `100 kpps = 0.1 Mpps` odnosno `1000 kpps = 1 Mpps`

Kako smo već prije izračunali, za **1 Gbps** uz **64 *byte*ne** pakete znači kako je potrebna propusnost od **1.488 Mpps** (1.488 Milijuna paketa u sekundi) - za jedan port (mrežnu utičnicu).

Paket ili *Okvir* ?

Da budemo precizniji, pojam *Packet* ili paket, se odnosi na oblik podataka odnosno **PDU** (*protocol delivery unit*) u sloju 3 (*Layer 3*) (odnosno IP sloju *TCP/IP*a). Dok se pojam *Frame*, odnosno okvir, odnosi na oblik podataka odnosno **PDU** na OSI sloju 2 (*Layer 2*), dakle onome što će biti poslano na mrežu preko OSI sloja 1 (*Layer1*).

Dalje u tekstu ćemo, zbog jednostavnosti i dalje govoriti o mrežnim paketima.

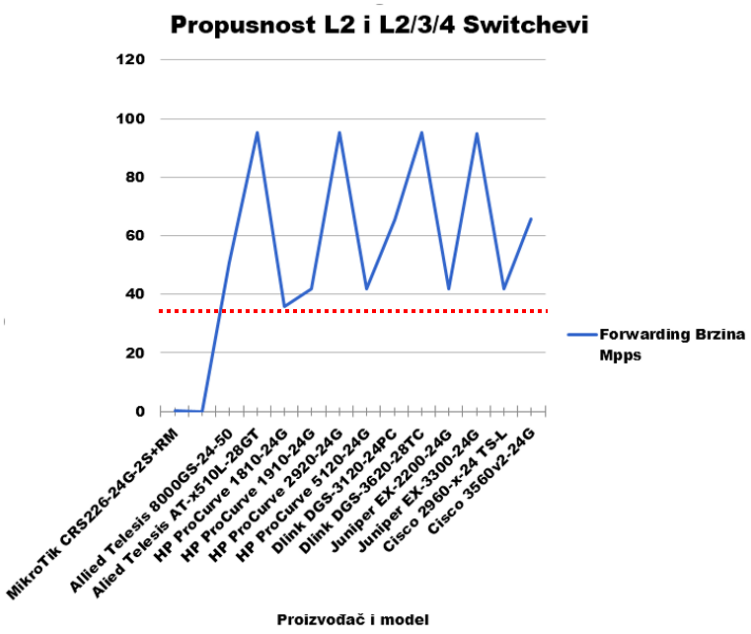
Što se događa u praksi

Događa se i to da proizvođač u raznim dokumentima za isti model preklopnika navodi totalno različite podatke, pa je najbolje provjeriti isti podatak u barem dva do tri službena dokumenta i/ili dodatno kontaktirati proizvođača (ovo nažalost nije rijetka pojava)
Oni manje "ozbiljni" navode puno veće pakete da bi im rezultati izgledali drastično bolji. Dakle mi ćemo dalje govoriti samo o **64** ***byte*** paketima i svim pokazateljima s tom veličinom paketa.
Kada gledamo rezultate mjerenja, trebamo tražiti broj **Mpps** za **64** *byte* pakete !.

Prema tome : 24 *portni* Gigabitni preklopnik mora imati minimalnu propusnost :
`24 x 1Gbps (1.488 Mpps) (duplex) = 36Mpps`

Pogledajmo sada karakteristike nekoliko modela preklopnika, raznih proizvođača, dostupnih kod nas , koje sam uzeo u razmatranje.
Napomene : ovdje je donja granica "ozbiljnosti" odnosno "upotrebe" koja se očekuje od 24 x 1Gbps preklopnika, minimalno 36Mpps .
Sve preko toga je nepotrebno osim:

- ako uređaj ima dodatnih SFP ili sličnih portova: svaki dodatni 1Gbps SFP dodaje potrebu za povećanjem propusnosti od 1.488 Mpps ili
- ako se koristi i kao platforma za snažnije modele preklopnika ,koji recimo ima dvostruko veći broj portova ili dodatne SFP ili SFP+ portove.



