KOMUTACIONI SISTEMI
– Poglavlje 9 –

9 Paketski komutatori

Tehnika komutacije kola se zasniva na principu zauzimanja resursa u mreži za vezu između korisnika koji žele međusobno komunicirati. Zauzete resurse potom koristi samo dotična veza tj. samo podaci koji se razmenjuju u dotičnoj vezi se prenose tim zauzetim resursima. Ovakav princip je bio adekvatan u početku razvoja telekomunikacionih mreža, jer je najkomplikovaniji posao bio upravo dotično zauzimanje resursa, ali s obzirom da su veze trajale dovoljno dugo, frekvencija zauzimanja resursa nije bila prevelika pa je bilo tehnološki moguće realizovati komutaciju kola u početnim fazama razvoja telekomunikacionih mreža. Prednost komutacije kola je pre svega mogućnost garancije kvaliteta servisa korisniku. Naime, onoga momenta kada su resursi za vezu zauzeti, mogu se veoma precizno odrediti svi parametri dotične veze poput protoka, kašnjenja, varijacije kašnjenja (koja je veoma mala u komutaciji kola), itd. Samim tim, lako je utvrditi kako će veza biti opslužena, tj. kakav kvalitet usluge će korisnik dobiti. Ali, postoji i velika mana tehnike bazirane na komutaciji kola, a to je slaba iskorišćenost zauzetih resursa. Naime, razmotrimo kao primer prenos govora. U slučaju digitalnog prenosa govora preko klasične telefonske mreže (fiksna telefonija) koristi se G.711 kodirani govorni signal (po A zakonu kompresije u Evropi) i tada naizgled nema neiskorišćenosti resursa jer se neprestano prenose osmobitni govorni odmerci preko zauzetih resursa za dotičnu govornu vezu. Međutim, ako se podsetimo rada drugih kodera govornog signala opisanih u prethodnom poglavlju, jedna od osobina govora je da postoje česte pauze u govoru i tada nema korisnog signala. Neki koderi su tu činjenicu koristili da bi na početku perioda pauze (tišine) preneli parametre šuma okoline i potom samo povremeno slali ove parametre tokom perioda pauze (ako se ti parametri ne menjaju drastično, u suprotnom se na drastičnu promenu nove vrednosti parametara odmah šalju). To znači da bi tokom perioda tišine mogao da se smanji protok govornog signala, čime zauzeti resursi ne bi bili dovoljno iskorišćeni jer su oni zauzeti po parametru vršnog protoka (najveća vrednost zahtevanog protoka u toku komunikacije). Pošto druge veze ne mogu da koriste te resurse oni ostaju neiskorišćeni što je loše za telekomunikacionog operatera jer mu se tako prihodi neće povećati koliko su mogli. Ova situacija je znatno izraženija u slučaju računarske komunikacije koja je po svojoj prirodi sporadična (*bursty*). U slučaju računarske komunikacije podaci se tipično šalju u naletima tzv. burstovima, koje potom slede relativno duži periodi pauze. U pauzi između burstova se ne bi ništa prenosilo jer u komutaciji kola druge veze nemaju pravo da koriste tuđe resurse. Ovaj problem je uočen razvojem računara i potrebe za njihovim povezivanjem, pa je otuda teorija paketske komutacije nastala veoma rano (šezdesetih godina prošlog veka). Naravno, sa stanovišta korisnika je možda bolja tehnika komutacije kola jer korisnik zna šta dobija za svoj novac, ali ipak je potrebno sagledati situaciju sa više strana. Sa manjom iskorišćenosti resursa, potrebna su znatno veća ulaganja operatera u proširenje svoje infrastrukture da bi se pokrio još veći broj korisnika, pa bi tako dodavanje novih korisnika bilo otežano i usporeno, a najverovatnije bi i razvoj novih servisa bio usporeniji, naročito ako bi oni zauzimali veće resurse i dodatno doprinosili slaboj iskorišćenosti resursa.

Tehnika komutacije paketa se zasniva na činjenici da se u komunikaciji između korisnika razmenjuju paketi, tj. da su podaci koje korisnici razmenjuju smešteni u pakete. Svaki paket se prenosi kroz mrežu prolazeći kroz mrežne čvorove. Po ulasku u mrežni čvor, određuje se na koji izlaz mrežnog čvora treba poslati dotični paket. Paket zatim čeka na svoj redosled za slanje zajedno sa paketima drugih veza koji se šalju preko istog izlaza mrežnog čvora. Očigledno,

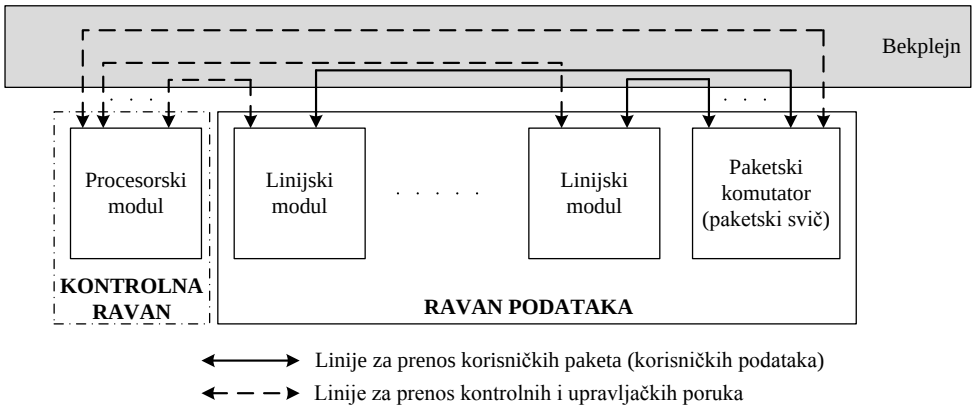
preko izlaza, tj. linka, koji je povezan na dotični izlaz, se uvek šalju paketi ako ih ima za slanje, tako da pauza u jednoj vezi ne utiče na iskorišćenost linka koji će tada da prenosi pakete drugih veza koje su takođe u toku. Komutacija paketa postiže visoku iskorišćenost resursa u mreži, ali sada nedostaje garancija kvaliteta servisa, tj. ne može se znati kakvu uslugu će dobiti korisnik. Na primer, usled komunikacije prevelikog broja korisnika može se generisati prevelik broj paketa koji prolaze kroz iste delove mreže pa samim tim može doći do preopterećenja pojedinih delova mreže, a samim tim i do potencijalnih gubitaka paketa u mreži. Očigledno, parametri poput kašnjenja, varijacije kašnjenja, ostvarenog protoka i dr. u velikoj meri zavise od trenutne situacije u mreži, odnosno trenutnih komunikacija u toku. Otuda se u mrežama uvode brojne tehnike koje pokušavaju da reše navedene probleme i da garantuju određeni kvalitet servisa korisniku (poput u prethodnim poglavljima pomenutih IntServ i DiffServ tehnika). Takođe, komutacija kola podrazumeva zauzimanje mrežnih resursa, pa je stoga neophodan proces uspostave veze u okviru koje se zauzimaju resursi, odnosno proces raskidanja veze u okviru kojeg se oslobađaju prethodno zauzeti mrežni resursi. U slučaju komutacije paketa proces uspostave veze nije neophodan, ali može da se izvrši. Na primer, u slučaju IP protokola nema uspostave veze, već se IP paketi direktno šalju, a u slučaju ATM mreža se vrši uspostava virtuelnog kola.

Veoma je bitno naglasiti da je na penetraciju paketske tehnologije na telekomunikaciono tržište dominantno uticao tehnološki razvoj. Naime, u slučaju paketske komutacije, mrežni čvor obrađuje svaki paket ponaosob, što znači da se obrađuje i svaki paket iste veze ponaosob. Na primer, u mrežnom čvoru je za svaki paket bitno odrediti na koji izlaz mora izaći i to se radi za svaki paket. Pored ovoga rade se i druge obrade paketa, poput provere ispravnosti paketa na osnovu polja za proveru i dr. Pošto se procesiranje radi na nivou paketa, što je protok paketa brži, potrebna je veća moć procesiranja. Ako se ima u vidu da se danas u mrežama standardno instaliraju linkovi čije su brzine jednake desetinama Gb/s, sa tendencijom da se u bliskoj budućnosti instaliraju i terabitski linkovi, jasno je da moć procesiranja mora biti ekstremno visoka. Na primer, najkraći eternet okvir (64B) na 40Gb/s linku traje svega 12.8ns, i ako se pretpostavi najgori slučaj da pristižu samo najkraći eternet okviri, tada mrežni čvor na svom ulazu mora da obrađuje te okvire za svega 12.8ns inače bi došlo do nagomilavanja okvira koji čekaju na obradu. S druge strane, kod komutacije kola se resursi zauzimaju na početku veze. Ako uzmemo da se veze generišu znatno sporije od generisanja paketa (što je i logično jedna veza generiše više paketa pa je očigledno generisanje paketa intezivnije od generisanja veza), jasno je da ovaj problem nije postojao u slučaju komutacije kola, pa je i to razlog zašto je prva tehnika koja se koristila u mrežama bila komutacija kola. Tek sa razvojem integrisanih čipova je bilo moguće efikasno procesirati pakete i omogućiti razvoj i penetraciju paketske komutacije do današnjeg nivoa koji omogućava rad mrežnih čvorova i sa linkovima čiji je protok i do 100Gb/s, pa i većim u bliskoj budućnosti.

Mreže bazirane na komutaciji paketa se tipično nazivaju paketskim mrežama. Mrežni čvorovi moraju obavljati veliki skup funkcija da bi uspešno izvršavali obradu korisničkih paketa i njihovo prosleđivanje do odgovarajućih krajnjih korisnika. Stoga se u mrežnim čvorovima paketskih mreža generalno razlikuju dve ravni funkcija - kontrolna ravan i ravan podataka. Ova podela je načinjena na osnovu poslova koje mrežni čvor mora da izvrši. S jedne strane je potrebno da se obavi usmeravanje korisničkih paketa kao osnovna funkcija mrežnog čvora. S druge strane je neophodan velik skup kontrolnih protokola neophodnih za ispravno i kvalitetno obavljanje osnovne funkcije mrežnog čvora. Kontrolni protokoli podrazumevaju razmenu kontrolnih paketa između mrežnih čvorova i čiji je intezitet razmene (protok) znatno manji od

inteziteta saobraćaja korisničkih paketa između mrežnih čvorova. Istovremeno, rezultat kontrolnih protokola se često aplicira na sve portove mrežnog čvora, pri čemu port predstavlja ulaz/izlaz mrežnog čvora. Ulazni port označava ulaz mrežnog čvora, a izlazni port označava izlaz mrežnog čvora. Na portove mrežnog čvora se priključuju linkovi za povezivanje mrežnog čvora sa drugim mrežnim čvorovima ili korisnicima. Pošto su linkovi tipično bidirekcioni, onda su i portovi tipično bidirekcioni tj. i ulazni i izlazni istovremeno. Pošto postoji značajna razlika u protoku kontrolnih informacija (paketa) od korisničkih informacija (paketa), u mrežnom čvoru se razdvajaju ravan podataka koja obuhvata obradu korisničkih paketa, i kontrolna ravan koja podrazumeva obradu kontrolnih paketa. Zbog svojih različitih priroda, ravan podataka se tipično implementira u hardveru radi ostvarivanja što bržeg procesiranja korisničkih paketa, a kontrolna ravan u softveru radi ostvarivanja što veće fleksibilnosti. Naime, protokoli kontrolne ravni se izvršavaju na procesoru opšte namene čime kontrolna ravan postaje laka za održavanje i unapređivanje, a funkcije kontrolne ravni koje mrežni čvor treba da podrži se lako dodaju ili modifikuju (lakše je modifikovati softver nego hardver na gotovom uređaju). Tipični primeri funkcija ravni podataka su ispitivanje ispravnosti pristiglog paketa, komutacija paketa sa ulaznih na odgovarajuće izlazne portove, mehanizmi odbacivanja paketa u slučaju zagušenja, baferisanje paketa, lukap funkcija kojom se određuje na koji izlazni port treba proslediti pristigli paket i dr. Tipični primeri funkcija kontrolne ravni su protokoli rutiranja i ažuriranje tabele usmeravanja u ravni podataka (lukap funkcija pretražuje tabelu usmeravanja), udaljeni pristup administratora, komandni interfejs, protokoli za rezervaciju resursa, protokoli za nadgledanje mreže, autentifikacija i dr.

Pojedini autori i proizvođači mrežne opreme dodatno razlikuju kontrolnu ravan i ravan upravljanja. One se izvršavaju na istom mestu u mrežnom čvoru (tipično softverski na procesoru opšte namene), tako da sa stanovišta fizičke pozicije ove dve ravni u mrežnom čvoru nema nikakve razlike, i jedina razlika je u skupu funkcija. Kontrolna ravan se odnosi na mehanizme i funkcije kontrole rada ravni podataka poput protokola rutiranja, mehanizama kvaliteta servisa, rezervacije resursa, multikast podrške i dr. Ravan upravljanja se odnosi na funkcije upravljanja mrežnim čvorom poput konfigurisanja rada mrežnog čvora, nadgledanja rada mrežnog čvora, alarmiranja u slučaju neispravnosti, omogućavanja udaljenog pristupa mrežnom čvoru i dr.



Slika 9.1. Principska arhitektura mrežnog čvora paketske mreže

Na slici 9.1 je prikazana principska arhitektura jednog mrežnog čvora paketske mreže. Mrežni čvor tipično sadrži sledeće delove:

- Linijski moduli

- Procesorski modul
- Paketski komutator (paketski svič)
- Bekplejn

Linijski moduli i paketski komutator pripadaju ravni podataka, dok procesorski modul pripada kontrolnoj ravni. Linijski moduli sadrže ulazne/izlazne portove i izvršavaju obradu paketa. Linijskih modula može biti više jer tipično jedan linijski modul podržava manji broj portova, pa je potreban veći broj linijskih modula da bi se podržao željeni broj portova (pri tome maksimalan kapacitet mrežnog čvora ograničava maksimalan broj portova, a time i linijskih modula). Paketski komutator je ekvivalent komutacionog polja telefonske centrale i izvršava komutaciju paketa sa ulaznih portova na odgovarajuće izlazne portove. Tipično se u praksi paketski komutator naziva terminom paketski svič. Korisnički paketi pristižu na ulazne portove linijskih modula i vrši se obrada paketa na ulazu (ulaznom portu) koja tipično podrazumeva proveru ispravnosti paketa i lukap funkciju da bi se utvrdilo na koji izlaz treba da se prosledi paket. Zatim, paketski komutator prosleđuje paket na odgovarajući izlazni port, gde se vrši eventualna obrada na izlazu (tipično ova obrada ima za cilj postizanje fer servisa opsluživanja tokova, kvalitet servisa i sl.) i potom prosleđivanje paketa preko izlaznog linka dalje u mrežu ka sledećem mrežnom čvoru ili odredišnom korisniku. U slučaju pristizanja kontrolnih paketa, oni se sa ulaznog porta prosleđuju procesorskom modulu. Isto tako, kontrolni paketi koje generiše procesorski modul se prosleđuju od procesorskog modula na odgovarajuće izlazne portove.

Procesorski modul sadrži procesor na kome se instalira operativni sistem mrežnog čvora u okviru koga se izvršavaju funkcije kontrolne ravni. Procesorski modul je ekvivalent upravljačkog bloka iz telefonske centrale. Uobičajeno, mrežni čvorovi sadrže samo jedan procesorski modul, ali ih može biti i više ukoliko kapacitet mrežnog čvora to zahteva. Preko kontrolnih linija procesorski modul upravlja radom (preciznije konfiguriše rad) linijskih modula i paketskog komutatora. Svi moduli se hardverski prave u vidu kartica koje se povezuju na bekplejn.

Bekplejn ili bekpanel predstavlja štampanu ploču koja definiše pozicije kartica (gde se utiskuju linijski moduli, a gde procesorski modul i paketski komutator) i koji vrši međusobno povezivanje modula. Na ovaj način je omogućen modularan i skalabilan dizajn mrežnih čvorova, čime je olakšano da operateri koji instaliraju mrežne čvorove mogu lako da ih prilagode svojim potrebama, a takođe mogu veoma lako da ih nadograđuju i unapređuju. Naravno, u slučaju mrežnih čvorova malih kapaciteta tipično su navedeni moduli integrisani na jednu zajedničku štampanu ploču ili manji broj štampanih ploča povezanih bekplejnom. Napomenimo, da u mrežnom čvoru uvek postoji i blok za napajanje sa istom ulogom kao i kod telefonske centrale (obezbeđivanje snage za rad svih delova mrežnog čvora).

U okviru ovoga poglavlja pažnja je posvećena paketskim komutatorima. Većina karakteristika i aspekata bitnih za komutatore na bazi komutacije kola su bitni i za paketske komutatore (sa eventualnim sitnijim razlikama), ali postoje i dodatne karakteristike koje su važne samo za paketske komutatore. Neke od karakteristika bitnih za paketske komutatore su:

- Podrška multikast saobraćaja
- Blokada
- Skalabilnost

- Ubrzanje komutatora
- Propusnost komutatora
- Cena
- Kompleksnost
- Tip paketa koji se komutira (paketi promenljive ili fiksne dužine)
- Baferi

Što se tiče osobina skalabilnosti, cene i kompleksnosti one su sličnog tumačenja kao i kod komutatora na bazi komutacije kola. Skalabilnost određuje koliko je lako proširivati kapacitet komutatora (što je lakše proširivati kapacitet komutatora to je bolji (skalabilniji) komutator), cena predstavlja ekonomičnost izrade komutatora (što manje elemenata (i što jednostavnijih elemenata) se upotrebi u izradi komutatora, komutator će biti ekonomičniji), a kompleksnost definiše kako kompleksnost implementacije komutatora (koja je u svezi sa cenom komutatora), tako i složenost upravljanja komutatorom.

Paket koji se prosleđuje kroz paketski komutator može biti unikast (jedan na jedan) ili multikast. Multikast paketi predstavljaju dodatno opterećenje za mrežne čvorove. Neki paketski komutatori mogu lako da naprave replike paketa i tako proslede simultano paket na više izlaza, ali i pored toga, veoma je složeno upravljati konfiguracijom paketskog komutatora (čak i onih koji lako fizički prosleđuju multikast pakete) u slučaju prisustva multikast saobraćaja. Naime, multikast paket može u opštem slučaju da se prosledi na proizvoljan broj izlaza, pa kada se tome doda i podatak da istovremeno može da postoji velik broj multikast tokova, postaje veoma složeno konfigurisati paketski komutator tako da ne dođe do sudara prilikom komutacije paketa na izlaznom portu tj. na izlazu paketskog komutatora, i da se istovremeno ostvari što veći protok paketa kroz komutator. Otuda se neretko koristi tehnika da se na ulaznom portu naprave kopije paketa za svaki od izlaznih portova na koje multikast paket treba proslediti i da se potom tako kreirani paketi tretiraju kao unikast paketi. Jasno je da se na ovaj način protok paketa značajno uvećava jer sada dobijamo situaciju kao da je odjednom došlo više paketa na istom ulaznom portu, pa samim tim ovakvo rešenje nije efikasno iako je jednostavno za realizaciju. Postoji velik broj predloženih rešenja u naučnim radovima, ali multikast podrška u mrežnim čvorovima paketskih mreža i dalje predstavlja velik izazov i otvorenu problematiku. U slučaju komutacije kola ovaj problem je bio manje izražen jer se prilikom uspostave veze vršilo zauzimanje resursa, ali potom kada su oni zauzeti više nije bilo problema u samom procesu komutacije (u komutaciji kola je problem bio izražen jedino ako sam komutator nije imao podršku za ostvarivanje multikast veza). Napomenimo da generalno ne postoji multipoint veza u komutatorima paketa što je logično jer paket pristiže na jedan ulazni port, ali može se reći da svojevrsna multipoint veza sa stanovišta čitavog mrežnog čvora može da postoji, poput miksera u audio konferenciji gde bi se miksovali paketi sa više ulaza (od više učesnika konferencijske veze) koji bi se potom slali na više izlaza ka ostalim učesnicima konferencije. Međutim, ovde se vrši prosleđivanje paketa sa ulaznih portova na isto mesto u mrežnom čvoru (na primer, dodatni modul koji vrši funkciju miksovanja) gde se vrši miksovanje pa tek potom se sa tog mesta vrši multikast paketa ka odgovarajućim izlaznim portovima tako da se ne može reći da je u pitanju multipoint veza kroz paketski komutator.

Mrežni čvorovi u paketskim mrežama su potpuno dostupni tako da se osobina dostupnosti uglavnom ni ne spominje. Sa stanovišta blokade razlikujemo neblokirajuće i

blokirajuće paketske komutatore sa istom definicijom kao kod komutacije kola. Neblokirajući komutator je onaj kod kojeg se može uspostaviti veza između proizvoljnog slobodnog ulaza i proizvoljnog slobodnog izlaza bez obzira na postojeće uspostavljene konekcije, u suprotnom komutator je blokirajući. Naravno, efekat blokade nije identičan kao kod komutacije kola. Kod komutacije kola veza se uspostavi i traje dok traje i komunikacija tj. dok se komunikacija između korisnika ne raskine. Ako se usled blokade ne može izvršiti komutacija, veza će biti odbijena (neće se uspostaviti). Kod komutacije paketa se konfiguracija komutatora vrši na nivou paketa, pa na primer pojam uslovne blokade nije toliko bitan jer se konfiguracija komutatora menja od paketa do paketa i nema potrebe prerasporediti postojeće veze jer se sledeća konfiguracija opet radi od nule pa postojeće veze praktično ni ne postoje. Isto tako, ako se prilikom jedne konfiguracije blokira komutacija za neki paket, u sledećoj konfiguraciji će se opet pokušati tj. paket neće biti odbačen.

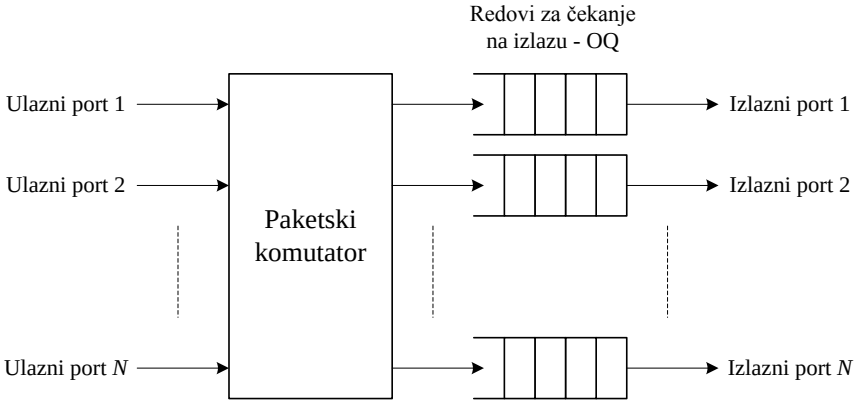
Efekti blokade se mogu umanjiti upotrebom principa ubrzanja (*speedup*) u paketskim komutatorima. Ubrzanje *s* paketskog komutatora označava da paketski komutator radi na *s* puta većoj brzini od ulaznih/izlaznih linkova čime je moguće ostvariti veći broj konfiguracija paketskog komutatora u jedinici vremena, a samim tim i proslediti veći broj paketa sa ulaza na izlaze u jedinici vremena.

Komutacija paketa može da se odnosi na pakete promenljive dužine ili pakete fiksne (konstantne) dužine. Paketi fiksne dužine se označavaju terminom ćelije. Pokazalo se da paketski komutator mnogo efikasnije radi i da ga je znatno jednostavnije konfigurisati kada su paketi fiksne dužine. Neke paketske mreže podrazumevaju pakete fiksne dužine (ATM mreže), a neke paketske mreže podrazumevaju pakete promenljive dužine (IP mreže). Otuda se u slučaju paketa promenljive dužine vrši njihova segmentacija na ćelije fiksne dužine na ulaznim portovima mrežnih čvorova. Potom se ćelije komutiraju, pa se na izlaznom portu rekonstruiše originalni paket i onda se prosleđuje na izlazni link. Ukoliko paketski komutator radi sa paketima fiksne dužine (najčešći slučaj) tada se kaže da on vrši komutaciju ćelija, u suprotnom ako radi sa paketima promenljive dužine tada se kaže da komutator vrši komutaciju paketa (ovo je često zbunjujuće jer se komutator paketa koristi i kao zbirni termin koji obuhvata i komutaciju ćelija i komutaciju paketa promenljive dužine). Očigledno, komutacija paketa promenljive dužine ima određene elemente koji liče na komutaciju kola jer se komutator konfiguriše za prosleđivanje novog paketa dok su još u toku prosleđivanja nekih paketa sa drugih ulaza, ali je učestanost konfigurisanja komutatora znatno veća u slučaju komutacije paketa od slučaja komutacije kola. Takođe, u slučaju komutacije paketa promenljive dužine pojam uslovne blokade ima određenog značaja, ali ipak malog praktičnog efekta jer su trajanja paketa isuviše kratka da bi preraspodela veza kroz komutator (rekonfiguracija komutatora) imala praktičnog smisla. Napomenimo da ćemo u ovom poglavlju podrazumevati komutaciju ćelija čak i ako se koristi termin paket i komutacija paketa, sem ako se drugačije ne naglasi u samom tekstu.

Propusnost komutatora određuje efikasnost, odnosno kapacitet komutatora. Ona se računa za slučaj kada su svi ulazi 100% opterećeni (neprestano dolaze paketi) i svi izlazi 100% opterećeni (pristigli paketi su namenjeni odgovarajućim izlazima tako da nijedan izlaz nije preopterećen, što znači da ako je broj ulaza i izlaza jednak, svi izlazi su 100% opterećeni). Ukupan ostvareni protok na svim izlazima zajedno podeljen sa ukupnim protokom pristižućih paketa sa svih ulaza zajedno predstavlja propusnost komutatora. Cilj je da propusnost komutatora bude što veća, idealno 100%.