Što to znači za preklopnike iz ove kategorije koji nisu u stanju isporučiti minimalno 36Mpps ?.

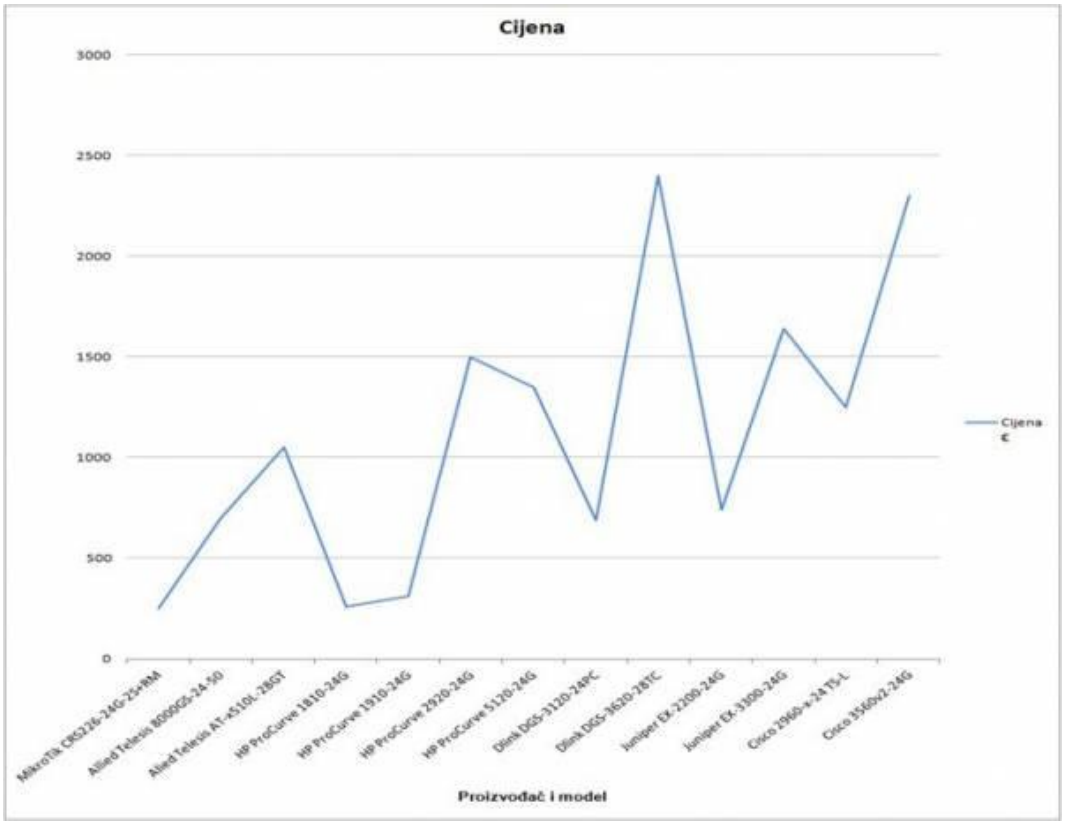
Preklopnik koji je primjerice u mogućnosti isporučiti samo do 10Mpps, punom brzinom može *opskrbiti* maksimalno do 7 x 1Gbps.

Dakle ako imamo do maksimalno sedam (7) mrežnih uređaja (računala/poslužitelja i sl.) spojenih na preklopnik, sve će raditi dovoljno brzo, ali već spajanjem osmog (8) uređaja dolazi do usporavanja.

Naime platili smo preklopnik sa 24 x 1Gbps a dobili smo sedam (7) *portni* preklopnik s dodatnih 17 portova.

Ovo nekada nije loše – ako je on cijenom vrlo prihvatljiv kao usmjerivač i preklopnik, na kojemu neće ukupno biti puno mrežnog prometa (u ovom slučaju ne više od ukupno 10Mpps) ali u većini primjena je to pristup/uređaj koji treba izbjegavati, što zbog buduće potrebe porasta brzine ili broja spojenih novih (dodatnih) uređaja.

I na kraju pogledajmo i cijene svih navedenih uređaja:



Svi odabrani uređaji su odabrani na osnovi dostupnosti (u našim trgovinama) i stanja koje sam često zatekao u upotrebi. Točan odabir proizvođača i modela ne želim komentirati u ovom članku.

Konačna preporuka i odabir bi bili ovisni o svim parametrima koje sam do sada naveo te o praktičnom iskustvu za točno određenog proizvođača, za točno određenu seriju i model (po nekada i za točno određenu verziju firmware-a).

Ovo možda zvuči smiješno ali doživio sam slučajeve u kojima određene novije verzije firmware-a ispravljaju stare greške i otvaraju nove koje nikako da se zatvore.

Bilo je i slučajeva gdje proizvođači navode listu grešaka za koje znaju ali ih godinama (ili nikada) ne isprave.

Do te je dokumentacije malo teže doći jer se baš ne reklamira. Bilo je i slučajeva u kojima ne radi određena kombinacija protokola ili postavki koja bi po svakoj logici morala raditi a i radi inače (ali baš na određenom modelu ne radi – što shvatite nakon pola dana surfanja i pretraživanja interneta).

Postojali su i slučajevi u kojima sam proizvođač nudi kao rješenje da se u slučaju upotrebe protokola A,B i C zajedno u određenoj konfiguraciji ne koriste određeni portovi na preklopniku…

Na granici 1Gbps i više (10/40/50/100Gbps)

Prednost ali i mana ASIC (Application-specific integrated circuits) je u njihovom dizajnu. Implementira se hardverska podrška (zapravo hardverska akceleracija) za svaki pojedini mrežni protokol ili funkcionalnost.
Ovo osnovno obilježje *ASIC*a daje mu iznimnu brzinu, koja je nedostižna klasičnim procesorima (CPU) ali je i mana jer dodavanje novog mrežnog protokola ili nove funkcionalnosti znači potrebu da se projektira i proizvede posve novi *ASIC chip*. Ovo u praksi znaci potrebu za kupnjom novog uređaja.
Nove generacije ASIC-a su programabilne verzije *ASIC* chipova koje rješavaju ovaj problem. Odnosno oni su neovisni o mrežnom protokolu ili funkcionalnosti ali zadržavaju ekstremne brzine obrade mrežnih paketa. Pojam **Programabilni ASIC** često možemo vidjeti pod nazivom **Software Defined Networking** (**SDN**) – ako govorimo o ASIC chipovima.

S obzirom na to da je važnost *ASIC*a jasna, mnogi veliki proizvođači poput tvrtki **Cisco Systems** i **Juniper Networks** ulagali su i ulažu u razvoj svojih ASIC rješenja. U toku 2009 godine tvrtka **Juniper** obavila je da će u periodu od tri godine uložiti nekoliko stotina milijuna dolara u razvoj ASIC chipova nove generacije a slično je i s tvrtkom **Cisco**.
Juniper je nakon toga 2013 godine izbacio na tržište prvi preklopnik s njihovim novim programabilnim ASIC-om. Radi se o modelu *EX9200* (nazivi su im *I-Chip* i *Trio chipset*). **Cisco** je početkom iste godine također krenuo s upotrebom svog ASIC riješenja pod nazivom **UADP Unified Access Data Plane** koji je prvo ugrađen u **Cisco Catalyst 3850** seriju 10Gbps preklopnika.
Cisco 3850 koristi *UADP ASIC* koji podržava do 24 x 10Gbps portova. Dakle modeli sa do 24 porta imaju jedan UADP ASIC dok modeli sa 48 portova imaju 2 UADP ASIC-a.
Kasnije je izašla nova serija 3650 (10/100/1000 Mbps) koja također koristi UADP ASIC.
Zbog vrlo skupog razvoja svojih rješenja i **Cisco** i **Juniper** u određenim ("slabijim") modelima uređaja koriste ASIC chipove specijaliziranih proizvođača *ASIC*a poput tvrtki **Broadcom**, **Marwell** i **Fulcrum microsystems** (2011 ih je kupio **Intel**) a koje nude svoja *ASIC* rješenja i drugim proizvođačima mrežne opreme (poput tvrtki: **Brocade**, **HP**, **DELL** i drugih).
Na zahtjevnijim platformama oba proizvođača (**Cisco** i **Juniper**) uz upotrebu univerzalnih *ASICa* poput gore navedenih, koriste i svoja rješenja ili ih kombiniraju sa svojima da bi dobili sve potrebne funkcionalnosti i brzinu u odnosu na konačnu cijenu proizvoda. Proizvođači poput tvrtki **Cisco** i **Juniper** na svojim najzahtjevnijim platformama najčešće koriste svoja rješenja koja prema njima nude najveće performanse (u ovoj kategoriji se baš i ne pita previše za cijenu).
S obzirom na činjenicu da tvrtka **Broadcom** drži 65% tržišta *ASIC*a, pogledajmo i što oni nude.
Pogledajmo proizvode koji pokrivaju gornji segment tržišta (10Gbps i 40Gbps).
Trident [BCM56840] (pojavio se u 2010.g.)