Pošto je paketski saobraćaj po prirodi sporadičan (*bursty*) povremeno dolazi do preopterećenja izlaznih portova. Naime, ukupan protok pristiglih paketa namenjenih nekom određenom izlaznom portu prevazilazi brzinu dotičnog porta, odnosno linka koji je povezan na taj port. Na primer, dva paketa namenjena istom izlaznom portu mogu istovremeno pristići na dva ulazna porta. Očigledno, ova dva paketa ne mogu istovremeno i izaći, već jedan za drugim. Otuda je neophodno instalirati bafere u mrežnom čvoru koji će čuvati pakete da ne bi došlo do njihovih nepotrebnih gubitaka. Potreba za baferima takođe predstavlja bitnu razliku u odnosu na komutaciju kola. Baferi u komutaciji kola praktično ne treba da postoje jer resurse koristi samo jedna veza tako da podaci samo protiču kroz dotične resurse bez zadržavanja. Ako i postoji zadržavanje poput T komutatora u TDM komutacionim poljima, ona su veoma kratka i sami ti baferi su veoma malih kapaciteta. Gde će baferi biti pozicionirani u mrežnom čvoru paketske mreže zavisi od same arhitekture mrežnog čvora i tipa paketskog komutatora kojeg koristi. Baferi mogu da se postave na ulazima (ulaznim portovima), na izlazima (izlaznim portovima) ili u paketskom komutatoru. Dozvoljene su i sve međusobne kombinacije. Najpoznatije varijante su (u opisima ćemo podrazumevati pod pojmom paket da je u pitanju paket fiksne dužine tj. ćelija):

- Baferi na izlaznim portovima (*OQ - Output Queuing*)
- Baferi na ulaznim portovima (*IQ - Input Queuing*)
- Baferi na ulaznim i izlaznim portovima (*CIOQ - Combined Input and Output Queuing*)
- Zajednički bafer (*Shared Memory Queuing*)
- Baferi u paketskom komutatoru (*CQ - Crosspoint Queuing*)

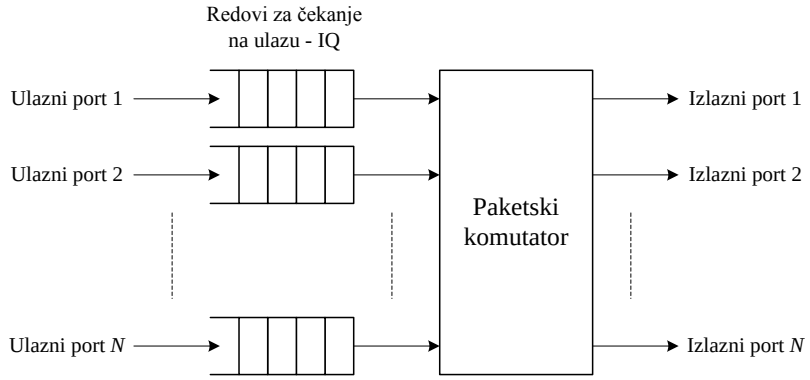


Slika 9.2. Baferi na izlaznim portovima

Principska struktura komutacije paketa u slučaju kada su baferi samo na izlaznim portovima je prikazana na slici 9.2. U slučaju kada su baferi samo na izlaznim portovima, paketi se sa ulaznih portova direktno prosleđuju na izlazne portove i upisuju u bafere na odgovarajućim izlaznim portovima. Prosleđivanja se vrše u takozvanim slotovima, gde je trajanje jednog slota određeno trajanjem jednog paketa (preciznije ćelije). Pošto paketi sa više ulaznih portova mogu istovremeno (u istom slotu) biti namenjeni istom izlaznom portu, paketski komutator mora da obezbedi da se ti paketi mogu proslediti izlaznom portu bez odbacivanja. U suprotnom, arhitektura prikazana na slici 9.2 bi bila loša jer bi dolazilo do prečestih i nepotrebnih odbacivanja paketa, što bi dovelo do veoma loših performansi kako mrežnog čvora, tako i čitave mreže. Postoji više načina da se postigne prosleđivanje paketa bez gubitaka do izlaznih portova

čak i u najgorem slučaju u kome su istovremeno paketi sa svih ulaznih portova namenjeni istom izlaznom portu. Jedan način je da se paketski komutator na svaki izlazni port poveže preko *N* svojih izlaza, gde je *N* broj ulaznih portova (što je ujedno i broj izlaznih portova). U ovom slučaju paketski komutator treba da bude neblokirajući, tj. da obezbedi da se svih *N* ulaznih portova može istovremeno povezati na *N* izlaza komutatora koji idu ka istom izlaznom portu. Problem ovakvog rešenja je prevelika cena i kompleksnost komutatora. Drugo rešenje predviđa upotrebu ubrzanja komutatora. Komutator mora da radi *N* puta brže od dolaznih/odlaznih linkova, što znači da komutator u jednom slotu može da prosledi do *N* paketa sa istog ulaza, odnosno do *N* paketa na isti izlaz. Stoga komutator može sa svih ulaznih portova da prosledi pakete na isti izlazni port bez odbacivanja. Problem ovog rešenja je što postoji tehnološko ograničenje u brzini rada komutatora, tako da u slučaju velikog broja portova i/ili velikih brzina dolaznih/odlaznih linkova postaje tehnološki neizvodljivo postići zahtevano ubrzanje koje bi pokrilo bez gubitaka i najgori slučaj (paketi sa svih ulaznih portova namenjeni istom izlaznom portu). Što se tiče realizacije bafera postoje dva moguća pristupa. Jedan pristup je upotreba FIFO memorije za svaki tok ponaosob. Pod tokom se tipično podrazumeva par (*i*, *j*), gde je *i* ulazni port na koji je došao paket, a *j* izlazni port sa koga će se poslati paket dalje u mrežu. Otuda je broj tokova minimalno *N*. Međutim, u slučaju da postoje i klase saobraćaja (DiffServ) ili dodatne identifikacije tokova (IntServ), tada broj tokova može biti i veći od *N*. Očigledno, u toku slota može doći do maksimalno jednog upisa u FIFO memoriju. Pošto se sa izlaznog porta u trajanju slota može poslati samo jedan paket, to znači da će iz neke od FIFO memorija biti pročitan jedan paket. Otuda najgori slučaj za FIFO memoriju je jedan upis i jedno čitanje u toku jednog slota, što znači da nisu potrebne brze memorije. Međutim, očigledan problem je što je broj FIFO memorija prevelik (minimalno N^2), što je neekonomično rešenje. Pri tome je i iskorišćenje FIFO memorija slabo jer će većina memorija u prozvoljnom posmatranom trenutku biti prazno ili veoma slabo popunjeno, a svaka FIFO memorija ponaosob je dimenzionisana za najgori slučaj (naravno, kada se prepuni FIFO memorija dolazi do odbacivanja). Drugi pristup je upotreba jedne memorije (bafera) na izlaznom portu u koji će da se upisuju paketi svih tokova koji izlaze na dotični izlazni port. U ovom slučaju, u toku jednog slota može da se upiše maksimalno *N* paketa (slučaj kada su u jednom slotu svi ulazni portovi slali paket na isti izlazni port) i da se pročita jedan paket, pa je očigledno da su potrebne veoma brze memorije što predstavlja ozbiljan problem u slučaju brzih linkova i/ili velikog broja portova *N*. Unutar bafera, tokovi su organizovani u vidu tzv. virtuelnih redova za čekanje (*VOQ - Virtual Output Queuing*) koji predstavljaju liste u memoriji. Svaka lista (virtuelni red za čekanje) odgovara jednom toku. Pošto se koriste liste potrebno je čuvati i dodatne informacije pored paketa radi ulančavanja članova liste što povećava memorijske zahteve. Međutim, poredeći sa FIFO varijantom ukupni memorijski zahtevi su značajno manji jer tokovi dele isti bafer pa se dodela memorijskog prostora tokovima dinamički prilagođava trenutnoj saobraćajnoj situaciji pa nema slabog iskorišćenja memorije kao kod FIFO varijante. Takođe ukupan broj memorija je *N*, što je znatno manje od FIFO varijante. Na osnovu navedenog, može se zaključiti da je najveći problem bafera na izlaznim portovima, slaba skalabilnost, odnosno teško je kreirati mrežni čvor sa velikim brojem portova zasnovan na ovoj strukturi. Prednost ove strukture je dobra podrška za kvalitet servisa. Naime, paketi se direktno prosleđuju na izlazne portove, pa izlazni portovi imaju odmah na raspolaganju pakete svih tokova kako oni stignu. Stoga je veoma lako opsluživati tokove u skladu sa željenim ciljevima u pogledu kvaliteta servisa, poput fer opsluživanja u slučaju preopterećenja izlaznog porta (protok paketa koji sa ulaznih portova pristižu na izlazni port

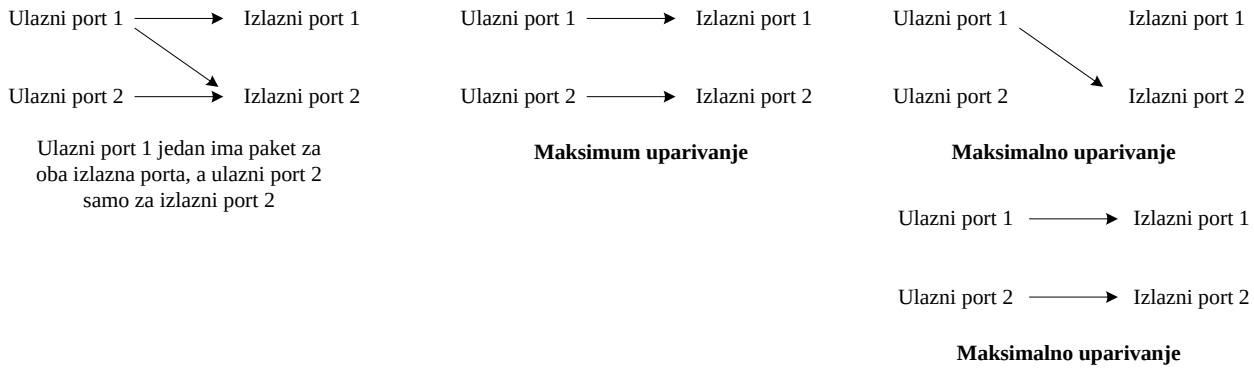
prevazilaze protok izlaznog porta tj. izlaznog linka koji je povezan na njega), prioritetnog opsluživanja tokova osetljivih na kašnjenje i sl.



Slika 9.3. Baferi na ulaznim portovima

Principska struktura komutacije paketa u slučaju kada su baferi samo na ulaznim portovima je prikazana na slici 9.3. U ovom slučaju, paketi se čuvaju u baferima koji su na ulaznim portovima. Koriste se raspoređivači (*scheduler*) koji vrše za svaki slot uparivanje ulaznih portova i izlaznih portova. Upareni par (*i, j*) označava da će sa ulaznog porta *i* biti poslat paket koji će kroz paketski komutator biti prosleđen na izlazni port *j*. Na osnovu rada raspoređivača se vrši konfigurisanje paketskog komutatora tako da poveže uparene parove ulaz/izlaz. Ukoliko paketski komutator radi sa ubrzanjem *s* tada raspoređivač vrši *s* puta proces uparivanja u toku slota, tj. sa jednog ulaza može da se pošalje maksimalno *s* paketa u jednom slotu i na jedan izlaz može da stigne maksimalno *s* paketa u jednom slotu. Za neke algoritme raspoređivanja je teorijski ili eksperimentalno utvrđeno da je ubrzanje 2 dovoljno za odlične performanse i postizanje optimalne propusnosti strukture sa baferima na ulaznim portovima, što je veoma bitna osobina jer ubrzanje 2 je tehnološki izvodljivo. Svaki ulazni port ima jedan bafer. U toku jednog slota se vrši jedan upis (eventualnog paketa ako je pristigao), i vrši se jedno čitanje paketa (ako je ulazni port uparen sa nekim izlaznim portom) ako nema ubrzanja, odnosno maksimalno *s* čitanja ako paketski komutator ima ubrzanje *s*. Čak i ako se koristi ubrzanje njegova vrednost je niska (tipično 2), tako da se ne moraju koristiti preterano brzi baferi, odnosno njihova realizacija je tehnički izvodljiva i ne zavisi od broja portova *N*. Očigledno, mrežni čvorovi sa baferima na ulazu su skalabilni i mogu da podrže i velik broj portova. Prilikom realizacije bafera na ulaznom portu, može se koristiti FIFO bafer ili bafer sa VOQ redovima za čekanje. FIFO bafer je jednostavniji (čuvaju se samo paketi i nema dodatnih informacija), ali njegova upotreba dovodi do tzv. HOL (*Head Of Line*) blokade. Naime, u najgoroj situaciji na vrhu FIFO bafera se na svim ulaznim portovima može naći paket koji namenjen istom izlaznom portu. To znači da će za neki ulazni port trebati da se proslede paketi svih ostalih ulaznih portova, pre nego što se prosledi i njegov paket (*N*-1 konfigurisanja paketskog komutatora će čekati). Ako su iza tog paketa postojali neki drugi paketi namenjeni drugim izlaznim portovima, bilo bi zgodno kada bi se oni mogli poslati, da ulazni port ne bi nepotrebno bio besposlen, ali to nije moguće zbog FIFO bafera (isto važi i za druge ulazne portove koji su takođe čekali na slanje svog paketa). Usled HOL blokade može doći do značajnog obaranja propusnosti paketskog komutatora naročito pri velikim opterećenjima ulaznih portova, tako da se iz tog razloga FIFO baferi više ne koriste. Bafer sa VOQ redovima za čekanje je nešto komplikovaniji za realizaciju od FIFO bafera, ali nema problem HOL blokade, pošto se paketi nalaze u listama pa se paket iz bilo koje liste može proslediti. Takođe, VOQ

redovi za čekanje komplikuju algoritme za raspoređivanje. U slučaju FIFO bafera, algoritmi za raspoređivanje rade sa N podataka (paketi koji su prvi u FIFO baferima), ali u slučaju VOQ redova za čekanje rade sa N^2 podataka jer se svi tokovi (tj. redovi za čekanje) uzimaju u obzir pri proračunu konfiguracije paketskog komutatora, tj. uparivanja ulaznih i izlaznih portova. Što se tiče samog uparivanja razlikuju se maksimum (*maximum*) i maksimalno (*maximal*) uparivanje. Maksimum uparivanje pokušava naći maksimalan broj uparivanja koji može da se ostvari, dok maksimalno uparivanje podrazumeva inkrementalno dodavanje parova (po nekom algoritmu) sve dok se novi parovi mogu dodati. U opštem slučaju maksimalno uparivanje kreira manje parova od maksimum uparivanja. Maksimalno uparivanje je jednostavije za realizaciju, ali u proseku kreira manje parova od maksimum uparivanja. S druge strane, maksimum uparivanje može da izazove u određenim saobraćajnim slučajevima opsluživanje tokova koje nije fer, a u ekstremnim slučajevima može čak da dovede i do potpunog neopsluživanja pojedinih tokova. Jednostavan primer maksimum i maksimalnog uparivanja je dat na slici 9.4. Ulazni port 1 ima pakete za oba izlazna porta, dok ulazni port 2 ima paket samo za izlazni port 2. Maksimum uparivanje će da postigne maksimalan broj uparivanja, a to je povezivanje ulaza 1 sa izlazom 1 i ulaza 2 sa izlazom 2 kao na slici. Maksimalno uparivanje će da vrši inkrementalno formiranje parova. U gornjem primeru za maksimalno uparivanje sa slike je prvo uparen ulaz 1 sa izlazom 2. Nakon ovog uparivanja ne mogu da se dodaju novi parovi. Očigledno, broj uparivanja je manji od maksimum uparivanja. Međutim, algoritam maksimalnog uparivanja je mogao da spoji ulaz 1 sa izlazom 1. Tada bi mogao da se doda i par ulaz 2/izlaz 2 i postigne isto rešenje kao kod maksimum uparivanja (donji primer za maksimalno uparivanje). Otuda, u opštem slučaju maksimalno uparivanje postiže manji ili jednak broj parova u odnosu na maksimum uparivanje. Lako je uočiti u datom primeru da ako neprestano ima paketa na ulaznom portu 1 za izlazni port 1, i ulaznom portu 2 za izlazni port 2, i pri tome nikad nema paketa za izlazni port 1 na ulaznom portu 2, da nikad neće doći do spajanja ulaznog porta 1 i izlaznog porta 2, čime nikad ne bi bio opslužen tok (1,2).



Slika 9.4. Maksimum i maksimalno uparivanje

Algoritmi raspoređivanja tipično rade maksimalno uparivanje. Poznati algoritmi raspoređivanja su, na primer, PIM (*Parallel Iterative Matching*) i iSLIP, ali postoje i mnogi drugi algoritmi raspoređivanja predloženi u literaturi. PIM algoritam, kojim se proračunava uparivanje ulaza sa izlazima, se sastoji iz k iteracija (broj iteracija je $k \leq N$), pri čemu se svaka iteracija sastoji iz 3 koraka:

1. Svaki neupareni ulaz šalje zahteve svim izlazima za koje ima pakete u svojim VOQ redovima za čekanje.

2. Svaki neupareni izlaz na slučajan način bira primljeni zahtev koji će potvrditi iz skupa zahteva koje je primio. Pri tome, svaki primljeni zahtev ima podjednaku verovatnoću da će biti izabran. Šalje se odobrenje ulazu čiji je zahtev izabran.
3. Svaki neuparen ulaz bira na slučajan način odobrenje koje je primio od neuparenih izlaza. Pri tome, svako primljeno odobrenje ima podjednaku verovatnoću da će biti izabrano. Vrši se uparivanje dotičnog ulaza sa izlazom čije je odobrenje izabrano.

PIM algoritam je jednostavan za implementaciju (sem donekle kvalitetne funkcije za slučajan odabir), ali problem je što ne postiže veoma dobru propusnost paketskog komutatora, a takođe može da dovede i do nefer opsluživanja u slučaju velikih saobraćajnih opterećenja. iSLIP algoritam predstavlja unapređenje PIM algoritma. Takođe se sastoji iz k iteracija (broj iteracija je $k \leq N$), pri čemu se svaka iteracija sastoji iz 3 koraka:

1. Svaki neupareni ulaz šalje zahteve svim izlazima za koje ima pakete u svojim VOQ redovima za čekanje.
2. Svaki neupareni izlaz bira po round-robin principu (počev od najprioritetnijeg ulaza) primljeni zahtev koji će potvrditi iz skupa zahteva koje je primio. Šalje se odobrenje ulazu čiji je zahtev izabran.
3. Svaki neuparen ulaz bira po round-robin principu (počev od najprioritetnijeg izlaza) primljeno odobrenje koje će potvrditi iz skupa odobrenja koje je primio. Vrši se uparivanje dotičnog ulaza sa izlazom čije je odobrenje izabrano. U slučaju prve iteracije za upareni ulaz se podešava da najprioritetniji izlaz bude prvi sledeći izlaz (po internoj kružnoj numeraciji portova u mrežnom čvoru, na primer, ako je upareni izlaz 5, tada će najprioritetniji izlaz da postane 6) iza izlaza koji je uparen sa dotičnim ulazom. Isto tako, u slučaju prve iteracije za upareni izlaz se podešava da najprioritetniji ulaz bude prvi sledeći ulaz (po internoj kružnoj numeraciji portova u mrežnom čvoru, na primer, ako je upareni ulaz 2, tada će najprioritetniji ulaz da postane 3) iza ulaza koji je uparen sa dotičnim izlazom. Ovo ažuriranje prioritetnih ulaza i izlaza na uparenim ulazima/izlazima se radi samo u prvoj iteraciji jer je primećeno da u suprotnom može da se desi da neki tokovi nikada ne budu opsluženi u određenim saobraćajnim slučajevima.

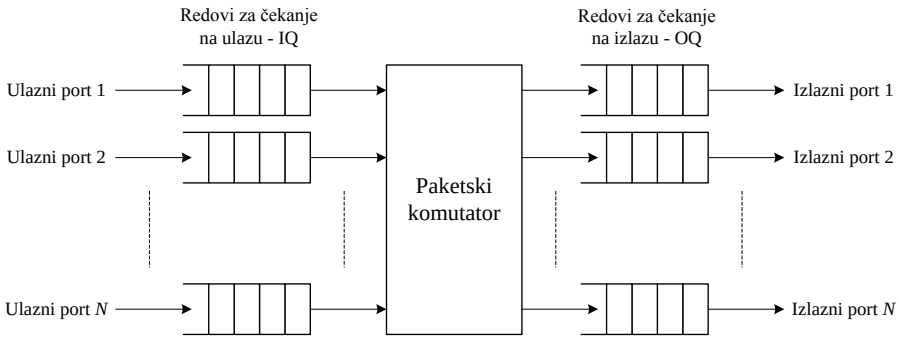
iSLIP postiže značajno bolje perfomanse od PIM algoritma za uparivanje, pri čemu iSLIP ima nešto složeniju implementaciju (moraju da se čuvaju informacije o prioritetnim ulazima/izlazima na izlazima/ulazima). Inače, iSLIP ne predstavlja akronim, već je autor algoritma (Nick McKeown) nadenuo ovaj naziv algoritmu.

Algoritmi raspoređivanja trebaju da proračunaju konfiguraciju paketskog komutatora, odnosno uparivanja ulaza/izlaza za trajanje jednog slota ako nema ubrzanja, odnosno za trajanje s-tog dela slota ako paketski komutator ima ubrzanje s. U slučaju veoma brzih portova tj. linkova trajanje vremenskog slota je veoma kratko (par desetina ns, pa čak i manje) pa je bitno u tim situacijama da algoritmi raspoređivanja brzo rade. Otuda, pojedini algoritmi raspoređivanja proširuju svoj proračun na više slotova, ali se radi pajplajn, tj. kao na serijskoj traci se započinju proračuni za buduće slotove jedan za drugim. Proračun traje više slotova, ali pošto se radi više proračuna u isto vreme, postiže se da se rezultati proračuna konfiguracije komutatora ređaju jedan za drugim u potrebnom tempu (jedan po slotu ako nema ubrzanja, odnosno s po slotu ako

ima ubrzanja *s*). SGS (*Sequential Greedy Scheduler*) algoritam raspoređivanja radi po ovom principu.

Treba napomenuti i da mrežni čvorovi kod kojih su baferi na ulazima imaju slabiju podršku za kvalitet servisa od mrežnih čvorova sa baferima na izlazu jer su paketi tokova na ulazima tj. distribuirani po ulaznim portovima pa je teže organizovati podršku za kvalitet servisa. Postoje algoritmi raspoređivanja koji uzimaju u obzir određene aspekte kvaliteta servisa prilikom proračuna uparivanja. Na primer, težinski iSLIP (*Weighted iSLIP*) algoritam raspoređivanja uzima u obzir i težine tokova prilikom proračuna uparivanja radi postizanja fer opsluživanja tokova u slučaju pojave preopterećenja pojedinih izlaza u skladu sa težinama tokova.

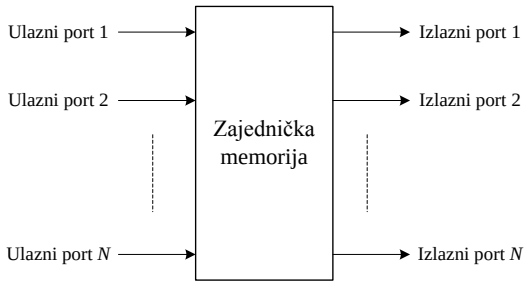
Na kraju navedimo i jednu važnu činjenicu vezanu za strukture sa baferima na ulazu. Naime, ako se koristi ubrzanje *s* paketskog komutatora, tada do *s* paketa u jednom slotu može da stigne na jedan izlaz kao što smo već napomenuli ranije u tekstu. Pošto izlaz može da pošalje samo jedan paket u jednom slotu, očigledno je da izlazi takođe moraju da implementiraju bafere jer bi u suprotnom došlo do gubitaka paketa. Stoga, ako se koristi ubrzanje *s*, u suštini je u pitanju struktura sa baferima na ulaznim i izlaznim portovima koja je prikazana na slici 9.5. Međutim, u literaturi se i pored toga često pojedini algoritmi raspoređivanja, odnosno strukture označavaju kao strukture sa baferima na ulazu iako definišu i upotrebu ubrzanja *s*, čime ustvari ukazuju na činjenicu da baferi postoje i na izlazu. Kao što se vidi i iz ovog primera, termini u literaturi neretko mogu da budu zbunjujući, što je često posledica da u istom periodu više različitih autora obrađuju istu oblast pa često dođe do pojave istih termina sa nešto različitim tumačenjima, ali i različitih termina koji se odnose na istu stvar.



Slika 9.5. Baferi na ulaznim i izlaznim portovima

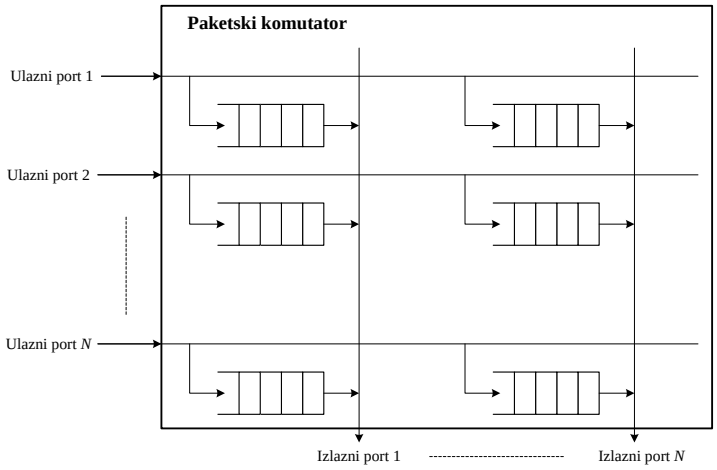
Principska struktura komutacije paketa u slučaju kada su baferi i na ulaznim portovima i na izlaznim portovima je prikazana na slici 9.5. Ideja ove strukture je ublažavanje problema koje su originalno imale strukture sa baferima samo na izlazu, odnosno sa baferima samo na ulazu. Paketski komutator radi sa ubrzanjem *s*, ali to ubrzanje je manje od *N* tj. vrednost *s* je dovoljno mala da bude tehnološki moguće da se realizuje. Na ovaj način se omogućavalo prosleđivanje do *s* paketa na jedan izlaz u jednom slotu, a oni paketi koji nisu mogli da se proslede su zadržavani u baferima na ulazu. Time je postignut približan rad kao kod strukture koja je imala bafere samo na izlazu. U početku su i na ulazu i na izlazu korišćeni FIFO baferi. Ubrzanje *s* je označavalo da može više paketa da se prosledi sa jednog ulaza u toku slota, čime je umanjen efekat HOL blokade, jer je paket kraće vremena pravio blokadu. Međutim, i pored toga su kreirane i varijante koje su ulazu koristile bafer sa virtuelnim redovima za čekanje. Ovo je omogućavalo da algoritmi raspoređivanja efikasnije raspoređuju pakete za slanje tj. da efikasnije vrše uparivanje

ulaza i izlaza i postignu visoku propusnost paketskog komutatora, što smo već opisali kod struktura sa baferima na ulazu.



Slika 9.6. Zajednički bafer

Principska struktura komutacije paketa u slučaju kada se koristi zajednički bafer (memorija) je prikazana na slici 9.6. U ovom slučaju paketi sa svih ulaznih portova se upisuju u istu, zajedničku memoriju tj. bafer. Istovremeno, i paketi koji se prosleđuju na izlazne portove se čitaju iz iste, zajedničke memorije. Paketi se upisuju u odgovarajuće liste tj. VOQ redove za čekanje u zajedničkoj memoriji koje su pridružene odgovarajućim izlaznim portovima, pa stoga postoji *N* listi u zajedničkoj memoriji. Naravno, ako se koriste i mehanizmi kvaliteta servisa broj tokova može biti i veći, tj. jednom izlaznom portu može biti pridruženo i više od jedne liste, pa bi tada ukupan broj listi bio veći od *N*. Pošto je svakom izlaznom portu pridružena odgovarajuća lista koja sadrži sve pakete namenjene dotičnom izlaznom portu, moguće je postići dobru kontrolu kvaliteta servisa kao i fer opsluživanja koja je ekvivalentna onoj koju je postizala struktura sa baferima samo na izlaznim portovima. Takođe, upotreba zajedničkog bafera obezbeđuje optimalnu upotrebu memorijskih resursa, tj. ova struktura ima najmanje memorijske zahteve jer obezbeđuje dinamičko prilagođavanje memorijskog prostora svim listama odjednom (ne mogu sve biti istovremeno prepunjene tj. isto popunjene). Ali, postoji i velika mana ove strukture. U toku jednog slota mora biti moguće upisati *N* paketa i pročitati *N* paketa, što znači da bafer mora biti veoma brz, pa u slučaju veoma brzih portova/linkova i/ili velikog broja portova može postati tehnološki neizvodljivo formirati tako brz bafer, što znači da ova struktura ima problem sa skalabilnošću.



Slika 9.7. Baferi u paketskom komutatoru

Principska struktura komutacije paketa u slučaju kada se baferi nalaze u paketskom komutatoru je prikazana na slici 9.7. Na slici 9.7 je prikazana krosbar struktura gde se u ukrsnim

tačkama nalaze baferi, međutim, baferi se mogu staviti i u druge realizacije paketskih komutatora. U slučaju kada se baferi nalaze samo u paketskom komutatoru, logično je da će u pojedinim slotovima kada se paketi prosleđuju sa ulaznih na izlazne portove, neki od njih usmeravati ka istim izlaznim portovima, što će dovesti do kolizije na izlazu komutatora. Otuda se stavljaju baferi u komutator i u njih se smeštaju paketi koji izgube u borbi koja određuje koji od njih će biti prosleđen na izlaz. Isto tako, pojedini komutatori su interno blokirajući pa tako čak i kada paketi nisu namenjeni istom izlaznom portu može doći do kolizije unutar samog komutatora. Da ne bi došlo do prosleđivanja samo jednog paketa i odbacivanja svih ostalih paketa u koliziji, mogu se ugraditi baferi u koje se smeštaju paketi koji su izgubili borbu za prosleđivanje prilikom kolizije. Problem ove strukture je prevelik broj bafera koje treba ugraditi u slučaju komutatora sa velikim brojem ulaza i izlaza, pa su tipično baferi veoma mali i sadrže prostor za svega nekoliko paketa, do eventualno stotinjak paketa (što je retko sem kod komutatora sa manjim brojem ulaza i izlaza). Otuda, se ove strukture ipak kombinuju sa baferima na ulazu koji su većih kapaciteta i koji će zadržavati pakete na ulazu da ne bi došlo do preopterećivanja bafera u komutatoru i time do nepotrebnih odbacivanja. Inače, u strukturi prikazanoj na slici 9.7, paketi se direktno upisuju u odgovarajuće bafere. Paket sa nekog ulaza se ispisuje u bafer koji se nalazi u tački ukrštanja horizontalne linije koja odgovara dotičnom ulazu i vertikalne linije koja odgovara izlazu kome je paket namenjen. Svaki izlazni port kontroliše čitanje bafera iz svoje vertikale jer ti baferi sadrže pakete namenjene tom izlazu. Opsluživanje bafera zavisi od algoritma opsluživanja koji može biti round-robin princip, opsluživanje najpopunjenijeg bafera i dr. Takođe, treba primetiti da u toku jednog slota može da bude maksimalno jedan upis i jedno čitanje iz bafera, što znači da ne postoji problem brzine bafera kao kod nekih drugih struktura. Isto tako, broj bafera je N^2, što jeste problem u slučaju velikog broja ulaznih i izlaznih portova, odnosno ovakva struktura nije skalabilna.

Napomenimo još jednu bitnu razliku između komutacije kola i paketa. U komutaciji kola korisnička informacija ne treba da nosi dodatne informacije neophodne za procesiranje unutar mreže jer je put kojim ide informacija unapred zauzet i samim tim poznat. U slučaju komutacije paketa neophodno je dodati informacije neophodne za obradu paketa u mrežnim čvorovima. Ove dodatne informacije se stavljaju u zaglavlje paketa. Na primer, neophodna je informacija koja će mrežnim čvorovima pomoći da odrede na koji izlaz da proslede paket - odredišna IP adresa u slučaju IP mreža.

Paketski komutatori mogu da se podele na:

- Vremenski komutatori
- Prostorni komutatori

Vremenski komutatori izvršavaju komutaciju paketa u vremenskom domenu, a prostorni u prostornom domenu. Vremenski komutatori se dalje dele na:

- Komutatori sa zajedničkim medijumom za prenos
- Komutatori sa zajedničkom memorijom

Napomenimo da u slučaju komutacije ćelija kod prostornih komutatora, ćelije moraju da budu fazno sinhronisane na ulazima u komutator, tj. sve moraju da započnu u istom momentu da se prosleđuju ka komutatoru. Prostorni komutatori se dele na:

- Komutatori sa jednostrukim putanjama

- Komutatori sa višestrukim putanjama

Komutatori sa jednostrukim putanjama predstavljaju komutatore kod kojih postoji samo jedan put između ulaza *i* i izlaza *j*. Ako postoji više putanja, tada je u pitanju komutator sa višestrukim putanjama. Komutatori sa jednostrukim putanjama se dalje dele na:

- Krosbar komutatori
- Potpuno povezani komutatori
- Banyan komutatori

Komutatori sa višestrukim putanjama se dalje dele na:

- Umnoženi Banyan komutatori
- Klosovi komutatori
- Višeravanski komutatori
- Recirkulacioni komutatori

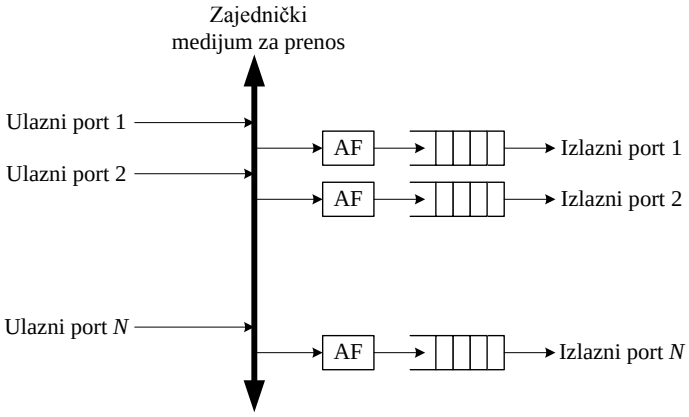
U nastavku ovog poglavlja će biti objašnjeni navedeni tipovi komutatora. Napomenimo da postoje i druge podele paketskih komutatora u literaturi. Takođe, veoma je teško napraviti preciznu podelu paketskih komutatora s obzirom na njihovu brojnost, ali isto tako i često preklapanje osobina pojedinih komutatora, pa je teško za pojedine tipove komutatora utvrditi kojoj tačno grupi pripadaju što će se i videti u nastavku ovog poglavlja. Još jednom naglasimo da ćemo u opisima komutatora u nastavku poglavlja podrazumevati da se radi komutacija ćelija i da se termin paket sa stanovišta komutatora odnosi na ćeliju tj. paket fiksne dužine, sem ako se ne naglasi drugačije u samom tekstu.

Na kraju, rezimirajmo još jednom osnovne razlike komutacije kola i paketa:

- Komutacija kola ima slabije iskorišćenje mrežnih resursa od komutacije paketa.
- Komutacija kola ima bolju podršku za kvalitet servisa od komutacije paketa.
- Komutacija kola konfiguriše komutator na bazi konekcija, a komutacija paketa na bazi paketa (koji su sadržaj konekcije).
- Komutacija paketa zahteva veću moć procesiranja jer se obrada vrši na bazi paketa.
- Komutacija paketa dodaje dodatne informacije neophodne za procesiranje paketa u mrežnim čvorovima (zaglavlje paketa), dok komutacija kola ne mora da dodaje dodatne informacije.
- Komutacija paketa zahteva upotrebu bafera (u suštini, baferi ne moraju da se koriste, ali tada bi komutacija paketa praktično bila neupotrebljiva), dok komutacija kola ne zahteva upotrebu bafera.
- Komutacija kola zahteva proces uspostave veze jer je ona neophodna za zauzimanje resursa, dok kod komutacije paketa proces uspostave veze nije neophodan (ali može da se izvrši).

9.1. Komutatori sa zajedničkim medijumom za prenos

U ovom slučaju, komutator je, ustvari, medijum za prenos kojeg dele svi ulazni i izlazni portovi. Struktura ovog komutatora je prikazana na slici 9.1.1. Svakom ulaznom portu je dodeljen jedan kanal u vremenskom (TDM) multipleksu na zajedničkom medijumu za prenos, pri čemu jedna perioda TDM multipleksa sadrži *N* vremenskih kanala, gde je *N* broj ulaznih/izlaznih portova. Ulazni portovi šalju pakete u svojim vremenskim kanalima. Svi izlazni portovi su priključeni takođe na zajednički medijum i istovremeno svi izlazni portovi primaju TDM multipleks, a samim tim i pakete koje šalju ulazni portovi. Svaki izlazni port implementira adresni filtar (AF) koji propušta samo one pakete namenjene dotičnom izlaznom portu. Propušteni paketi se upisuju u bafer (tipično je u pitanju FIFO bafer, ali može da se koristi i bafer koji sadrži VOQ redove za čekanje ako je potrebno implementirati mehanizme za fer opsluživanje i kvalitet servisa). Pošto primerak paketa sa ulaznog porta stiže do svih izlaznih portova, ovaj komutator ima internu podršku za multikast. Sam medijum za prenos može biti magistrala kao na slici 9.1.1, ali i prsten.



Slika 9.1.1. Komutator sa zajedničkim medijumom za prenos

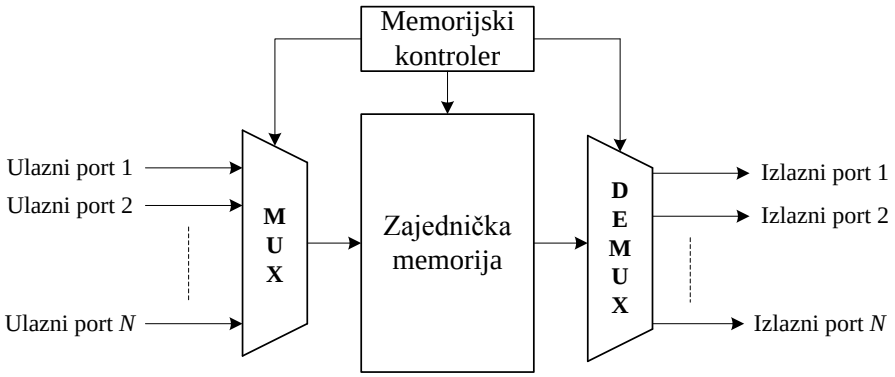
Međutim, postoje i značajni problemi ovog komutatora, koji su ekvivalentni problemima struktura sa baferima samo na izlaznim portovima (kod ovog komutatora su baferi samo na izlaznim portovima). Očigledno, medijum za prenos mora da radi na brzini *N* puta većoj od brzine portova da bi bio moguć TDM multipleks od *N* kanala, pošto jedan kanal mora da ima protok jednak protoku jednog porta. Takođe, u najgorem slučaju za vreme jedne periode TDM multipleksa, može da se desi da se svi paketi (iz svih kanala, tj. sa svih ulaznih portova) upišu u bafer istog izlaznog porta. Otuda, bafer mora biti sposoban da u jednom slotu, koji odgovara jednoj periodi TDM multipleksa, izvrši *N* upisa i jedno čitanje (paket koji se prosleđuje na izlazni link), tj. moraju se implementirati veoma brzi baferi. Na osnovu navedenih problema, jasno je da ovaj komutator nije skalabilan i ne može da podrži velik broj portova u slučaju brzih linkova.

9.2. Komutatori sa zajedničkom memorijom

Komutatore sa zajedničkom memorijom smo već opisali prilikom opisa struktura sa zajedničkim baferom. Kao što se može videti sa slike 9.6, svi ulazni i izlazni portovi dele zajedničku memoriju tj. bafer. Međutim, memorije tipično imaju ograničen broj portova za upis i čitanje (jedan ili dva) i očigledno je nemoguće da se svi ulazni i izlazni portovi zasebno spoje na

memorijske portove za upis, odnosno čitanje. Zato se koristi vremenski (TDM) multipleks za organizovanje pristupa za upis ulaznim portovima, odnosno pristupa za čitanje izlaznim portovima. Realizacija komutatora sa zajedničkom memorijom je prikazana na slici 9.2.1. Ulazi se vezuju preko multipleksera na port za upis zajedničke memorije. Port za čitanje zajedničke memorije se vezuje na demultiplekser, a izlazi demultipleksera se vezuju na izlazne portove. Memorijski kontroler u TDM maniru kontroliše rad multipleksera tako što u toku trajanja slota (paketa) povezuje svaki ulazni port jednom na port za upis zajedničke memorije. Takođe, memorijski kontroler određuje gde će biti upisan dotični paket (adresa upisa), na osnovu podataka sa odgovarajućeg ulaznog porta (tipično rezultat lukap funkcije na ulaznom portu, ali mogu biti i dodatni podaci ako se implementira podrška za kvalitet servisa). Na slici nisu prikazane linije kojima se šalju ovi podaci sa ulaznih portova do memorijskog kontrolera. Na identičan način, memorijski kontroler u TDM maniru kontroliše i rad demultipleksera tako što port za čitanje zajedničke memorije spaja sa svakim izlaznim portom tačno jednom u toku trajanja slota. Na portu za čitanje se nalazi paket pročitan iz memorije.

Prednost ove strukture je mogućnost ostvarivanja kvalitetne kontrole fer opsluživanja i postizanja željenog kvaliteta servisa (ekvivalentno strukturi sa baferima samo na izlaznim portovima), a takođe ovaj komutator optimalno troši memorijske resurse kao što je objašnjeno kod struktura sa zajedničkim baferom. Takođe, ovaj komutator ima podršku za multikast saobraćaj jer paket upisan u zajedničku memoriju može biti prosleđen proizvoljnim izlaznim portovima. Naravno, kao što je već rečeno, velika mana je potreba za veoma brzim memorijama jer u jednom slotu (vreme trajanja paketa) treba omogućiti N upisa i N čitanja u najgorem slučaju. Otuda, ovaj komutator nije skalabilan.

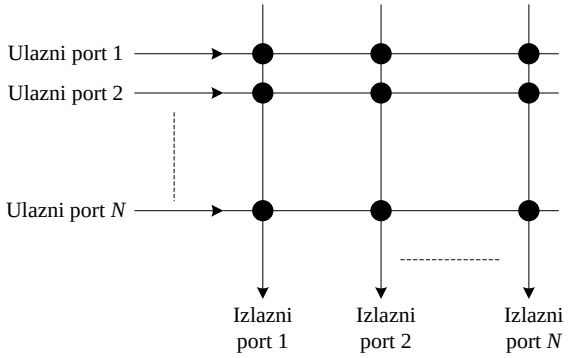


Slika 9.2.1. Komutator sa zajedničkom memorijom

9.3. Krosbar komutatori

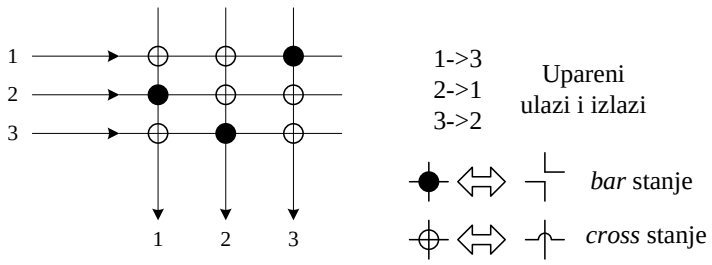
Krosbar komutatori su veoma slični S komutatorima iz TDM komutacionih polja telefonskih centrala. Ovaj komutator, prikazan na slici 9.3.1, predstavlja rešetkastu strukturu, gde svaki red (horizontalna linija) odgovara jednom ulaznom portu, a svaka kolona (vertikalna linija) odgovara jednom izlaznom portu. U preseku linija se nalaze prekidači koji kontrolišu rad krosbar komutatora tj. njegovu konfiguraciju (parove uparenih ulaza i izlaza). Po difoltu prekidači su otvoreni (*cross* stanje). Ako se želi spojiti ulaz i sa izlazom j, tada će se zatvoriti prekidač u preseku horizontalne linije i i vertikalne linije j. Stanje zatvorenog prekidača se još označava i terminom *bar* stanje. Primer konfiguracije 3x3 krosbara je dat na slici 9.3.2 gde je izvršeno spajanje sledećih parova ulaz-izlaz: (1,3), (2,1), (3,2). Na slici 9.3.2 je prikazan interni izgled

prekidača kad je on otvoren (*cross* stanje), odnosno zatvoren (*bar* stanje). Očigledno, u jednoj koloni ne sme biti više zatvorenih prekidača jer bi došlo do grešaka u prosleđivanju paketa. Takođe, nema smisla i da više prekidača bude zatvoreno na jednoj horiznotalnoj liniji, jer ako se na slici 9.3.2 pogleda interno stanje zatvorenog prekidača vidi se da bi samo prvi zatvoreni prekidač imao efekta.



Slika 9.3.1. Krosbar komutator

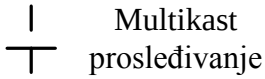
Krosbar komutator je neblokirajući pa ne zahteva interno ubrzanje. Veoma je jednostavan za realizaciju i takođe je modularan tj. veoma lako se mogu dodati novi portovi. Međutim, broj prekidača raste sa N^2, gde je N broj ulaznih, odnosno izlaznih portova, što predstavlja problem u slučaju velikog broja portova. Otuda se krosbar komutator uglavnom koristi za manje i srednje veličine komutatora. Takođe, krosbar komutator se može koristiti kao gradivna jedinica u višestepenim komutatorima poput Klosovih komutatora (slično kao kod analognih komutacionih polja u komutaciji kola). Naravno, pošto istovremeno (u istom slotu) više paketa može biti namenjeno istom izlaznom portu, neophodno je implementirati bafere da bi se sprečili nepotrebni gubici paketa. Baferi mogu da se implementiraju na ulaznim portovima, unutar komutatora ili na izlaznim portovima (mogu se kreirati i kombinacije prethodno tri nabrojane varijante), a osobine ovih varijanti smo već objasnili ranije u poglavlju. U slučaju implementacije bafera samo na izlazu, krosbar mora raditi sa ubrzanjem jer jedino tako može proslediti više paketa na isti izlaz u okviru jednog slota.



Slika 9.3.2. Primer konfiguracije 3x3 krosbar komutatora

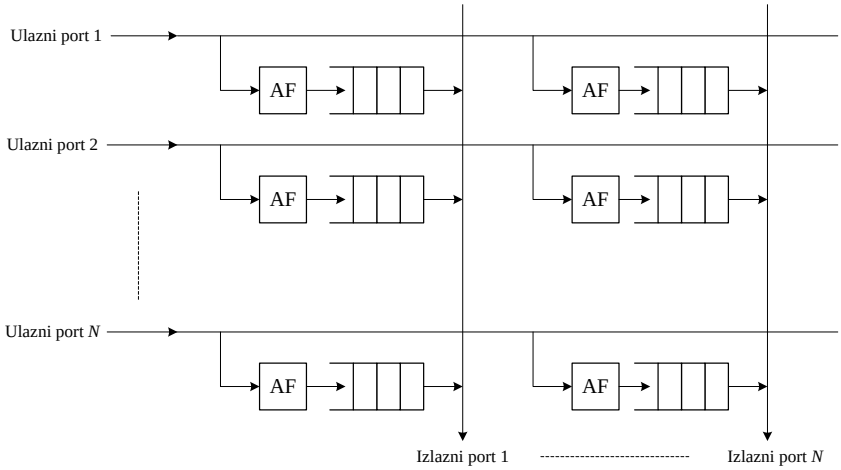
Napomenimo da krosbar komutator ima i osobinu samorutiranja. Naime, krosbar komutator može da se konfiguriše kontrolom spolja, ali ima i mogućnost da se samostalno konfiguriše - osobina samorutiranja. Naime, paketu koji se prosleđuje kroz krosbar dodaje se interno zaglavlje (koje postoji i koristi se samo na nivou mrežnog čvora) koje sadrži identifikaciju izlaznog porta na koji treba paket da se prosledi. Svaka ukrsna tačka ima prostu logiku kojom se na osnovu vrednosti zaglavlja (tipično samo jednog bita iz zaglavlja) vrši otvaranje ili zatvaranje prekidača čime se, ustvari, vrši samorutiranje paketa kroz krosbar

komutator. U slučaju krosbar komutatora, paket vrši konfiguraciju prekidača na horizontalnoj liniji na kojoj je i ušao jer onog momenta kada se skrene na odgovarajućoj vertikali tada se direktno spušta do odgovarajućeg izlaznog porta (nijedan drugi ulazni port neće na dotičnoj vertikali zatvoriti prekidač jer bi u suprotnom došlo do kolizije, naravno, ovo pod uslovom da se vodi računa da se u istom trenutku ne šalju paketi ka istom izlaznom portu sa dva ili više ulaznih portova).



Slika 9.3.3. Interno stanje prekidača za multikast prosleđivanje

Ako se koriste samo dva stanja prekidača (*bar* i *cross*) tada krosbar komutator nema internu podršku za multikast pakete. Da bi je ostvario moralo bi se dodati još jedno stanje prekidača prikazano na slici 9.3.3 koje bi pored prosleđivanja paketa ka izlazu, prosleđivalo paket i dalje na horizontalnu liniju. Naravno, i dalje na nivou kolone sme biti zatvoren maksimalno jedan prekidač (u *bar* stanju ili stanju za multikast prosleđivanje).



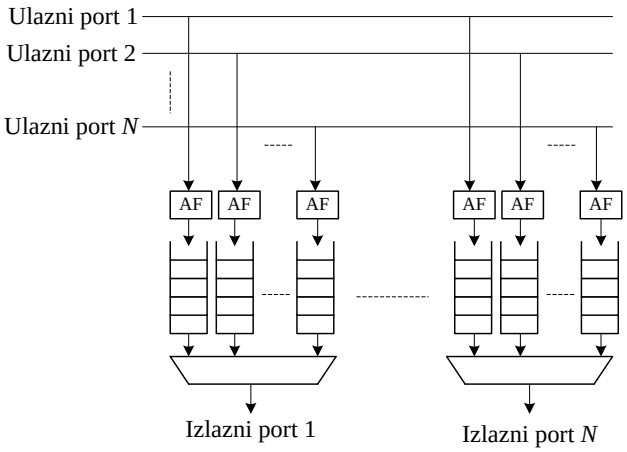
Slika 9.3.4. Krosbar komutator sa baferima u ukrsnim tačkama

U varijanti kada se u krosbar komutatoru koriste baferi, baferi se stavljaju u ukrsne tačke na način prikazan na slici 9.3.4. Ispred svakog bafera se postavlja adresni filtar kojim se određuje da li će se paket upisati u bafer ili ne. Očigledno, u ovom slučaju je multikast saobraćaj podržan jer je moguć upis u više bafera odjednom (na istoj horizontalnoj liniji) za paket sa jednog ulaznog porta. Izlazni port kontroliše čitanje iz bafera na svojoj vertikalnoj liniji jer ti baferi sadrže pakete namenjene tom izlaznom portu. Algoritam opsluživanja tih bafera određuje način na koji će se paketi čitati iz njih (na primer, round-robin princip, opslužiti bafer koji sadrži najviše paketa i dr.), pri čemu je moguća dobra kontrola kvaliteta servisa i fer opsluživanja približna onoj kod struktura sa baferima na izlazu (približna, a ne identična iz razloga što su u praksi baferi u ukrsnim tačkama ipak manji od onih koji bi se implementirali na izlaznim portovima pa je veća verovatnoća odbacivanja paketa). Dobra strana je i to što nisu potrebni brzi baferi, maksimalno jedan upis i jedno čitanje po slotu. Mana ovog komutatora je veliki broj ukrsnih tačaka za velik broj portova *N*, a samim tim i velik broj bafera, pa se ova vrsta komutatora koristi samo za manje i srednje veličine komutatora. Otuda su baferi tipično mali i mogu sadržati manji broj paketa, pa ako se ne bi koristili baferi na ulazu dolazilo bi potencijalno do velikih gubitaka paketa u mrežnom čvoru. Pri tome je važno uočiti da se postiže slaba iskorišćenost bafera, jer svaki bafer

odgovara samo jednom toku, tj. nema deljenja bafera između više tokova čime bi se optimizovalo memorijsko iskorišćenje. Stoga se najčešće krosbar komutatori sa baferima u ukrsnim tačkama koriste u kombinaciji sa baferima na ulaznim portovima, pri čemu baferi na ulaznim portovima mogu biti FIFO baferi ili baferi sa VOQ redovima za čekanje. Pošto FIFO baferi mogu da izazovu HOL blokadu (situacija da je za određeni ulazni port neki bafer u krosbaru prepunjen i prvi paket u FIFO dotičnog ulaznog porta je namenjen baš tom baferu), češće se koristi varijanta sa VOQ redovima za čekanje.

9.4. Potpuno povezani komutatori

Potpuno povezani komutatori podrazumevaju da postoji zaseban fizički put za svaki par ulaz-izlaz. Struktura ovog komutatora je prikazana na slici 9.4.1.

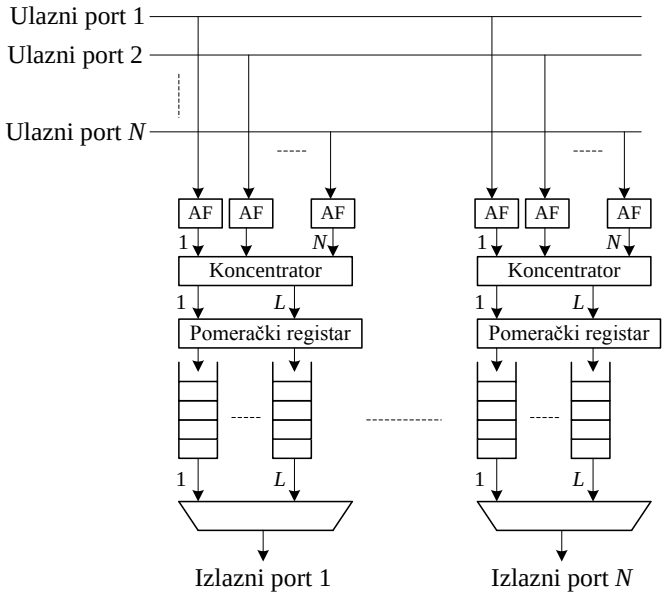


Slika 9.4.1. Potpuno povezani komutator

Kao što se vidi, postoji posebna linija od bilo kog ulaznog porta do bilo kog izlaznog porta. Potpuno povezani komutatori koriste FIFO bafere na izlazu, jer u okviru jednog slota može biti namenjeno više paketa za isti izlazni port. Ulazni port šalje paket na sve svoje linije tj. ka svim izlaznim portovima, a adresni filtri (AF) određuju u koji bafer će paket biti upisan. Ova struktura ima podršku za multikast jer paket dolazi uvek do svih izlaznih portova. Ovi komutatori imaju dobru podršku za kvalitet servisa i fer opsluživanje jer su ekvivalentni strukturi sa baferima samo na izlazu. Struktura komutatora je jednostavna i nisu potrebni brzi baferi (maksimalno jedan upis i jedno čitanje po slotu), a takođe nije potrebno ni ubrzanje rada komutatora. Očigledan problem je prevelik broj bafera (N^2), pa ovaj komutator nije skalabilan. Takođe, pošto je svakom toku dodeljen zaseban bafer, iskorišćenje bafera je slabo. Umesto N bafera na izlaznom portu, može se koristiti samo jedan bafer (tada se postiže bolje iskorišćenje bafera), ali tada to mora biti brz bafer koji može za vreme jednog slota da upiše N paketa i ispiše jedan paket. Fizička realizacija takvog bafera bi podrazumevala multiplekser na koji bi se vodile sve linije iz adresnih filtara dotičnog izlaznog porta, a izlaz multipleksera bi se vodio na port za upis bafera. Naravno, multiplekser bi po TDM principu omogućavao upis paketa sa svakog ulaza po istom principu kao kod komutatora sa zajedničkom memorijom sa slike 9.2.1. Pri tome može da se koristi i FIFO bafer i VOQ bafer, pri čemu bi VOQ bafer obezbedio bolju implementaciju kvaliteta servisa i fer opsluživanja.

Implementacija sa slike 9.4.1 implementira N bafera na izlaznom portu da bi se pokrio i najgori slučaj, a to je situacija kada u istom slotu svi ulazni portovi šalju paket na isti izlazni port.

Međutim, ovakva situacija je retka, pa se u literaturi pojavila ideja da se umesto *N* implementira *L* bafera, gde je *L*<<*N* pri čemu je *L* fiksan broj. Time bi se izbegao problem slabe skalabilnosti jer *L* ima relativno nisku vrednost, pa bi *N*x*L* bafera bilo prihvatljivo sa stanovišta fizičke implementacije. Ovakav komutator se naziva nokaut komutator. Struktura nokaut komutatora je prikazana na slici 9.4.2.



Slika 9.4.2. Nokaut komutator

Nokaut komutator ubacuje koncentrator između adresnih filtara i bafera. Uloga koncentratora je da propusti sve pakete namenjene dotičnom izlaznom portu ako ih je maksimalno *L*, a ako paketa ima više onda propušta tačno *L* paketa, a višak paketa odbacuje. Realizacija koncentratora tipično koristi nokaut princip selekcije pa otuda i naziv komutatora - nokaut komutator. Nokaut princip selekcije se zasniva na izvršavanju *L* kup takmičenja (ekvivalent sportskog kup takmičenja). Svako kup takmičenje se sastoji iz odgovarajućeg broja rundi, a runda se sastoji iz duela paketa koji učestvuju u kup takmičenju (u jednoj rundi paket ima jedan duel, i ishod duela za paket je ili pobednik ili gubitnik). Pobednici duela iz jedne runde, prelaze u sledeću rundu dotičnog kup takmičenja, dok gubitnici prelaze u sledeće kup takmičenje. Konačni pobednik jednog kup takmičenja predstavlja paket koji će se proslediti na izlaz koncentratora. Gubitnici iz runde *i* kup takmičenja prelaze u rundu *i* sledećeg kup takmičenja (sem ako nema više kup takmičenja na raspolaganju tj. u pitanju je poslednje *L*-to kup takmičenje) gde se nadmeću međusobno i sa pobednicima iz prethodne runde tog sledećeg kup takmičenja (ako je prethodna runda postojala). Dueli se obavljaju implementacijom 2x2 svič elementa, pri čemu se na ulaze sviča dovode duelisti, a izlazi su definisani kao pobednik i gubitnik. Ako postoji samo jedan duelist (paket) na ulazu, on je automatski pobednik i prosleđuje se na izlaz pobednik. Ako postoje dva duelista, onda se prosleđuje jedan na izlaz pobednik, a drugi na izlaz gubitnik. Prosleđivanje može biti fiksno, na primer, desni ulaz je uvek gubitnik, ili alternativno (u svakom narednom duelu se alternira koji ulaz će biti pobednik).

Na izlasku iz koncentratora, paketi se ravnomerno upisuju po FIFO baferima da bi se postigla njihova ravnomerna popunjenost i to se postiže upotrebom pomeračkog registra koji se koristi za kontrolu upisa (koristi se kao pokazivač na prvi sledeći bafer u koji treba upisati paket, pri čemu se pokazivač kružno pomera, tj. posle poslednjeg bafera prelazi se opet na prvi). FIFO

baferi kao i u slučaju sa slike 9.4.1 moraju da podrže maksimalno jedan upis i jedno čitanje tokom slota. VOQ baferi nemaju smisla, sem ako se ne izvrši fiksno mapiranje tokova na svaki od bafera, što bi moglo dovesti i do neoptimalnog korišćenja bafera, ali i do potrebe za bržim baferima (eventualno je izvodljivo ulančavanje tako da bude moguće da članovi iste liste budu u različitim baferima, ali to bi značajno komplikovalo implementaciju i trošilo značajno više hardverskih resursa). Očigledno, potencijalna disperzija paketa istog toka po baferima dovodi do otežane implementacije mehanizama za kvalitet servisa i fer opsluživanje.

9.5. Banyan komutatori

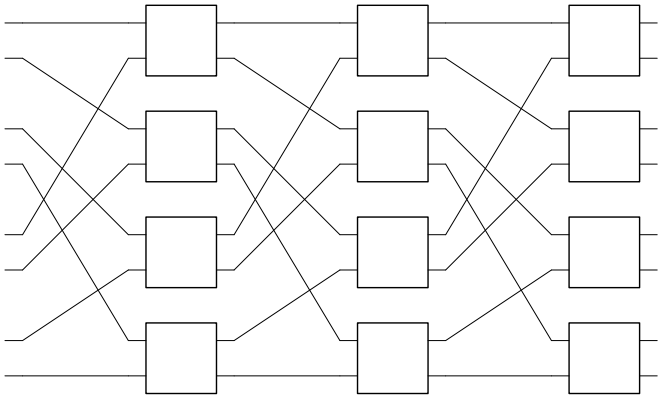
Krosbar komutatori su imali dobre osobine poput jednostavnosti i neblokiranja, ali su bili problematični za realizaciju velikih komutacionih polja zbog potrebe za velikim brojem prekidača (N^2, gde je N broj ulaznih, odnosno izlaznih portova). Otuda je nastala potreba za optimalnijim strukturama koje bi omogućavale efikasniju konstrukciju velikih komutatora (sa velikim brojem ulaza i izlaza). Kao što se vidi identičan problem, kao i kod analognih komutacionih polja, se rešavao i kod paketskih komutatora, pa su otuda usvojene i višekaskadne strukture koje su se razvile kod analognih komutacionih polja poput Banyan i Klosovih struktura. Banyan strukture nismo opisivali kod analognih komutacionih polja jer su se češće koristile Klosove strukture, dok su Banyan strukture postale popularne u ATM mrežama, odnosno koristile su se za realizaciju paketskih komutatora u ATM mrežnim čvorovima.