- Podržava do 48 x 10Gbps na jedom ASIC chipu
  - Koriste ga :
    - Juniper (QFX3500)
    - Cisco (Nexus 3064)
    - Dell Networks (bivši Force 10) – S4810
    - HP (5900 AF 48XG)
    - IBM (BNT RackSwitch G8264)
    - … i još nekoliko manjih proizvođača

Trident+ [BCM56840 (+)]

- Podržava do 64 x 10Gbps na jednom ASIC chipu,
  - Koriste ga:
    - Cisco (Nexus serija preklopnika)
    - Juniper (QFX3500 serija preklopnika)
    - Dell (kupnjom tvrtke "Force 10" u toku 2011. godine)

Trident II XGS [BCM56850] (2012.g.) (za 10Gbps i 40 Gbps)

- Podržava 32 x 40G porta ili 104 x 10G sve na jednom ASIC chipu.
- Switching propusnost ovog ASIC-a je 1280 Gbps.
  - Koriste ga :
    - Cisco (Nexus 9000)
    - Dell Networking S6000
    - i nekoliko manjih proizvođaća

Tomahawk (2014.g)

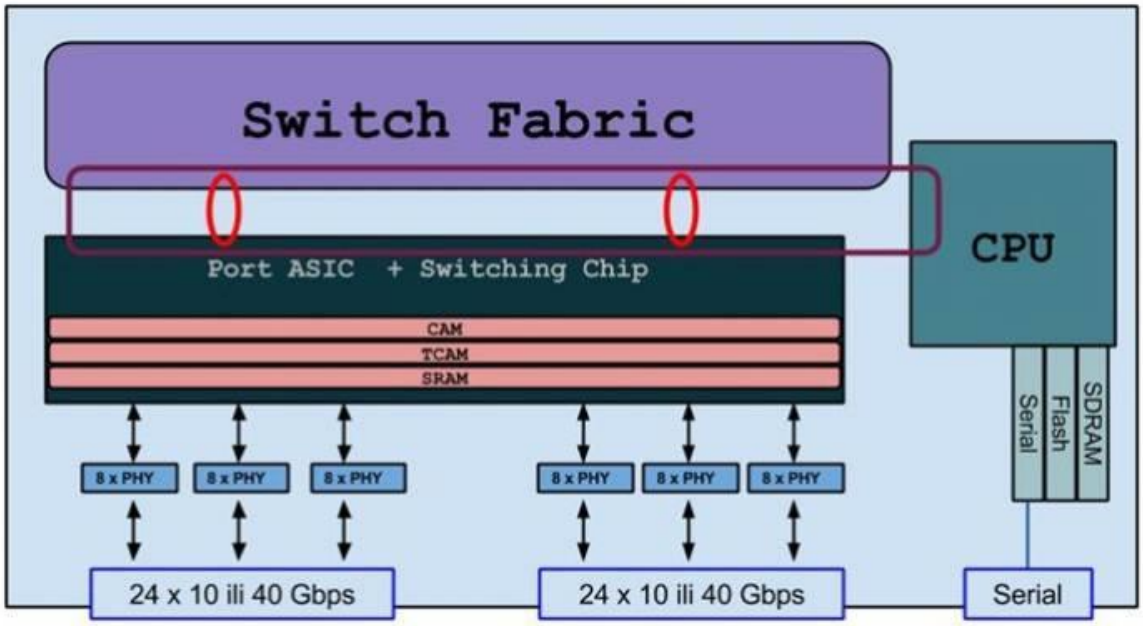
- Ima switching propusnost od 3200 Gbps (3.2Tbps)
- Jedan chip podržava do 128 x 25Gbps portova
- Ovo je prvi programabilni ASIC tvrtke Broadcom (ovu tehnologiju **Broadcom** zove **BroadView**).

Tomahawk II (2016.g.)

- Ima propusnost od 6400 Gbps (6.4 Tbps) i to Multilayer L2/L3/MPLS
- Jedan chip podržava 64 x 100Gbps portova ili 128 x 40Gbps/50Gbps
- Hardverski podržava mnoge protokole za tuneliranje
- Podržava OpenFlow
- Podržava do 256 x 25Gbps SERDES (Serijalizaciju/Deserijalizaciju podataka)
- …

Definitivno možemo reći da se sve više ide prema modelu u kojemu jedan *ASIC* i *Switching chip* podržavaju dovoljan broj *portova* za cijeli preklopnik. Ovim dizajnom se rješava problem sporosti sabirnice između više *ASIC chipova*, koja postaje sve veći problem na sve većim brzinama (10G/40G/100G).

Prema ovom modelu referentni dizajn bi izgledao ovako (slika).



Što je tu Open Source ?

Kao što smo spomenuli i svaki usmjerivač i preklopnik se sastoje od gotovo istih komponenti kao i računala.
Najveća razlika je u *Switching chipu* i *ASIC*u. Nakon što je **Broadcom** ponudio *ASIC* (+ Switching chip) generacije **Trident II**, stvari su se drastično promijenile. Upravljački programi za ovaj *ASIC* – i za Linux postali su dostupni. Bilo je samo pitanje vremena kada će se netko sjetiti napraviti svoj preklopnik (praktično računalo) sa spojenim ovim *ASIC*om i dobili ste nevjerojatno snažnu platformu. Ostaje napraviti svoju distribuciju linuxa i optimizirati postojeće mrežne servise da znaju iskoristiti snagu ovog novog hardvera.
Došli smo do *OpenSource* preklopnika koji se, barem u teoriji, može natjecati sa proizvođačima poput **Cisco** i **Juniper** (i ostalih) – zanimljivo.
Sredinom 2014 se pojavila tvrtka **Pica8** koja je napravila upravo to – uzeli su *bootloader* od *Open Network Install Environment*a (koji razvija tvrtka **Cumulus Networks**) a koji je dio *Open Compute Project*a te dodali i druge komponente Linuxa, kao i potrebne mrežne servise i nastala je njihova distribucija Linuxa koja se zove *PicOS.*

Vrlo brzo nakon toga uslijedila je dobra podrška i za **Trident**, **Trident+** i za najnovije **Tomahawk** i **Tomahawk II**.
⇒ Ali na koji hardver (ipak neće svatko kod kuće sastavljati svoj 10Gbps preklopnik).
Bez brige oni nude i upravo ovakve "prazne" preklopnike :
http://www.pica8.com/products/pre-loaded-switches
Ubrzo su se pojavili i drugi – čije distribucije Linuxa se isto mogu instalirati na ove preklopnike :

- Cumulus Networks (Open Network Install Environment),
- Vyatta (kupila ih je tvrtka "Brocade"),
- On.Lab ONOS.