Pod Banyan strukturom se podrazumeva mreža kroz koju postoji tačno jedan put između bilo kog ulaza i bilo kog izlaza (potpuna dostupnost), pri čemu je mreža sačinjena od manjih svičeva (komutatora - koristićemo termin svič u nastavku ovog poglavlja za označavanje gradivne jedinice Banyan komutatora). Pri tome, u slučaju regularne Banyan mreže (strukture) koriste se identični svičevi, a u slučaju neregularne Banyan mreže koristi se više različitih svičeva. U nastavku poglavlja ćemo razmatrati regularne Banyan strukture, pošto su one češće i u praksi, a isto tako su češće razmatrane u literaturi (u nastavku teksta ćemo pod Banyan strukturom podrazumevati regularnu Banyan strukturu).

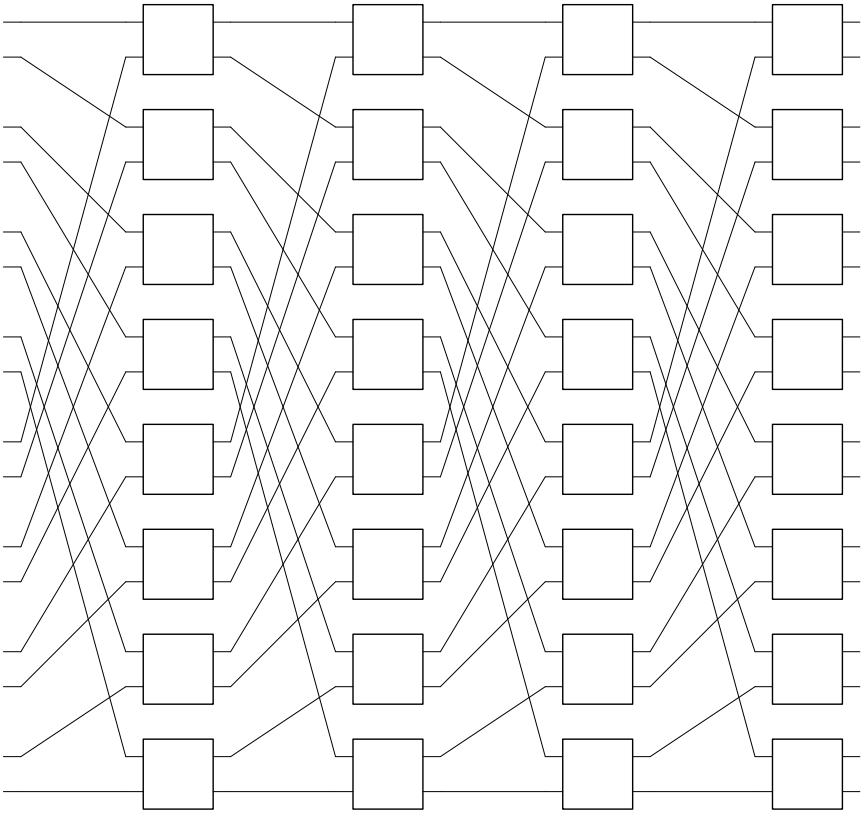
Banyan strukture poseduju sledeće osobine:

- Postoji tačno jedan put od bilo kog ulaza do bilo kog izlaza.
- Ukupan broj kaskada je jednak $N_k = \log_b N$, gde je N broj ulaznih, odnosno izlaznih portova, a b predstavlja broj ulaza, odnosno izlaza jednog sviča (svič je bxb). Broj svičeva u jednoj kaskadi je jednak $N_s = N/b$. U praksi se tipično koriste 2x2 svičevi pa je broj kaskada $\log_2 N$, a broj svičeva u kaskadi $N/2$. Takođe, u praksi se u slučaju većih svičeva tipično koriste vrednosti za b koje su stepena dvojke (4, 8, ...).
- Banyan strukture poseduju osobinu samorutiranja. Dodavanjem binarne vrednosti (u slučaju 2x2 svičeva, za bxb svičeve se dodaje vrednost iz brojnog sistema čija je osnova b) adrese izlaznog porta ispred paketa je moguće interno samorutiranje paketa kroz Banyan strukturu tako što se u sviču ispituje odgovarajuća cifra dodate vrednosti. Naravno, prilikom same implementacije je moguće kreirati varijantu u kojoj se konfigurišu svičevi spolja bez korišćenja samorutiranja, ali se u praksi, ipak, tipično koristi osobina samorutiranja jer je time pojednostavljena kompletna implementacija komutatora. Napomenimo da nemaju sve regularne

Banyan strukture osobinu samorutiranja, ali se u praksi uglavnom one koriste (podskup regularnih Banyan struktura sa osobinom samorutiranja se označava terminom delta mreže).



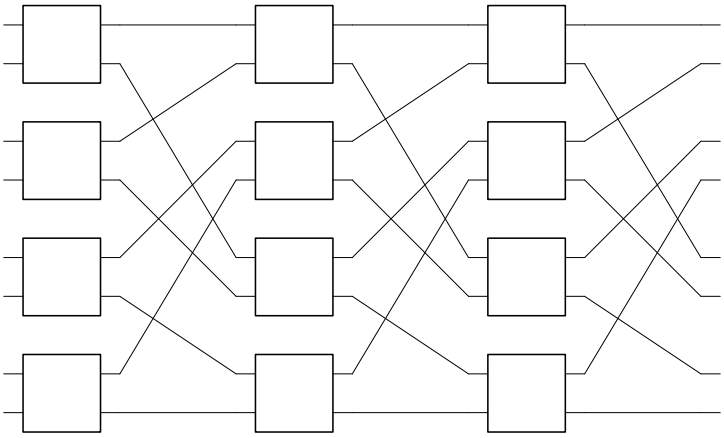
Slika 9.5.1. Omega 8x8 komutator



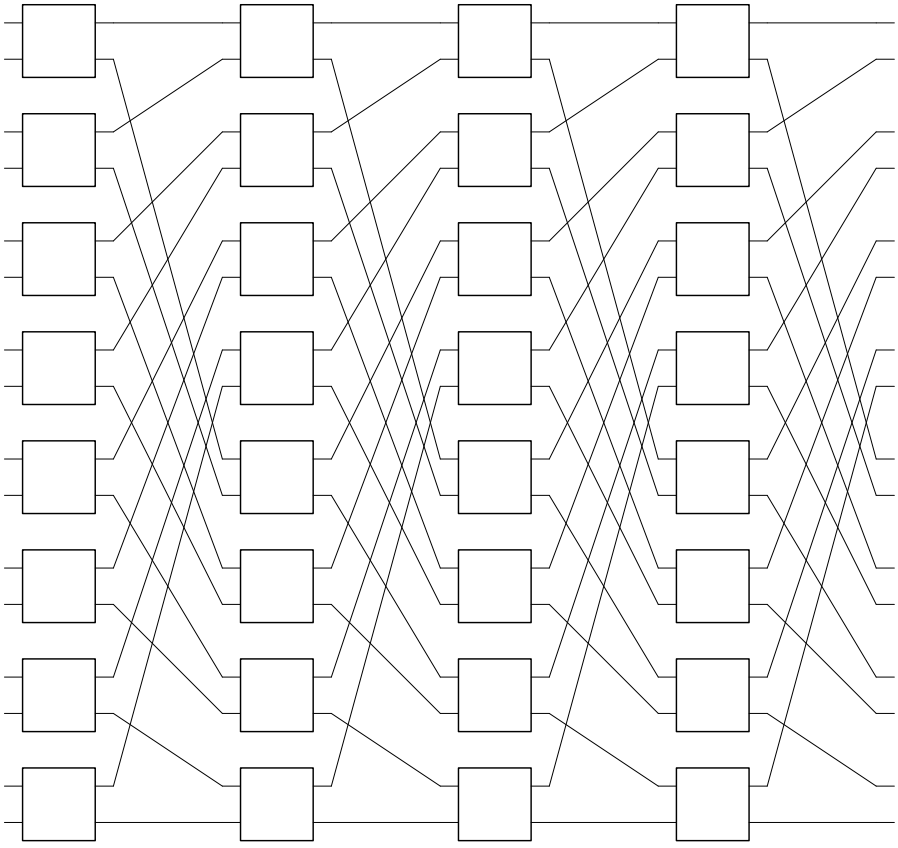
Slika 9.5.2. Omega 16x16 komutator

Razmotrimo sada najpoznatije Banyan strukture. Napomenimo da za svaku Banyan strukturu postoji i njena inverzna struktura (koja je takođe Banyan struktura) koja odgovara odrazu u ogledalu dotične Banyan strukture. To je i logično ako se uzme u obzir da postoji samo jedan put između bilo kog ulaza i izlaza, pa ako obrnemo uloge ulaza i izlaza (ulazi postanu izlazi i obrnuto), put će i dalje biti isti samo će smer prolaska kroz put da se obrne. Omega (*shuffle exchange*) struktura za 8x8 i 16x16 veličine komutatora je prikazana na slikama 9.5.1 i 9.5.2. Veze između susednih kaskada su identične i formiraju se na sledeći način. Izlazi prvog

sviča se redom vezuju za prve ulaze prvih *b* svičeva iz sledeće kaskade. Izlazi drugog sviča se redom vezuju za prve ulaze sledećih *b* svičeva iz sledeće kaskade. Ovaj proces se ponavlja dok se ne povežu svi prvi ulazi svičeva sledeće kaskade. Potom se proces nastavlja od prvog sviča tekuće kaskade čiji izlazi još nisu povezani. Njegovi izlazi se redom povezuju na druge ulaze prvih *b* svičeva sledeće kaskade. Pa sledeći svič tekuće kaskade redom povezuje svoje izlaze sa drugim ulazima sledećih *b* svičeva, itd. Opisani postupak se ponavlja sve dok se ne povežu svi izlazi svičeva tekuće kaskade, odnosno svi ulazi svičeva sledeće kaskade. Treba primetiti da se ulazni portovi na identičan način vezuju na ulaze svičeva prve kaskade.

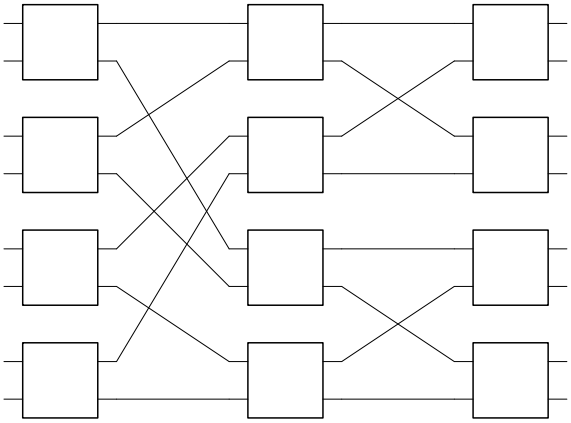


Slika 9.5.3. ***Reverse shuffle exchange*** **8x8 komutator**

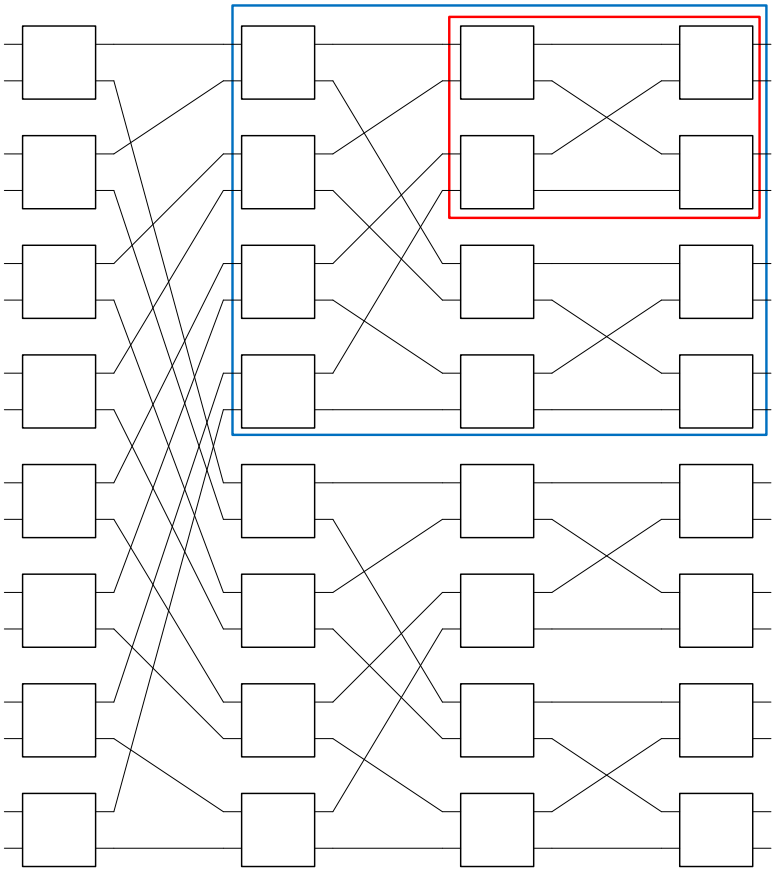


Slika 9.5.4. ***Reverse shuffle exchange*** **16x16 komutator**

Struktura inverzna omega strukturi se naziva *reverse shuffle exchange* struktura. Ova struktura predstavlja odraz u ogledalu omega strukture. Na slikama 9.5.3 i 9.5.4 su prikazane *reverse shuffle exchange* strukture za 8x8 i 16x16 komutatore. Jedan način kreiranja ove strukture je da se kreira omega struktura pa napravi njen odraz u ogledalu, a drugi način kreiranja je upotreba principa kreiranja omega strukture samo u suprotnom smeru od izlaza ka ulazu. Princip povezivanja na poslednje kaskade sa izlaznim portovima je identičan povezivanju susednih kaskada.

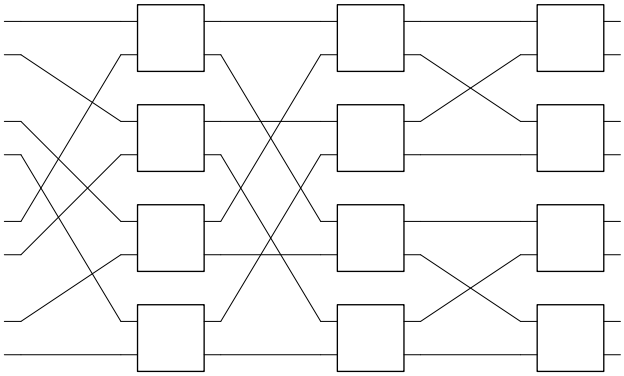


Slika 9.5.5. *Baseline* 8x8 komutator

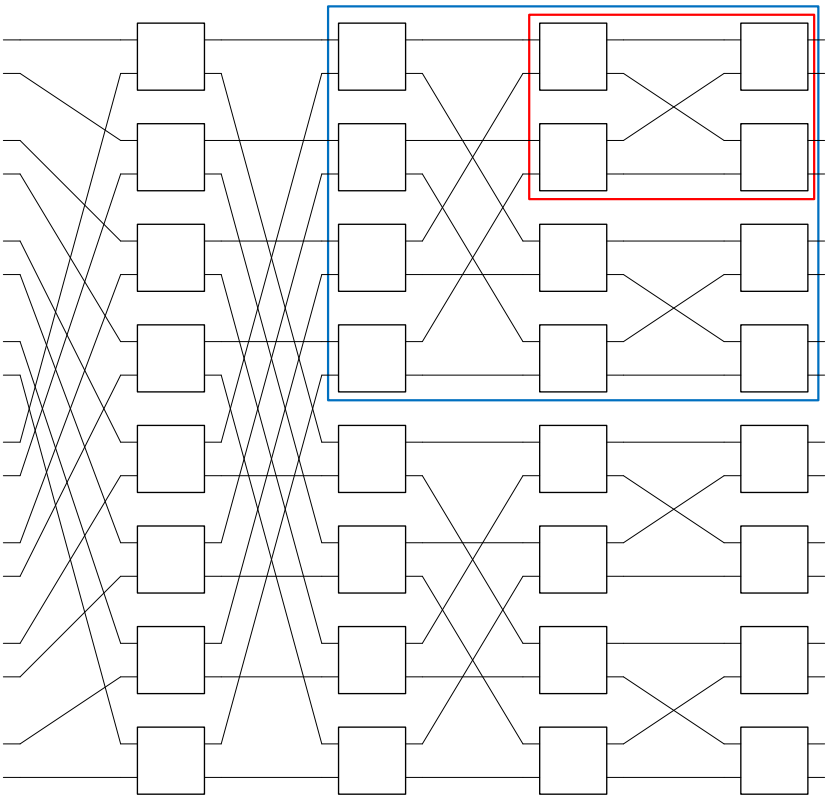


Slika 9.5.6. *Baseline* 16x16 komutator

Baseline struktura je prikazana na slikama 9.5.5 i 9.5.6 za 8x8 i 16x16 komutatore. *Baseline* struktura se kreira primenom rekurzije. Na slici 9.5.6 su uokvirene *baseline* strukture za 4x4 i 8x8 dimenzije. Struktura $b{\cdot}n$x$b{\cdot}n$ se kreira upotrebom nxn strukture na sledeći način. Postavlja se b nxn struktura jedna iznad druge. Prva kaskada se povezuje na te nxn strukture na sledeći način. Prvih b svičeva se redom povezuju na prvih b ulaza svake od nxn struktura (prvi izlaz prvog sviča na prvi ulaz prve nxn strukture, prvi izlaz drugog sviča na drugi ulaz prve nxn strukture,..., drugi izlaz prvog sviča na prvi ulaz druge nxn strukture, drugi izlaz drugog sviča na drugi ulaz druge nxn strukture, ...). Sledećih b svičeva prve kaskade se na isti način povezuju na sledećih b ulaza svake od nxn struktura. Ovaj princip se ponavlja dok se ne izvrši kompletno povezivanje izlaza prve kaskade sa ulazima iz svih nxn struktura.

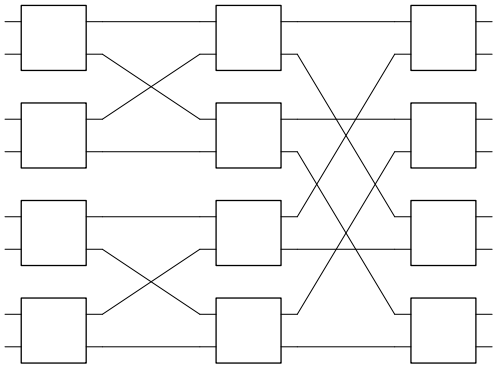


Slika 9.5.7. 3-*cube* (8x8) komutator

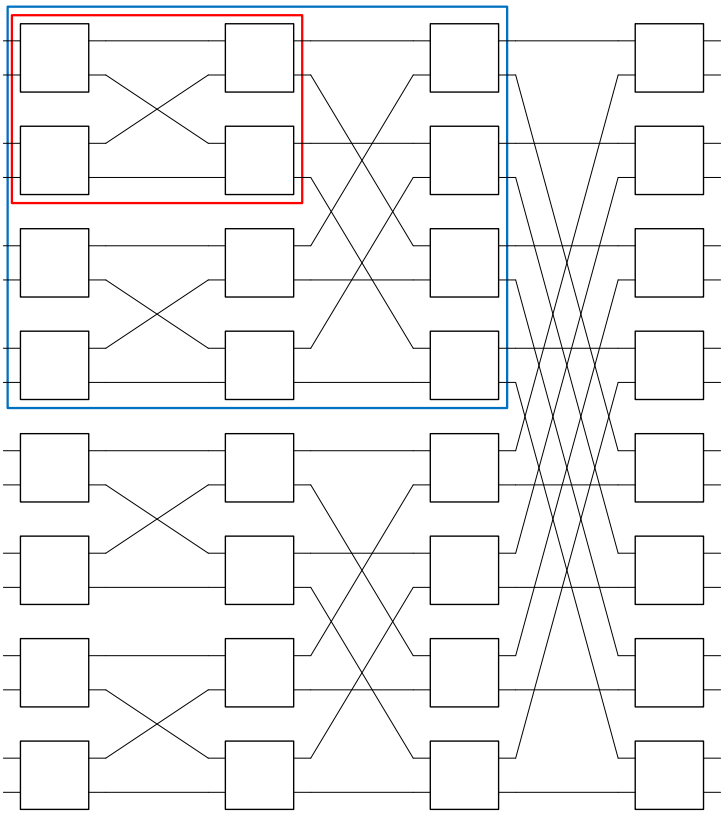


Slika 9.5.8. 4-*cube* (16x16) komutator

Struktura *n-cube* je prikazana na slikama 9.5.7 i 9.5.8 za 8x8 i 16x16 komutatore, pri čemu *n* označava broj kaskada. Struktura *n-cube* se kreira primenom rekurzije. Na slici 9.5.8 su uokvirene *n-cube* strukture za 4x4 i 8x8 dimenzije. Struktura $b \cdot n \times b \cdot n$ se kreira upotrebom $n \times n$ strukture na sledeći način. Postavlja se b $n \times n$ struktura jedna iznad druge. Prva kaskada se povezuje na te $n \times n$ strukture na sledeći način. Izlazi prvog sviča prve kaskade se redom povezuju na prve ulaze prvih svičeva svih $n \times n$ struktura. Izlazi drugog sviča prve kaskade se redom povezuju na prve ulaze drugih po redu svičeva svih $n \times n$ struktura. Proces se ponavlja dok se ne povežu svi prvi ulazi svičeva prve kaskade $n \times n$ strukture. Postupak se ponavlja sa preostalim svičevima prve kaskade, samo se sada oni povezuju na druge (pa treće i tako redom) ulaze svičeva prve kaskade svih $n \times n$ struktura. Sami ulazni portovi se povezuju na ulaze prve kaskade *n-cube* strukture na identičan način kao kod omega strukture.

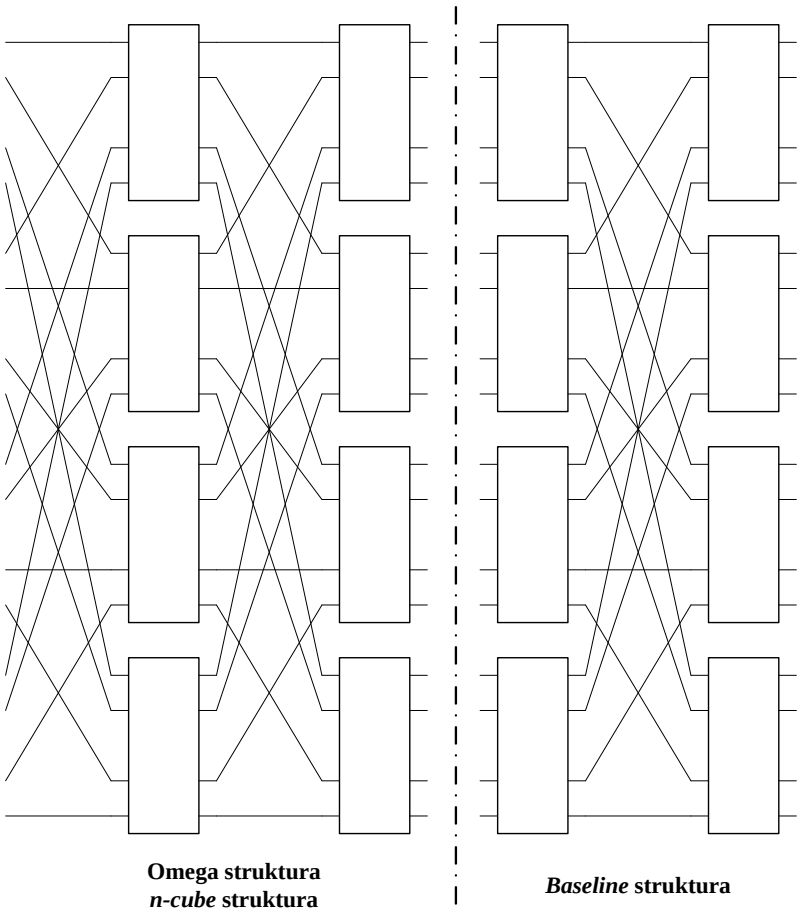


Slika 9.5.9. SW-Banyan 8x8 komutator



Slika 9.5.10. SW-Banyan 16x16 komutator

SW-Banyan struktura je veoma slična *n-cube* strukturi i dobija se kao njen odraz u ogledalu uz izuzetak da nema mešanog povezivanja izlaza sa izlaznim portovima, kao što su ulazni portovi bili povezani sa ulazima *n-cube* strukture. Otuda je princip povezivanja kaskada identičan onome iz *n-cube* strukture, ali u suprotnom smeru. SW-Banyan struktura je prikazana na slikama 9.5.9 i 9.5.10 za 8x8 i 16x16 komutatore.

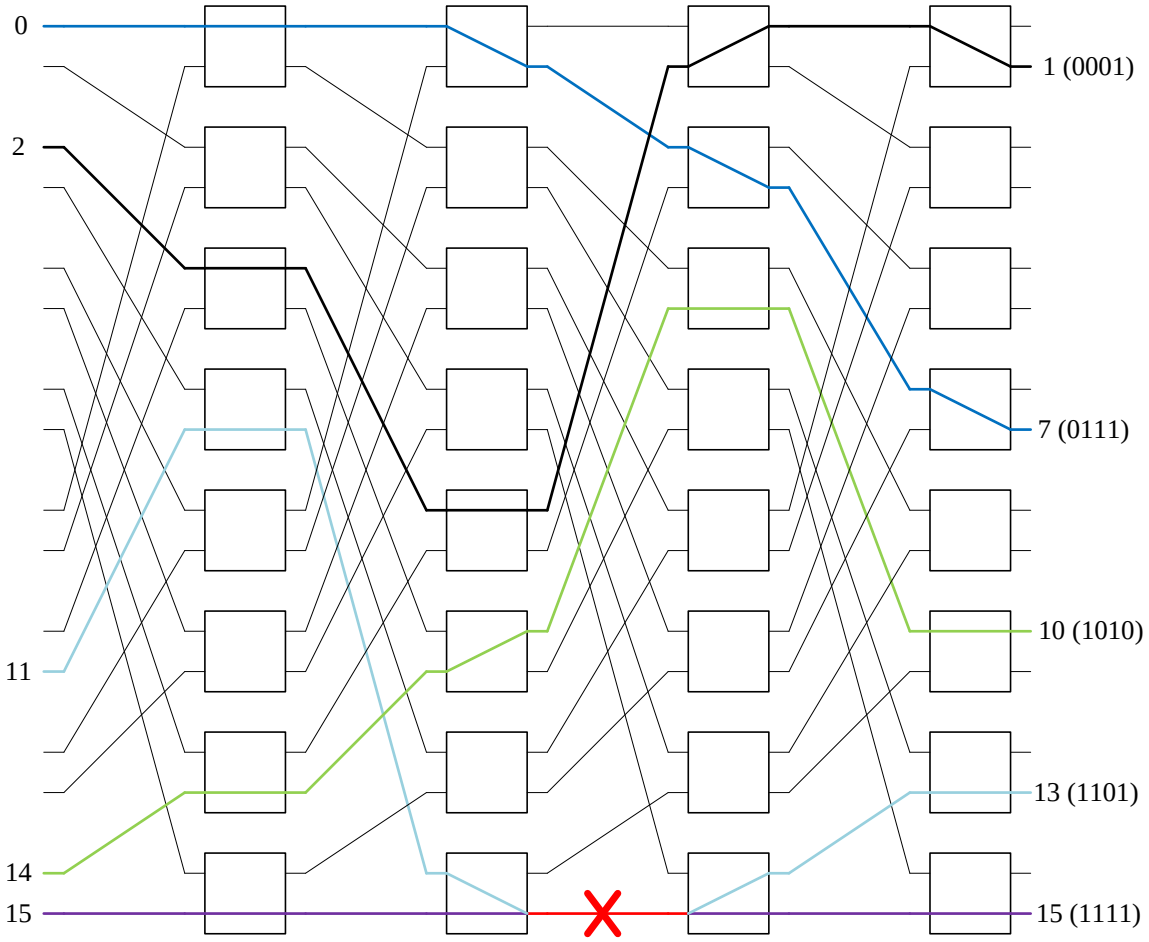


Slika 9.5.11. Omega, *baseline*, *n-cube* 16x16 komutatori primenom 4x4 svičeva

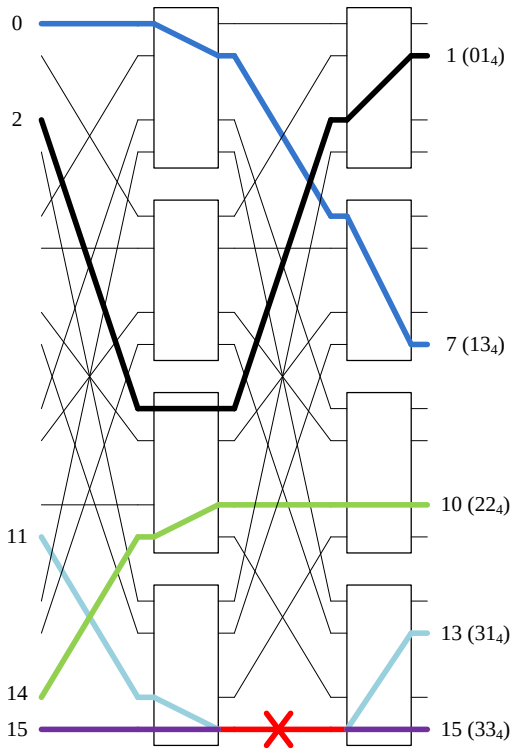
Kao što smo rekli Banyan strukture se mogu kreirati i većim svičevima od 2x2 svičeva, ali 2x2 svičevi se, ipak, najčešće koriste za kreiranje Banyan struktura. Na slici 9.5.11 su prikazane omega, *baseline* i *n-cube* strukture za 16x16 komutator kreirane upotrebom 4x4 svičeva (tipično se za dimenzije svičeva koriste vrednosti koje odgovaraju stepenu dvojke, ali teoretski mogu da se koriste proizvoljne celobrojne vrednosti, poput 3x3 sviča). Može se uočiti da su omega i *n-cube* strukture identične za 16x16 komutator, kada se koriste 4x4 svičevi.