Izvori informacija

(**1**) https://www.opensource-osijek.org/wordpress/2016/02/16/switching-i-routing-jucer-danas-sutra/ : Izvorni članak autora
(**2**) https://en.wikipedia.org/wiki/Twisted_pair : TP kabeli
(**3**) https://en.wikipedia.org/wiki/Category_5_cable : Cat 5 kabeli
(**4**) https://en.wikipedia.org/wiki/Category_6_cable : Cat 6 kabeli
(**5**) https://en.wikipedia.org/wiki/Medium-dependent_interface : MDI, MDI-X i Auto MDI-X
(**6**) http://netoptimizer.blogspot.hr/2014/05/the-calculations-10gbits-wirespeed.html : Kernel network parameters
(**7**) http://networkengineering.stackexchange.com/questions/5057/what-is-the-actual-size-of-an-ethernet-mtu : MTU
(**8**) http://www.cisco.com/c/dam/global/da_dk/assets/docs/presentations/CVU-juni_X_S-series.pdf : Cisco 3750-X design
(**9**) https://ciscointerworking.wordpress.com/2015/06/14/asic-redundancy/ : Cisco ASIC and Design
(**10**) http://www.cisco.com/networkers/nw03/presos/docs/RST-2011.pdf : Cisco Design
(**11**) http://rabdoul.free.fr/PUBLIC/Cisco%20slides/QOS%202.3/BRKRST-3437-Marc%20Alonso.pdf : Cisco Arhitektura switcheva
(**12**) http://networking.ventrefamily.com/2010/08/asic-to-port-mappings.html : ASCI to Port - Cisco
(**13**) https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000599-en.pdf : Dizajn Juniper QFX 10000 *switcheva*
(**14**) http://www.cisco.com/assets/global/DK/seminarer/pdfs/Catalyst_Switching_Deep_Dive_Feb2016.pdf : Cisco Dizajn "snažnijih" kategorija *switcheva*
(**15**) https://calomel.org/freebsd_network_tuning.html : FreeBSD Network Tuning
(**16**) http://kb.pert.geant.net/PERTKB/MultiThreadDemux : Multi Thread Net
(**17**) https://github.com/gokzy/freebsd-rps/wiki/Receive-Packet-Steering-on-FreeBSD : RPS Tunning FreeBSD
(**18**) https://wiki.freebsd.org/201305DevSummit/NetworkReceivePerformance/ComparingMutiqueueSupportLinuxvsFreeBSD : RFS FreeBSD
(**19**) http://www.chelsio.com/wp-content/uploads/resources/T5-40Gb-FreeBSD-Netmap.pdf : Chelsio T5 + FreeBSD
(**20**) https://www.mjmwired.net/kernel/Documentation/networking/scaling.txt#308 : Scaling in the Linux Networking Stack
(**21**) https://conferences.oreilly.com/oscon/oscon-tx/public/schedule/detail/56727 : pfSense najave - Network optimizacija
(**22**) https://lwn.net/Articles/382428/ : Članak o RFS tehnologiji
(**23**) http://www.dell.com/downloads/global/power/1q04-her.pdf : Članak o **TOE**
(**24**) http://www.chelsio.com/wp-content/uploads/resources/T6-100G-DDP-FreeBSD.pdf : Chelsio i FreeBSD: 100Gbps Net i TOE (TCP Offload Engine)
(**25**) http://www.chelsio.com/wp-content/uploads/resources/T5-40Gb-FreeBSD-TOE-DDP.pdf : Chelsio 40Gbps, FreeBSD i TOE
(**26**) https://www.opensource-osijek.org/dokuwiki/wiki:knjige:uvod_u_linux#numa : Uvod u Linux - NUMA arhitektura
(**27**) https://calomel.org/network_performance.html : FreeBSD Network Performance Tuning
(**28**) https://en.wikipedia.org/wiki/PCI_Express : PCI Express sabirnica
(**29**) http://luca.ntop.org/10g.pdf : 10 Gbit/s Line Rate Packet Processing Using Commodity Hardware
(**30**) https://pdfs.semanticscholar.org/cf0c/c4bdabc3e5f04221ee08ae31bc8714f2367d.pdf : netmap: Memory Mapped Access To Network Devices
(**31**) https://www.netgate.com/blog/building-a-behemoth-router.html : FreeBSD/pfSense 10Gbps Ethernet overview
(**32**) http://dpdk.org/ : DPDK biblioteke
(**33**) http://info.iet.unipi.it/~luigi/netmap/ : **netmap** Framework
(**34**) https://www.linux-kvm.org/images/c/c5/Kvm-forum-2013-High-Performance-IO-for-VMs.pdf : netmap + Linux QEMU/KVM