Prikažimo sada princip samorutiranja kroz Banyan strukturu. Na slici 9.5.12 je prikazan primer ruta za pojedine parove ulaz/izlaz kroz omega 16x16 komutator. Princip samorutiranja u slučaju 2x2 svičeva uzima bit po bit odredišne adrese prilikom prolaska kroz svičeve da bi odredio na koji izlaz 2x2 sviča treba usmeriti paket. Ako je vrednost bita 0 ide se na izlaz 0 (gornji izlaz) 2x2 sviča, a ako je vrednost bita 1 ide se na izlaz 1 (donji izlaz) 2x2 sviča. Na primer, put označen tamno-plavom bojom spaja ulaz 0 sa izlazom 7. Odredišna adresa izlaza 7 u binarnom formatu je 0111, što znači da će odgovarajući svič prve kaskade da usmeri paket na

svoj gornji izlaz, a odgovarajući svičevi sledeće tri kaskade će da usmere paket na svoj donje izlaze. Ako se pogleda slika 9.5.12 može se videti da se navedeni princip koristi u svim prikazanim uparenim parovima ulaz/izlaz. Očigledno, da bi samorutiranje moglo da se izvrši, neophodno je da se paketu doda zaglavlje sa odredišnom adresom izlaznog porta. Odredišna adresa izlaznog porta u stvari predstavlja identifikaciju (redni broj) izlaznog porta. Princip samorutiranja je prikazan i na slici 9.5.13 za omega 16x16 komutator kada se koriste 4x4 svičevi. Sada se adresa predstavlja u brojnom sistemu sa osnovom 4 (implementaciono bi se i dalje adresa predstavljala binarno, ali bi se uzimalo više bita odjednom za određivanje na koji izlaz sviča treba usmeriti paket). Princip samorutiranja je identičan kao kod 2x2 svičeva, samo se sada izlaz određuje na osnovu broja iz sistema sa osnovom 4 (vrednosti cifara 0, 1, 2 i 3). Na primer, u slučaju prosleđivanja sa ulaza 0 na izlaz 7 (13_4), odgovarajući svič prve kaskade će proslediti paket na izlaz 1 (drugi izlaz po redu jer numeracija izlaza ide od 0 do 3), a odgovarajući svič druge kaskade će proslediti paket na izlaz 3. Napomenimo još jednom da implementacija može da definiše da se konfigurisanje komutatora, a time i rada svičeva u komutatoru vrši spolja, ali to se u praksi uglavnom ne radi jer se time komplikuje implementacija i troši više hardverskih resursa.



Slika 9.5.12. Princip samorutiranja



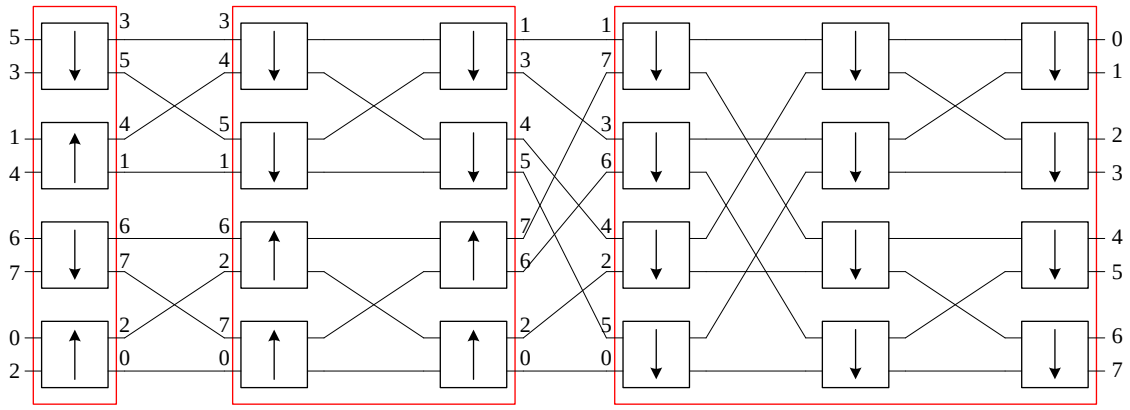
Slika 9.5.13. Princip samorutiranja za 4x4 svičeve

Napomenimo da se redosled upotrebe bita odredišne adrese prilikom prolaska kroz svičeve u procesu samorutiranja može razlikovati za različite Banyan strukture. Na primer, omega, *baseline* i *n-cube* strukture koriste princip opisan za omega komutator - biti se uzimaju redom kako su i navedeni u odredišnoj adresi (od najvišeg do najnižeg bita). U slučaju *reverse-shuffle* strukture biti se uzimaju obrnutim redosledom, od najnižeg bita (u prvoj kaskadi) do najvišeg bita (u poslednjoj kaskadi) odredišne adrese. U slučaju SW-Banyan strukture biti se uzimaju takođe obrnutim redosledom, pri čemu se najniži bit ne uzima u obzir tj. on ostaje na svom (najnižem) mestu (od adrese $a_{n-1}a_{n-2}... a_2a_1a_0$ se kreira informacija za samorutiranje u obliku $a_1a_2... a_{n-2}a_{n-1}a_0$).

U oba data primera samorutiranja prikazana na slikama 9.5.12 i 9.5.13, dolazi do kolizije označene crvenom bojom. Naime, Banyan komutatori su interno blokirajući komutatori, što znači da za pojedine parove ulaz/izlaz nije moguće istovremeno proslediti pakete usled unutrašnje blokade. U datim primerima u pitanju su parovi (11,13) i (15,15). U slučaju kolizije, samo jedan paket se propušta što znači da će drugi paket biti izgubljen, što bi trebalo izbeći. Jedan način je da se koristi ubrzanje u komutatoru pa da bude moguće proslediti u različitim iteracijama u okviru istog slota pakete koji su u koliziji, što nije praktično rešenje jer se zahteva i određivanje koji paketi će biti u koliziji što može biti nepraktično za proračun u slučaju velikog broja ulaza/izlaza. Drugi način je da se implementira povratna sprega kojom bi izlazni portovi obaveštavali ulazne portove o uspešnom pristizanju njihovih poslatih paketa. Ulazni portovi bi čuvali poslate pakete u baferima pa u slučaju potvrde o uspešnom slanju bi ih brisali iz bafera, u suprotnom bi ih ponovo slali. Ovaj način je isto pomalo nepraktičan za slučaj velikog broja portova jer bi bilo komplikovano ekonomično implementirati efikasnu povratnu spregu. Međutim, primećeno je jedno veoma važno svojstvo omega i *n-cube* struktura, a to je da su one interno neblokirajuće ako su zadovoljena sledeća dva uslova:

- Odredišne adrese paketa (odredišta paketa) su sortirane u opadajućem ili rastućem redosledu na ulazima.
- Nema neaktivnog ulaza između aktivnih ulaza. Pod aktivnim ulazom se podrazumeva ulaz koji ima paket za prosleđivanje kroz komutator, a u suprotnom se ulaz smatra neaktivnim.

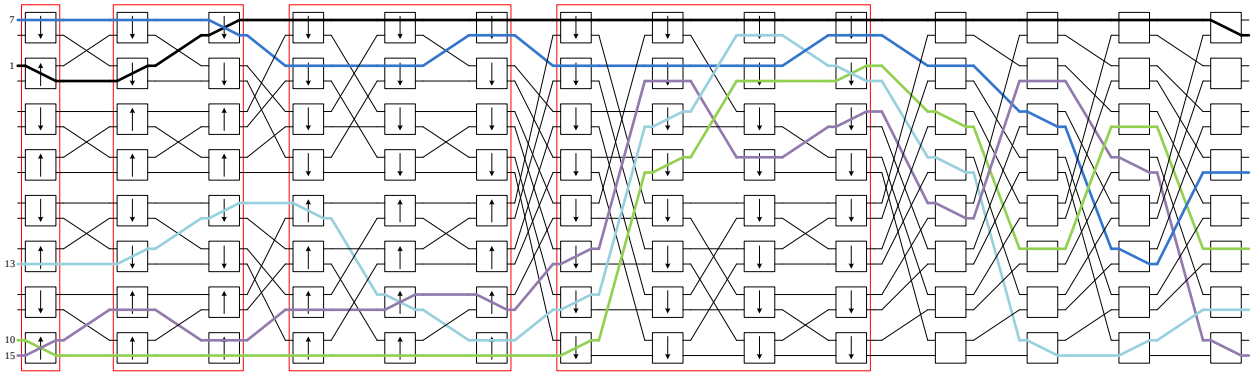
Očigledno bi bilo onda poželjno izvršiti sortiranje paketa po njihovim odredištima čime bi se bez bojazni od kolizije mogle koristiti navedene dve Banyan strukture. Otuda se uz te strukture tipično koriste i mreže (strukture) za sortiranje koje dovode pakete na ulaze omega ili *n-cube* strukture u sortiranom redosledu (bez neaktivnih ulaza (rupa) između aktivnih ulaza). Pre nego što objasnimo strukture za sortiranje, navedimo još par važnih napomena vezanih za upotrebu bafera uz (sve) Banyan strukture. Pošto više paketa istovremeno (u istom slotu) može biti namenjeno istom izlaznom portu potrebno je baferisati pakete. Baferi se mogu stavljati i na izlaznim portovima (ako su samo na izlaznim portovima tada je neophodno ubrzanje komutatora) i na ulaznim portovima, ali i u samim Banyan komutatorima radi čuvanja paketa koji su izgubili duel usled kolizije. Sve napomene date ranije u poglavlju vezane za pozicije bafera u mrežnom čvoru generalno važe i u ovom slučaju kada se kao komutator koristi Banyan struktura.



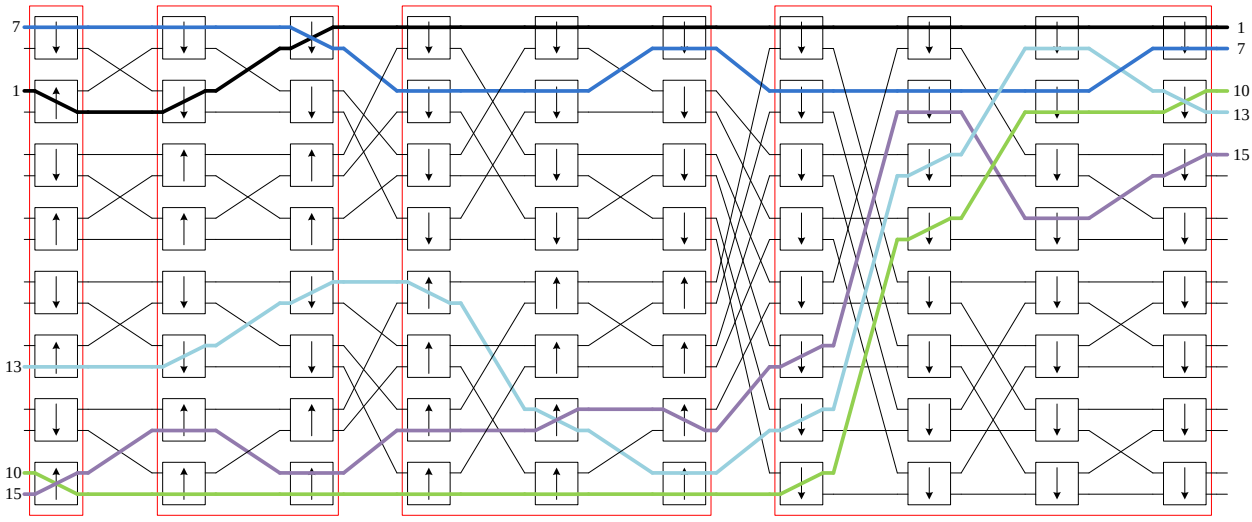
Slika 9.5.14. Batcher 8x8 struktura za sortiranje

Postoji više struktura za sortiranje. J edna od poznatijih struktura je Batcher struktura koja se često koristi u kombinaciji sa Banyan strukturama, pa se često koristi termin Batcher-Banyan struktura (komutator) čime se označava Banyan struktura (omega ili *n-cube*) ispred koje se nalazi Batcher struktura za sortiranje. Batcher struktura se sastoji iz delova (tzv. *merge* kaskada), pri čemu su svi delovi sastavljeni od 2x2 svičeva. Ukupan broj delova je jednak $\log_2 N$, gde je N broj ulaznih, odnosno izlaznih portova. Prvi deo ima samo jednu kaskadu, drugi deo ima dve kaskade, treći deo tri kaskade i tako redom do poslednjeg dela koji ima $\log_2 N$ kaskada. Svaka kaskada sadrži $N/2$ svičeva. Na slici 9.5.14 je prikazana Batcher struktura 8x8 koja sortira 8 ulaznih portova, tačnije pakete koje oni šalju. Kao što vidimo Batcher 8x8 struktura se sastoji iz tri dela, čiji se broj kaskada uvećava za jedan sa rednim brojem dela (prvi deo - jedna kaskada, drugi deo - dve kaskade i treći deo - tri kaskade). Ideja Batcher strukture je jednostavna. Prvi deo ima za zadatak da formira $N/2$ sortiranih listi od 2 člana (na osnovu $N/2$ parova sortiranih listi od jednog člana), drugi deo formira $N/4$ listi od 4 člana (na osnovu $N/4$ parova sortiranih listi od dva člana) i tako redom do poslednjeg dela koji formira jednu sortiranu listu od N članova (na osnovu 1 para sortiranih listi od $N/2$ članova). Ako pogledamo sliku 9.5.14, videćemo da upravo tako funkcioniše Batcher struktura. Smer strelice u sviču ukazuje na koji izlaz sviča se usmerava paket sa većom vrednošću odredišne adrese. Ukoliko je na ulazu prisutan samo jedan paket, tada

se on usmerava na izlaz koji odgovara manjoj vrednosti odredišne adrese. Prvi deo formira 4 sortirane liste od po dva člana, pri čemu smer strelice u dotičnom redu dela određuje tip sortiranja - opadajući (strelica nagore) ili rastući (strelica nadole) redosled. Drugi deo formira 2 sortirane liste od po četiri člana, pri čemu smer strelice u dotičnom redu dela određuje tip sortiranja na identičan način kao kod prvog dela. Poslednji (treći) deo formira finalnu sortiranu listu paketa po rastućem redosledu koja se prosleđuje na ulaze Banyan strukture. Struktura samih delova je jednaka odgovarajućoj *n-cube* strukturi (i ulazi se vezuju na identičan način) čiji su primeri prikazani na slikama 9.5.7 i 9.5.8. Tako se treći deo sastoji od jedne 3-*cube* strukture, drugi deo od dve 2-*cube* strukture, a prvi deo od četiri 1-cube strukture (koje odgovaraju samim svičevima). Pri tome neparne *n-cube* strukture u delovima imaju svičeve kod kojih su strelice orijentisane nadole, a parne *n-cube* strukture u delovima imaju svičeve kod kojih su strelice orijentisane nagore (pretpostavka je da se *n-cube* strukture u delu broje od 1, pri čemu se broje u smeru nadole). Engleski termin *merge network* se koristi za označavanje mreža koje vrše spajanje dveju sortiranih listi sa ulaza u *merge* mrežu u jednu sortiranu listu. Kao što smo naveli Batcher struktura se sastoji od *merge* delova koji sadrže odgovarajući broj *merge* mreža - vidimo da se broj listi dvostruko smanjuje na izlazu iz svakog od delova, i takođe broj *merge* mreža koje rade u paraleli u jednom delu zavisi od broja parova listi koje se spajaju.



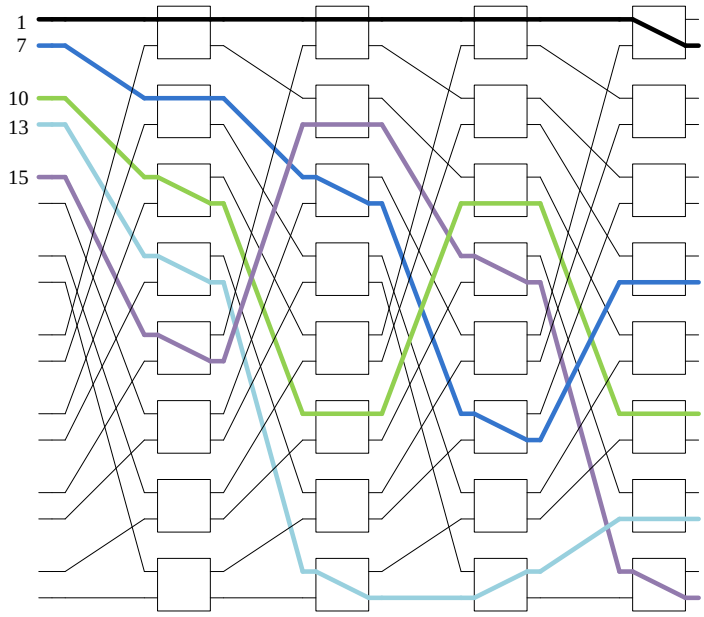
Slika 9.5.15. Primer Batcher-Banyan 16x16 strukture



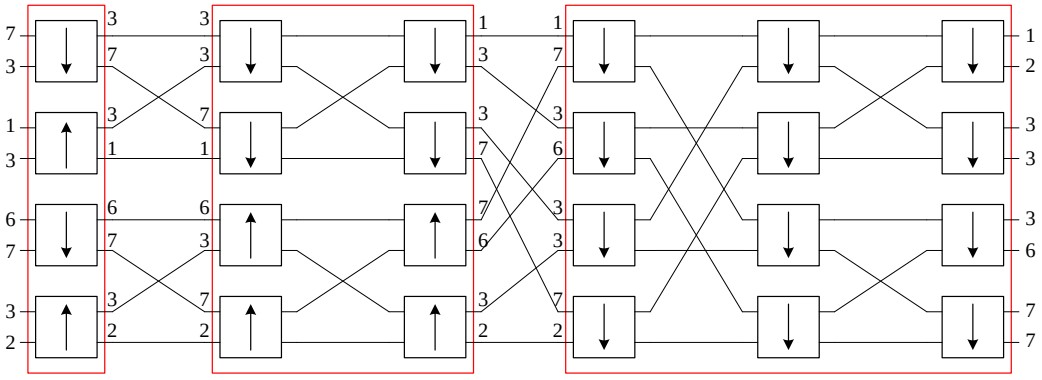
Slika 9.5.16. Batcher struktura iz primera

Prikažimo sada Batcher-Banyan strukturu za 16x16 komutator sa parovima ulaz-izlaz kao sa slike 9.5.12 da proverimo da li je izbegnuta pojava interne blokade. Na slici 9.5.15 je prikazana kompletna Batcher-Banyan struktura i može se videti da u ovom slučaju nema interne

blokade, upravo zbog izvršenog sortiranja u okviru Batcher strukture. Na slikama 9.5.16 i 9.5.17 su prikazane zasebno Batcher i Banyan (omega) struktura iz primera sa slike 9.5.15 radi bolje preglednosti.



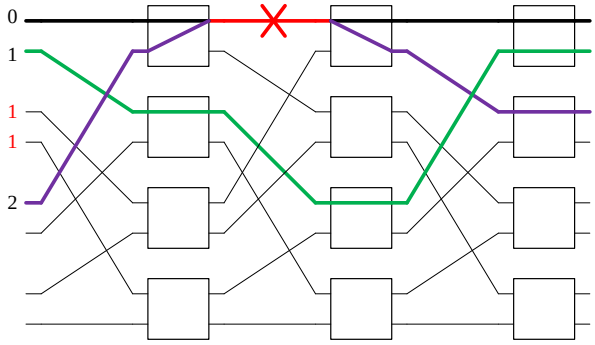
Slika 9.5.17. Banyan struktura iz primera



Slika 9.5.18. Primer Batcher strukture kada ima više paketa namenjenih istom izlaznom portu

U slučaju da nema spoljašnje kontrole (raspoređivača) koja bi nadgledala da ulazni portovi ne šalju istovremeno pakete na isti izlazni port, može se desiti situacija da na ulazu u Batcher mrežu uđu paketi namenjeni istom izlaznom portu. Kada dva paketa namenjena istom izlaznom portu dođu na ulaze istog sviča u Batcher strukturi, svejedno je kako će ta dva paketa biti prosleđena na izlaze sviča. Nakon sortiranja, paketi namenjeni istom izlaznom portu će biti jedni do drugih. Primer ove situacije je prikazan na slici 9.5.18 za 8x8 Batcher strukturu. Kao što se vidi paketi namenjeni izlaznom portu 3 su sortirani jedni do drugih, a isto važi i za pakete namenjene izlaznom portu 7. Lista je i dalje sortirana u neopadajućem redosledu (nije rastuća jer postoje isti brojevi u nizu). Ovakva situacija može da izazove interno blokiranje jer će da se naruši uslov da ne sme da postoji neaktivan ulaz između aktivnih ulaza. Naime, samo jedan od paketa namenjenih istom izlaznom portu sme da se prosledi što znači da će nastati eventualna rupa između paketa koji se prosleđuju (rupa neće nastati samo ako nema većih elemenata u listi tj. paketa se većom vrednošću odredišne adrese jer će tada neprosleđeni elementi biti na samom

kraju sortirane liste), a rupu će da formiraju neprosleđeni paketi. Primer blokiranja koji se javlja u ovakvoj situaciji je dat na slici 9.5.19 (paketi čije je odredište označeno crvenom bojom se ne prosleđuju jer su namenjeni istom odredištu kao paket sa drugog ulaza). Stoga je veoma važno izbeći ovakvu situaciju i obezbediti da se ne prosleđuje više paketa namenjenih istom portu kada se koristi struktura za sortiranje jer njena primena ne bi imala željenog efekta.



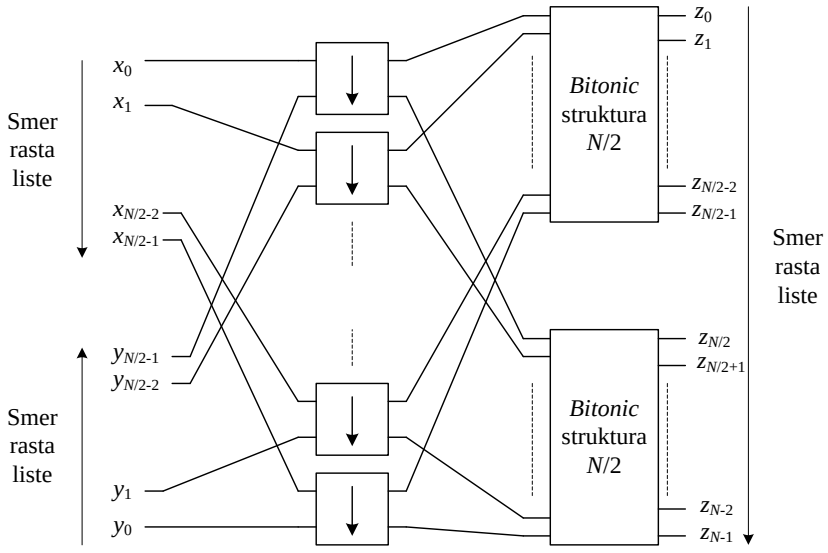
Slika 9.5.19. Primer internog blokiranja usled pojave rupe između sortiranih paketa zbog neprosleđivanja paketa namenjenih istom portu

Da bi se sprečila situacija da se više paketa namenjenih istom portu pošalje istovremeno, predložena su različita rešenja u literaturi. Jedan način je upotreba algoritama raspoređivanja ako se Batcher-Banyan komutator koristi u strukturi sa baferima na ulazu. Druga varijanta predložena u literaturi se sastoji iz tri faze. U prvoj fazi se šalju samo odredišne adrese. Na izlazu Batcher strukture se vrši provera za svaki član sortiranog niza da li je njegov prethodnik namenjen istom izlaznom portu ili ne. Ako se prethodnik razlikuje, dotičnom ulaznom portu se šalje pozitivna potvrda (druga faza). Samo ulazni portovi koji su primili pozitivnu potvrdu će poslati svoje pakete kroz Batcher-Banyan komutator (treća faza), dok će drugi baferisati dotične pakete i pokušati u narednim slotovima. Pošto se u trećoj fazi ne šalju paketi namenjeni istom portu, Batcher struktura za sortiranje će kreirati rastuću listu i samim tim će Banyan struktura moći bez internog blokiranja da komutira sve pakete sa svojih ulaza. Ova varijanta je jednostavna za implementaciju jer koristi resurse koji se ionako koriste za komutaciju paketa, ali je problem što nije efikasna kao algoritmi raspoređivanja, ukoliko se na ulazu koriste VOQ redovi za čekanje (ako su u pitanju FIFO baferi tada je ova varijanta optimalna tj. ne mogu se postići bolji rezultati od nje, ali tada može doći do HOL blokade). Postoje i druge varijante predložene u literaturi za sprečavanje da više paketa bude poslato na isti izlazni port.

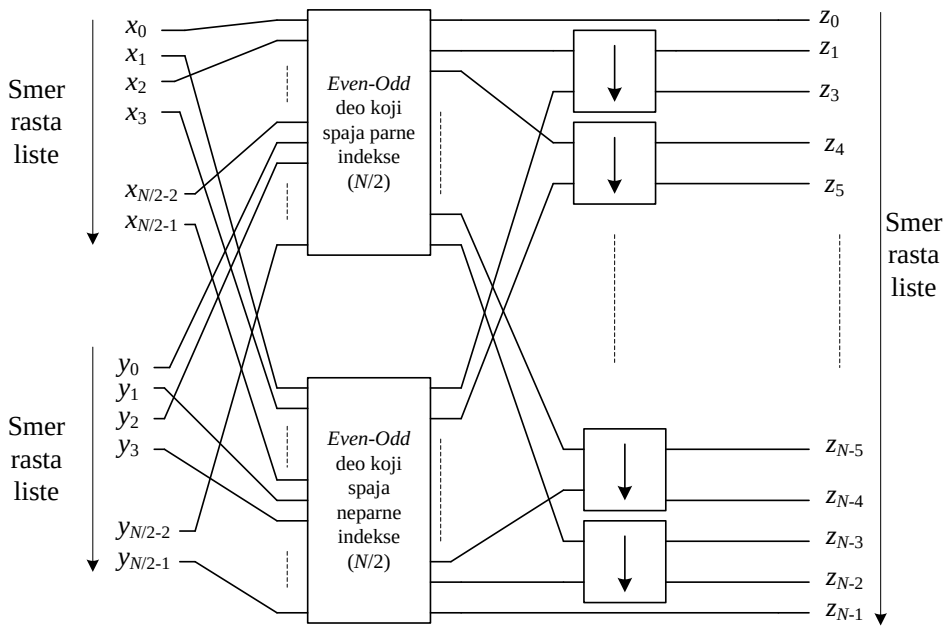
Napomenimo da je Batcher struktura za sortiranje zasnovana na algoritmu za spajanje dve sortirane liste u jednu zajedničku sortiranu listu pod nazivom bitoničko spajanje (*bitonic merge*). Pored ovog algoritma postoji i algoritam par-nepar spajanje (*odd-even merge*). Principske rekurzivne šeme za spajanje dve sortirane liste od $N/2$ članova u jednu sortiranu listu od N članova su prikazane na slikama 9.5.20 i 9.5.21 za bitoničko spajanje i par-nepar spajanje, respektivno.

Ideja bitoničkog spajanja je da poredi članove liste u obrnutom redosledu (najmanji član jedne liste sa najvećim članom druge liste, drugi po redu najmanji član liste sa drugim po redu najvećim članom liste,...) i da one koji su manji šalje u jednu bitoničku strukturu dimenzije $N/2$, a one koji su veći u drugu bitoničku strukturu dimenzije $N/2$. Na ovaj način će biti prosleđeni u gornju bitoničku strukturu dimenzije $N/2$ elementi koji će pripadati nižoj polovini konačnog sortiranog niza, a u donju bitoničku strukturu dimenzije $N/2$ elementi koji će pripadati višoj

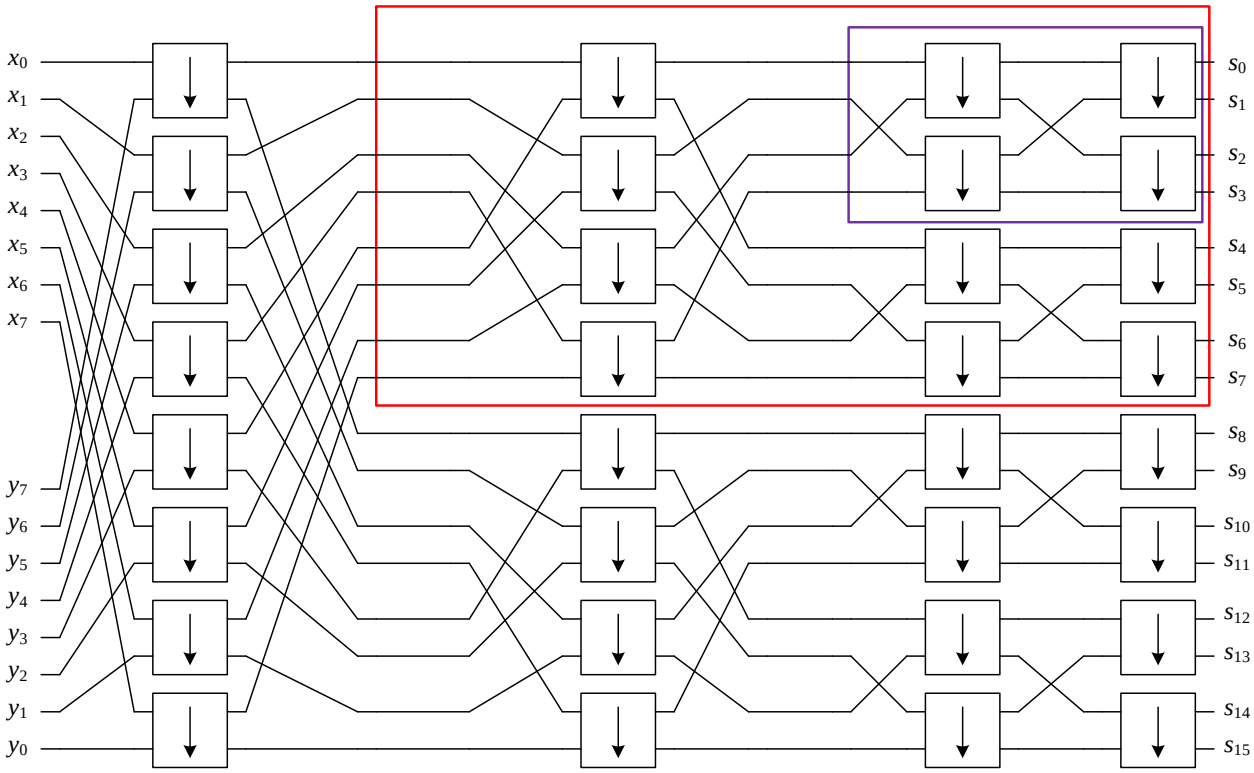
polovini konačnog sortiranog niza. Naime prvo poređenje gde x_i bude veći od odgovarajućeg člana niza y s kojim se poredi, x_i i svi članovi niza x iza njega (sa većim indeksom) će biti prosleđeni u donju bitoničku strukturu jer će oni sigurno biti veći u preostalim poređenjima (y članovi će biti sve nižih vrednosti, a x članovi sve viših vrednosti). Pre te situacije su se prosleđivali članovi y niza na donju bitoničku strukturu. To znači da će sigurno na ulaz donje bitoničke strukture da bude dovedeno $N/2$ najviših vrednosti gledajući oba niza zbirno (član x niza najmanje vrednosti a da je prosleđen na donju bitoničku strukturu ima sigurno veću vrednost od svih članova x niza prosleđenih na gornju strukturu, a isto važi i za y članove koji su prosleđeni na gornju strukturu jer se sa njihovim najvećim predstavnikom upravo poredio član x niza najmanje vrednosti koji je prosleđen na donju bitoničku strukturu; analogan zaključak se može izvesti i za y člana najmanje vrednosti a da je prosleđen na donju bitoničku strukturu). Pošto se bitoničke strukture dimenzije $N/2$ formiraju na analogan način, svaka od bitoničkih struktura dimenzije $N/2$ će sortirati u rastućem redosledu članove sa svojih ulaza i time će se spajanjem ta dva niza sa izlaza bitoničkih struktura dimenzije $N/2$ dobiti kompletno sortiran niz dimenzije N (sortiran u rastućem redosledu). Inače, termin bitonički niz označava niz kod koga postoji indeks k takav da je niz do tog člana sa indeksom k rastući, a posle njega niz je opadajući (na primer, niz 0,1,2,3,4,2,1 je bitonički). Bitonički niz je i onaj niz kod kog je niz prvo opadajući (do člana sa indeksom k) pa rastući. Ulaz u bitoničku strukturu dimenzije N je bitonički (ulaz podrazumeva gledanje svih ulaza kao jednog niza $x_0x_1...x_{N/2-1}y_{N/2-1},..,y_1,y_0$) jer x niz raste i potom će rasti za još jedan član ako je najveći član y niza veći od najvećeg člana x niza i nakon toga će doći do opadanja vrednosti. Ako je najveći član y niza manji od najvećeg člana x niza tada će opadanje krenuti odmah sa prelazom na članove y niza. Da bi bitonička struktura radila korektno neophodno je da njeni ulazi zbirno gledano čine kružni bitonički niz. Kružni bitonički niz je niz dobijen od bitoničkog niza rotiranjem za k mesta ($0 \leq k \leq N-1$). Očigledno, bitonički niz je ujedno i kružni bitonički niz (rotiranje za 0 mesta bitoničkog niza). Rastući i opadajući nizovi su takođe kružni bitonički nizovi. Dokazano je da ulazi u bitoničku strukturu dimenzije $N/2$ takođe čine kružni bitonički niz (ako je ulaz u bitoničku strukturu dimenzije N bitonički), čime je omogućeno rekurzivno formiranje bitoničke strukture po principu prikazanom sa slike 9.5.20.



Slika 9.5.20. Principska rekurzivna šema za bitoničko spajanje



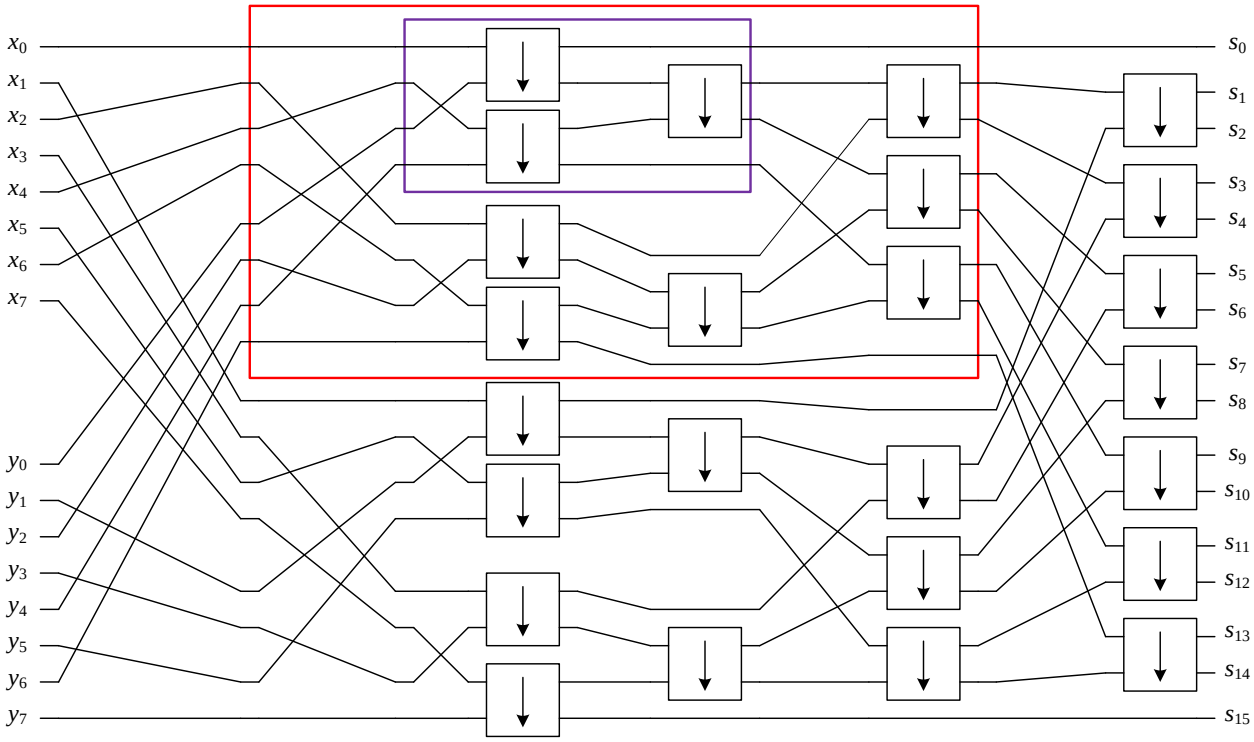
Slika 9.5.21. Principska rekurzivna šema za par-nepar spajanje



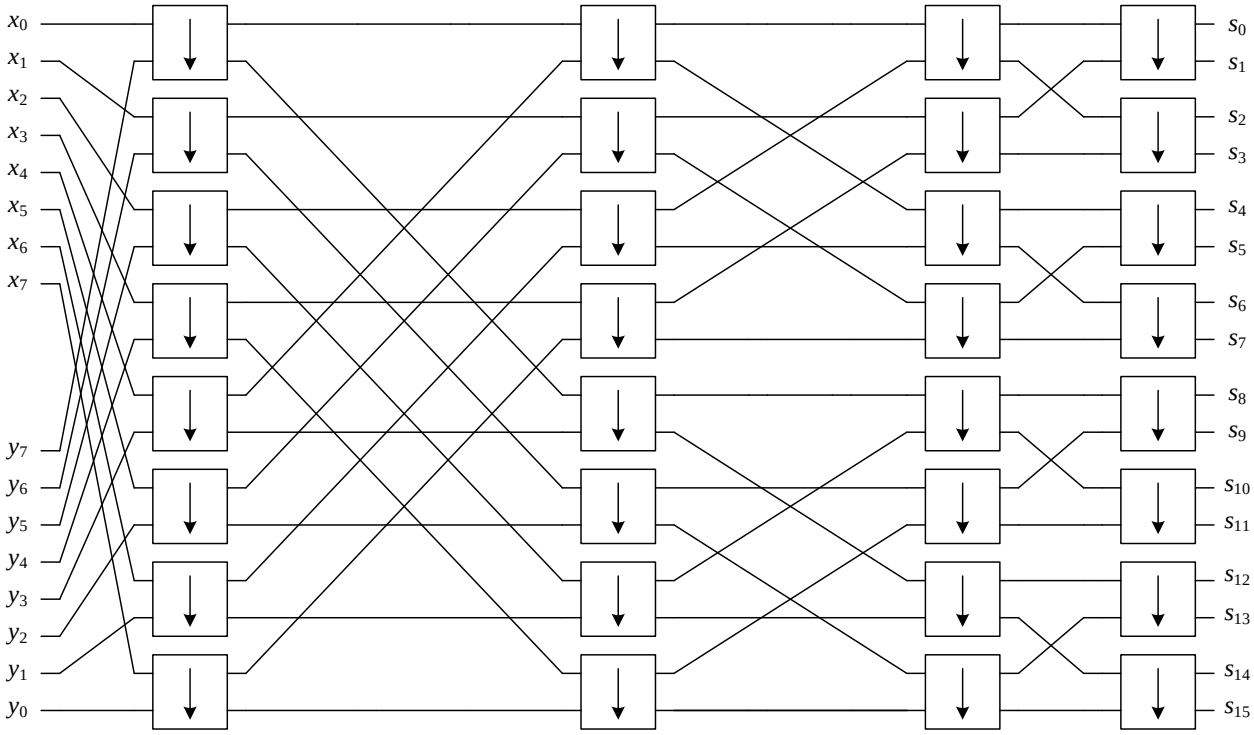
Slika 9.5.22. Struktura za bitoničko spajanje dve sortirane osmočlane liste u jednu šesnaestočlanu listu

Par-nepar spajanje listi tj. nizova je nešto intuitivnije. Naime, vrši se učešljavanje članova niza tako što se parni indeksi šalju u gornju strukturu dimenzije $N/2$, a neparni u donju strukturu dimenzije $N/2$. Svaka od struktura izvrši sortiranje članova i potom se vrši poređenje članova na izlazu. Najniži član sortirane liste gornje strukture se ne poredi jer je on sigurno najmanji (najmanji članovi obe liste su završili u gornjoj strukturi), a isto važi i za najveći član sortirane

liste donje strukture (najveći članovi obe liste su završili u donjoj strukturi). Preostali članovi se redom porede i dodatno preuređuju za jedno mesto na sviječevima za poređenje koji se nalaze na izlazima. Na taj način se dobija kompletno sortirana lista u rastućem redosledu na izlazu.



Slika 9.5.23. Struktura za par-nepar spajanje dve sortirane osmočlane liste u jednu šesnaestočlanu listu



Slika 9.5.24. Struktura za bitoničko spajanje odgovara *n-cube* strukturi

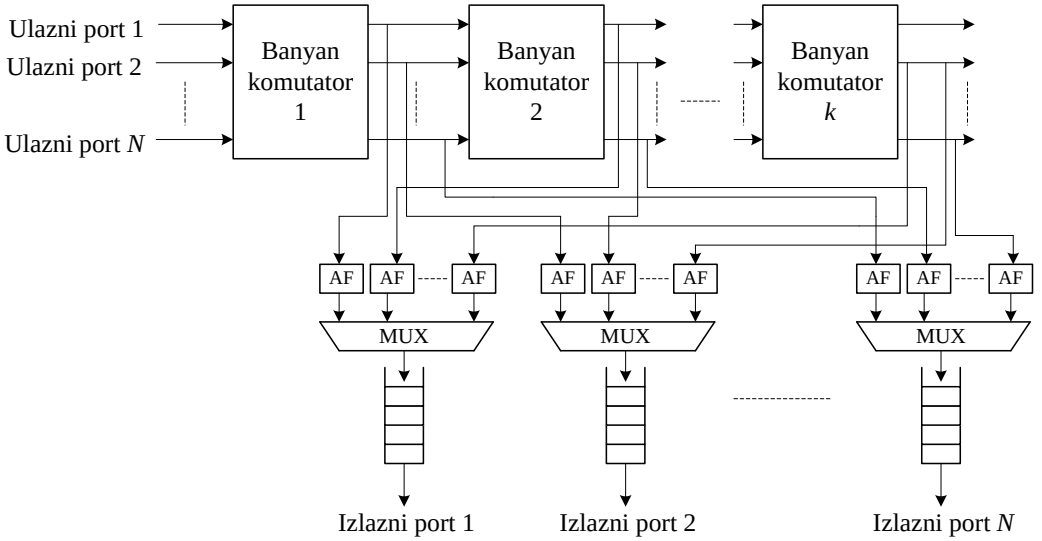
Na slikama 9.5.22 i 9.5.23 su prikazane strukture za spajanje dve sortirane liste dimenzije 8 u jednu sortiranu listu dimenzije 16 strukturama koje su dobijene rekurzivnim principom sa slika 9.5.20 i 9.5.21. Na slikama je označena crvenom bojom jedna struktura za spajanje dve sortirane liste dimenzije 4 u jedni sortiranu listu dimenzije 8, a ljubičastom bojom jedna struktura za spajanje dve sortirane liste dimenzije 2 u jedni sortiranu listu dimenzije 4.

Napomenimo da struktura za bitoničko povezivanje ima šemu povezivanja identičnu kao *n-cube* struktura. Na slici 9.5.24 je prikazana struktura sa slike 9.5.22 sa direktnijim iscrtavanjem linija (na slici 9.5.22 su linije crtane tako da se jasnije vide ulazi u bitoničke strukture manjih dimenzija) i može se jasnije videti da je u pitanju *n-cube* struktura. Upotrebom svič elemenata sa suprotnim prosleđivanjem (svič element sa strelicom usmerenom nagore) dobija se opadajuća (sortirana) spojena lista na izlazu. Pri tome, u slučaju par-nepar spajanja se moraju obrnuti smerovi lista na ulazu (što je logično jer sada najveći elementi obeju listi moraju završiti u gornjoj par-nepar strukturi koja će sada da prima neparne indekse, a najmanji elementi obeju listi moraju završiti u donjoj par-nepar strukturi koja će sada da prima parne indekse). Upravo u tu svrhu se koriste takvi svičevi (sa strelicom usmerenom nagore) u pojedinim bitoničkim strukturama u Batcher sorteru da bi se formirale opadajuće liste u skladu sa zahtevom bitoničke strukture - da jedna lista bude rastuća, a druga opadajuća na njenom ulazu. Takođe, sa slika 9.5.22 i 9.5.23 se može primetiti i razlika u broju svič elemenata. Par-nepar struktura ima manje svič elemenata, dok bitonička struktura ima jednak broj svič elemenata u svim kaskadama (kolonama).

Strukture za sortiranje mogu da poprave performanse Banyan komutatora (pod uslovom da se koristi omega ili *n-cube* struktura), ali unose dodatne kaskade, tj. troše dodatne hardverske resurse. Pri tome, broj dodatnih kaskada koje unosi struktura za sortiranje je jednaka $1+2+...+\log_2 N=[\log_2 N\cdot(1+\log_2 N)]/2$, što je za veliko N ipak relativno velik broj kaskada. Otuda se koriste i druge tehnike za unapređivanje performansi Banyan komutatora i one se tipično zasnivaju na umnožavanju Banyan struktura tako što se kreira više putanja između proizvoljnog ulaznog i proizvoljnog izlaznog porta čime se smanjuje verovatnoća interne blokade (ili se u potpunosti ukida problem interne blokade). Umnoženi Banyan komutatori spadaju u grupu komutatora sa višestrukim putanjama kao što je već rečeno na početku poglavlja, za razliku od Banyan komutatora koji spadaju u grupu komutatora sa jednostrukim putanjama.

Tandem Banyan struktura se zasniva na rednom vezivanju Banyan struktura po principu prikazanom na slici 9.5.25. Ideja je veoma jednostavna, kada se desi kolizija dva paketa usled interne blokade, jedan paket će se korektno rutirati na željeni izlaz sviča, a drugi paket će se rutirati na preostali izlaz sviča. Ovim principom svi paketi će stići do izlaza Banyan strukture. Oni koji su usled kolizije bili pogrešno usmereni će izaći na pogrešan izlaz. Svi ispravno rutirani paketi će biti prosleđeni na odgovarajuće izlazne portove, a oni paketi koji nisu ispravno rutirani će pokušati u sledećoj Banyan strukturi. Na svaki izlazni port se dovode odgovarajući izlazi Banyan struktura, a adresni filtri se koriste da bi se sprečio upis pogrešno rutiranog paketa koji se usled toga pojavio na pogrešnom izlazu Banyan strukture. Verovatnoća uspeha će se povećavati u narednim Banyan strukturama jer će konkurencija biti sve manja pošto će sve veći broj paketa već biti uspešno prosleđen. Dodavanjem dovoljnog broja Banyan struktura se može postići željena verovatnoća gubitaka paketa usled interne blokade. Očigledna mana ove strukture je dodavanje dodatnih Banyan struktura što kao posledicu ima veću potrošnju hardverskih resursa. Treba napomenuti da se u zaglavlje paketa mora dodati informacija o tome da li je paket već doživeo pogrešno usmeravanje unutar Banyan komutatora ili ne. Ovo je bitno ako pogrešno

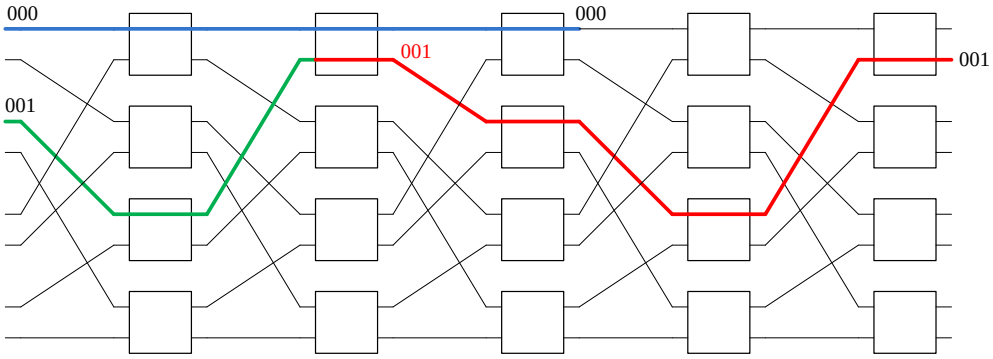
usmereni paket ponovo doživi koliziju sa ispravno rutiranim paketom. U ovoj situaciji nije svejedno koji paket će se pogrešno rutirati, kao u slučaju kolizije dva ispravna paketa ili dva već pogrešno rutirana paketa, već se ispravno rutiran paket mora usmeriti na željeni izlaz sviča. Naravno, ova indikacija će se obrisati na ulasku u novi Banyan komutator jer tamo svi pogrešno rutirani paketi pokušavaju iznova.



Slika 9.5.25. Tandem Banyan struktura

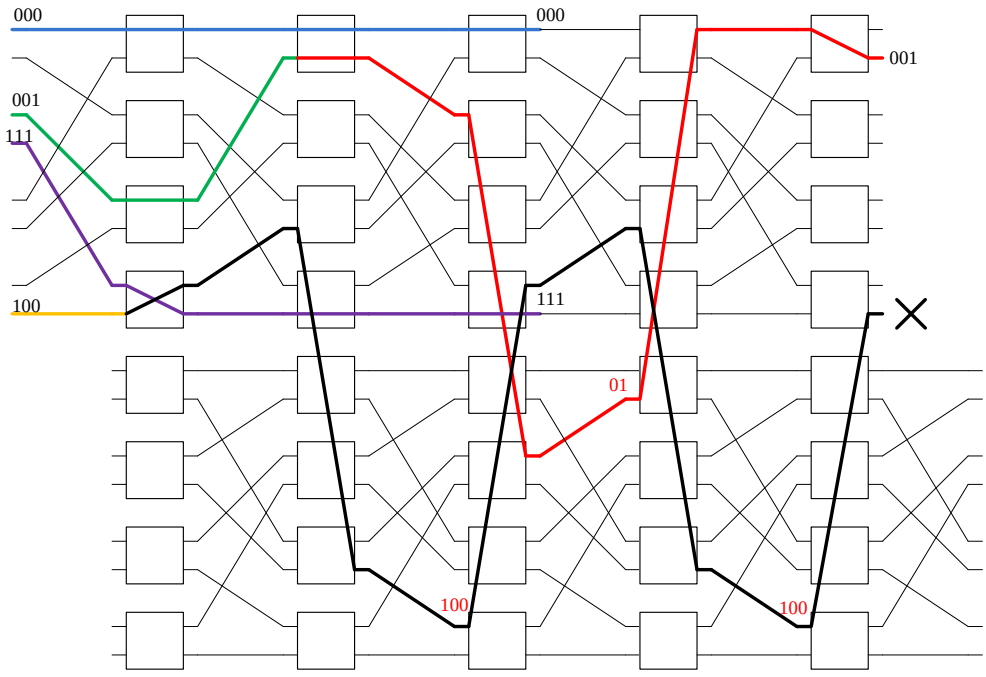
Drugi način za ublažavanje interne blokade je umnožavanje kaskada Banyan strukture. Kao što znamo broj kaskada Banyan strukture je jednak $\log_2 N$ ako se koriste 2x2 svičevi što je najčešće i slučaj. Ako pogledamo omega strukturu sa slika 9.5.1 i 9.5.2 vidimo da nema razlika u povezivanju susednih kaskada. To znači da paket možemo ponovo početi usmeravati ka željenom izlazu u bilo kojoj kaskadi. Stoga možemo proširiti omega strukturu dodatnim kaskadama. Kada se desi blokada, jedan od paketa će da se usmeri ka pogrešnom izlazu sviča i resetovaće se njegova informacija koja se koristi za samorutiranje kroz omega strukturu na početnu vrednost (tj. odredišnu adresu izlaza kome je paket namenjen). Već od sledeće kaskade počinje ponovno rutiranje paketa. Ako zbog nedovoljnog broja dodatnih kaskada paket ne uspe da dođe do odredišta, onda se paket odbacuje na izlazu iz ovako proširene strukture. Nedostatak ove strukture je slaba efikasnost, naročito u slučaju relativno velike vrednosti $\log_2 N$, a razlog je što paket koji se ponovo rutira može ponovo doživeti blokadu, pa ponovo mora da se restartuje rutiranje paketa kroz mrežu što znači da je potrebno od tog momenta još $\log_2 N$ kaskada da bi paket mogao da stigne do svog odredišta (ako ne doživi novu koliziju). Takođe, pošto paket može doći do svog odredišta u različitim kaskadama, izlazi originalne omega strukture i svih dodatnih kaskada se vode na odgovarajuće izlazne portove kao kod tandem Banyan struktura sa slike 9.5.25 (koriste se adresni filtri i multiplekser na identičan način).

Primer ove varijante sa dodavanjem dodatnih kaskada omega strukturi je prikazan na slici 9.5.26. Paket sa ulaza 000 se šalje na izlaz 000, a paket sa ulaza 010 se šalje na izlaz 001 (na slici su na ulazima navedene odredišne adrese paketa). U drugoj kaskadi dolazi do kolizije i paket namenjen izlazu 001 se rutira na pogrešan izlaz. Crvenom bojom je označeno resetovanje informacije za samorutiranje, tj. ona je ponovo postavljena na odredišnu adresu 001. Od tog momenta počinje ponovno rutiranje paketa koje označeno crvenom bojom. Na izlazu 001 pete kaskade paket napokon dolazi do svog ispravnog odredišta.



Slika 9.5.26. Primer rada omega strukture sa dodatnim kaskadama

Prethodno opisana varijanta ima svoju dodatnu modifikaciju koja se sastoji od dodavanja *reverse shuffle* strukture koja je inverzna omega strukturi (i dalje postoje dodatne kaskade koje su sada prisutne u obe strukture), pri čemu se dodavanje vrši tako što se ove dve strukture lepe jedna preko druge, tako da se dobijaju svičevi sa četiri ulaza i četiri izlaza i time praktično operišu u obe ravni tj. strukture. Omega struktura je zadužena za prosleđivanje paketa, a *reverse shuffle* struktura je zadužena za korekciju grešaka u rutiranju paketa koje se javljaju usled kolizija tj. interne blokade. Ideja je veoma jednostavna, paket koji doživi koliziju se vrati za jedan korak unazad koristeći *reverse shuffle* strukturu čime se ustvari simulira vraćanje unazad u omega strukturi. Nije moguće vraćati se unazad kroz kaskade, ali na ovaj način se prolaskom kroz jednu iteraciju *reverse shuffle* strukture vrši vraćanje unazad za jedak korak u omega strukturi, pri čemu se čitavo vreme ide unapred kroz kaskade. Ovime je postignuto da se ne vrši reset čitave informacije za samorutiranje, već se informacija resetuje na ono mesto gde je doživljena greška i pokušava se ponovo. Na ovaj način broj dodatnih kaskada kroz koje se mora proći do ispravnog odredišta je smanjen, čime se podiže efikasnost kompletne strukture. Primer rada ove varijante sa dodavanjem dodatnih kaskada omega strukturi je prikazan na slici 9.5.27.

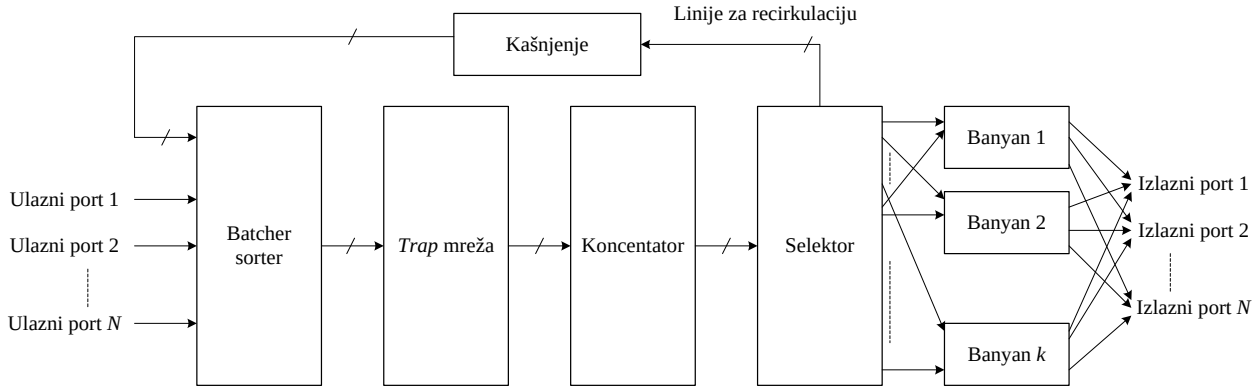


Slika 9.5.27. Primer rada omega strukture sa dodatnim kaskadama i dodatom *reverse shuffle* strukturom

Na slici 9.5.27 je prikazan slučaj koji je prikazan i na slici 9.5.26 gde su paketi namenjeni odredištima 000 i 001 doživeli koliziju u drugoj kaskadi. Omega ravan i *reverse shuffle* ravan su prikazani razdvojeni radi bolje preglednosti dešavanja. Sada se paket, namenjen odredištu 001, nakon pogrešnog rutiranja prosleđuje u svič treće kaskade još uvek u omega ravni (crvena linija pokazuje kretanje paketa nakon kolizije). Međutim, svič treće kaskade sada prebacuje paket u *reverse shuffle* ravan i u okviru te ravni kao da se vraća paket unazad u omega strukturi. Paket ulazi u svič četvrte kaskade koji ima istu poziciju sviča u okviru kaskade kao svič u kome je paket doživio koliziju. Sada se paket rutira počev od bita kod kojeg je doživljena kolizija (drugi bit po redu u adresi 010) i kao što vidimo adresa za samorutiranje je sada 10 tj. deo adrese, a ne kompletna adresa. Paket se prebacuje opet u omega ravan i nastavlja rutiranje od mesta gde je stao zbog kolizije i kao što vidimo u petoj kaskadi dolazi do željenog odredišta (001). Na slici je dat i primer gde paket biva odbačen jer nije uspeo da stigne do željenog odredišta pre završetka strukture. U pitanju je paket za odredište 100 poslat sa ulaza 111. Ovaj paket dva puta doživljava koliziju i iz tog razloga mu je bilo potrebno još dodatnih kaskada da bi došao do željenog odredišta (100). I u ovoj varijanti, pošto paket može doći do svog odredišta u različitim kaskadama, izlazi originalne omega strukture i svih dodatnih kaskada u omega ravni se vode na odgovarajuće izlazne portove kao kod tandem Banyan struktura sa slike 9.5.25 (koriste se adresni filtri i multiplekser na identičan način).

Sunshine komutator je poznati komutator zasnovan na Banyan strukturi, pri čemu koristi tehniku recirkulacije, ali i višeravansku tehniku. Ako se podsetimo klasifikacije komutatora sa početka poglavlja, u grupu komutatora sa višestrukim putanjama spadaju klase višeravanskih i recirkulacionih komutatora, kao i umnoženih Banyan struktura. Očigledno veoma je teško klasifikovati u koju tačno grupu spada *Sunshine* komutator jer ima karakteristike sve tri navedene klase, čime se ilustruje napomena sa početka poglavlja da u mnogim situacijama nije lako precizno klasifikovati komutatore. Struktura *Sunshine* komutatora je prikazana na slici 9.5.28.

Princip rada *Sunshine* komutatora je sledeći. Na ulaz Batcher mreže za sortiranje dolaze paketi poslati sa *N* ulaznih portova, kao i recirkulisani paketi. Recirkulisani paketi imaju prioritet u prosleđivanju. Batcher mreža vrši sortiranje i sortirani paketi se šalju u tzv. *trap* mrežu, ova mreža utvrđuje da li dolazi do kolizija izlaznog porta u smislu da postoji više od *k* paketa namenjenih istom izlaznom portu. Naime, pošto se koristi u paraleli *k* Banyan struktura, moguće je na isti izlazni port proslediti do *k* paketa u jednom slotu. Ako se detektuju kolizije tj. da je jednom portu namenjeno više od *k* paketa, suvišni paketi se obeležavaju da su namenjeni za recirkulaciju. Koncentrator formira sortiranu listu paketa za prosleđivanje i prosleđuje je selektoru koji šalje pakete ka *k* Banyan struktura vodeći računa da ne pošalje u istu Banyan strukturu pakete namenjene istom izlaznom portu (to se radi tako što se svaki *k*-ti izlaz selektora spaja na istu Banyan strukturu, a pošto u sortiranoj listi maksimalno *k* uzastopnih članova može imati istu vrednost time obezbeđujemo da sigurno neće paketi sa istom odredišnom adresom završiti u istoj Banyan strukturi). Takođe, koncentrator formira sortirane liste paketa po prioritetima koje se prosleđuju za recirkulaciju, pri čemu se sortiranje vrši da bi se u slučaju prekoračenja maksimalnog broja paketa koji može da se recirkuliše odbacili oni paketi najmanjeg prioriteta. Linija za recirkulaciju se kasni da bi recirkulisani paketi na ulazu u Batcher sorter bili poravnati sa paketima koje šalju ulazni portovi. Paketi se sa izlaza Banyan strukture prosleđuju na izlazne portove koji moraju biti spremni da prime *k* paketa u slotu (dovoljno brzi baferi).



Slika 9.5.28. *Sunshine* komutator

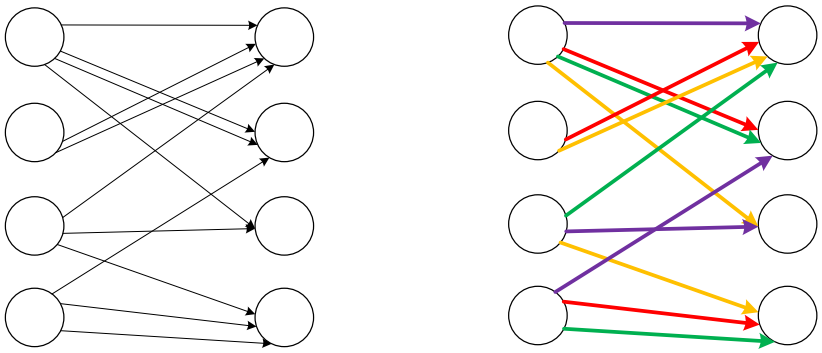
Na kraju napomenimo da Banyan strukture inherentno podržavaju multikast pakete, jer svičevi mogu da kopiraju paket na sve svoje izlaze. Pri tome, informacija za samorutiranje se mora modifikovati da bi svičevi bili svesni da je u pitanju multikast paket koji moraju da kopiraju na sve svoje izlaze.

9.6. Klosovi komutatori

Kao što smo rekli u prethodnom potpoglavlju, krosbar komutatori nisu bili pogodni za kreiranje komutatora sa velikim brojem ulaza i izlaza, pa su iz tog razloga kreirane višekaskadne strukture poput Banyan i Klos struktura koje su trošile manje prekidača i samim tim bile pogodnije za kreiranje većih komutatora. Klos strukture smo već opisali u okviru analognih komutacionih polja u potpoglavlju 4.2 i sve navedeno tamo važi i za paketske Klosove komutatore, pri čemu su najvažnije stavke iz tog poglavlja sa stanovišta paketske verzije Klosovog komutatora način kreiranja Klosovog komutacionog polja (tj. komutatora), kao i uslovi za neblokirajuće i uslovno blokirajuće polje (izrazi (4.2.4.1) i (4.2.4.3), respektivno). Teoretski gledano mogu se koristiti i Klosove strukture sa više od 3 kaskade kao paketski komutatori, ali u praksi se, ipak, koriste samo trokaskadne Klosove strukture. Pošto sve navedeno za Klosove strukture iz poglavlja 4.2 važi i za paketske komutatore, osvrnimo se sada na razlike koje proističu iz činjenice da se Klosova struktura koristi za paketsku komutaciju.

U komutaciji kola Klosova struktura se koristila za komutaciju govorne veze koja je zauzimala resurse (put) u Klosovoj strukturi koji su se potom koristili za prenos govornih signala. U slučaju neblokirajućeg Klosovog komutacionog polja veza se mogla ostvariti bez ikakvih preuređivanja postojećih veza, a u slučaju uslovno blokirajućeg polja potencijalno su se morale preurediti postojeće veze ako je veza koja se uspostavlja bila uslovno blokirana. Za preuređivanje se mogao koristiti Paulov algoritam. U slučaju paketske komutacije, okolnosti konfigurisanja Klosovog komutatora su malo drugačije. Naime, tipično se vrši komutacija paketa fiksne dužine (komutacija ćelija), pri čemu se u svakom sukcesivnom slotu parovi ulaz i izlaz koji se uparuju menjaju. Pri tome, trajanje slota u kome mora da se izvrši kompletna konfiguracija Klosovog komutatora je znatno kraće nego vreme potrebno za dodavanje nove veze u slučaju komutacije kola, naročito ako se koriste veoma brzi linkovi. Otuda, nema potrebe za vršenjem preuređivanja veza da bi se dodala nova veza ili inkrementalnog dodavanja nove veze, jer se svaki put konfiguracija radi od nule. Stoga je posebna pažnja posvećena algoritmima raspoređivanja koji vrše ne samo uparivanje ulaz-izlaz koji će se povezati u dotičnom slotu, već i put kojim će ići paket za upareni par pošto postoji više puteva na raspolaganju (puteva ima

onoliko koliko ima i svičeva u srednjoj tj. drugoj kaskadi - ovde smo kao i kod Banyan struktura svičevima označili gradivne jedinice Klos komutatora). Prilikom izbora puteva mora se voditi računa da se ne preklope putevi za pojedine uparene parove ulaz-izlaz. Pošto se u praksi češće koriste uslovno blokirajuće strukture, problem uparivanja i određivanja puta postaje složeniji jer je teže odrediti optimalan skup parova da se postigne što veća propusnost Klosovog komutatora usled postojanja mogućnosti pojave uslovne blokade za neke potencijalne parove, a razrešavanje uslovne blokade bi usporilo proces konfigurisanja komutatora. Kod kreiranja algoritama raspoređivanja treba imati u vidu da oni moraju veoma brzo da rade (u trajanju jednog slota moraju da izvrše proračun, a dužina slota može da bude svega nekoliko desetina ns, pa čak i manja), što znači da je idealno da oni budu što jednostavniji sa jedne strane, a sa druge strane da daju dobre rezultate konfigurisanja komutatora. Razlog zašto se koriste uslovno blokirajući Klosovi komutatori leži u ekonomičnosti (ako se pretpostavi simetričan komutator i pogledaju izrazi (4.2.4.1) i (4.2.4.3), može se videti da je kod uslovno blokirajućeg komutatora, broj svičeva u srednjoj kaskadi gotovo dvostruko manji, a to ujedno znači i manje linkova u komutatoru za povezivanje svičeva). U literaturi je predložen velik broj algoritama raspoređivanja koji se mogu primeniti na Klosove komutatore. Pri tome, u slučaju trokaskadnog Klosovog komutatora, izlaz sviča iz prve kaskade određuje put kojim će paket ići, jer svič druge kaskade mora da usmeri paket ka sviču treće kaskade na koji je priključen odredišni izlazni port, a sam svič treće kaskade usmerava paket na odgovarajući (odredišni) izlazni port. Stoga se izbor puta vrši samo u svičevima prve kaskade i algoritmi raspoređivanja se koncentrišu na njih. Pošto je put određen izlaznim linkom iz sviča prve kaskade, a samim tim svičem druge kaskade preko kojeg će paket ići, rezultat rada algoritama raspoređivanja za svaki upareni par ulaz-izlaz definiše, ustvari, preko kog komutatora srednje (druge) kaskade će paket ići.



Zadata uparivanja za koja treba odrediti put tj. svič druge kaskade preko kojih će ići

Rešenje problema

Primer problema nalaženja puteva za $C_3(4,4,4,4,4)$

Slika 9.6.1. Rešavanje problema određivanja puteva kroz Klosovu strukturu

Problem koji algoritmi raspoređivanja treba da reše se može podeliti u dve etape. U prvoj etapi se nalaze uparivanja ulaznih portova sa izlaznim portovima i ovaj problem je identičan problemu koji rešavaju strukture sa baferima na ulazu i koje nalaze maksimum ili maksimalno uparivanje ulaza i izlaza u zavisnosti od primenjenog algoritma za rešavanje tog problema. Kada se odrede parovi ulaz-izlaz, u drugoj etapi treba rešiti sledeći problem, a to je konfigurisanje komutatora, odnosno određivanje puta za svaki upareni par tako da ne dođe do kolizije puteva. Problem druge etape se može prikazati preko grafa koji sadrži čvorove podeljene u dve kolone,

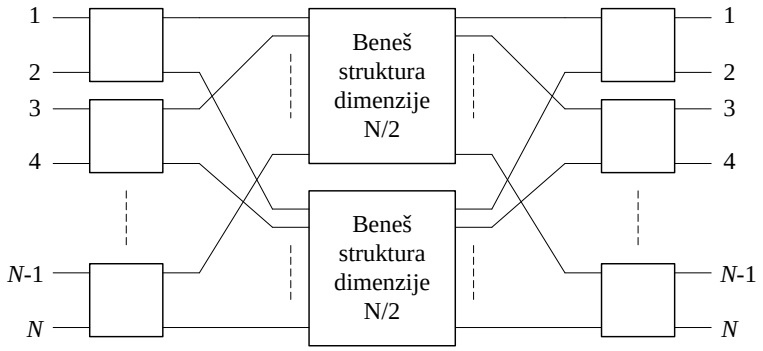
prvu kolonu čine čvorovi koji predstavljaju svičeve prve kaskade, a drugu kolonu predstavljaju čvorovi koji predstavljaju svičeve treće kaskade (slika 9.6.1). Broj linija između odgovarajućeg čvora prve i druge kolone predstavlja broj uparenih parova ulaza i izlaza, gde su ulazi povezani na dotični čvor prve kolone tj. svič prve kaskade, odnosno izlazi na dotični čvor druge kolone, tj. svič treće kaskade. Problem koji treba rešiti u ovom grafovskom prikazu je bojenje linkova tako da linkovi koji izlaze iz nekog čvora budu svi različito obojeni, odnosno da linkovi koji ulaze u neki čvor budu svi različito obojeni, pri čemu treba upotrebiti minimalan broj boja (ili preciznije broj upotrebljenih boja ne sme prevazići broj svičeva srednje kaskade). Svaka boja, ustvari, odgovara jednom sviču srednje kaskade i određuje put kojim će paket između odgovarajućih svičeva prve i treće kaskade ići. Boje linkova koje izlaze iz jednog čvora prve kolone moraju da se razlikuju jer je svič prve kaskade povezan sa svakim svičem druge kaskade preko samo jednog linka. Boje linkova koje ulaze u jedan čvor druge kolone moraju da se razlikuju jer je svič treće kaskade povezan sa svakim svičem druge kaskade preko samo jednog linka. Na slici 9.6.1 je prikazan primer rešavanja problema nalaženja puteva kroz $C_3(4,4,4,4,4)$ Klosovu strukturu pomoću grafa. Kao što vidimo u rešenju problema korišćene su četiri boje, a svaka boja odgovara jednom sviču druge kaskade. Pri tome, nijedan čvor prve kolone nema linkove sa istom bojom, a isto važi i za čvorove druge kolone što znači da je problem uspešno rešen. Na primer, svič druge kaskade, kojem odgovara link zelene boje, će povezati prvi svič prve kaskade sa drugim svičem treće kaskade, treći svič prve kaskade sa prvim svičem treće kaskade i četvrti svič prve kaskade sa četvrtim svičem treće kaskade. Problem druge etape se može predstaviti i u vidu matrice nxn, gde je n broj svičeva prve, odnosno treće kaskade (broj svičeva prve i treće kaskade je tipično jednak jer su Klosovi komutatori tipično simetrični usled istog broja ulaznih i izlaznih portova u mrežnim čvorovima). Član matrice (i, j) predstavlja broj paketa koje treba proslediti između sviča i prve kaskade i sviča j treće kaskade (vrednosti članova matrice su određeni rezultatom prve etape). Rešavanje problema druge etape se u ovoj varijanti može predstaviti dekompozicijiom matrice na k matrica u kojima će svaki red i kolona sadržati maksimalno jednu jedinicu (a sve ostale vrednosti u matrici će biti nula), pri čemu je zbir ovih k matrica jednak originalnoj matrici. Parametar k predstavlja broj svičeva u drugoj kaskadi, pa otuda svaka od k matrica predstavlja jedan svič druge kaskade i vrednosti koje odgovaraju jedinici u njima određuju koji svičevi prve i treće kaskade će biti spojene preko dotičnog sviča druge kaskade (red određuje svič prve kaskade, a kolona svič treće kaskade). Važno je napomenuti da pošto se koriste uslovno blokirajući (ili neblokirajući što je redak slučaj) Klosovi komutatori, rešenje problema druge etape sigurno postoji, ali ga treba i naći.

Sa stanovišta multikast paketa, svičevi u Klosovom komutatoru mogu da prave kopije i šalju ih na više svojih izlaza, čime Klosov komutator inherentno podržava multikast pakete, ali kao i kod Banyan komutatora, potrebno je kreirati mehanizam za efikasnu konfiguraciju svičeva, što je složen problem. Međutim, multikast problem je i inače složen kao što je već rečeno na početku ovog poglavlja.

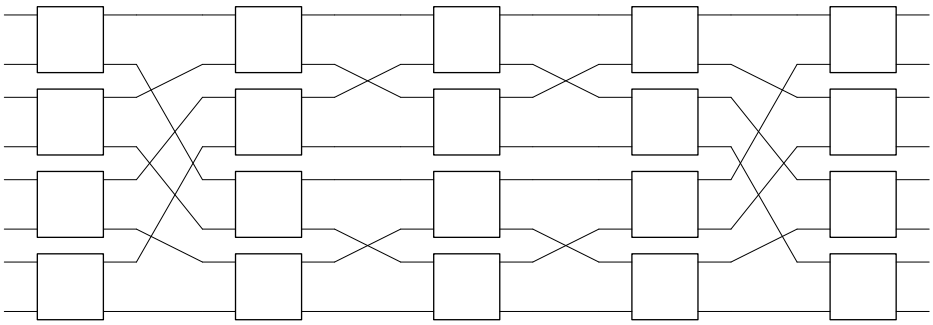
U praksi Klosovi komutatori su veoma popularni za kreiranje mrežnih čvorova velikih kapaciteta koji zahtevaju komutatore sa velikim brojem ulaza i izlaza. Kao što smo već naveli, koriste se tipično samo trokaskadne strukture. Osnovni razlog je što je za njih lakše kreirati algoritam raspoređivanja koji bi ih optimalno konfigurisao. U suštini za komutatore sa brojem ulaza većim od 200 bi čak optimalnije bile petokaskadne strukture ako se podsetimo grafika sa slike 4.2.6.4 koji prikazuje zavisnost broja prekidača od broja ulaza u Klosov komutator, ali bi upotreba petokaskadnih Klosovih komutatora unela dodatnu kompleksnost sa stanovišta konfiguracije pa se, ipak, koriste trokaskadne strukture.

Kao i kod Banyan komutatora, baferi se mogu dodati na ulazne portove, na izlazne portove ili u sam komutator (mogu i kombinacije navedenih pozicija bafera). Pri tome, dodavanje bafera u komutator olakšava proces raspoređivanja (pa se mogu koristiti jednostavniji algoritmi) jer sada mogu i da se dozvole potencijalne kolizije puteva paketa, jer će se paketi koji izgube u duelu moći sačuvati u baferima u komutatoru. Baferi se mogu postaviti u svičeve bilo koje od kaskada, pri čemu može baferima biti pokriveno i više kaskada, pa i sve tri. U suštini ako se baferi postave u svičeve prve kaskade, to bi odgovaralo strukturi sa baferima na ulazu, a ako se baferi postave u svičeve treće kaskade to bi odgovaralo strukturi sa baferima na izlazu. Baferi u srednjoj kaskadi bi odgovarali strukturi sa baferima u komutatoru.

Ono što predstavlja svojevrsnu manu sa stanovišta hardverske implementacije je činjenica da se tipovi svičeva u kaskadama Klosovog komutatora tipično razlikuju po svojim dimenzijama (broju ulaza i izlaza). Sa stanovišta hardverske implementacije je uglavnom poželjnije imati iste tipove svičeva jer se time omogućava efikasnija hardverska implementacija. Benešov komutator predstavlja modifikaciju Klosove strukture i kod njega se u svim kaskadama koristi isti tip sviča dimenzije bxb. Pri tome, najčešća varijanta podrazumeva 2x2 svičeve. Benešov komutator ima ukupno $2\log_b N-1$ kaskada, gde je N broj ulaza, odnosno izlaza u komutator. Svaka kaskada sadrži ukupno N/b svičeva. Ako se koriste 2x2 svičevi tada je ukupan broj kaskada $2\cdot\log_2 N-1$, a svaka kaskada sadrži $N/2$ svičeva. Benešov komutator se takođe često koristi u praksi za formiranje paketskih komutatora velikog kapaciteta tj. paketskih komutatora koji podržavaju velik broj ulaza i izlaza.



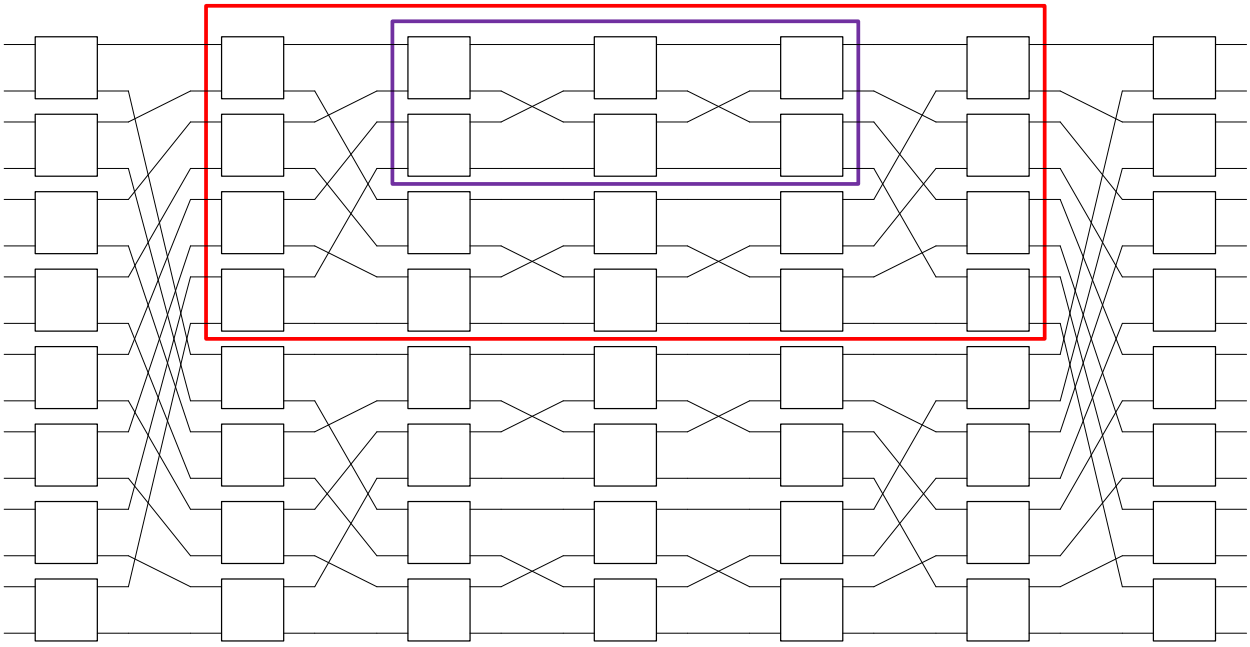
Slika 9.6.2. Rekurzivno kreiranje Benešovog komutatora dimenzije *N*x*N*



Slika 9.6.3. Benešov komutator 8x8

Benešov komutator se formira primenom principa rekurzije prikazanom na slici 9.6.2 za slučaj kada se koriste 2x2 svičevi. Princip je veoma jednostavan. Gornji izlazi svičeva prve kaskade se redom vezuju na gornju (prvu) Beneš strukturu dimenzije $N/2$, a donji izlazi svičeva prve kaskade na identičan način se vezuju na donju (drugu) Beneš strukturu dimenzije $N/2$. U

slučaju *b*x*b* svičeva bi princip bio isti - prvi izlazi svičeva prve kaskade bi se vezivali redom na prvu Beneš strukturu dimenzije *N*/*b*, drugi izlazi na drugu Beneš strukturu dimenzije *N*/*b*,... poslednji izlazi na poslednju Beneš strukturu dimenzije *N*/*b*. Ulazi svičeva treće kaskade se vezuju na izlaze druge kaskade tj. Benešove strukture dimenzije *N*/2 (*N*/*b* u opštem slučaju) po istom principu pošto je Benešova struktura simetrična.

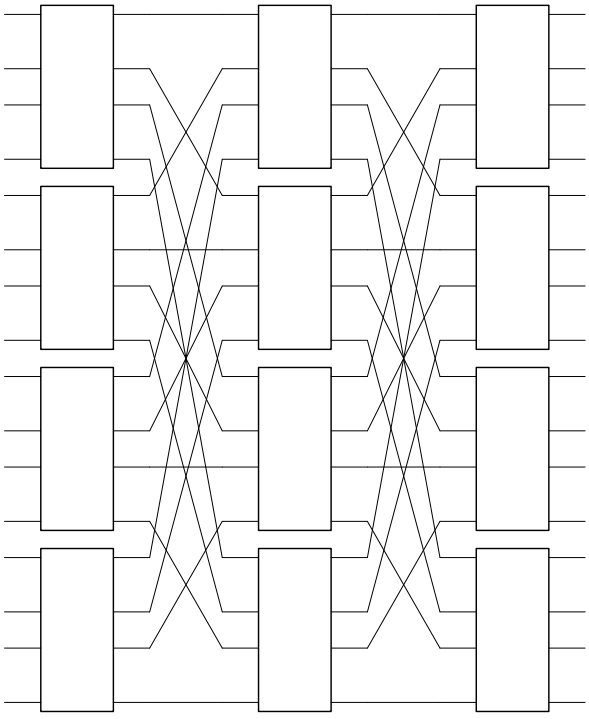


Slika 9.6.4. Benešov komutator 16x16

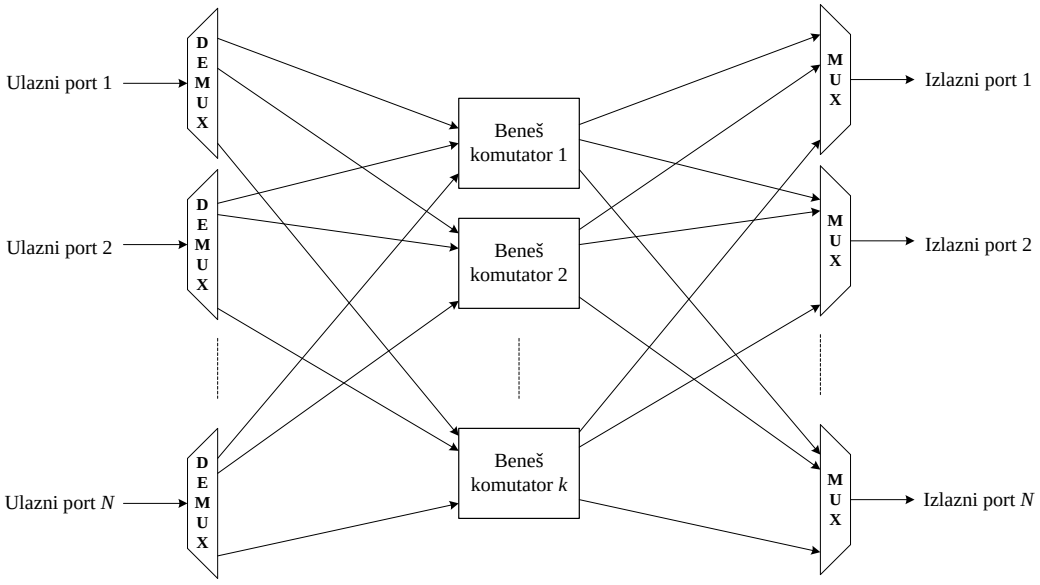
Na slikama 9.6.3 i 9.6.4 su prikazani Benešovi komutatori za dimenzije 8x8 i 16x16 kada se koriste svičevi 2x2. Na slici 9.6.4 su prikazani uokvireni po jedan Benešov komutator dimenzije 4x4 i 8x8 da bi se bolje sagledao princip rekurzije. Ako uporedimo strukturu Benešovog komutatora sa *baseline* komutatorom sa slika 9.5.5 i 9.5.6 možemo videti da je Beneš struktura veoma slična *baseline* strukturi. U slučaju 8x8 Beneš komutatora, prve tri kaskade su identične *baseline* komutatoru, a poslednje tri kaskade odgovaraju inverznoj *baseline* strukturi. Analogno važi za prve četiri, odnosno poslednje četiri kaskade Benešovog 16x16 komutatora. Na slici 9.6.5 je prikazan Benešov komutator dimenzije 16x16 za slučaj kada se koriste 4x4 svičevi. Možemo videti da je, u stvari, u pitanju Klosova trokaskadna struktura $C_3(4,4,4,4,4)$, a razlog je što se trokaskadne Benešove strukture direktno poklapaju sa Klosovim strukturama. Isto tako struktura sa slike 9.6.5 je veoma slična *baseline* strukturi po istom principu kao Benešove strukture sa 2x2 svičevima. Prve dve kaskade su identične *baseline* strukturi, a poslednje dve inverznoj *baseline* strukturi.

Videli smo da je Benešova struktura veoma slična *baseline* strukturi kada se koriste 2x2 svičevi, odnosno da se *baseline* i inverzna *baseline* struktura međusobno spajaju pri čemu su svičevi poslednje kaskade *baseline* strukture i prve kaskade inverzne *baseline* strukture međusobno preklopljeni. Otuda se u Benešovoj strukturi može koristiti samorutiranje na osnovu odredišne adrese u poslednjih $\log_2 N$ kaskada, pošto će paket u njima prolaziti kroz inverznu *baseline* strukturu tj. inverzni *baseline* komutator. Isto važi i za slučaj kada se koriste *b*x*b* svičevi, samo je tada u pitanju poslednjih $\log_b N$ kaskada. Prvih $\log_2 N$-1 kaskada ($\log_b N$-1 kaskada u opštem slučaju) se mogu koristiti za distribuisanje paketa po mogućim putanjama radi

smanjenja verovatnoće interne blokade jer do ulaska u inverznu *baseline* strukturu paket ima više opcija na raspolaganju. Tek kad se paket nađe na ulazu u inverznu *baseline* strukturu će paketu preostati samo jedna putanja na raspolaganju. Princip prosleđivanja (usmeravanja) paketa u kaskadama pre ulaska u inverznu *baseline* strukturu zavisi od usvojenog algoritma. Na primer, paket se može usmeravati po slučajnom principu u svakom od svičeva kroz koje prolazi; može se koristiti i tehnika plavljenja kojom će paket da se šalje na sve raspoložive izlaze sviča, a da oni nisu zauzeti prosleđivanjem nekog drugog paketa; može se koristiti i tehnika kojom se na ulazu u sam Benešov komutator odredi čitav put paketa kroz Benešov komutator.



Slika 9.6.5. Benešov komutator 16x16 pomoću 4x4 svičeva



Slika 9.6.6. Kantor komutator

Kantor komutator koristi više Benešovih komutatora upotrebom principa višeravanske tehnike. Prikaz Kantor komutatora je prikazan na slici 9.6.6. Ulazni portovi su povezani na sve Benešove komutatore, kao i izlazni portovi. Na ovaj način se povećava broj mogućih putanja paketa do odredišta tj. izlaznog porta. Kantor komutator je neblokirajući ako je ispunjeno da je $k \geq \log_2 N$. Ovo svojstvo omogućava lakšu implementaciju algoritama za raspoređivanje, jer su algoritmi jednostavniji i efikasniji ako je komutator neblokirajući, od slučaja kada je komutator uslovno blokirajući što smo napomenuli na početku ovog potpoglavlja.

9.7. Višeravanski komutatori

Višeravanski komutatori se zasnivaju na tehnici paralelne upotrebe komutatora. Naime, višeravanski komutator implementira više kopija komutatora, poput na primer, *sunshine* komutatora ili Kantor komutatora koje smo ranije opisali. Svaki taj gradivni komutator za sebe predstavlja jednu ravan komutiranja, pa otuda i naziv višeravanski komutatori. Višeravanski komutatori na ovaj način obezbeđuju bolje performanse pre svega u pogledu interne blokade. Pri tome nema potrebe koristiti ubrzanje internih linkova tj. komutatora (ili bar nema potrebe koristiti isuviše veliko ubrzanje u zavisnosti od broja ravni) jer je moguće proslediti više paketa odjednom sa različitih ulaznih portova na isti izlazni port (tada bafer na izlaznom portu treba da bude dovoljno brz). U slučaju ovih komutatora, moraju se implementirati algoritmi koji će vršiti distribuciju paketa po ravnima tj. gradivnim komutatorima da se ne bi desilo, na primer, da se dva paketa namenjena istom izlaznom portu proslede u isti gradivni komutator jer bi tada sigurno došlo do njihove međusobne kolizije. Višeravanski komutatori imaju i veću pouzdanost, jer otkazom jedne ravni tj. gradivnog komutatora, višeravanski komutator i dalje radi (sa nešto slabijim performansama) jer su ostale ravni funkcionalne i mogu da vrše komutaciju.

9.8. Recirkulacioni komutatori

Recirkulacioni komutatori vrše recirkulaciju paketa koji bi inače bili odbačeni usled kolizije. Ovi recirkulisani paketi se dovode na zasebne ulaze komutatora, što je jedna od mana ovog komutatora jer broj ulaza i izlaza komutatora mora biti veći od broja ulaznih i izlaznih portova, jer moraju da postoje dodatni izlazi za odbijene pakete koji će se recirkulisati, kao i dodatni ulazi za recirkulisane pakete koji će ponovo pokušati da se komutiraju do odgovarajućeg izlaznog porta. Recirkulacioni komutatori podižu performanse komutatora jer povećavaju propusnost i smanjuju procenat gubitaka paketa, ali mogu da izazovu poremećaj u redosledu paketa nekog toka što može biti problem u određenim situacijama (na primer, u TCP toku usled načina rada mehanizma za potvrđivanje paketa i kontrolu zagušenja može doći do smanjenja protoka podataka u TCP toku) i tada se moraju implementirati mehanizmi na izlaznom portu koji bi ispravili ovo narušavanje redosleda paketa. Primer komutatora koji koristi recirkulaciju je *sunshine* komutator.

9.9. Ostale tehnike komutacije

U okviru ovog potpoglavlja ćemo navesti neke od poznatih komutatora, kao i arhitektura mrežnih čvorova prilagođenih efikasnijoj komutaciji, a koje nismo svrstali u prethodno izložene tipove komutatora.

Birkhoff-Von Neumann komutator u originalnoj varijanti, u stvari, predstavlja način kontrolisanja komutatora poput, na primer, krosbar komutatora. U slučaju Birkhoff-Von

Neumann komutatora se koristi matrica saobraćaja A dimenzije NxN, gde je N broj ulaznih, odnosno izlaznih portova. Član matrice (i, j) predstavlja normalizovani protok saobraćaja između ulaznog porta i i izlaznog porta j (normalizovan u odnosu na brzine linkova tj. portova). Pri tome se smatra da nema preopterećenja izlaza (zbir svake kolone je maksimalno 1). Zbir svakog reda je takođe maksimalno 1 jer bi inače ulazi imali veći saobraćajni protok nego što protoci ulaznih portova to fizički dozvoljavaju. Pokazano je da se saobraćajna matrica A može razložiti na tzv. matrice permutacije P_k na sledeći način:

$$A \leq \sum_{k=1}^{K} \alpha_k P_k$$
$$\sum_{k=1}^{K} \alpha_k = 1 \qquad (9.9.1)$$
$$K \leq N^2 - 2N + 2$$

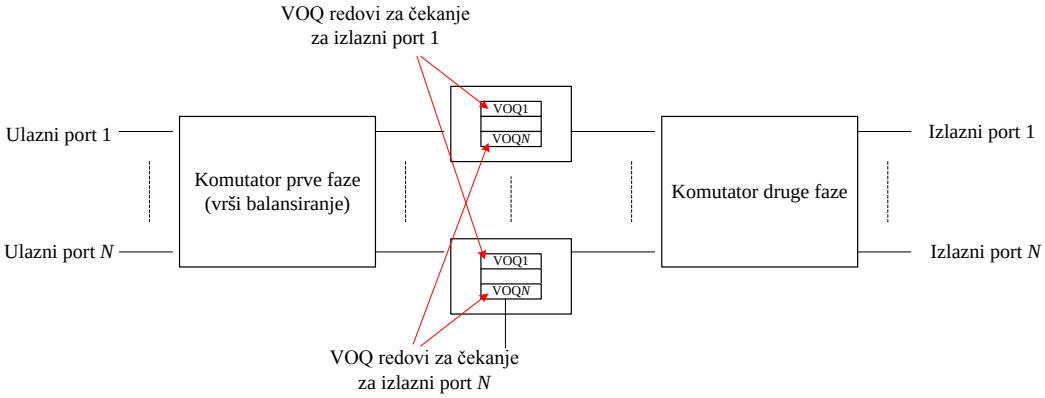
gde su α_k pozitivni težinski koeficijenti. Matrica permutacije podrazumeva da njeni članovi imaju samo vrednosti 0 ili 1, pri čemu u svakoj koloni, kao i redu može biti maksimalno jedna jedinica.

Svaka matrica permutacije P_k predstavlja jednu konfiguraciju komutatora tj. predstavlja skup uparenih parova ulaza i izlaza. Težinski koeficijent α_k predstavlja učestanost kojom će se javljati dotična konfiguracija komutatora. Na primer, ako je težinski koeficijent 1/4, tada će se dotična konfiguracija javljati u 25% slučajeva. Na ovaj način će biti zadovoljena relacija u prvom redu izraza (9.9.1) tj. biće opslužen saobraćaj predstavljen saobraćajnom matricom A. Kao što se vidi, onog momenta kada se izvrši proračun i odrede matrice permutacije i njihovi težinski koeficijenti, može se formirati TDM princip konfigurisanja komutatora, gde će svaka konfiguracija (matrica permutacije) dobiti broj kanala koji je u srazmeri sa odgovarajućim težinskim koeficijentom (jedan kanal traje jedan slot, a slot odgovara trajanju jednog paketa). U slučaju uniformne distribucije saobraćaja na svim ulazima, sve matrice permutacije će imati isti težinski faktor pa će se oneređati jedna za drugom. Pri tome, matrice permutacije se tada ciklično formiraju. Na primer, prva konfiguracija bi bila uparivanje (1,1), (2,2)...(N,N). Sledeća bi bila (1,2), (2,3)...(N,1), pa potom (1,3), (2,4)...(N,2) itd. Kao što se vidi, bilo bi ukupno N konfiguracija koje je jednostavno odrediti.

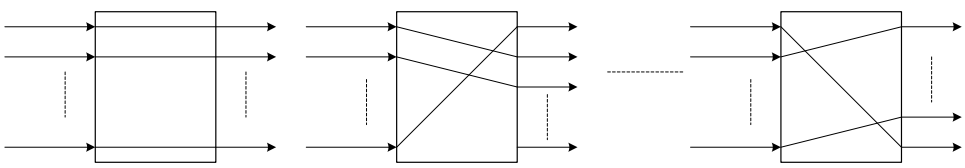
Naravno, postoje i problemi vezani za ovakvu strukturu. Saobraćaj tipično nema uniformnu distribuciju pa je određivanje matrica permutacije komplikovano i zahteva komplikovan proračun, odnosno mnogo vremena traje proračun. Takođe, matrica saobraćaja nije konstantna već se menja u vremenu pa se složeni proračun mora često i izvršavati. Nije ni jednostavno dinamički određivati trenutnu matricu saobraćaja jer se ona brzo i menja.

Birkhoff-Von Neumann komutator sa balansiranjem saobraćaja pokušava da reši ovaj problem tako što kreira dve faze komutacije (slika 9.9.1). U prvoj fazi se vrši balansiranje saobraćaja sa ulaza po VOQ redovima za čekanje koji su pridruženi ulazima u komutator druge faze (svakom ulazu u komutator druge faze je pridruženo N VOQ redova za čekanje, po jedan za svaki izlazni port). Komutator druge faze je konfigurisan kao da radi sa uniformnom distribucijom saobraćaja (konfiguracija opisana nešto ranije u ovom potpoglavlju i koja se sastoji iz N konfiguracija koje se ciklično ponavljaju), pa je stoga princip konfigurisanja komutatora druge faze veoma jednostavan i nema nikakvih proračuna. Ideja je jednostavna. Cilj prve faze je da razbaca pakete sa ulaznih portova po ulazima komutatora druge faze (tj. po njihovim VOQ

redovima za čekanje) i time obezbedi da čak i u slučaju *bursty* saobraćaja ne budu dominantno prisutni paketi samo za nekoliko parova ulaz-izlaz i gde bi se skup tih parova dinamički menjao vremenom upravo usled *bursty* prirode tokova čime bi se neprestano menjale optimalne konfiguracije komutatora. Na ovaj način bi svaki ulaz komutatora druge faze imao pakete za svaki izlazni port pa bi se dobio približno uniforman saobraćaj po ulazima komutatora druge faze. Otuda bi svaka konfiguracija iz ranije pomenutih *N* cikličnih konfiguracija imala maksimalno iskorišćenje komutatora druge faze koji bi tako ostvarivao maksimalnu propusnost. Očigledna prednost ove ideje je veoma jednostavno konfigurisanje komutatora druge faze bez ikakvih proračuna (praktično se prolazi kroz *N* unapred određenih konfiguracija komutatora). Komutator prve faze radi po istom principu konfigurisanja (*N* cikličnih konfiguracija) jer se na taj način obezbeđuje ciklično povezivanje ulaznog porta sa ulazima u komutator druge faze i time se očekuje da će paketi istih tokova (tok = par ulazni port-izlazni port) završiti na različitim ulazima komutatora druge faze, tj. da će biti razbacani podjednako po ulazima komutatora druge faze, čime će se postići za komutator druge faze uniformna distribucija saobraćaja na njegovim ulazima. Mana ovog rešenja je što u nekim saobraćajnim situacijama može doći do toga da se i pored upotrebe komutatora prve faze paketi namenjeni istom izlaznom portu gomilaju na istom ulazu komutatora druge faze (ili manjim skupovima ulaza komutatora druge faze) i time smanji efikasnost rada (u nekim situacijama smanjenje može biti drastično) komutatora druge faze, a time i čitavog Birkhoff-Von Neumann komutatora sa balansiranjem saobraćaja. Smanjenje efikasnosti potiče od toga što u takvim situacijama komutator druge faze ne ostvaruje dobar protok za određeni izlaz za koji su paket nagomilani u VOQ redovima na malom broju ulaza komutatora druge faze (najgori slučaj samo jedan ulaz) jer će u veoma malom broju konfiguracija biti prosleđivani paketi na dotični izlaz (mali ostvareni protok tog izlaza, a protok kojim ulaze ti paketi namenjeni dotičnom izlazu je znatno veći).



Birkhoff-Von Neumann komutator sa balansiranjem saobraćaja

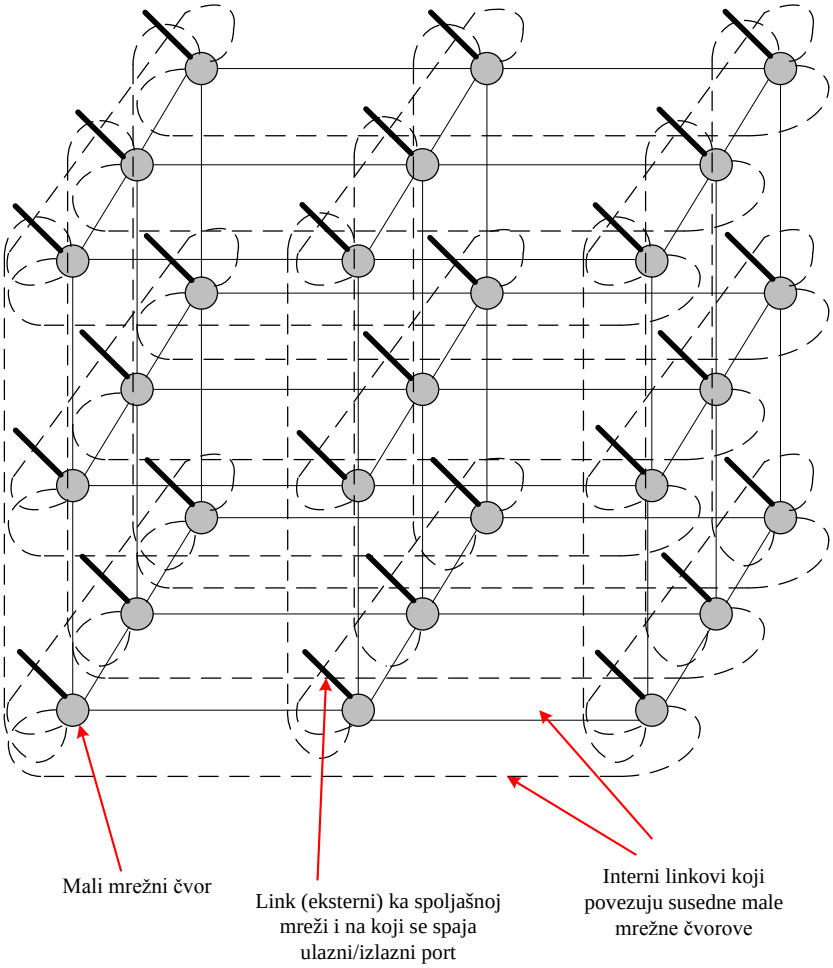


Ciklične konfiguracije komutatora

Slika 9.9.1. Birkhoff-Von Neumann komutator sa balansiranjem saobraćaja

Interesantna arhitektura korišćena u pojedinim mrežnim uređajima je torusna arhitektura prikazana na slici 9.9.2. Ideja ove arhitekture je da se teret komutacije prebaci na same portove.

Naime, svaki port sada predstavlja jedan mali mrežni čvor koji je povezan sa šest susednih mrežnih čvorova tj. portova (levi sused/desni sused po *x* osi, sused ispred/sused iza po *y* osi, gornji sused/donji sused po *z* osi). Paket se rutira kroz ove male mrežne čvorove da bi došao do odredišta - izlaznog porta kome je paket namenjen. Pri tome, postoji više putanja po kojima paket može stići od ulaznog porta do odredišnog izlaznog porta. Rutiranje kroz torusnu arhitekturu može biti fiksno (na primer, prvo po *x* osi, pa po *y* osi pa po *z* osi dok se ne dostigne odredište), može biti adaptivno (u zavisnosti od opterećenosti putanja bira se manje opterećena putanja), kao i balansirano (sve potencijalne putanje se koriste u skladu sa svojom dodeljenom težinom). Sama komutacija se odvija u tim malim mrežnim čvorovima, a pošto je broj ulaza/izlaza u komutator mali, komutatori su veoma jednostavni, pa je očigledno da je kompletna težina komutacije preusmerena na samu arhitekturu mrežnog čvora tj. torusnu mrežu (komutatori su praktično distribuirani po mrežnom čvoru). Na ovaj način je postignuta visoka skalabilnost mrežnog čvora jer kompleksnost komutacije ne zavisi od broja ulaznih/izlaznih portova, pa je lako povećavati broj portova. Naravno, postoje i mane. Kašnjenje kroz ovakav mrežni čvor zavisi od međusobne udaljenosti portova u torusnoj mreži, a takođe može biti i veliko u slučaju velikih torusnih mreža sa velikim brojem čvorova. Takođe, postoji opasnost od zagušenja internih linkova čak i u situacijama kada nijedan izlazni port nije preopterećen, što može dovesti do nepoželjnog i nepotrebnog odbacivanja paketa.



Slika 9.9.2. Torusna arhitektura

Za ostvarivanje velikih protoka između mrežnih čvorova, kao i korisnika i mrežnih čvorova, koje su reda veličine desetine i stotine Gb/s, pa čak i Tb/s, neophodna je upotreba optičkih linkova. Pri tome, koristi se tehnika DWDM (*Dense Wavelength Division Multiplex*) koja koristi multipleksiranje signala po talasnim dužinama (WDM tehnika), a termin gust (*dense*) potiče od činjenice da se u multipleksu koristi nekoliko desetina, pa i stotina talasnih dužina (u komercijalnoj upotrebi su DWDM sistemi sa nekoliko desetina talasnih dužina, a veći sistemi su ili u fazi prototipova ili postignuti eksperimentalno u lab okruženju), čime se naglašava pomak u gustini multipleksiranja u odnosu na početne optičke WDM sisteme. Napomenimo da je WDM, u suštini, isto što i FDM (*Frequency Division Multiplex*), samo se koristi parametar talasne dužine umesto frekvencije nosioca signala.

Pošto se koriste optički linkovi za ostvarivanje velikih protoka u prenosu signala, cilj je da i mrežni čvorovi postanu optički, tj. da signal ostane u optičkom domenu i u njemu se procesira unutar mrežnog čvora. Razlog je što električni domen ima nižu brzinu u odnosu na optički domen, a isto tako u slučaju visokih protoka i velikog broja portova, broj linija unutar mrežnog čvora je ogroman, kao i energetska potrošnja čvora, pri čemu dolazi i do problema elektromagnetskih smetnji između linija što komplikuje izradu mrežnog čvora. Stoga postoji opasnost da komutacija u električnom domenu postane prepreka za dalje povećanje kapaciteta mrežnih čvorova i da postane prepreka u praćenju povećanja brzina linkova koji se ostvaruju u optičkom domenu. Međutim, postoje i veliki problemi u ostvarivanju cilja da mrežni čvorovi postanu potpuno optički. Neki od njih su:

- Problem baferisanja sadržaja paketa u optičkom domenu. Tehnike koje su trenutno u praksi na raspolaganju za formiranje bafera su kreiranje optičke linije za kašnjenje (u suštini, optičko vlakno) i optičke petlje (koja je isto optičko vlakno). Obe tehnike se zasnivaju na tome da se u njih pusti jedan ili više paketa da prođu kroz optičko vlakno u vidu linije ili petlje i dok se paket/paketi nalaze u njima oni su baferisani. Na izlazu iz linije za kašnjenje paket se mora ili proslediti dalje ili vratiti ponovo na liniju za kašnjenje ako se želi i dalje baferisati. U slučaju petlje, paket/paketi kruže po petlji dok ne dođe vreme za njihovo dalje prosleđivanje. Očigledno, struktura bafera je u optičkom domenu znatno manje funkcionalna nego u električnom domenu. Dužina linije/petlje određuje kapacitet bafera.
- Problem efikasne implementacije logičkih funkcija u optičkom domenu. Paketi zahtevaju intezivno procesiranje, na primer, lukap funkcija za određivanje izlaznog porta na koji paket treba da izađe ili provera ispravnosti paketa, pa je otuda bitno imati tehnološku mogućnost implementiranja složenih logičkih i aritmetičkih funkcija koje omogućavaju efikasno procesiranje paketa. Trenutno, električni domen omogućava efikasno kreiranje takvih složenih funkcija upotrebom ASIC ili FPGA čipova, dok optički domen nema još uvek tako napredne mogućnosti obrade.
- Problem segmentacije paketa promenjive dužine. Paketi promenjive dužine se koriste na Internetu tj. u okviru IP tehnologije. Kao što smo već ranije naveli, paketski komutatori znatno efikasnije rade ako komutiraju ćelije tj. pakete fiksne dužine. U električnom domenu nije problem izvršiti segmentaciju paketa promenjive dužine na ćelije fiksne dužine, ali u optičkom domenu to nije jednostavno izvesti.

Sa stanovišta prosleđivanja paketa u optičkom domenu kroz mrežu postoje tri moguća pristupa:

- Komutacija kola
- Komutacija paketa
- Komutacija burstova

Komutacija kola je trenutno najizvodljivija. Ona se zasniva na sledećem principu. U električnom domenu se izvrši zauzimanje resursa u optičkoj mreži (optičkom domenu) koje se zasniva na konfigurisanju mrežnih čvorova, na primer, tako da optički komutiraju proizvoljnu talasnu dužinu DWDM signala sa proizvoljnog ulaznog porta na istu talasnu dužinu DWDM signala proizvoljnog izlaznog porta. Napomenimo da je moguće raditi i konverzije talasne dužine, tako da uspostavljeno kolo na jednom linku koristi jednu talasnu dužinu, a na drugom neku drugu talasnu dužinu, čime se olakšava uspostava kola i smanjuje verovatnoća blokade usled nepostojanja slobodne i iste talasne dužine koja bi se koristila na čitavom putu kroz mrežu. Uglavnom se konverzija radi tako što se signal konvertuje u električni domen i potom prevede opet u optički domen, ali u drugu talasnu dužinu. Moguća je konverzija talasne dužine i direktno u optičkom domenu, ali to je još uvek prilično skuplje rešenje. Uspostava veze, tj. zauzimanje resursa se vrši u oba smera. Nakon toga se preko uspostavljenog puta razmenjuju paketi u optičkom domenu, a pošto se koristi komutacija kola tj. zauzeto kolo nema potrebe za baferisanjem ni za procesiranjem paketa u optičkom domenu. Naravno, mana je slabo iskorišćenje zauzetog optičkog puta što je posledica komutacije kola. Komutacija kola može biti statička tako što administrator mreže uspostavi određene putanje u skladu sa zahtevima korisnika i procenjenog trenda saobraćaja u mreži, a može biti i dinamička u smislu da se uspostavljaju i raskidaju kola po trenutnoj saobraćajnoj situaciji pri čemu ta dinamika ne može biti preterano brza zbog procesa uspostave kola.

Komutacija paketa podrazumeva da se u optičkom domenu obrađuje paket po paket. Ako uzmemo u obzir trenutne probleme koje smo naveli u postizanju cilja da mrežni čvorovi postanu potpuno optički, očigledno je da je trenutno tehnološki nemoguće efikasno i komercijalno isplativo kreirati potpuno optički mrežni čvor. Tipično se u praksi vrši na ulaznom portu konverzija paketa iz optičkog u električni domen, pa se vrši komutacija paketa u električnom domenu (pošto su u ovom domenu razvijena pouzdana i efikasna rešenja komutacije kao što smo videli u okviru ovog poglavlja), pa se na izlaznim portovima vrši ponovo konverzija paketa, ali iz električnog u optički domen. Drugi vid rešenja je da se samo zaglavlja paketa konvertuju u električni domen i procesiraju, a da se paket komutira u potpunosti u optičkom domenu bez konverzije (u električni domen), čime se izbegavaju u najvećem delu skupe konverzije čitavog paketa iz optičkog domena u električni i obrnuto. U ovom slučaju se koriste optička rešenja za baferisanje paketa, a za konfigurisanje optičkog komutatora se i dalje koristi električni domen jer je trenutno bolji zbog efikasnije implementacije kontrolne logike.

Komutacija burstova predstavlja svojevrsni kompromis između komutacije kola i paketa. Naime, na ivici optičkog domena, mrežni čvor sakuplja burstove paketa prema drugim mrežnim čvorovima na ivici dotičnog optičkog domena. Kada nakupi dovoljno veliki burst, vrši zauzimanje puta kroz optički domen (mrežu) i potom šalje burst po zauzetom putu čime se, kao kod komutacije kola, ne vrši procesiranje i baferisanje paketa iz bursta u okviru mrežnih čvorova koji se nalaze na zauzetom putu. Put se zauzima samo u jednom smeru čime je proces uspostave puta jednostavniji nego kod komutacije kola. Očigledno, ovaj princip unosi dodatno kašnjenje

usled procesa formiranja burstova, ali i zauzimanja puta, a takođe je neophodno razviti efikasne algoritme koji će proceniti kada burst treba da se pošalje. U slučaju postojanja velikog broja odredišta, memorijski zahtevi za realizaciju bafera, koji bi čuvali burstove koji se formiraju, mogu postati problematični jer bi broj istovremenih burstova koji se formira mogao postati prevelik.

Uređaji koji se koriste u optičkim mrežama i vrše ulogu komutacije su optički *add-drop* multiplekser (*OADM - Optical Add/Drop Multiplexer*) i OXC (*Optical Cross-Connect*). Optički *add-drop* multiplekser radi na sledeći način. U njega pristiže WDM ili DWDM multipleksiran signal. Vrši se demultipleksiranje signala na sastavne komponente tj. talasne dužine. Talasne dužine koje su stigle na odredište se odvode dalje (odvode se signali koje su prenosile dotične talasne dužine ili u električnom obliku ako se vrši konverzija u električni domen ili u originalnom obliku ako se i dalje koristi ista talasna dužina) ka korisniku ili na procesiranje. Potom se vrši multipleksiranje signala (koji nisu stigli na odredište) koji se provode dalje, zajedno sa novim signalima za koje dotični čvor predstavlja izvorište (ovi signali koriste talasne dužine koje su oslobodili signali koji su stigli na odredište, tj. ne mogu koristiti talasne dužine signala koji tranzitiraju kroz čvor). Ovako multipleksiran signal (WDM ili DWDM) se prosleđuje dalje u mrežu. Osim opisanog načina realizacije postoje i nešto drugačiji načini realizacije OADM uređaja, ali opisana funkcija je ista. OXC predstavlja komutator (mrežni čvor) koji vrši komutaciju signala sa ulaznih optičkih portova na odgovarajuće izlazne optičke portove. Pri tome razlikuju se električni OXC (koji se još naziva i hibridni) i potpuno optički OXC (koji se još naziva i transparentni OXC), gde u slučaju električnog OXC se vrši konverzija iz optičkog domena u električni i obrnuto radi obavljanja komutacije u električnom domenu, a u slučaju potpuno optičkog OXC nema konverzije već se komutacija vrši takođe u optičkom domenu. Pod terminom optički svič tj. optički komutator se tipično podrazumeva OXC funkcionalnost. Međutim, treba biti oprezan sa tim terminom jer postoje različita tumačenja ovog pojma. Na primer, proizvođači često tim terminom (optički svič) označavaju mrežne čvorove čiji su ulazni/izlazni portovi optički i gde mrežni čvor vrši komutaciju između dotičnih ulaznih/izlaznih portova. Ali, sam komutator u mrežnom čvoru može biti i električni i optički, tj. proizvođači i jedan i drugi slučaj ubrajaju u optički svič. U praksi je komutator unutar mrežnog čvora najčešće realizovan u električnom domenu, pa se vrši konverzija iz optičkog u električni domen na ulaznim portovima, odnosno konverzija iz električnog u optički domen na izlaznim portovima. S druge strane, neki autori pod terminom optički svič podrazumevaju potpuno optički svič koji vrši komutaciju paketa u optičkom domenu (uz procesiranje zaglavlja i konfigurisanje komutatora u električnom domenu), pri čemu se koristi i termin transparentni optički svič da bi se naglasilo da se paket komutira u potpunosti u optičkom domenu. U slučaju da se koriste konverzije iz optičkog u električni domen i obrnuto, ti autori koriste termin električni ili hibridni optički svič da bi označili da se komutacija vrši u električnom domenu. Takođe, napomenimo da se u literaturi i patentima opisuju i tzv. optoelektronski komutatori koji vrše optičku komutaciju (komutator radi u optičkom domenu), ali se procesiranje i baferisanje paketa vrši u električnom domenu, kao i konfigurisanje komutatora (sami portovi tj. linkovi mogu da rade u električnom ili optičkom domenu).

Optički komutatori predstavljaju oblast koja se intezivno istražuje. Trenutno postoje mnogobrojne tehnike optičke komutacije (akusto-optička tehnika, MEMS (*Micro-Electro-Mechanical Systems*) tehnika, tehnika tečnih kristala, i mnoge druge). Na primer, interesantna je MEMS tehnika koja se zasniva na upotrebi malih ogledala koja usmeravaju (komutiraju) svetlost sa ulaza komutatora na željene izlaze. 2D MEMS komutator je sličan po funkcionalnosti krosbar

komutatoru i ogledala imaju dve moguće pozicije koje odgovaraju *cross* i *bar* stanju. 3D MEMS komutator omogućava da ogledala menjaju svoj ugao proizvoljno pa se time sa manjim brojem ogledalaca (do 2*N*) postiže mogućnost komutacije sa potpunom dostupnošću. Međutim, postoje još mnogi problemi koji moraju da se reše da bi optički komutatori u potpunosti zaživeli i u praksi tj. komercijalnoj upotrebi i da zaista postignu efikasnost i performanse koje se od njih očekuju u budućnosti. Takođe, ostaje da se vidi koja tehnika ili tehnike u optičkoj komutaciji će postati dominantne u praksi s obzirom na veliki broj predloženih tehnika koje se i dalje usavršavaju.