(**35**) https://pdos.csail.mit.edu/papers/click:tocs00/paper.pdf : Click Router - opis
(**36**) http://openvswitch.org/support/ovscon2014/18/1630-ovs-rizzo-talk.pdf : SW Data plane performance - QEMU, DPDK, OVS, netmap and VALE
(**37**) http://www.ntop.org/products/packet-capture/pf_ring/ : PF RING zadužen za brzo dohvaćanje, analizu i obradu mrežnih paketa
(**38**) https://en.wikipedia.org/wiki/TCP_offload_engine#Freed-up_CPU_cycles : Odnos: CPU ciklusi (Hz)/bitovi podataka na mreži (TCP/IP)
(**39**) http://www.nanogrids.org/jaidev/papers/ispass03.pdf : TCP/IP performanse
(**40**) https://lwn.net/Articles/551284/ : Linux Low-latency Ethernet device polling
(**41**) https://www.freebsd.org/cgi/man.cgi?query=polling&apropos=0&sektion=0&manpath=FreeBSD+11.0-stable&arch=default&format=html :FreeBSD Pooling
(**42**) http://www.science.unitn.it/~fiorella/guidelinux/tlk/node86.html : Pooling & Interrupts (IRQ)
(**43**) https://en.wikipedia.org/wiki/Interrupt_coalescing :Interrupt moderation
(**44**) https://en.wikipedia.org/wiki/Ethernet Ethernet Wiki
(**45**) https://en.wikipedia.org/wiki/Cut-through_switching Cut-through metoda
(**46**) https://en.wikipedia.org/wiki/Store_and_forward Store and forward metoda
(**47**) https://medium.com/speedtest-by-ookla/engineer-maximizes-internet-speed-story-c3ec0e86f37a
(**48**) https://www.pfsense.org/hardware/ : *hardware sizing*
(**49**) https://docs.opnsense.org/manual/hardware.html *hardware sizing*
(**50**) https://www.sophos.com/en-us/products/unified-threat-management/tech-specs.aspx#software : *Sophos UTM hardware*
(**51**) http://www.brocade.com/en/products-services/software-networking/network-functions-virtualization/vrouter.html *Brocade* **Vyatta/vRouter**
(**52**) https://www.netgate.com//products/sg-1000.html : *pfSense Router* na *ARM* arhitekturi quick
(**53**) http://www.androidauthority.com/arm-vs-x86-key-differences-explained-568718/
(**54**) https://en.wikipedia.org/wiki/New_API: Linux NAPI (Network New API)
(**55**) https://software.intel.com/en-us/articles/how-intel-quickassist-technology-accelerates-nfv-use-cases Intel Quick Assist tehnologija
(**55.1**)http://ark.intel.com/products/77988/Intel-Atom-Processor-C2758-4M-Cache-2_40-GHz Intel Atom C 2758
(**56**) https://fd.io/news/announcement/2017/06/fast-data-fdio-project-issues-fourth-release-furthers-position-universal Fast Data IO Framework
(**57**) https://wondernetwork.com/pings/ World ping latencije
(**58**) http://www.dell.com/en-us/work/shop/productdetails/force10-s2410 Dell Force 10 S2410 preklopnik - specifikacije
(**59**) http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5020-switch/white_paper_c11-465436.html Cisco Nexus 5020 preklopnik - specifikacije
(**60**) https://en.wikipedia.org/wiki/IEEE_802.1Q 802.1Q - VLANovi
(**61**) https://wiki.wireshark.org/MTU MTU - najčešće veličine
(**62**) https://en.wikipedia.org/wiki/Maximum_transmission_unit MTU - definicija
(**63**) http://bradhedlund.com/2008/12/19/how-to-calculate-tcp-throughput-for-long-distance-links/ Latencija, propusnost i veličina TCP prozora
(**64**) https://www.opensource-osijek.org/dokuwiki/wiki:knjige:uvod_u_linux#ip_adrese Knjiga: Uvod u Linux i Linux napredno - IP protokol

Više informacija o knjizi

Više informacija o knjizi možete dobiti na stranici :

https://www.opensource-osijek.org/wordpress/kratka-prica-o-mrezama-preklopnici-i-usmjerivaci/

QR Code - poveznica na stranicu: