УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Срђан Р. Дурковић

АРХИТЕКТУРА ПАКЕТСКОГ СВИЧА ЗА ЕФИКАСНО КОМУТИРАЊЕ УНИКАСТ И МУЛТИКАСТ САОБРАЋАЈА

докторска дисертација

Beograd, 2022.

UNIVERSITY OF BELGRADE
SCHOOL OF ELECTRICAL ENGINEERING

Srđan R. Durković

PACKET SWITCH ARCHITECTURE FOR EFFICIENT UNICAST AND MULTICAST TRAFFIC SWITCHING

Doctoral Dissertation

Belgrade, 2022.

ПОДАЦИ О МЕНТОРУ И ЧЛАНОВИМА КОМИСИЈЕ:

МЕНТОР:

др Зоран Чича, ванредни професор,
Универзитет у Београду, Електротехнички факултет

ЧЛАНОВИ КОМИСИЈЕ:

др Зоран Чича, ванредни професор,
Универзитет у Београду, Електротехнички факултет

др Милан Бјелица, редовни професор,
Универзитет у Београду, Електротехнички факултет

др Марија Малнар, ванредни професор,
Универзитет у Београду, Саобраћајни факултет

др Младен Копривица, доцент,
Универзитет у Београду, Електротехнички факултет

Датум усмене одбране:_____

ЗАХВАЛНИЦА

Овом приликом изражавам неизмјерну захвалност ментору др Зорану Чичи, ванредном професору на Електротехничком факултету у Београду за вишегодишњу несебичну пријатељску помоћ, савјете и усмјеравање приликом израде ове дисертације. Такође, зхвалност дугујем и члановима комисије чији су добронамјерни савјети допринијели да ова дисератција добије свој коначни изглед.

АРХИТЕКТУРА ПАКЕТСКОГ СВИЧА ЗА ЕФИКАСНО КОМУТИРАЊЕ УНИКАСТ И МУЛТИКАСТ САОБРАЋАЈА

САЖЕТАК:

У дисертацији је предложена једноставна архитектура свича као и алгоритми за ефикасно распоређивање и комутацију уникаст и мултикаст саобраћаја, што је од великог значаја за савремене телекомуникационе мреже у којима количина саобраћаја константно расте.

Први дио доприноса ове дисертације чини предлог рјешења свича за ефикасно управљање уникаст саобраћајем. Ово рјешење је развијено комбинујући најбоље особине постојећих рјешења, при том избјегавајући одређене њихове недостатке. Циљ је да се омогући што брже прослијеђивање пакета уз прихватљив ниво хардверске комплексности. Свич који је развијен у овој дисертацији представља комбинацију свичева са баферима на улазу и свичева који користе Биркхоф вон Нојман принцип детерминистичког конфигурисања комутационог модула па се не захтијева прорачун конфигурација комутатора. При томе, за разлику већине рјешења који користе Биркхоф вон Нојман принцип конфигурисања, у предложеном рјешењу могуће је користити само један физички комутациони модул који би обављао функције оба логичка комутациона модула. Да би се гарантовало да није дошло до поремећаја редослиједа пакета, предложен је и једноставан алгоритам за одабир пакета за слање. Такође, дат је и предлог унапријеђења подршке за фер сервис првобитно предложеног рјешења за комутацију уникаст саобраћаја.

У другом дијелу дисертације, пажња је посвећена унапријеђењу предложеног рјешења за ефикасно управљање и мултикаст саобраћајем. Потреба за овим се јавила као посљедица развоја нових сервиса (нпр. IPTV, онлајн игре итд.) који генеришу такав тип саобраћаја. Како је удио мултикаст саобраћаја у мрежи постао незанемарљив, перформансе свичева који су развијени примарно за уникаст саобраћај значајно опадају. Рјешење које је предложено у првом дијелу дисертације је унапређено додавањем модула који служи за управљање мултикаст саобраћајем. Овдје је идеја да се оптерећење са улазног порта који прима мултикаст пакете распореди на више портова који треба да приме те пакете. Овако је на релативно једноставан начин омогућено ефикасно управљање мултикаст саобраћајем.

У оквиру дисертације су урађене софтверске симулације које су показале да ова рјешења постижу врло добре перформансе у односу на постојећа. Такође, урађена је и хардверска имплементација предложеног основног уникаст рјешења која је показала релативно скромне захтијеве у погледу хардверских ресурса.

КЉУЧНЕ РЕЧИ: пакетска комутација, распоређивање пакета, фер сервис, мултикаст, интернет рутер, архитектура свича, саобраћајни сценарији

НАУЧНА ОБЛАСТ: Техничке науке - Електротехника и рачунарство

УЖА НАУЧНА ОБЛАСТ: Телекомуникације - комутациони системи

УДК БРОЈ: 621.3

PACKET SWITCH ARCHITECTURE FOR EFFICIENT UNICAST AND MULTICAST TRAFFIC SWITCHING

ABSTRACT:

The dissertation proposes a simple switch architecture as well as algorithms for efficient scheduling and switching of unicast and multicast traffic, which is of great importance for modern telecommunication networks because their traffic load is constantly and rapidly increasing.

The first part of the dissertation's contributions comprises a proposed switch which efficiently manages unicast traffic. The proposed switch is developed by using the best characteristics of the existing solutions while avoiding some of their drawbacks. The aim is to enable fast packet forwarding while achieving an acceptable level of hardware complexity. The proposed solution combines architecture with buffers at input ports and Birkhoff-von Neumann architecture based on deterministic switch module configurations. Hence, calculation of switch module configurations is not needed. Also, folded architecture is possible, which means that only one physical switching module is used for both switching stages of Birkhoff-von Neumann architecture. A simple algorithm for packet scheduling has been developed in order to avoid packet out-of-sequence problems. Finally, fair service support improvement is introduced for the originally proposed switch solution.

The second part of the dissertation is devoted to the enhancement of the proposed unicast switch for efficient management of multicast traffic. The need for multicast support has emerged as a consequence of the development and introduction of new services (such as IPTV, online gaming, etc.) that generate multicast traffic. As the amount of multicast traffic is not negligible anymore, the performance of packet switches that were primarily developed for the unicast traffic is significantly degraded. The solution proposed in the first part of the diseration is enhanced with the module used for multicast traffic management. Here, the idea is that the multicast load at some input port is distributed over ports that are also destination for the multicast packets. This approach enables relatively simple but efficient management of multicast traffic.

In this dissertation, software simulations were conducted, which confirmed that proposed solutions achieve very good performances compared to existing solutons. Furthermore, hardware implementation of the proposed basic unicast switch solution shows modest requirements in terms of needed hardware resources.

KEYWORDS: packet switching, packet scheduling, fair service, multicast, internet router, switch architecture, traffic scenarios

SCIENTIFIC FIELD: Technical sciences - Electrical and computer engineering

SCIENTIFIC SUBFIELD: Telecommunications - switching systems

UDK NUMBER: 621.3

САДРЖАЈ

1.Увод

У последњих неколико деценија Интернет је на револуционаран начин промијенио дотадашњи начин живота. Првобитна идеја америчке агенције за развојне пројекте је била да се направи телекомуникациона мрежа која би повезивала неколико војних база. Међутим, прва веза је остварена између неколико америчких универзитета. Убрзо су научници који су радили на развоју овог пројекта уочили могућности његове примјене и у цивилне сврхе, чиме је започела историја савременог Интернета. Ипак, како су у том периоду рачунари били врло масивни и скупи, Интернет веза је постојала само између државних установа или приватних корпорација које су имале сопствене рачунаре. Убрзани развој електронике је довео до развоја персоналних рачунара који су постали доступни већем броју људи. То је уједно и прекретница у развоју Интернета. Наиме, од мреже која је повезивала мали број институција број прикључака је почео нагло да расте мјерећи се милонима или стотинама милиона корисника. Да би све функционисало, било је неопходно развити и одговарајуће протоколе за комуникцију као и мрежне уређаје који ће омогућити ефикасну комуникацију између великог броја различитих корисника.

У првој фази, пренос података се махом вршио преко постојеће телефонске инфраструктуре, тј. упредене бакарне парице. Бежична комуникација је врло ријетко коришћена и то обично за повезивање удаљених корисника гдје није било исплативо и ефикасно градити жичану инфраструктуру. Касније су и коаксијални каблови, који су инцијално коришћени за диситрибуцију телевизијског сигнала постали медијум за пренос података преко Интернета. Осим линкова за пренос података, кључну улогу у раду Интернета имају мрежни уређаји чији је задатак да успоставе везу између рачунара који размјењују информације. У телефонским мрежама, ову улогу су имале централе које су радиле на принципу комутације кола. Принцип рада ових централа је подразумијевао да се између два корисника успостави веза и да им се додијеле ресурси који су на располагању само њима. Ово омогућава висок квалитет сервиса, али с друге стране утиче на неефикасно коришћење ресурса јер се размјена информација врши само повремено. На тај начин, ресурси које би могли бити на располагању другим корисницима остају неискоришћени. Зато се на Интернету користи принцип комутације пакета. Овдје се подаци прије слања дијеле на пакете који се затим независно прослеђују кроз мрежу, док се не пријему врши њихова реконструкција. Ово омогућава знатно ефикасније коришћење мрежних ресурса.

Овакав начин комутације је захтијевао и развој нових мрежних уређаја који га подржавају, тј. рутера и свичева који су у себи садржали пакетске комутаторе. Наиме, у телефонским централама се конфигурисање врши само приликом успостављања везе између два корисника. У пакетским мрежама, конфигурисање пакетских комутатора у мрежним уређајима се врши у сваком временском слоту (што је једнако дужини трајања пакета) што је знатно чешће него у телефонским мрежама. Из овога разлога, кључни утицај на перформансе мреже имају управо ови уређаји. У пакетским мрежама може доћи до ситуације да у мрежни уређај истовремено стигне више пакета које треба прослиједити на исти излазни порт. Да не би дошло до губитка пакета, неопходно је да постоје бафери гдје би се они привремено смјестили док не буду прослијеђени на одговарајући излазни порт. У првој фази Интернета,

развијени су комутатори који су имали бафере на излазним портовима. Ово рјешење подразумијева да се сви пакети одмах прослијеђују на одговарајуће излазне портове. Ако је на неки излазни порт стигло више пакета, они се привремено смјештају у бафер. Ово омогућава 100% пропусност, као и добру подршку за квалитет сервиса јер се на излазном порту може имплементирати логика за одабир пакета за слање у мрежу. Међутим, кључни недостатак овог рјешења је што се захтијева да бафер мора да у једном временском слоту изврши до *N* уписа, тј. да ради *N* пута брже од линкова, гдје је *N* број портова. Како је брзина линкова стално расла, ово рјешење је постало неефикасно.

У сљедећој фази, фокус истраживања је био на комутаторима који користе бафере на улазним портовима. У овом случају од бафера се захтијева да у сваком временском слоту изврше један упис и једно читање, што је знатно повољније него код комутатора са баферима на излазним портовима. Међутим, овај тип комутатора захтијева да се у сваком слоту врши прорачун конфигурације комутатора, што је комплексан задатак. Предложен је велики број алгоритама који су задужени за прорачун конфигурација. Такође, предложена су и рјешења која комбинују бафере и на улазним и на излазним портовима, па чак и у самим комутаторима.

Развој технологије је омогућио да корисници приступају Интеренету не само преко персоналних рачунара већ и преко лаптопова и мобилних телефона. Истовремено, појавио се велики број апликација и сервиса који су постали дио свакодневнице што је све заједно довело до експоненцијалног раста саобраћаја на Интеренту. Постављање каблова са оптичким влакнима је обезбиједио огроман капацитет, па су уско грло постали мрежни уређаји од којих се захтијевало да раде знатно брже. Да би се избјегао проблем прорачуна конфигурација, предложено је рјешење које користи детерминистичке конфигурације. Оно се заснива на теореми Биркхофа и вон Нојмана која омогућава да се за познати шаблон саобраћаја унапријед изврши прорачун конфигурације комутатора. Овај комутатор се састоји од два комутациона модула, при чему је први задужен за равномјерну дистрибуцију саобраћаја по свим портовима другог комутационог модула који затим прослеђује пакете на одгварајуће излазне портове. Међутим, уочено је да овакав начин рада доводи до поремећаја редослиједа пакета, што је у *IP* (*Internet Protocol*) мрежама недопустиво јер би изазавало велики број ретрансмисија пакета и самим тим до деградације перформанси мреже. Зато је предложен велики број рјешења са циљем да се овај проблем избјегне или ријеши.

Један од доприноса ове дисертације јесте да се нађе рјешење које би комбиновало најбоље особине постојећих рјешења, а да се при том избјегну њихови недостаци. Идеја је да предложено рјешење користи бафере на улазним портовима као и архитектуру комутатора који су развијени на бази Биркхоф вон Нојманове теореме. На овај начин је избјегнута потреба да се у сваком слоту врши прорачун конфигурација, што је ограничавало скалабилност комутатора са баферима на улазним портовима. С друге стране, иако се користе детерминистичке конфигурације, проблем поремећаја редослиједа пакета је ријешен коришћењем бафера на улазним портовима као и развојем алгоритма за одабир пакета за слање. Алтернативно рјешење не користи овај алгоритам већ ресеквенционе бафере на излазним портовима за исправљање поремећаја у редоследу, чија је величина ограничена на само *N* пакета, гдје је *N* број портова комутатора. Сва остала рјешења која користе ресеквенционе бафере захтијевају знатно веће димензије бафера.

Битан аспект у раду пакетских комутатора је и понашање током повремених загушења, односно преоптерећења излазних портова. Тада је пожељно да токови који пролазе кроз загушени излазни порт добију фер третман. Многа рјешења не остварују добре

перформансе у сценаријима загушења што утиче и на рад саме мреже, али и задовољство корисника. У оквиру ове дисертације је предложено унапријеђење фер сервис подршке уникаст рјешења предложеног у овој дисертацији. Предложено унапријеђење остварује веома добру фер сервис подршку што је потврђено кроз неколико испитиваних сценарија загушења излазних портова.

Коначно, у посљедње вријеме је дошло до миграције бројних сервиса на Интернет, као што су нпр. телевизија, онлјан играње и слично. За ове типове сервиса је карактеристично да се исти садржај дистрибуира ка већем броју корисника. Овакав тип саобраћаја се назива мултикаст саобраћај. Јасно је да би слање више копија једног истог садржаја било неефикасно, па је идеја да се шаље само једна копија до одговарајућих мрежних уређаја, који затим прослеђују више копија до крајњих корисника. Како је количина овог саобраћаја у мрежи постала незанемарљива, перформансе уређаја који су развијени за управљање уникаст саобраћајем су лошије. Зато је пажња истраживача усмјерена на унапређења ових комутатора да би могли да ефикасно управљају и уникаст и мултикаст саобраћајем. Већина постојећих рјешења користи комутаторе са баферима на улазним портовима, при чему се већина тих рјешења заснива на додавању посебног бафера гдје би се смјештали мултикаст пакети.

Један од доприноса дисератције је унапријеђење комутатора који је развијен за уникаст саобраћај, како би на ефикасан начин управљао и мултикаст саобраћајем. Наиме, развијен је модул који се заснива на распоређивању мултикаст саобраћаја на више портова који су укључени у тај саобраћај. На овај начин су задржане предности оригиналног рјешења уз додатак који је релативно једноставан.

У оквиру дисертације урађене су софтверске симулације ради упоређивања перформанси предложених рјешења са постојећим. Резултати су показали да ова рјешења постижу врло добре перформансе за различите саобраћајне сценарије, и димензије комутатора што потврђује добру скалабилност предложених рјешења. Такође, урађена је и хардверска имплементатција предложеног уникаст рјешења која је потврдила да су захтијеви у погледу хардверских ресурса на прихватљивом нивоу.

Остатак дисертације је организован на сљедећи начин. У другом поглављу су изложене основе комутације пакета, идентификовани кључни изазови при дизајнирању пакетских комутатора и представљени су досадашњи приступи у развоју пакетских комутатора. У трећем поглављу су представљена најпопуларнија постојећа рјешења уникаст комутатора. У четвртом поглављу су представљена рјешења до којих се дошло током рада на овој дисертацији, при чему је на крају дато и поређење њихових перформанси. У петом поглављу је извршено поређење перформанси коначно изабраног рјешења до ког се дошло у оквиру истраживачког рада на овој дисертацији са најбољим постојећим рјешењима. У шестом поглављу је урађена процјена хардверске комплексности предложеног рјешења за уникаст саобраћај. У седмом поглављу је изложено унапријеђење фер сервис подршке за уникаст рјешење предложено у овој дисертацији. У осмом погављу су представљени комутатори који су развијени за управљање мултикаст саобраћајем. У деветом поглављу је представљено унапријеђење рјешења које је у оквиру дисертације развијено за уникаст саобраћај за ефикасно управљање мултикаст саобраћајем. У десетом поглављу је извршено поређење перформанси предложеног мултикаст рјешења са другим, постојећим мултикаст рјешењима. Коначно у једанаестом поглављу је дат закључак, након које слиједи списак коришћене литературе, списак скраћеница, списак слика и списак табела.

2. ОСНОВИ КОМУТАЦИЈЕ ПАКЕТА

Развој Интернета је започео 1960-их година као пројекат америчке агенције за истраживачке пројекте. У првој фази, Интернет, тада под називом Арпанет, је био намијењен за војне сврхе, тј. за повезивање војних база. Међутим, прва Интернет веза је успостављена између четири рачунара који су били смјештени у америчким универзитетима: Калифорнијски универзитет у Лос Анђелесу, Универзитет Стенфорд, Калифорнијски универзитет у Санта Барбари и Универзитет у Јути [1]. Веза је остварена помоћу модема, а капацитет мреже је био 50 kb/s. До тада се комуникација на даљину заснивала на телефону и телеграфу који су користили комутацију кола као технологију за пренос сигнала. Технологија комутације кола захтијева успоставу директне везе (кола) између два корисника да би они могли међусобно да комуницирају. Док траје комуникација између два корисника, одговарајући ресурси су резервисани само за њих и самим тим су недоступни за остале кориснике.

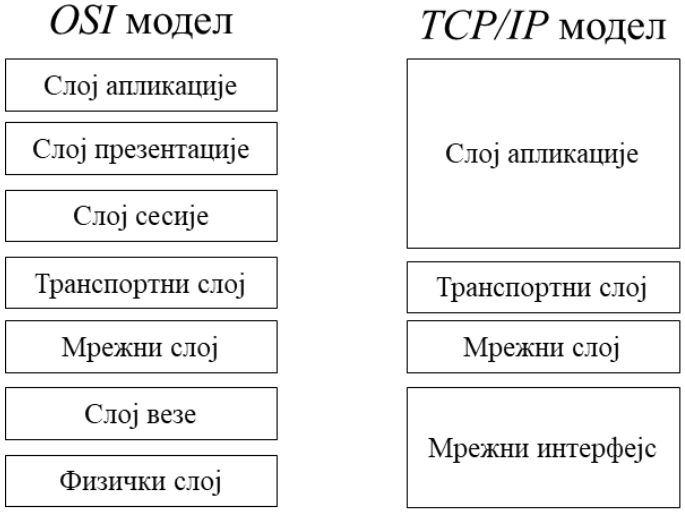
Интернет је унио револуцију у технологију преноса информација, јер је умјесто комутације кола заснован на комутацији пакета. Ову идеју је у свом раду први изложио Леонард Клајнрок, професор на Универзитету MIT, 1961. године [2]. Комутација пакета подразумијева да се подаци прије слања сегментирају на пакете, који се затим независно преносе кроз мрежу. За разлику од комутације кола, у мрежама заснованим на комутацији пакета се није вршило успостављање везе и резервација ресурса између пошиљаоца и примаоца, већ су се пакети података независно прослеђивали кроз мрежу. Касније су развијене технологије попут *MPLS* (*Multi Protocol Label Switching*), које и у пакетским мрежама омогућавају да се за неке кориснике и/или сервисе резервишу одређени ресурси.

У класичним телефонским мрежама заснованим на комутацији кола кључну улогу играју телефонске централе које омогућавају успостављање везе. У пакетским мрежама ову улогу имају свичеви (за размјену пакета у оквиру једне мреже) и рутери (за размјену пакета између више мрежа). Сваки пакет података осим корисног садржаја садржи и одговарајуће заглавље. У заглављу пакета се налазе информације које рутерима и свичевима омогућавају да уз помоћ одговарајућих протокола рутирања прослиједе пакете података до једног или више примаоца. На пријему се врши реконструкција послате информације која је била подијељена у више пакета.

Пренос података комутацијом пакета омогућава знатно ефикасније коришћење мрежних ресурса. Као што је већ речено, у случају комутације кола у току успостављања везе се ресурси резервишу само за два корисника. Ово доводи до слабе искоришћености ресурса нарочито у случају комуникације у којој постоје честе паузе у слању података што је карактеристично за рачунарску комуникацију где је саобраћај спорадичне (*bursty*) природе. У случају пауза у комуникацији резервисани ресурси су неискоришћени иако можда постоје други корисници који би могли у том моменту да искористе те ресурсе. У случају комутације пакета нема резервације ресурса (за већину сервиса) тако да корисници заузимају само онолико капацитета колико им је заиста неопходно за комуникацију. Прецизније, ресурси су искоришћени само док траје слање пакета, када нема пакета који се преносе (пауза у комуникацији) ти ресурси су слободни за друге кориснике. Овај принцип својеврсног

мултиплексирања веза са становишта коришћења ресурса омогућава висок степен искоришћења ресурса мреже што је главна предност комутације пакета у односу на комутацију кола. Наравно, постоје и мане које се огледају у чињеници да може доћи и до загушења у мрежи услед чега може доћи до губитака пакета, већих кашњења и варијације кашњења. Додатно, да би се смањили губици пакета у мрежи, неопходни су бафери у мрежним чворовима. Такође, мрежни чворови морају вршити процесирање на нивоу пакета, а не на нивоу веза као код комутације кола што изискује већу процесорску моћ мрежних чворова у комутацији пакета. Међутим, све ове мане су оправдане кад се сагледа добитак који се остварује у повећању искоришћења мрежних ресурса.

Архитектура Интернета је први пут дефинисана 1983. године када је међународна организација за стандардизацију (*ISO*) усвојила *OSI* (*Open System Interconnection*) модел како би се стандардизовала комуникација на Интернету са циљем да се омогући комуникација између разних мрежа као и коришћење опреме различитих произвођача. Овај модел дефинише архитектуру Интернета у 7 слојева, као што је показано на слици 2.1. Међутим, овај модел се показао као превише комплексан за практичну примјену због претјераног броја заглавља која би се морала користити па је Интернет усвојио де факто архитектуру која је названа *TCP/IP* архитектура по два протокола по којима је Интернет препознатљив. Код овог модела број слојева је мањи као што се види са слике 2.1. Тиме се решава проблем великог броја слојева код *OSI* модела. Наиме, код *TCP/IP* модела дио слоја сесије из *OSI* модела је интегрисан у транспортни слој *TCP/IP* модела, док се прва два слоја *OSI* модела у *TCP/IP* моделу посматрају као један (мада могу и одвојено). Разлог је што ова два слоја дефинишу конкретну технологију док су мрежни и виши слојеви у највећој мјери технолошки независни.



Слика 2.1. Архитектура *OSI* и *TCP/IP* модела

Током 1990-их година је дошло до развоја различитих протокола апликационог слоја као што су: *HTTP* (*Hypertext Transfer Protocol*), *FTP* (*File Transfer Protocol*), *SMTP* (*Simple Mail Transfer Protocol*) итд. Сви ови протоколи су омогућили развој различитих апликација које су захваљујући већој доступности рачунара постали врло популарни код великог броја људи. Све ово је утицало на готово експоненцијалан раст саобраћаја на Интернету. Такође, последњих година су постали популарни сервиси попут *IPTV*, онлајн игара, мултикаст сервиса, сервиса у облаку итд. Како мултимедијални садржаји који имају велике захтјеве у погледу протока чине велики удио саобраћаја у мрежи, неопходно је сталним унапређењем

инфраструктуре пратити непрестани тренд раста количине података која се размењује. Изградња инфраструктуре са оптичким влакнима је омогућила пренос огромних количина података тако да су мрежни уређаји постали уско грло. Разлог је што ови уређаји треба да у кратком временском периоду испитају заглавље пакета, затим да изврше претрагу табеле прослеђивања да би се одредило на који излазни порт треба проследити пакет и коначно да проследе пакет на одговарајући излазни порт. Рутери на окосници савремених мрежа би требало да подржавају велики број портова велике брзине. На пример, тренутно су на тржишту доступни уређаји који подржавају протоке ~400 Gb/s [3-5]. Узимајући у обзир пакете величине 64 B (минимална величина етернет оквира који садржи *IP* пакет), портови би требало да обраде и проследе пакете на излазне портове у року краћем од 1 ns. На основу овога јасно се уочава колико је изазовно конструисати рутер великог капацитета и приступачне цијене који би могао да подржи захтјеве савремених мрежа [6].

2.1. Архитектура рутера

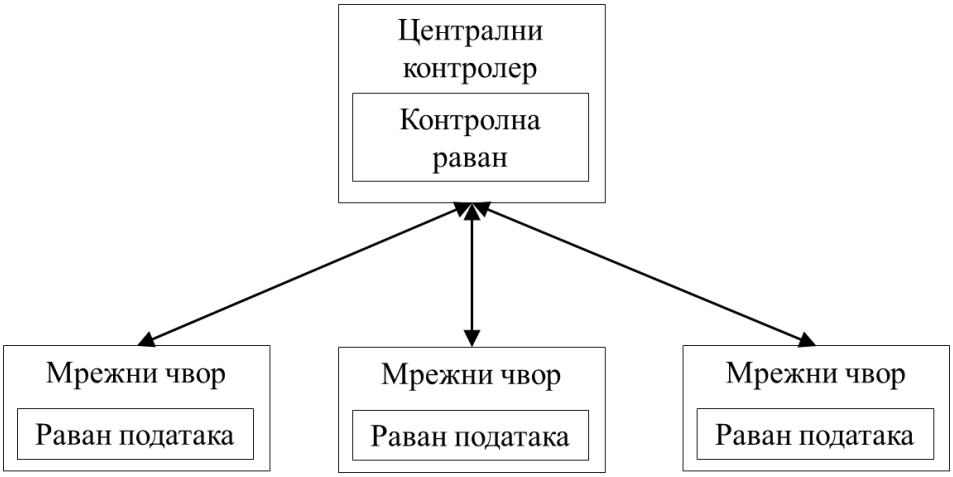
У интернет мрежи која је главни представник мрежа заснованих на комутацији пакета, рутери су окосница мреже, односно омогућавају глобалну повезаност. Рутери треба да обаве велики број функција да би се ефикасно процесирали и прослеђивали пакети. Генерално, све функције рутера се могу подијелити у двије категорије: функције равни података и функције контролне равни [7]. Ова подјела је извршена на основу врсте послова које рутер треба да изврши.

Функције равни података обухватају испитивање исправности пакета, одлучивање о одбацивању пакета у случају загушења, одабир пакета за слање, одређивање на који излазни порт треба проследити сваки пакет који пролази кроз мрежни чвор као и комутацију пакета унутар мрежног чвора. Када *IP* пакет дође у рутер прво се испитује његова исправност, декрементира вриједност *time-to-live* поља и врши се прорачун контролне суме заглавља пакета. Затим се на основу одредишне адресе ради претрага табеле прослеђивања рутера. Ако постоји више кандидата у табели прослеђивања, онда се за одређивање излазног порта користи правило најдужег преклапања префикса. Такође, у рутерима се могу извршавати и неке додатне функције као што је подршка за квалитет сервиса или филтрирање саобраћаја. На примјер, на основу информација из заглавља, пакетима се може дати виши или нижи приоритет приликом одабира пакета за слање. Неке апликације попут *Voice over IP* су осетљиве на кашњење па се пакетима који припадају тој врсти саобраћаја даје предност приликом прослеђивања. Осим тога, на рутерима се може инсталирати *firewall* апликација која служи за одбацивање и пропуштање саобраћаја по одређеним критеријумима које дефинише администратор мреже. Тада се обично врши тзв. дубока инспекција пакета (*deep packet inspection*) где се испитују и заглавље транспортног слоја, а неретко и апликационог слоја [8-12]. Након што се заврши са обрадом пакета, врши се његово прослеђивање на излазни порт која такође представља велики изазов и чиме се и бави ова теза. Пошто је све ове функције неопходно извршавати у врло кратком времену, оне се ради бржег извршавања углавном имплементирају хардверски, мада се на портовима мањих брзина могу имплементирати и софтверски пошто су на њима временска ограничења лабавија.

Функције контролне равни обухватају извршавање протокола рутирања, протокола за резервацију ресурса, протокола за надгледање мреже, размјену информација из табела рутирања, ажурирање табела прослеђивања рутера, удаљени приступ администратора, командни интерфејс, аутентификацију итд. Такође, типично је да се резултати рада контролне равни примјењују на све портове рутера. Ове функције се извршавају периодично и проток контролних порука које се размјењују је много мањи од протока корисничких

пакета. Како брзина извршавања ових функција није ограничавајући фактор оне се имплементирају софтверски на процесору опште намјене. Софтверска имплементација омогућава много већу флексибилност у раду, јер је на овај начин много лакше одржавати, додавати и модификовати функције које рутер треба да изврши. На основу резултата рада контролне равни врши се конфигурисање равни података.

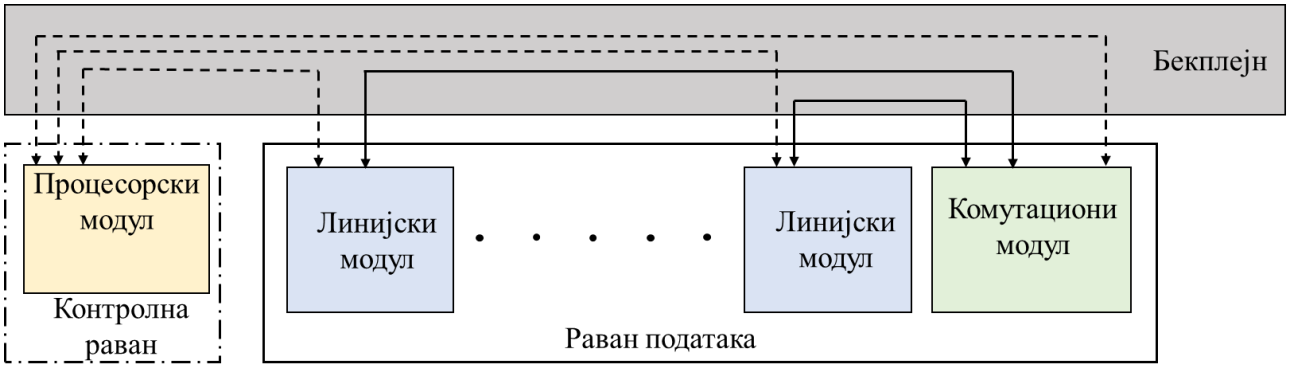
У традиционалним мрежама функције контролне равни су, као што је претходно описано, имплементиране у самим мрежним уређајима. Међутим, у претходној деценији је предложен и развијен концепт софтверски дефинисаних мрежа (*SDN - Software Defined Networking*) који се све више примјењује у пракси [13-16]. Идеја је да се функције контролне равни имплементирају на једном уређају тј. централном контролеру, који затим преко мреже комуницира са мрежним уређајима и врши њихову конфигурацију. Централни контролер може бити хардверски уређај или виртуелна машина. Дакле, сада се на мрежним уређајима имплементирају само функције равни података. Архитектура овакве мреже је дата на слици 2.1.1. Предност оваквог концепта је што се сада цијела мрежа може конфигурисати са једног уређаја, умјесто да се сваки мрежни уређај посебно конфигурише. Ово омогућава бољу, бржу и флексибилнију контролу мреже, лакше управљање мрежом, оптимизацију перформанси мреже. Централизовано управљање мрежом може уједно представљати и проблем, јер уколико дође до проблема у раду централног контролера, цијела мрежа ће имати проблем у раду.



Слика 2.1.1. Архитектура *SDN* мреже

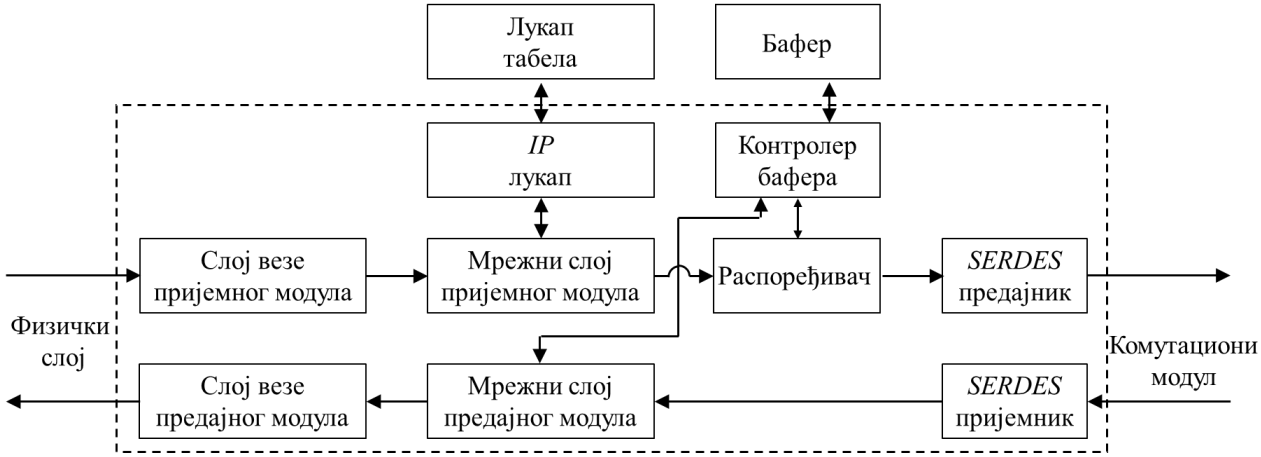
На слици 2.1.2 је дата типична архитектура рутера, који се састоји од сљедећих дијелова [17]:

- Процесорски модул
- Линијски модули
- Комутациони модул
- Бекплејн



Слика 2.1.2. Принципска архитектура мрежног чвора пакетске мреже

Линијски модули и комутациони модул припадају равни података, док процесорски модул припада контролној равни. Комуникација између процесорског модула и линијског модула је типично остварена преко бекплејна или кроз мрежу у случају *SDN* мрежа. Процесорски модул се састоји од процесора на коме се инсталира оперативни систем мрежног чвора у оквиру кога се извршавају функције контролне равни. Мрежни чворови типично садрже само један процесорски модул, али их може бити и више ако је то неопходно. Процесорски модул преко контролних линија (или мреже у случају *SDN* мрежа) конфигурише рад линијских модула и комутатора. Процесорски модул се хардверски прави као картица која се повезује на бекплејн, али суштински је у питању рачунар [6]. У *SDN* мрежама процесорски модул преко мреже комуницира са мрежним уређајима и врши њихово конфигурисање. Такође, концепт виртуелизације мрежних функција често иде у пару са *SDN* приступом. У овом концепту се мрежне функције реализују софтверски употребом виртуелних рачунара, чиме се постиже флексибилна реализација контролне равни која може да се имплементира и у облак окружењу. На примјер, 5G технологија користи овај концепт и укључен је у одговарајуће стандарде 5G технологије [18-22].



Слика 2.1.3. Принципска архитектура линијског модула

На слици 2.1.3 је представљена принципска архитектура линијског модула описана у [23]. Линијски модули су улазна/излазна тачка пакета у/из рутера. Линијски модул се састоји од неколико мањих модула, при чему сваки од њих врши одређени скуп функција. Примљени оквири прво стижу на слој везе пријемног модула. Овај модул обрађује примљени оквир (провјера *CRC* контролне суме, екстракција *IP* пакета). У случају да је оквир исправан, *IP* пакет се прослеђује мрежном слоју пријемног модула који извршава два скупа задатака.

Прво, испитује се *IP* заглавље, декрементира се *time-to-live* поље, ажурира се контролна сума *IP* заглавља и *IP* адреса дестинације се шаље *IP* лукап модулу. Затим, уколико је пакет исправан врши се сегментација на ћелије фиксне дужине. Ћелије су привремено бафероване док чекају резултат *IP* лукап модула који показује на који излазни порт треба прослиједити ове ћелије. *IP* лукап модул за дату *IP* адресу дестинације претражује лукап табелу да би одредио на који излазни порт ћелије треба да буду прослијеђене [24-28]. Лукап табела садржи информације о свим познатим мрежним дестинацијама и обично се налази у екстерној, брзој *SRAM* меморији. Након пријема резултата *IP* лукап модула, мрежни слој пријемног модула прослеђује ћелије у распоређивач, заједно са информацијом о излазном порту на који их треба прослиједити. Распоређивач одређује када ће ћелије кроз комутациони модул бити прослијеђене на одговарајући излазни порт. Ћелије су смјештене у бафер док чекају да буду одабране за слање. Како бафер мора бити у стању да прими велику количину ћелија он је обично имплементиран у екстерним *DRAM* меморијама које имају велики капацитет [29-32]. Контролер бафера је одговоран за ишчитавање и упис ћелија у бафер. Ћелије које су одабране за слање се прослеђују ка комутационом модулу преко брзих серијских линкова, тако да је неопходна употреба *SERDES* (*Serializer*/*Deserializer*) модула. *SERDES* предајни модул врши паралела-у-серија конверзију када се ћелије шаљу кроз комутациони модул, док *SERDES* пријемни модул врши обрнуту конверзију када од комутационог модула прими ћелију. Ћелије примљене од комутационог модула су прослијеђене мрежном слоју излазног линијског модула који их смјешта у бафер. Када су све ћелије једног пакета примљене, пакет се реконструише у мрежном слоју излазног модула. Реконструисани *IP* пакет се прослеђује мрежном слоју излазног модула који додаје заглавље мрежног слоја и формирани оквир прослеђује физичком слоју ради даљег слања у мрежу.

Дакле, лукап табела и бафер заправо нису модули који врше обраду пакета. Наиме, лукап табела је табела у којој се налазе подаци о мрежној топологији које је рутер прикупио од протокола рутирања из своје контролне равни. *IP* лукап модул приступа лукап табели да би извршио свој задатак. Бафер на улазном порту служи за смјештање ћелија које чекају да буду одабране за слање, док бафери на излазном порту служе за смјештање ћелија који чекају реконструкцију одговарајућег *IP* пакета. Лукап табела и бафери су типично смјештени у екстерним меморијама [23].

Комутациони модул врши прослеђивање пакета са улазних на излазне портове у складу са конфигурацијом која дефинише који парови улаз-излаз су повезани. У тези ћемо комутациони модул заједно са распоређивачем који дефинише које ћелије се прослеђују кроз њега означавати термином пакетски комутатор јер они здружено представљају функцију пакетске комутације. У случају да на улазни порт стигне контролни пакет који је намијењен самом мрежном чвору, комутациони модул их прослеђује процесорском модулу. У *SDN* мрежама пакети који се не могу класификовати се прослеђују централном конролеру тј. на одговарајући излазни порт (преко кога рутер комуницира са *SDN* контролером). Такође, када процесорски модул генерише неки контролни пакет он се преко комутационог модула прослеђује на одговарајуће излазне портове [6]. Функције равни података се извршавају у сваком временском слоту, гдје временски слот представља вријеме трајања јединице података фиксне дужине (ћелија). Отуда се и обавља раније поменута функција сегментације пакета у линијским модулима рутера [33]. Ако претпоставимо минималну величину Етернет оквира од 64 B у који је енкапсулиран *IP* пакет и брзину портова од 400 Gb/s, обраду и прослеђивање пакета је неопходно извршити за свега ~1.25 ns. Узимајући у обзир да рутери могу имати и велик број портова, проблем имплементације пакетских комутатора постаје још изазовнији јер је неопходно ефикасно бирати и распоређивати пакете за комутацију што није

лако ако се узме да има N^2 потенцијалних пакета између којих се бира N пакета посматрано из угла комплетног рутера [34,35]. Додатно, мултикаст пакети и гаранције квалитета појединим токовима повећавају сложеност проблема дизајна пакетског комутатора. Дакле, јасно је да је неопходно да пакетски комутатор ради веома брзо и ефикасно. Тема ове дисертације јесу управо пакетски комутатори, односно предлог рјешења за унапређење њихових перформанси и то пре свега у аспекту дела који дефинише одабир пакета за слање, а не реализацију самог комутационог модула. У сљедећем поглављу су представљени кључни захтјеви које пакетски комутатори морају да задовоље.

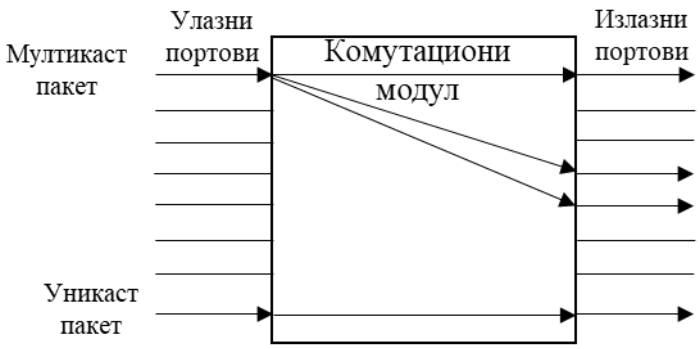
2.2. Кључне особине пакетских комутатора

Перформансе интернет мрежа, као и квалитет сервиса крајњих корисника у великој мјери зависи од перформанси мрежних уређаја. Перформансе мрежних уређаја највише зависе од начина рада пакетског комутатора. Зато је од великог значаја да пакетски комутатори што ефикасније обављају свој задатак. Неке кључне особине пакетских комутатора од којих зависе њихове перформансе су дате у наставку:

- **Пропусност**. Пакетски комутатори би требало да буду у стању да прослеђују велики број пакета у јединици времена. Пропусност пакетског комутатора одређује његову ефикасност, односно капацитет [6]. Пропусност се рачуна претпостављајући да су сви улазни портови 100% оптерећени (у сваком временском слоту стиже по један пакет на сваки од улазних портова) и да су сви излазни портови 100% оптерећени (пристигли пакети су равномјерно распоређени по свим излазним портовима, тако да су сви 100% опетерећени). Укупан број пакета пристиглих на све излазне портове подијељен са укупним бројем пакета пристиглих на све улазне портове представља пропусност пакетског комутатора. Пропусност је позитивна вриједност не већа од 1. Што је већа пропусност, то је боља ефикасност и перформансе пакетског комутатора.
- **Скалабилност**. Скалабилност одређује колико је лако проширивати капацитет пакетског комутатора у смислу броја подржаних портова као и њихове брзине (што је лакше проширивати капацитет пакетског комутатора, то је он скалабилнији). Раст саобраћаја захтијева од провајдера да на сваких 3-5 година унапређују своју опрему. Рутери који су дизајнирани тако да се релативно лако и јефтино могу унаприједити у погледу капацитета су пожељнији за оператере [36].
- **Убрзање**. Убрзање s значи да пакетски комутатор ради на s пута већој брзини од улазних/излазних линкова [6]. У овом случају могуће је остварити већи број (s>1) конфигурација пакетског комутатора у току једног временског слота, а самим тим и прослиједити већи број пакета.
- **Цијена и комплексност**. Пакетски комутатор би требало да има што једноставнију имплементацију, али и управљање. Ово даље има директан утицај на цијену уређаја [6].
- **Тип пакета који се прослеђује**. Величина пакета може бити фиксна или промењива, зависно од технологије која се користи. На примјер, *ATM* (*Asynchronous Transfer Mode*) мреже раде са пакетима фиксне дужине, док *IP* мреже раде са пакетима променљиве дужине [6]. Како се у пракси показало да пакетски комутатори много ефикасније раде, и да су знатно једноставнији за конфигурисање када су пакети фиксне дужине, типично се у интернет рутерима имплементира додатни модул који на улазним портовима врши сегментацију

пакета на ћелије фиксне дужине. Такође, на излазним портовима се имплементира модул који врши ресегментацију ћелија у пакете.

- **Уникаст и мултикаст**. Већина мрежних конекција су уникаст типа, гдје се комуникација одвија између нека два крајња уређаја, тј. корисника. Дакле, код оваквог типа саобраћаја, пакете који долазе на неки улазни порт треба прослиједити на само један излазни порт. Међутим, у последње вријеме велику популарност су стекле апликације попут гледања видео садржаја на захтјев, телевизије преко интернета, учења на даљину, онлајн играња игара, аудио/видео конференција, *data broadcasting*-а итд. гдје се информације из једног извора шаљу до више корисника. Овај вид саобраћаја се зове мултикаст саобраћај. Да би се ефикасно подржао овакав саобраћај у мрежи неопходно је да се у рутерима имплементира механизам који омогућава слање истих пакета с једног улазног порта на више излазних портова. Једна могућност за управљањем овом врстом саобраћаја је да се на улазном порту направи одговарајући број копија мултикаст пакета и да се они потом третирају као уникаст пакети. Међутим, на овај начин се проток пакета значајно повећава јер сада добијамо ситуацију као да је дошло више пакета на истом улазном порту чиме значајно може бити прекорачено максимално ограничење од 100% оптерећења улазног порта које важи за уникаст саобраћај. Самим тим, ово рјешење није ефикасно иако је једноставно за реализацију. Други приступ је употреба комутационог модула који подржава мултикаст комутацију, односно спајање улаза са више излаза у истом моменту. Примјер је дат на слици 2.2.1. Али, мултикаст комутација пакета усложњава конфигурисање комутационог модула. У поглављу 8 ће бити ријечи о предложеним рјешењима за ефикасно управљање мултикаст саобраћајем, а то је и један од проблема који се покушава ријешити у оквиру ове дисертације.



Слика 2.2.1. Мултикаст прослеђивање

- **Бафери.** Интернет саобраћај је по природи спорадичан (*bursty*) па у одређеним моментима долази до преоптерећења излазних портова. Ово се дешава када укупна количина пристиглих пакета намијењених неком одређеном излазном порту привремено превазилази капацитет дотичног излазног порта, односно линка који је повезан на тај порт. На примјер, може се десити да у истом временском слоту на два улазна порта стигну пакети које је потребно прослиједити на исти излазни порт. Како излазни порт у току једног временског слота не може да прими и пошаље више од једног пакета, неопходно је имплементирати бафере у којима ће се смјестити пакети који не могу да буду тренутно прослијеђени. Бафери могу бити

смјештени на излазним портовима, на улазним портовима или у самом комутатору о чему ће бити речи касније у тези [6].

2.3. Изазови при дизајнирању пакетског комутатора

Са порастом Интернет саобраћаја од рутера се захтијева да подржавају капацитете и до неколико стотина Tb/s. Кључни изазови које треба ријешити приликом дизајнирања рутера јесу обрада пакета (нпр. класификација пакета, претрага табеле прослеђивања, инспекција и ажурирање заглавља пакета), баферовање и распоређивање (*scheduling*) пакета [36]. Како се брзина линкова повећава сразмјерно се скраћује вријеме за које треба извршити ове операције. Подршка мултикаст саобраћају и квалитет сервиса додатно подижу сложеност претходно наведених изазова. У наставку су укратко описани кључни фактори који ограничавају перформансе пакетских комутатора:

- **Брзина рада меморије**. Брзина рада линкова данас је већ 400 Gb/s. Самим тим и меморије на портовима морају да подржавају ове брзине рада. Меморијске технологије које тренутно имају најбоље перформансе су:
  - *HBM* (*High Bandwidth Memory*) која подржава брзине до 819 Gb/s [37,38].
  - *RLDRAM* (*Reduced Latency Dynamic Random-Access Memory*) која подржава брзине 76.8 Gb/s [39, 40].
  - *QDR SRAM* (*Quad Data Rate Static Random-Access Memory*) која подржава брзине 35 Gb/s [41].
- **Распоређивач**. Задатак распоређивача је избор пакета на улазним портовима који ће се прослиједити на излазне портове. Циљ распоређивања пакета јесте унапређење перформанси пакетског комутатора, односно његове пропусности. Са порастом брзине линкова, неопходно је одабир пакета извршити у врло кратком времену. Примјера ради, ако узмемо у обзир брзину линка од 400 Gb/s, минималну величину *IP* пакета од 40 B, и интерно убрзање 2, распоређивач мора да у року од 0.4 ns одабере пакет који ће бити прослијеђен на излазни порт. Такође, што је већи број портова повећава се сложеност конфигурисања свича јер распоређивач мора да донесе одлуку за већи број пакета, а скуп из кога бира пакете за прослеђивање је такође већи. Уколико се користи централизована архитектура за распоређивач онда се комуникација одвија између њега и свих улазних портова, што може бити комплексно за имплементацију, због потребе повезивања распоређивача са свим портовима (*N* конекција, где је *N* број портова). С друге стране, код дистрибуиране архитектуре сви портови учествују у процесу одабира пакета што поједностављује имплементацију јер је распоређивач дистрибуиран по портовима [36]. Овакав принцип рада може довести до смањења пропусности пакетског комутатора и повећања просјечног кашњења пакета услед недостатка информација о стању на осталим портовима. Зато се типично користи интерно убрзање да би се побољшале перформансе.
- ***QoS* (*Quality of Service*) контрола**. У пакетским мрежама се врши пренос различитих врста саобраћаја, који имају различите захтјеве. На примјер, неке врсте саобраћаја (попут говора, *streaming* аудио и видео садржаја) су осетљиве на кашњење, док је *TCP* саобраћај осетљив на губитак пакета. Да би се задовољили захтјеви различитих класа саобраћаја у погледу квалитета сервиса неопходно је пажљиво распоређивати пакете што може бити врло захтијевно услед различитих захтијева различитих класа саобраћаја [36]. Неколико алгоритама за управљање и

одабир пакета подразумијева временско означавање пакета и њихов одабир на основу ове ознаке. Проналажење пакета с најмањом временском ознаком у року од неколико ns може направити уско грло у систему. Такође, одабрати пакет који би требало послати или одбацити, у року од ~1 ns није једноставно. Примјери алгоритама за одабир пакета су: *Max-Min* одабирање [42], *WRR* (*Weighted Round-Robin*) сервис [43], *Stop and Go* [44], *HRR* (*Hierarchical Round-Robin*) [45,46], *EDD* (*Earliest Due Date*) [47-49], *DRR* (*Deficit Round-Robin*) [50,51], *GPS* (*Generalized Processor Sharing*) [52-54], *WFQ* (*Weigthed Fair Queuing*) [55], *SCFQ* (*Self-Clocked Fair Queuing*) [56,57], *WF2Q* (*Worst-Case Fair Weighted Fair Queuing*) [58], *WF2Q+* [59,60], *fEDF* (*flit-based Earliest Deadline First*) [61]. Постоје различите метрике за оцјену ових алгоритама као што су: границе кашњења, остваривo искоришћење мреже, фер сервис, заглавље које уноси протокол, рачунарска комплексност, робусност итд. Осим одабира пакета, неопходно је имплементирати и механизме за управљање баферима, како би се у спрези са *TCP* протоколом обезбиједио квалитетан сервис у свим околностима. Типично правило је да се чува саобраћај који стигне у току 100 ms [39]. Ако је брзина рада линка 400 Gb/s, то значи да је неопходно да бафер има капацитет од приближно 4.7 GB за један порт. Неки од алгоритама за управљање баферима су: *Tail Drop* [62,63], *Drop on Full* [64,65], *Random Early Detection* [66-68], *Fair Random Early Detection* [69,70], *Stabilized Random Early Detection* [70,71], *Effective Random Early Detection* [72], *Choose nad Keep for responsive flows, Choose and Kill for unresponsive flows* [73], *Comparison-Based Buffer Management in QoS Switches* [74].

- **Потрошња енергије**. Велики број портова који подржавају велике брзине рада утичу и на потрошњу енергије, која може износити и до 1000 W по једном реку [75-76]. Такође, неопходно је обезбиједити и климатизацију просторије у којој се налази рутер. Све то заједно утиче да се мора водити рачуна и о енергетској ефикасности.

2.4. Стратегије баферовања

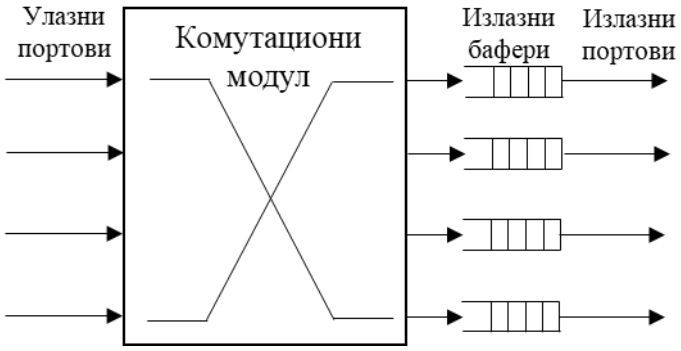
Као што је претходно већ речено, услед спорадичне природе интернет саобраћаја може се десити загушење излазних портова и/или унутрашњих линкова. Ово се дешава, на примјер, када у току једног временског слота, пристигне више пакета које треба прослиједити на исти излазни порт. Како сваки излазни порт може у току једног временског слота да пошаље само један пакет, неопходно је остале пакете који се услед загушења не могу одмах прослиједити привремено смјестити. Ови пакети се смјештају у одговарајуће бафере, који могу бити имплементирани на различитим позицијама у оквиру пакетског комутатора. Најчешће се користе сљедеће варијанте [6]:

- Бафери на излазним портовима
- Заједничка меморија
- Бафери на улазним портовима
- Бафери и на улазним и на излазним портовима
- Бафери унутар комутатора

Свака од ових варијанти има одређене предности и мане. У наставку ће бити представљене свака од њих.

2.4.1. Баферовање на излазним портовима

Код пакетских комутатора који користе бафере на излазним портовима пакети се одмах прослеђују на излазне портове. Зато је неопходно користити бафере на излазним портовима. Архитектура овог комутатора је дата на слици 2.4.1. У најгорем случају, може се десити да је на исти излазни порт потребно прослиједити *N* пакета. Да би то било могуће, неопходно је да комутатор ради са унутрашњим убрзањем *N*, што у суштини значи да комутатор у току једног слота може да пошаље *N* пакета на један излазни порт. Недостатак код оваквог приступа је што је технолошки веома тешко реализовати комутатор који би могао толико брзо да ради за велики број портова велике брзине чиме је уочљива главна мана овог рјешења - слаба скалабилност [6].



Слика 2.4.1. Пакетски комутатор са баферима на излазним портовима

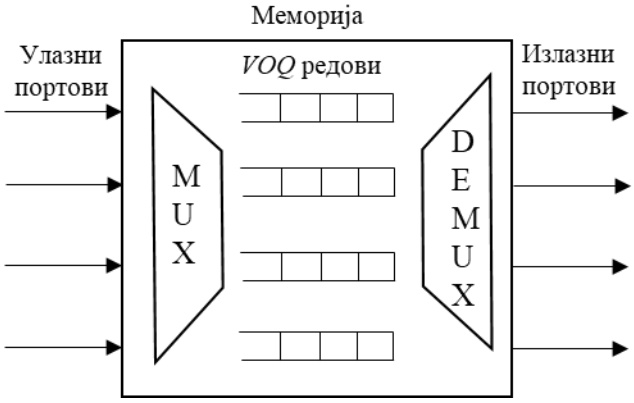
Постоји неколико начина да се реализују бафери на излазним портовима. Један је употреба *First-In-First-Out* (*FIFO*) бафера за сваки ток посебно, гдје ток представља све пакете који се шаљу с једног улазног порта на један излазни порт. То значи, да на сваком излазном порту постоји *N* бафера, што је неекономично и нескалабилно. Када се има у виду да у оквиру истог тока, постоје пакети различитих класа односно приоритета онда број бафера на једном излазном порту може бити и већи. При том, искоришћеност бафера је генерално мала, јер неће сви бафери бити истовремено попуњени. Друга опција је да се на излазима користи један бафер у оквиру кога постоји *N* логичких редова за чекање тј. *Virtual Output Queue* (*VOQ*) редова, по један за сваки ток на том излазном порту. Наравно, и овде се број редова за чекање повећава ако су праве засебни редови за класе саобраћаја. У овом случају, постоји само једна меморија, али меморија мора бити у стању да у току једног слота изврши *N* уписа и једно читање, што је опет врло проблематично када су брзине линкова или број портова велики што опет доводи до слабе скалабилности. Искоришћеност меморије је знатно боља, јер сви токови дијеле исту меморију, па се меморијски простор динамички дијели у складу са тренутним захтијевима. Још једна предност је што је сада потребно укупно *N* оваквих меморија, односно једна по излазном порту што је знатно мање него у претходном случају.

Коришћење бафера на излазним портовима омогућава лакшу *QoS* (*Quality of Service*) контролу, јер су пакети увијек доступни на излазним портовима. На тај начин, пакети могу да буду одабрани на основу њиховог приоритета, алоцираних ресурса и *QoS* захтјева. Као што је већ речено, кључни недостатак је слаба скалабилност оваквог рјешења.

2.4.2. Заједничка меморија

Архитектура комутатора са заједничком меморијом је приказана на слици 2.4.2. Овдје се иста меморија користи за смјештање пакета са свих улазних портова. Такође, и сви

излазни портови читају пакете из исте меморије. Меморија се састоји од *N VOQ* редова, гдје сваки *VOQ* ред одговара једном излазном порту. Такође, ако се користе и неки механизми за фер опслуживање или квалитет сервиса, број *VOQ* редова може бити и већи. На овај начин искоришћеност меморије је максимизована, јер се ресурси динамички дијеле између свих портова у складу са тренутним потребама [6]. Предложена су различита рјешења овог типа као што су: *Linked List* [77], *Content Addressable Memory* [78], *Space-Time-Space* [79], *Washington University Gigabit Switch* [80], *Concentartor-Based Growable Switch* [81], *Parallel Shared-Memory Switch* [82], *Shared Memory Buffer Management for Heterogeneous Packet Processing* [83] . У новије вријеме ова архитектура постаје популарна за рутере у тзв. *on-chip* мрежама (*NoC – Network on Chip*) [84-86].

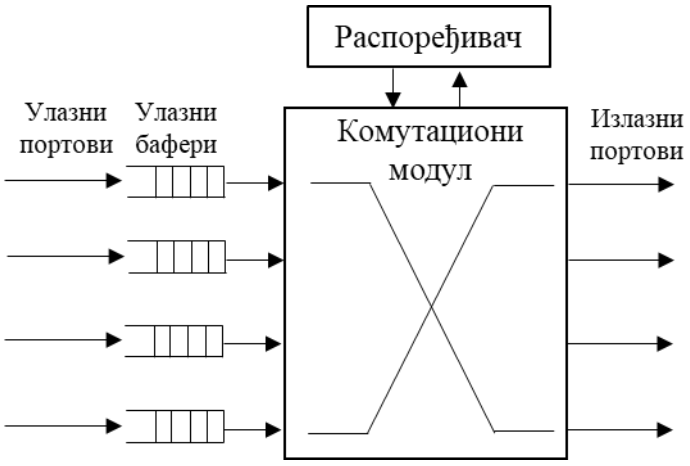


Слика 2.4.2. Пакетски комутатор са заједничком меморијом

Међутим, главни недостатак је што је у једном временском слоту неопходно истовремено извршити *N* уписа/исписа, што значајно ограничава скалабилност комутатора.

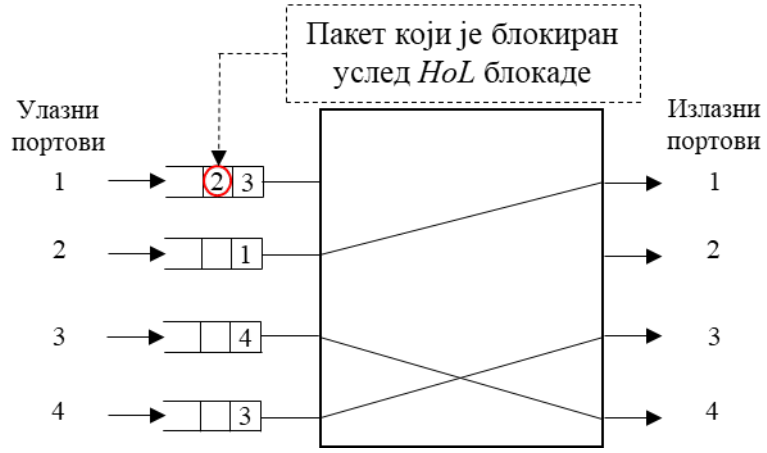
2.4.3. Баферовање на улазним портовима

Како су комутатори са баферима на излазним портовима ограничени бројем портова које могу подржати, алтернатива су комутатори са баферима на улазним портовима. Као што се види са слике 2.4.3 на сваком улазном порту постоји по један бафер. Код овог типа комутатора, неопходно је постојање распоређивача који ће вршити одабир пакета за прослеђивање, односно упаривање улазних и излазних портова. На основу рада овог распоређивача, врши се конфигурисање комутационог модула [6].



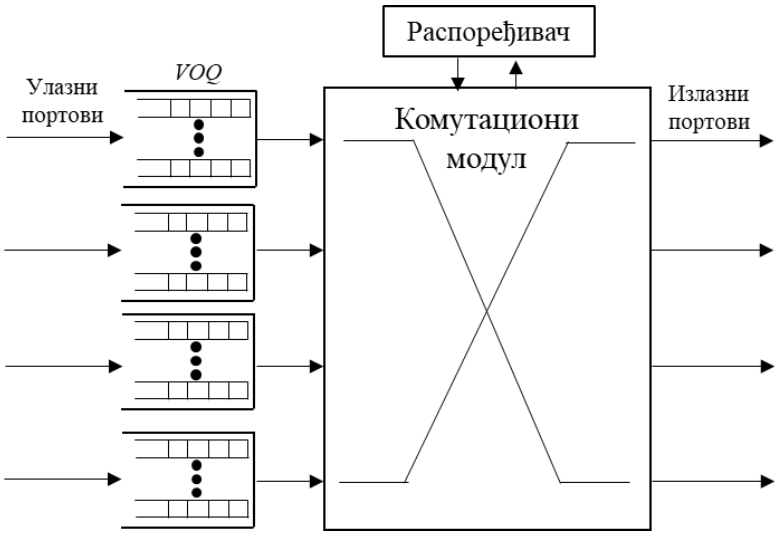
Слика 2.4.3. Пакетски комутатор са баферима на улазним портовима

Када се користи интерно убрзање *s*, тада распоређивач израчунава конфигурацију комутационог модула *s* пута у току једног временског слота. Бафери на улазним портовима треба да буду у стању да изврше један упис, односно један испис у току једног слота. Уколико се користи убрзање *s*, онда је неопходно да меморија може да подржи до *s* ишчитавања у једном слоту. Међутим, како је за већину алгоритама које користе распоређивачи утврђено да са убрзањем 2 постижу веома добре перформансе, онда је технолошки изводљиво да меморија подржи такво убрзање.



Слика 2.4.4. *HoL* блокада

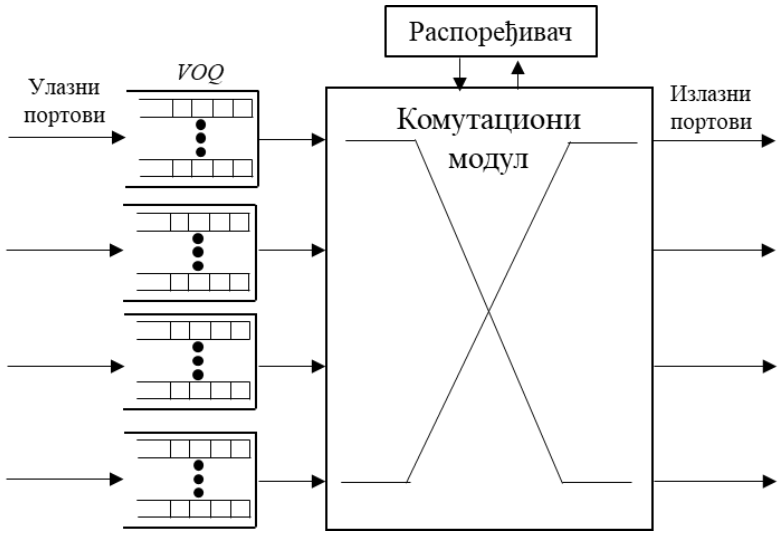
Уколико се на улазима користе *FIFO* бафери, долази до проблема *Head-of-Line* (*HoL*) блокаде. У примјеру са слике 2.4.4, улазни порт 1 не може да пошаље свој *HoL* пакет на излазни порт 3, јер је одабран пакет са улазног порта 4. Међутим, пакет који се налази иза *HoL* пакета на улазном порту 1 не може да буде прослијеђен на излазни порт 2, иако овај порт може да прими пакет. Овај проблем озбиљно утиче на пропусност комутатора. При униформном саобраћају, доказано је да је пропусност ограничена на 58.6% [87].



Слика 2.4.5. *VOQ* редови на улазним портовима

Алтернатива је коришћење *VOQ* редова, гдје на сваком улазном порту постоји *N VOQ* редова, по један за сваки излазни порт (слика 2.4.5). Међутим, у овом случају су и алгоритми за конфигурисање компликованији јер у случају *FIFO* бафера алгоритам ради са *N* података (челни пакети *FIFO* бафера), док у случају *VOQ* редова алгоритам ради са N^2 података (челни

пакети свих *VOQ* редова). Алгоритми за конфигурисање у овом случају се типично дијеле на *maximum* и *maximal* алгоритме, али ће више детаља о њима бити дато у поглављу 3.1.

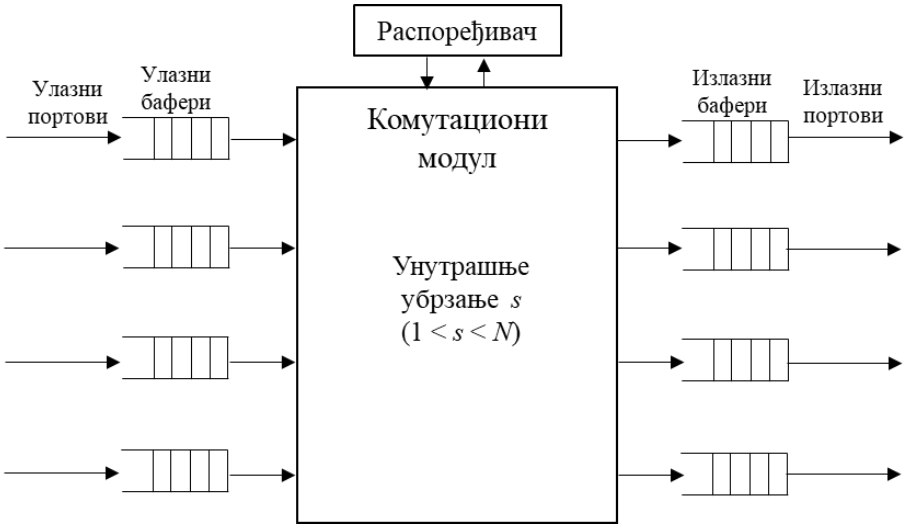


Слика 2.4.5. *VOQ* редови на улазним портовима

Главни недостатак овог типа комутатора је, што у веома кратком периоду трајања једног слота (~1 ns) треба извршити алгоритме за конфигурисање комутационог модула што може бити веома изазовно. Уколико се користи интерно убрзање *s*, онда је ово вријеме *s* пута краће. За ублажавање овог проблема се користи *pipeline* техника, када се алгоритам извршава за будуће слотове, један за другим. Иако прорачун траје више слотова, пошто се извршава више прорачуна у паралели, постиже се да се резултати прорачуна конфигурација комутационог модула ређају један за другим у потребном темпу. Још један недостатак овог типа комутатора је слабија подршка за квалитет сервиса у односу на комутаторе са баферима на излазним портовима. Такође, треба напоменути још једну битну ствар. Када комутатори са баферима на улазим портовима користе интерно убрзање, онда је неопходно имплементирати и бафере на излазним портовима, јер се може десити да неки излазни порт у једном слоту прими више пакета, а може да пошаље само један. Може се онда рећи да ови комутатори уствари припадају комутаторима са баферима и на улазним и на излазним портовима, о којима ће више ријечи бити у сљедећем одељку.

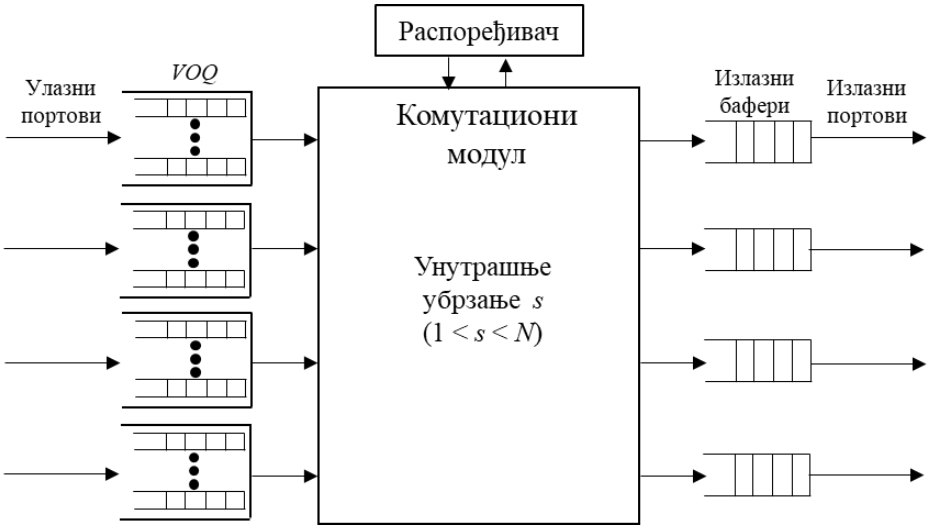
2.4.4. Комбиновано бафероване на улазним и излазним портовима

Архитектура комутатора са баферима и на улазним и на излазним портовима је дата на слици 2.4.6. Овај тип комутатора је развијен да би се ријешили проблеми које имају комутатори са баферима само на улазним или само на излазним портовима. Ови комутатори раде са унутрашњим убрзањем *s*, при чему је $1<s<N$, тј. *s* је довољно мало да је технолошки могуће реализовати комутатор [6]. На овај начин је могуће у току једног слота прослиједити до *s* пакета на један излазни порт, док се остали баферују на улазним портовима. На овај начин се постижу перформансе сличне оним које имају комутатори са баферима на излазним портовима. У првој фази развоја ових комутатора су се користили *FIFO* бафери. Доказано је да овакав комутатор достиже 99% пропусност, ако се користи убрзање 4 [88,89].



Слика 2.4.6. Пакетски комутатор са баферима на улазним и излазним портовима

Унапријеђено рјешење користи *VOQ* редове на улазним портовима и *FIFO* бафере на излазним портовима [91,92]. Архитектура овог комутатора је дата на слици 2.4.7. За ово рјешење је доказано да постиже исте перформансе као комутатори са баферима на излазним постовима, ако се користи убрзање 2 [92].

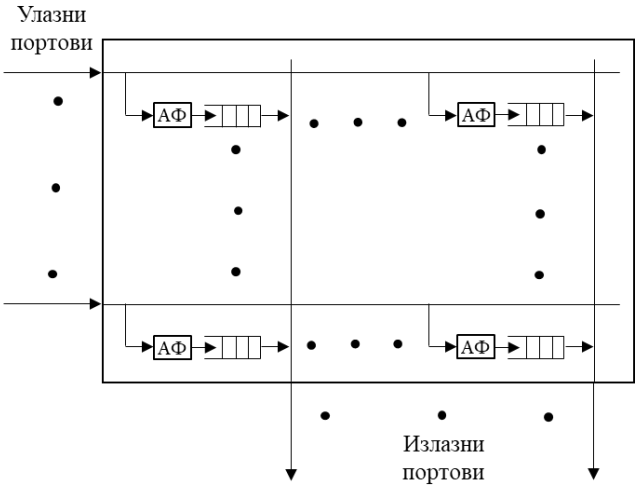


Слика 2.4.7. Пакетски комутатор са баферима са VOQ редовима на улазним портовима и баферима на излазним портовима

2.4.5. Баферовање унутар комутатора

У претходним одељцима је показано да су перформансе пакетских комутатора ограничене или брзином рада меморија или брзином извршавања алгоритама за конфигурисање комутационог модула. Комутатори са баферима на излазним портовима не захтијевају конфигурисање, већ се сви пристигли пакети одмах прослеђују на одговарајући излаз. Међутим, ови комутатори захтијевају коришћење веома брзих меморија, које треба да раде N пута брже од линкова. С друге стране, комутатори са баферима на улазним портовима користе меморије које раде на нивоу брзине линкова. Недостатак овог рјешења је што је у сваком временском слоту неопходно извршавање комплексних алгоритама за конфигурисање комутационог модула. Као компромисно рјешење су предложени

комутатори код којих су бафери смјештени у унутрашњости комутатора, као што је показано на слици 2.4.8. Пакети се прво смјештају у ове бафере (који се налазе у тачки укрштања хоризонталне линије која одговара улазном порту и вертикалне линије која одговара излазном порту) прије него буду прослијеђени на одговарајуће излазне портове. Дакле, ако је више пакета намијењено за исти излазни порт, они ће бити уписани у одговарајуће бафере, а затим распоређивач на сваком излазном порту бира један пакет из елемената који се налазе у одговарајућој колони. Главни недостатак код овог рјешења је потреба за коришћењем N^2 бафера, који су типично врло мале величине [6].



Слика 2.4.8. Кросбар комутатор са баферима у укрсним тачкама

Да би се избјегао губитак пакета, обично се користе и бафери на улазним портовима који су већег капацитета [93-96]. Сви ови бафери у току једног слота треба да изврше највише један упис, односно читање, што значи да не постоји проблем брзине рада меморија. Одабир пакета зависи од алгоритма који је имплементиран. Предожено је више рјешења: *Oldest Cell First* [97], *Longest Queue First* [98], *Most Critical Buffer First* [99]. Међутим, како је број бафера N^2, овакав тип комутатора је нескалабилан.

3. УНИКАСТ ПАКЕТСКИ КОМУТАТОРИ

У овом поглављу ће бити дат преглед постојећих рјешења уникаст пакетских комутатора. Узимајући у обзир дуг период развоја пакетских комутатора, а који је и даље у току, број постојећих рјешења је велик, али многа рјешења су превазиђена у смислу максималних перформанси које могу да остваре попут рјешења заснованих на архитектури са баферима на излазним портовима. Стога рјешења представљена у овом поглављу су одабрана на основу перформанси које постижу (пропусност комутатора као и средње кашњење пакета у комутатору), али и хардверске комплексности. Наравно, свако од представљених рјешења има своје предности и мане о којима ће такође бити ријечи.

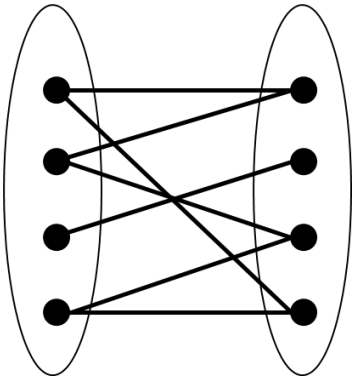
У раној фази развоја Интернета, капацитет пакетских комутатора је био реда величине до неколико стотина Mb/s. Имајући ово у виду, фокус истраживања је био доминантно на рјешењима заснованим на архитектури са баферима на излазним портовима јер су омогућавали добре у перформансе у смислу пропусности и кашњења пакета, али и фер опслуживања свих токова. Међутим, са изразитим повећањем саобраћаја у мрежи, порасли су и захтијеви које пакетски комутатори треба да подрже, како у смислу броја портова тако и њихове брзине. Пошто пакетски комутатори са баферима на излазним портовима нису могли да подрже брзине веће од неколико Gb/s по порту, фокус истраживања се промијенио на пакетске комутаторе са баферима на улазу који имају добру скалабилност. Ови комутатори захтијевају употребу распоређивача који у сваком временском слоту треба да изврши конфигурисање комутационог модула што је врло изазовно за имплементацију. Једно рјешење за овај проблем је коришћење пајплајн (*pipeline*) технике која омогућава да се прорачун извршава у току више временских слотова. Међутим, и даље остаје проблем што су прорачунате конфигурације по природи случајне, што ограничава перформансе и скалабилност овог типа комутатора. Предложени су различити алгоритми који служе за конфигурисање комутатора и они најпопуларнији ће бити описани у оквиру првог потпоглавља. Пошто извршавање ових алгоритама у сваком временском слоту може ограничити скалабилност комутатора са баферима на улазу, као алтернатива, предложен је *Load-Balanced Birkhoff-von Neumann* (*LB-BvN*) пакетски комутатор, који се садржи два комутациона модула, али не захтијева постојање распоређивача који би вршио њихово конфигурисање. Наиме, код овог рјешења конфигурације оба комутациона модула су детерминистичке, што је врло важно при великим брзинама линкова где се захтјева велика фреквенција конфигурисања комутационог модула. Такође, није потребно да се информације о прорачунатој конфигурацији преносе до самог комутационог модула да би се извршило његово конфигурисање. Кључни недостатак овог рјешења је поремећај редоследа пакета, што је проблематично у мрежама гдје се користи *TCP* протокол. Предложено је више рјешења овог проблема, о чему ће бити ријечи у другом потпоглављу.

У наставку рада се подразумијева да пакетски комутатори имају N улазних и N излазних портова. Такође, сматра се да сви комутатори раде са пакетима фиксне дужине, и да је трајање једног временског слота одређено дужином пакета. Под циклусом се сматра период од N узастопних временских слотова, осим ако није назначено другачије. Коначно,

пакети који долазе на улазни порт i и које треба прослиједити на излазни порт k припадају истом току пакета, и тај ток се означава са $F(i,k)$.

3.1. Пакетски комутатори са баферима на улазним портовима

О овом типу пакетских комутатора је већ било ријечи у одељку 2.4.3, гдје су објашњене њихове најважније карактеристике. Код ових комутатора, када пакет стигне на улазни порт смјешта се у одговарајући бафер на том порту, и затим чека да буде прослијеђен на одговарајући излазни порт. У наставку се разматрају комутатори који на улазним портовима користе *VOQ* редове за чекање. Ови комутатори имају распоређивач који у сваком временском слоту, одређује који улазни портови ће послати пакете и на које излазне портове. Ово уједно одређује и конфигурацију комутационог модула [36]. У наставку су описани најпознатији алгоритми који се користе за упаривање улазних и излазних портова. Иако код ових комутатора не постоји проблем брзине рада меморија који је постојао код пакетских комутатора са баферима на излазним портовима, главни недостатак је ограничена скалабилност, јер је изазовно извршити конфигурисање комутационог модула у току временског слота који траје ~1 ns.



Слика 3.1.1. Примјер упаривања бипартитног графа

Процес конфигурисања се може представити као проблем упаривања бипартитног графа. У математици се бипартитни граф дефинише као граф који се састоји од два дисјунктна скупа чворова (нпр. U и V), при чему нема пара чворова који припадају истом скупу, а да су суседни [100]. Ако једна група чворова представља улазне портове, а друга излазне портове, онда је циљ да се изврши њихово адекватно међусобно упаривање, тј. успостављање конекција између одговарајућих улазних и излазних портова. На слици 3.1.1, чворови са лијеве стране представљају улазне портове, док чворови с десне стране представљају излазне портове. Ивица која повезује улазни и излазни порт (тј. чвор с лијеве и чвор с десне стране) значи да постоји пакет за слање с тог улазног порта на тај излазни порт. Наравно, сваки улазни порт може имати пакете за слање на више различитих излазних портова, као што и сваки излазни порт може добити захтјеве са више улазних портова. Циљ упаривања је да се одабере скуп ивица од укупно N^2 могућих ивица, и то тако да сваки улазни порт буде повезан са највише једим излазним портом, као и да сваки излазни порт буде повезан са највише једним улазним портом. Алгоритми за решавање овог проблема, треба да задовоље неколико услова [36]:

- **Ефикасност.** Алгоритам треба да обезбиједи високу пропусност комутатора и мало кашњење пакета, тј. да се одабере скуп који има највише ивица у сваком временском слоту.

- **Фер сервис.** Алгоритам би требало на фер начин да опслужује све *VOQ* редове.
- **Стабилност.** Очекивани број пакета у сваком *VOQ* реду би требало да остане коначан за сваки дозвољен шаблон саобраћаја.
- **Комплексност имплементације.** Алгоритам би требало да има што мању хардверску комплексност. Повећање комплексности утиче на продужење периода упаривања што опет ограничава брзину рада распоређивача, а тиме и капацитет пакетског комутатора.

Предложено је више алгоритама за адекватно упаривање улазних и излазних портова. У суштини сви ови алгоритми се могу подјелити у двије групе: максимум упаривање и максимално упаривање. Даље, у оквиру ове двије групе постоји неколико типова алгоритама. У наредним одељцима ће бити представљени принципи рада ових алгоритама и нека рјешења која су базирана на њима која постижу добре перформансе уз прихватљиву комплексност.

3.1.1. Алгоритми максимум упаривања

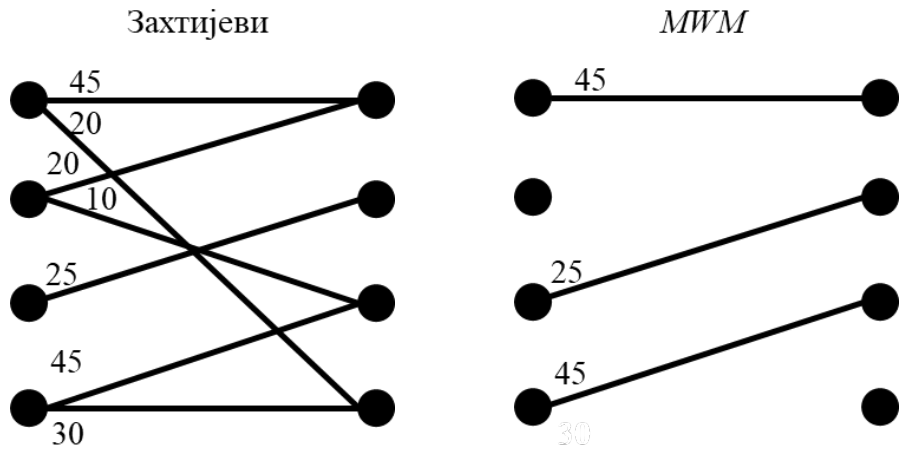
Класа максимум упаривања алгоритама омогућава налажење највећег могућег броја упарених улазних и излазних портова. То значи да се улазни и излазни портови не могу упарити ни на један другачији начин, а да се добије више упарених улазних и излазних портова него што је добијено на основу овог алгоритма. У наставку су објашњени најпознатији типови ове класе алгоритама.

i) Maximum Weight Matching (MWM)

У бипартитном графу, ивица која повезује улазни порт i и излазни порт j се означава са $e_{i,j}$, док се њена тежина означава са $w_{i,j}$. Тежина $w_{i,j}$ се обично, мада не и обавезно, односи на број пакета тренутно смјештених у $VOQ(i,j)$. *Maximum Weight Matching* (*MWM*) за бипартитни граф је онај који максимизује $\sum_{e(i,j)\in M} w_{i,j}$ [36]. На слици 3.1.2 је дат примјер резултата *MWM* алгоритма. Као што се види са слике, резултат *MWM* алгоритма је конфигурација комутационог модула у коме је укупна тежина ивица највећа могућа. У [101, 102] је доказано да *MWM* постиже 100% пропусност за сваки дозвољени шаблон саобраћаја. Под дозвољеним саобраћајем се подразумијева саобраћај гдје је улазни проток на свим улазним портовима мањи или једнак 100% капацитета, а при томе ниједан излазни порт није преоптерећен тј. проток пакета ка њима је мањи или једнак 100% њиховог капацитета. За мјерење пропусности се анализирају сценарији горње границе тј. 100% опетерећења. У [103] је доказано да је комплексност *MWM* алгоритма $O(N^3)$, што је исувише много за савремене пакетске комутаторе. Пажљивијим одабиром тежина ивица, или коришћењем апроксимација *MWM* алгоритма може се смањити комплексност израчунавања. *Longest Queue First* (*LQF*) и *Oldest Cell First* (*OCF*) су примјери *MWM* алгоритама [103]. *LQF* користи дужину редова $L_{i,j}(t)$ као тежину $w_{i,j}(t)$, док *OCF* користи вријеме чекања *HoL* пакета као тежину $w_{i,j}(t)$. За сваки дозвољени шаблон саобраћаја ови алгоритми постижу 100% пропусност. За недозвољени шаблон саобраћаја, код *LQF* алгоритма може доћи до нефер опслуживања портова, али код *OCF* овај проблем не постоји. Како би се смањила комплексност *LQF* алгоритма, предложен је *Longest Port First* (*LPF*) алгоритам код кога се тежина $w_{i,j}(t)$ дефинише као заузетост порта:

$$w_{i,j}(t) = \begin{cases} R_i(t) + C_j(t), & L_{ij}(t) > 0 \\ 0, & \text{, у супротном} \end{cases} \qquad (3.1.1)$$

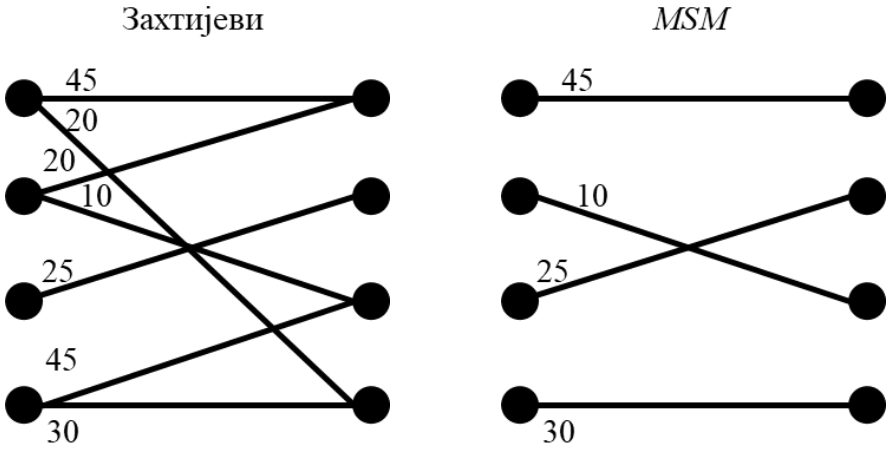
гдје је $R_i(t) = \sum_{j=1}^{N} L_{ij}(t)$, $C_j(t) = \sum_{i=1}^{N} L_{ij}(t)$. Комплексност *LPF* алгоритма је $O(N^{2.5})$, што је мање у односу на друге *MWM* алгоритме [104].



Слика 3.1.2. Примјер *MWM* алгоритма

ii) Maximum Size Matching (MSM) алгоритам

Maximum Size Matching (*MSM*) алгоритам налази конфигурацију која садржи максималан број ивица. Очигледно, *MSM* алгоритам је посебан случај *MWM* алгоритма кад је тежина свих ивица иста [36]. На слици 3.1.3 је дат примјер резултата *MSM* алгоритма. Са слике се види ће резултат овог алгоритма бити конфигурација са четири конекције, што је највећи могућ број за овај комутатор. У [105] је показано да је комплексност *MSM* алгоритма $O(N^{2.5})$, до к је у [106] показано да *MSM* постиже 100% пропусност за сваки дозвољен и униформан шаблон саобраћаја. Под униформним саобраћајем се подразумијева да је укупно оптерећење пакетског комутатора равномјерно распоређено међу свим улазним и излазним портовима. Међутим, у [101] је показано да за дозвољен, али неуниформан шаблон саобраћаја може доћи до нестабилности и нефер опслуживања токова, док у случају недозвољеног шаблона саобраћаја, може доћи до нефер опслуживања неких портова.

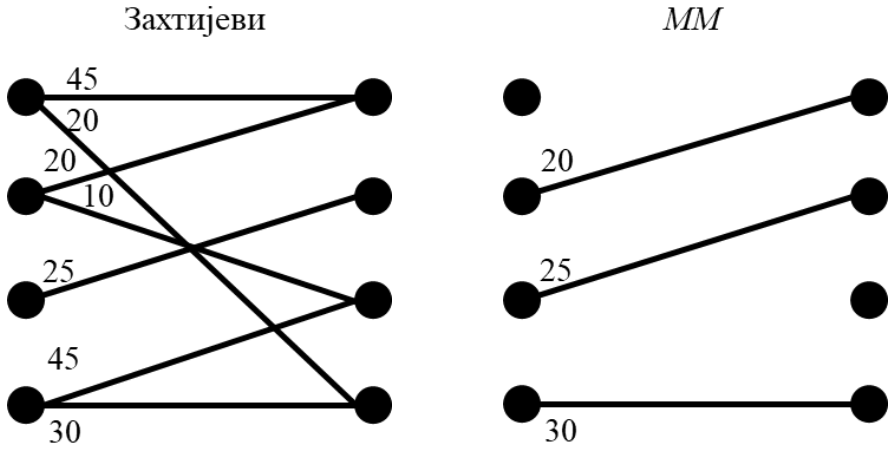


Слика 3.1.3. Примјер MSM алгоритма

3.1.2. Алгоритми максималног упаривања

Ова класа алгоритама доводи до максималног упаривања додајући нове конекције инкрементално, тј. без брисања претходно успостављених конекција. Код алгоритама максималног упаривања, ако неки улазни порт има пакете за слање, а није упарен ни са једним излазним портом на крају извршења алгоритма, онда сви излазни портови за које овај улазни порт има пакет морају бити упарени са другим улазним портовима. Када се користи алгоритам максималног упаривања, постоји могућност да резултат алгоритма буде слабији од оптималног, односно да се добије мање упарених улаза и излаза него што би се добило применом алгоритма максимум упаривања. С друге стране, алгоритми максималног упаривања су значајно једноставнији за имплементацију у односу на алгоритме максимум упаривања [36].

На слици 3.1.4 је дат примјер резултата алгоритма максималног упаривања. Са слике се види да ће улазни порт 1 и излазни порт 3 бити неупарени, али то је у реду јер на улазном порту 1 не постоји пакет који је намијењен за излазни порт 3. У [107] и [108] је показано да алгоритам максималног упаривања постиже 100% пропусност за униформни саобраћај, али је неопходно убрзање $s \geq 2$. У случају неуниформног саобраћаја не може се гарантовати 100% пропусност. Један начин имплементације алгоритма максималног упаривања је употреба итеративних алгоритама упаривања који користе више итерација до проналажења крајњег резултата. У свакој итерацији, најмање једна нова конекција је додата прије него је постигнута коначна конфигурација. Дакле, максимално упаривање се увијек може постићи у N итерација, мада неки алгоритми конвергирају брже и захтијевају нпр. $\log_2 N$ итерација. Главна разлика између различитих алгоритама максималног упаривања је у начину како улазни портови шаљу захтјеве, и како излазни портови прихватају исте током упаривања. У наставку су описани најпознатији представници ове класе алгоритама.



Слика 3.1.4. Примјер алгоритма максималног упаривања

i) Parallel Iterative Matching (PIM) алгоритам

PIM алгоритам који је предложен у [109] користи случајну селекцију током одређивања конфигурације комутационог модула. Овај алгоритам се може извршавати од једне до N итерација. У свакој итерацији учествују само они улазни и излазни портови који су у претходним итерацијама остали неупарени. Свака итерација се састоји из три корака:

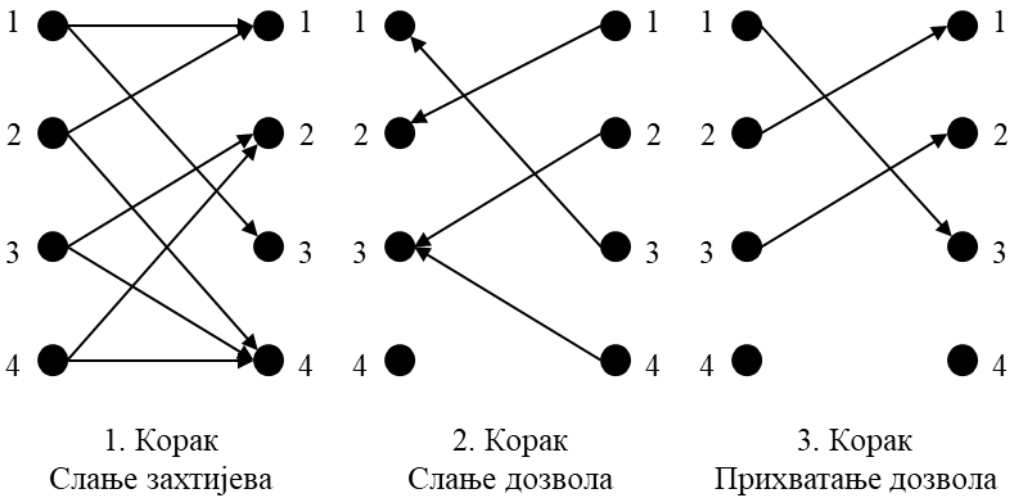
1) **Слање захтјева**. Сваки неупарени улазни порт шаље захтјев сваком излазном порту за који има пакет у својим *VOQ* редовима.

2) **Слање дозвола**. Ако неупарени излазни порт прими више захтјева, бира један од њих по случајном принципу. Сваки захтјев има једнаку вјероватноћу да буде одабран.
3) **Прихватање дозвола**. Ако један улазни порт прими дозволе од више излазних портова, потврђује се једна од њих на случајан начин. Свака дозвола има једнаку вјероватноћу да буде потврђена.

Поступак се понавља у више итерација док се не постигне максимално упаривање. На слици 3.1.5 је дат примјер једне итерације овог алгоритма. У првом кораку, сви улазни портови шаљу захтијеве према свим излазним портовима за које имају пакете. Улазни порт 1 ће послати захтјеве излазним портовима 1 и 3. Улазни порт 2 ће послати захтјеве излазним портовима 1 и 4. Улазни порт 3 ће послати захтјеве излазним портовима 2 и 4. Улазни порт 4 ће послати захтјеве излазним портовима 2 и 4.

У другом кораку се шаљу дозволе са излазних портова. Излазни порт 1 ће примити захтијеве са улазних портова 1 и 2, и рецимо да по случајном одабиру шаље дозволу улазном порту 2. Излазни порт 2 ће примити захтијеве са улазних портова 3 и 4, и рецимо да по случајном одабиру шаље дозволу улазном порту 3. Излазни порт 3 ће примити захтијев само са улазног порта 1, па ће њему и послати дозволу. Излазни порт 4 ће примити захтијеве са улазних портова 2, 3 и 4, и рецимо да по случајном одабиру шаље дозволу улазном порту 3.

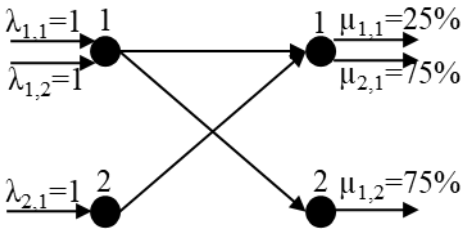
У трећем кораку се врши прихватање дозвола на улазним портовима. Улазни порт 1 ће примити дозволе са излазних портова 1 и 3, и рецимо да по случајном одабиру прихвата дозволу са излазног порта 3. Улазни порт 2 ће примити дозволу само са излазног порта 1, па ће њу и прихватити. Такође, улазни порт 3 ће примити дозволу са излазних портова 2 и 4, и рецимо да по случајном одабиру прихвата дозволу са излазног порта 2. Тиме је окончана прва итерација алгоритма, а по истом принципу се извршавају и остале итерације у којима учествују неупарени улазни и излазни портови. Очигледно је да ће се у сљедећој итерацији упарити улаз 4 и излаз 4 чиме ће бити окончан процес одређивања конфигурације.



Слика 3.1.5. Примјер *PIM* алгоритма

PIM алгоритам конвергира након $O(\log N)$ итерација [36]. Ако се користи само једна итерација, пропусност комутатора је свега 63% [36]. Уколико алгоритам има N итерација, пропусност ће бити 100% [36]. Такође, *PIM* алгоритам има проблем са нефер опслуживањем

токова, када је комутатор преоптерећен [110]. Примјер такве ситуације је дат на слици 3.1.6. Претпоставимо да улазни порт 1 има пакете за излазне портове 1 и 2 у сваком временском слоту, док улазни порт 2 има пакете само за излазни порт 2. У том случају, улазни порт 1 ће прихватити дозволу са излазног порта 1 свега једну четвртину слотова. Наиме, вероватноћа да излаз 1 пошаље дозволу улазу 1 је 50%. Пошто са излаза 2 увек стиже дозвола улазу 1, улаз 1 када стигну дозволе са оба излаза бира дозволу са улаза 1 са вероватноћом 50%. То укупно даје вероватноћу од 25% да ће у првој итерацији да се упари улаз 1 и излаз 1, односно 75% је вероватноћа да ће бити упарени улаз 1 и излаз 2. Фер опслуживање би подразумевало да у 50% временских слотова буду упарени улаз 1 и излаз 1. Дакле, јасно је да долази до нефер опслуживања токова. Осим тога, хардверска имплементација случајног избора може ограничити скалабилност овог рјешења.



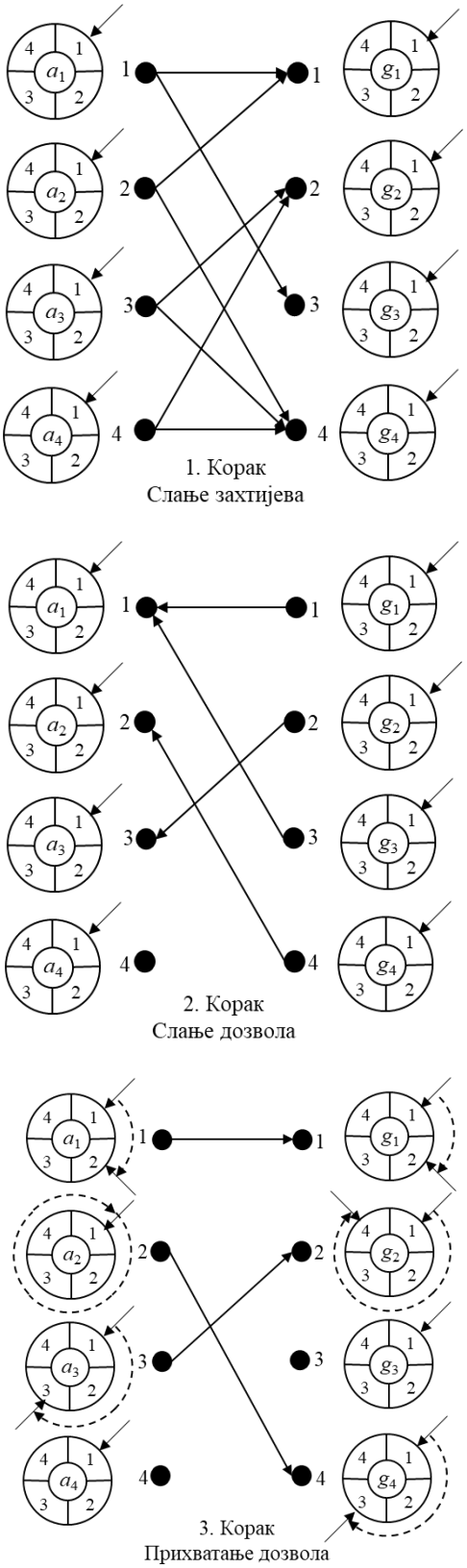
Слика 3.1.6. Примјер нефер сервиса *PIM* алгоритма

ii) i-Slip алгоритам

Овај алгоритам је предложен у [110]. Разлика је у начину на који се врши ажурирање показивача на улазним и излазним портовима. Процес упаривања улазних и излазних портова се састоји из три корака:

1) **Слање захтјева**. Сваки неупарени улазни порт шаље захтјев сваком излазном порту за који има пакет у својим *VOQ* редовима.
2) **Слање дозвола**. Ако неупарени излазни порт прими више захтјева, бира онај који је сљедећи у *round-robin* редоследу, почињући од улазног порта највећег приоритета. Излазни порт сваком улазном порту са којег је добио захтјев шаље обавјештење да ли је његов захтјев одобрен или не. Показивач на излазном порту је инкрементиран (по модулу *N*) на прву позицију иза улазног порта коме је захтјев одобрен, ако и само ако је та дозвола потврђена у трећем кораку прве итерације.
3) **Прихватање дозвола**. Ако један улазни порт прими дозволе од више излазних портова, он прихвата ону дозволу која је прва у *round-robin* редоследу, почињући од излазног порта највећег приоритета. Показивач на улазном порту је инкрементиран (по модулу *N*) на прву позицију иза излазног порта чија је дозвола прихваћена. Ови показивачи се ажурирају само у првој итерацији.

Поступак се понавља у више итерација док се не постигне максимално упаривање. Примјер рада овог алгоритма је дат на слици 3.1.9. У почетном тренутку сви показивачи на улазним портовима показују на излазни порт 1. Такође, сви показивачи на излазним портовима показују на улазни порт 1. У првом кораку, сви улазни портови шаљу захтијеве према свим излазним портовима за које имају пакете. Улазни порт 1 ће послати захтјеве излазним портовима 1 и 3. Улазни порт 2 ће послати захтјеве излазним портовима 1 и 4. Улазни порт 3 ће послати захтјеве излазним портовима 2 и 4. Улазни порт 4 ће послати захтјеве излазним портовима 2 и 4.

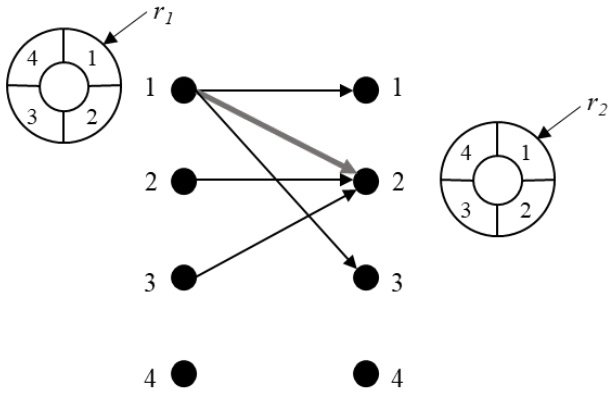


Слика 3.1.9. Примјер *iSlip* алгоритма

У другом кораку се шаљу дозволе са излазних портова. Излазни порт 1 ће примити захтијеве са улазних портова 1 и 2. Пошто показивач дозвола показује на улазни порт 1, онда

ће овај излазни порт послати дозволу улазном порту 1. Излазни порт 2 ће примити захтијеве са улазних портова 3 и 4. Пошто показивач дозвола показује на улазни порт 1, онда ће овај излазни порт послати дозволу улазном порту 3, јер је он први у *round-robin* редоследу. Излазни порт 3 ће примити захтијев само са улазног порта 1, па ће њему и послати дозволу. Излазни порт 4 ће примити захтијеве са улазних портова 2, 3 и 4. Пошто показивач дозвола показује на улазни порт 1, онда ће овај излазни порт послати дозволу улазном порту 2, јер је он први у *round-robin* редоследу.

У трећем кораку се врши прихватање дозвола на улазним портовима. Улазни порт 1 ће примити дозволе са излазних портова 1 и 3. Пошто показивач на улазном порту 1 показује на излазни порт 1, онда ће овај улазни порт прихватити дозволу са излазног порта 1. Показивач на улазном порту 1 се ажурира да показује на излазни порт 2, док се показивач на излазном порту 1 ажурира да показује на улазни порт 2. Улазни порт 2 ће примити дозволу само са излазног порта 4, па ће њу и прихватити. Показивач на улазном порту 2 ће и даље да показује на излазни порт 1, док се показивач на излазном порту 4 ажурира да показује на улазни порт 3. Такође, улазни порт 3 ће примити дозволу само са излазног порта 2, па ће њу и прихватити. Показивач на улазном порту 3 се ажурира да показује на излазни порт 3, док се показивач на излазном порту 2 ажурира да показује на улазни порт 4.



Слика 3.1.10. Нефер опслуживање у случају ажурирања показивача у свакој итерацији алгоритма

За *iSlip* алгоритам је карактеристично да се показивачи дозвола ажурирају само ако су њихове дозволе прихваћене. Такође, показивачи на улазним, односно излазним портовима су ажурирани само у току прве итерације, јер би у супротном могло доћи до нефер опслуживања неких токова. На слици 3.1.10 је дат примјер ове ситуације. Овдје се могу видјети захтјеви које улазни портови шаљу у току првог временског слота. Претпоставимо да показивач на улазном порту 1 показује на излазни порт 1, а да показивач на излазном порту 2 показује на улазни порт 1. Током прве итерације у овом временском слоту, улазни порт 1 ће прихватити дозволу са излазног порта 1, док ће у другој итерацији, излазни порт 2 послати дозволу улазном порту 2, јер улазни порт 1 већ заузет. Када улазни порт 2 прихвати дозволу са излазног порта 2, показивач дозвола на излазном порту 2 ће бити ажуриран да показује на улазни порт 3. Дакле, конекција између улазног порта 1 и излазног порта 2 ће можда бити одложена бесконачно. Међутим, код *iSlip* алгоритма, ажурирање се врши само током прве итерације, тако да је избјегнуто нефер опслуживање токова. У току $2 \cdot N$ временских слотова сваки пар улаз-излаз за који постоји захтјев ће бити опслужен, а последњој успостављеној конекцији се даје најнижи приоритет. Захваљујући *round-robin* механизму који се користи приликом ажурирања показивача, алгоритам обезбјеђује фер алокацију ресурса између свих токова. Овај алгоритам постиже 100% пропусност за униформни саобраћај за било који број

итерација, захваљујући ефекту десинхронизације [111,112]. Дакле, чак иако се користи само једна итерација, пропусност ће бити 100%, јер ће у коначном броју временских слотова, показивачи на излазним портовима показивати на другачији улазни порт у односу на све друге излазне портове. На тај начин, сваки улазни порт ће у сваком временском слоту послати један пакет.

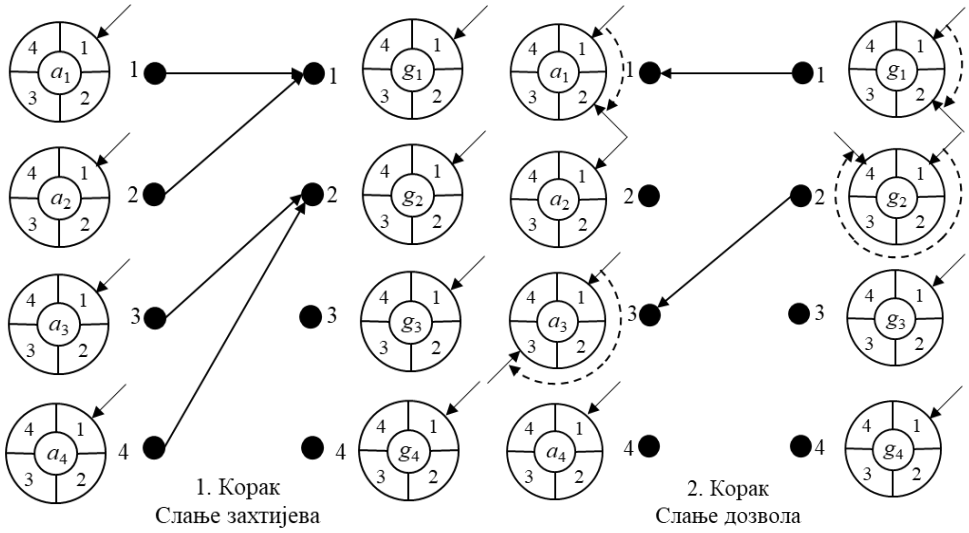
iii) Dual Round-Robin Matching (DRRM) алгоритам

DRRM алгоритам који је преложен у [113,114] такође користи *round-robin* механизам умјесто случајне селекције. Међутим, за разлику од претходно описаних алгоритама, овдје се *round-robin* селекција користи и приликом слања захтјева са улазних портова. Дакле, улазни портови не шаљу захтјеве на све излазне портове за које имају пакете, већ се по *round-robin* принципу бира само један од њих. Излазни порт може да прими највише *N* захтјева. Опет, користећи *round-robin* принцип шаље се дозвола само једном улазном порту. Дакле, овај алгоритам се састоји од два корака:

1) **Слање захтјева**. Сваки неупарени улазни порт шаље захтјев излазном порту који одговара првом непразном *VOQ* реду у *round-robin* редоследу. Показивач на улазном порту ће показивати на исти излазни порт, ако се у другом кораку не добије потврда с тог порта. У супротном, показивач се ажурира да показује на први сљедећи порт у *round-robin* редоследу.
2) **Слање дозвола**. Ако неупарени излазни порт прими више захтјева, бира онај који је сљедећи у *round-robin* редоследу. Излазни порт сваком улазном порту са којег је добио захтјев шаље обавјештење да ли је његов захтјев одобрен или не. Показивач дозвола на излазном порту је инкрементиран (по модулу *N*) на прву позицију иза улазног порта коме је захтјев одобрен. Ако порт није примио ниједан захтјев, показивач показује на исти порт.

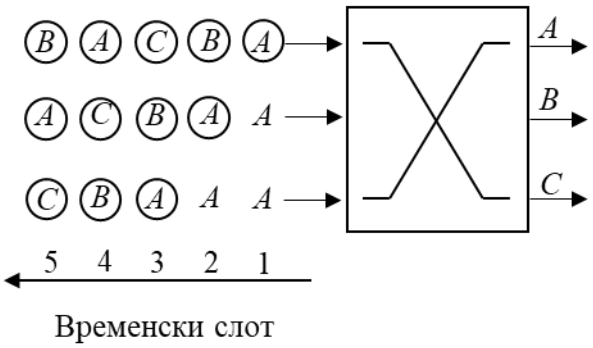
Пошто улазни портови шаљу само по један захтјев, и примају само по једну дозволу, трећи корак није неопходан. На слици 3.1.12 је дат примјер рада овог алгоритма. У почетном тренутку сви показивачи на улазним портовима показују на излазни порт 1. Такође, сви показивачи на излазним портовима показују на улазни порт 1. У првом кораку, сви улазни портови шаљу захтијеве према излазним портовима у складу са *round-robin* принципом. Улазни порт 1 има пакете за излазне портове 1 и 3. Пошто показивач на улазном порту 1 показује на излазни порт 1, онда ће послати захтјев овом излазу. Улазни порт 2 има пакете за излазне портове 1 и 4. Пошто показивач на улазном порту 2 показује на излазни порт 1, онда ће послати захтјев овом излазу. Улазни порт 3 има пакете за излазне портове 2 и 4. Пошто показивач на улазном порту 3 показује на излазни порт 1, онда ће послати захтјев излазном порту 2 у складу са *round-robin* принципом. Улазни порт 4 има пакете за излазне портове 2 и 4. Пошто показивач на улазном порту 4 показује на излазни порт 1, онда ће послати захтјев излазном порту 2 у складу са *round-robin* принципом.

У другом кораку се шаљу дозволе са излазних портова. Излазни порт 1 ће примити захтјеве са улазних портова 1 и 2. Пошто показивач на излазном порту 1 показује на улазни порт 1, онда ће овај излаз послати дозволу улазном порту 1. Показивач на излазном порту 1 се ажурира да показује на улазни порт 2. Такође, показивач на улазном порту 1 се ажурира да показује на излазни порт 2. Излазни порт 2 ће примити захтјеве са улазних портова 3 и 4. Пошто показивач на излазном порту 2 показује на улазни порт 1, онда ће овај излаз послати дозволу улазном порту 3, у складу са *round-robin* принципом. Показивач на излазном порту 3 се ажурира да показује на улазни порт 4. Такође, показивач на улазном порту 3 се ажурира да показује на излазни порт 3.



Слика 3.1.12. Примјер *DRRM* алгоритма

Пошто се размењује мања количина информација, *DRRM* алгоритму је потребно мање времена да изврши одабир пакета за слање, и лакши је за имплементацију. Слично као и *iSlip* алгоритам и *DRRM* има ефекат десинхроизације, тј. показивачи на улазним портовима који су примили дозволе у различитим временским слотовима, ће показивати на различите излазне портове. Примјер је дат на слици 3.1.13. Наиме, претпоставимо ситуацију гдје сви *VOQ* редови увијек имају бар један пакет. Са слике 3.1.13 се види који *HoL* пакети ће бити изабрани на сваком улазном порту у различитим временским слотовима. У временском слоту 1, сваки улазни порт бира пакет који треба послати на излазни порт *A*. Међу ова три захтјева, излазни порт ће потврдити само један (у овом конкретном примјеру, потврдиће се захтјев са улазног порта 1), док остали морају да чекају. *Round-robin* показивач на првом улазном порту ће се ажурирати да показује на излазни порт *B*. У временском слоту 2, улазни порт 1 ће послати пакет на овај излазни порт, јер ниједан други улазни порт неће послати захтјев ка порту *B*. Друга два улазна порта опет шаљу захтјеве излазном порту *A*, и улазни порт 2 ће добити дозволу и послати пакет, док ће улазни порт 3 морати да сачека сљедећи временски слот. Сада, *round-robin* показивачи на ова три улазна порта показују на излазне портове *C*, *B* и *A*, респективно. Последично, сва три улазна порта ће моћи да пошаљу пакете на одговарајуће излазне портове.



Слика 3.1.13. Ефекат десинхронизације *DRRM* алгоритма у случају сценарија пуног оптерећења

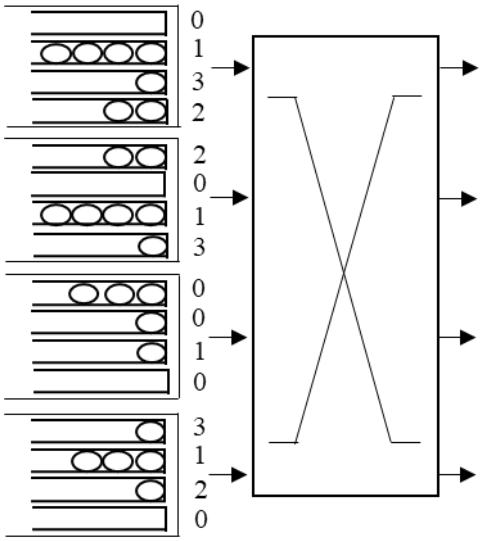
iv) Highest Rank First (HRF) алгоритам

HRF алгоритам је предложен у [115]. Процес упаривања улазних и излазних портова је нешто другачији него код осталих алгоритама максималног упаривања. Улазни порт додјељује одговарајуће рангове сваком излазном порту. Улазни порт ће додијелити ранг 1 оном излазном порту за који има највише пакета. Излазном порту који је други по броју пакета улазни порт ће додијелити ранг 2 итд. Излазни порт за који постоји најмање пакета за слање са тог улазног порта ће добити ранг *N*. Ако на улазном порту не постоји ниједан пакет за слање на одређени излазни порт, онда се њему додјељује ранг 0. Упаривање улазних и излазних портова се спроводи на сљедећи начин. Наиме, за сваки улазни порт *i* у временском слоту *t*, постоји преферирани излазни порт *j*, тако да важи:

$$j = (i+t) \bmod N \tag{3.1.2}$$

Даље се упаривање улазних и излазних портова врши у три корака:

1) **Слање захтјева.** Ако улазни порт *i* преферира излазни порт *j*, и ако у *VOQ*(*i*,*j*) постоји пакет, онда ће улазни порт *i* послати захтјев са рангом 1 излазном порту *j*, а свим осталим излазним портовима ће послати захтјев са рангом 0. Ако улазни порт *i* нема пакет за свој преферирани излазни порт онда свим излазним портовима шаље захтјеве са правим ранговима који су додијељени тим излазним портовима.
2) **Слање дозвола.** Ако излазни порт прими захтјев од свог префериираног улаза, онда њему шаље дозволу. У супротном, излазни порт шаље дозволу оном улазном порту који му је послао највећи ранг.
3) **Прихватање дозвола.** Ако улазни порт има пакет за свој преферирани излазни порт, онда ће прихватити дозволу са тог порта. У супротном, улазни порт ће прихватити дозволу са излазног порта који има највећи ранг, тј. најмању бројну вриједност ранга.



Слика 3.1.14. Примјер додјеле рангова код *HRF* алгоритма

На слици 3.1.14 је дат примјер рада овог алгоритма. Претпоставимо да посматрамо слот *t*=0. У овом слоту, за улазни порт 1 преферирани излазни порт је 1. Међутим, како не постоји пакет за овај излазни порт, онда ће се свим излазним портовима додијелити рангови на основу броја пакета у одговарајућим *VOQ* редовима. Излазни порт 1 ће добити ранг 1 јер за њега има највише пакета. Затим, излазни порт 4 ће добити ранг 2 јер је одговарајући *VOQ*

ред други по величини. Коначно, излазни порт 3 ће добити ранг 3 јер одговарајући *VOQ* ред има најмање пакета од свих непразних *VOQ* редова. Улазни порт 2 ће излазном порту 1 додијелити ранг 2, излазном порту 2 ранг 0, излазном порту 3 ранг 1 и излазном порту 4 ранг 3 у складу са дужинама одговарајућих *VOQ* редова. Пошто улазни порт 3 има пакет за свој преферирани излаз, њему ће додијелити ранг 1 а свим осталим ранг 0. Улазни порт 4 нема пакет за свој преферирани излаз, па ће излазним портовима додијелити рангове у складу са одговарајућим бројем пакета што је приказано на слици 3.1.14.

Након што излазни портови приме од улазних портова захтјеве са одговарајућим ранговима, у другом кораку се врши слање дозвола. Излазни порт 1 ће од улазног порта 1 добити захтјев са рангом 0, од улазног порта 2 захтјев са рангом 2, од улазног порта 3 захтјев са рангом 0 и од улазног порта 4 захтјев са рангом 3. Пошто је излазни порт 1 највећи ранг добио од улазног порта 2, њему ће се послати дозвола за слање. Излазни порт 2 ће од улазног порта 1 добити захтјев са рангом 1, од улазног порта 2 захтјев са рангом 0, од улазног порта 3 захтјев са рангом 0 и од улазног порта 4 захтјев са рангом 1. Пошто су улазни портови 1 и 4 послали захтјеве са истим рангом, излазни порт 2 на случајан начин бира коме ће од њих послати дозволу. Рецимо да ће одабрати улазни порт 1. Излазни порт 3 ће добити захтјев од свог преферираног улазног порта па ће њему и послати дозволу. Коначно, излазни порт 4 ће од улазног порта 1 добити захтјев са рангом 2, од улазног порта 2 захтјев са рангом 3, од улазних портова 3 и 4 захтјеве са рангом 0. Пошто је навећи ранг добио од улазног порта 1 њему ће бити послата дозвола.

У трећем кораку се врши прихаватње дозвола. Улазни порт 1 је добио дозволу од излазних портова 2 и 4. Пошто излазни порт 2 има већи ранг њему ће бити послат пакет. Улазни порт 2 је добио дозволу само од излазног порта 1 па ће њу прихаватити и послати пакет на тај порт. Улазни порт 3 ће прихватити дозволу од свог преферираног излаза и њему послати пакет. Улазни порт 4 није добио ниједну дозволу па неће слати пакет у овом слоту.

На основу формуле 3.1.2 је јасно да ће сваки улазни порт да преферира сваки излазни порт на сваких N слотова. Када је оптерећење комутатора велико, тј. када на свим улазним портовима постоје пакети за све излазне портове, онда ће се, захваљујући томе што се свим излазним портовима даје предност при слању пакета на сваких N слотова, постићи фер опслуживање свих токова. Када је оптерећење мало, онда је вјероватно да улазни портови често неће имати пакете за свој преферирани излазни порт па ће се одабир пакета вршити на основу испитивања рангова. Ово рјешење постиже боље перформансе у односу на остале алгоритме максималног упаривања [115]. Недостатак овог рјешења је што се захтијева сортирање N *VOQ* редова на сваком улазном порту ради додјељивања одговарајућих рангова што повећава сложеност имплементације.

v) Coded Highest Rank First (CHRF) алгоритам

Иако *HRF* алгоритам постиже врло добре перформансе, кључни недостатак је што се захтијева слање захтијева у виду $log(N+1)$ бита који садрже информацију о ранговима. Зато је у [115] предложен и *CHRF* алгоритам који шаље захтијеве од само једног бита. Овдје се бит користи као индикатор да ли је дошло до повећања или смањења ранга *VOQ* реда на улазном порту. Примјера ради, рецимо да је X_t једнобитни захтијев који је неки улазни порт послао одређеном излазном порту у слоту t. На основу његовог ранга у претходном слоту (t-1) X_t се поставља на вриједност 1 ако је тренутни ранг овог *VOQ* реда већи односно на вриједност 0 ако је тренутни ранг овог *VOQ* реда мањи. Међутим, поставља се питање шта ако је ранг неког *VOQ* реда остао исти. Како би се ријешио овај проблем сви *VOQ* редови су класификовани као: “празан”, “остали” и “најдужи”. Ако *VOQ* ред није празан, а није ни

најдужи онда се класификује као “остали”. Јасно је да може бити више *VOQ* редова који су “празни” или “остали”. Ако има више *VOQ* редова који су најдужи само један је означен као “најдужи”. Кључни изазов је да се избјегне да улазни порт добије више потврда за слање. Принцип рада алгоритма је сљедећи. Рецимо да је X_t једнобитни захтијев који улазни порт шаље у слоту *t*. Његова вриједност се одређује на основу ситуације у претходном слоту на сљедећи начин:

- Ако се статус дотичног *VOQ* реда промијени из “празан” (у слоту *t*-1) у “остали” или из “остали” у “најдужи” или из “празан” у “најдужи”, његов ранг се повећава и захтијев X_t=1 је послат.
- Ако се статус дотичног *VOQ* реда промијени из “најдужи” у “остали” или из “остали” у “празан” или из “најдужи” у “празан”, његов ранг се смањује и захтијев X_t=0 је послат.
- Ако дотични *VOQ* ред остаје у категорији “остали”, X_t=!X_{t-1} је послат (то јест послата је вриједност супртона од претходне)
- Ако дотични *VOQ* ред остаје “најдужи” шаље се X_t=1.
- Ако дотични *VOQ* ред остаје “празан” шаље се X_t=0.

Да би излазни порт могао да декодира захтијев неопходно је да чува информацију и о X_{t-1}. Када излазни порт прими X_t, ранг одговарајућег *VOQ* реда се одређује и на основу X_{t-1}. Шема декодирања је дата у табели 3.1.1. Како узастопне јединице значе повећање ранга, ако излазни порт прими $X_{t-1}X_t$=11 то значи примјену статуса у “најдужи”. На примјер, претпоставимо да је инцијални статус *VOQ* реда “празан”. Ако је X_{t-1}=1, *VOQ* ред ће да промијени статус у “остали” у слоту *t*-1. Даље, ако је X_t=1 статус *VOQ* реда у слоту *t* ће бити промијењен у “најдужи”. С друге стране, узастопне нуле значе смањење ранга, тако да $X_{t-1}X_t$=00 значи промјену статуса у “празан”. Дакле, могуће је недвосмислено утврдити да ли је статус *VOQ* реда “празан” или “најдужи”. Ово је нарочито важно јер слање потврде празном *VOQ* реду је губитак ресурса док је слање потврде *VOQ* реду који није “најдужи” неефикасно.

Табела 3.1.1. Принцип декодовања ранга VOQ реда

$X_{t-1}X_t$	00	01	10	11
Ранг	“празан”	“празан” “остали”	“остали” “празан”	“најдужи”

Ако излазни порт прими $X_{t-1}X_t$=01 онда тренутни статус *VOQ* реда може бити или “празан” или “остали”. Постоје сљедеће могућности:

- Ако је иницијално стање “празан”, X_{t-1}=1 значи промјену стања из “празан” у “остали” или из “празан” у “најдужи”. Дакле на крају слота *t*-1 постоје два могућа стања: “остали” или “најдужи”.
  - У случају да је статус “остали”, X_t=0 значи промјену статуса из “остали” у “празан” или из “остали” у “остали”.
  - У случају да је статус “најдужи”, X_t=0 значи промјену статуса из “најдужи” у “празан” или из “најдужи” у “остали”.

- Ако је иницијално стање “остали”, X_{t-1}=1 значи промјену стања из “остали” у “остали” или из “остали” у “најдужи”. Дакле на крају слота t-1 постоје два могућа стања: “остали” или “најдужи”.
  - У случају да је статус “остали”, X_t=0 значи промјену статуса из “остали” у “празан” или из “остали” у “остали”.
  - У случају да је статус “најдужи”, X_t=0 значи промјену статуса из “најдужи” у “празан” или из “најдужи” у “остали”.
- Ако је иницијално стање “најдужи”, X_{t-1}=1 значи промјену стања из “најдужи” у “најдужи”. Онда X_t=0 значи промјену из “најдужи” у “празан” или из “најдужи” у “остали”.

На сличан начин се закључује да ако излазни порт прими $X_{t-1}X_t$=10 онда тренутни статус *VOQ* реда може бити или “најдужи” или “остали”. Табела 3.1.1. даје шему декодовања на сваком излазном порту:

- Ако је $X_{t-1}X_t$=00 статус *VOQ* реда је “празан”.
- Ако је $X_{t-1}X_t$=01 статус *VOQ* реда је или “празан” или “остали”.
- Ако је $X_{t-1}X_t$=10 статус *VOQ* реда је или “остали” или “најдужи”.
- Ако је $X_{t-1}X_t$=11 статус *VOQ* реда је “најдужи”.

Дакле, на излазном порту приоритет при одабиру *VOQ* реда коме ће се послати потврда је сљедећи 11>10>01>00. На овај начин ако излазни порт прими (највише) N захтијева лако може да идентификује *VOQ* ред са највећим рангом користећи само двобитни компаратор. У складу с тим, *CHRF* алгоритам је имплементиран на сљедећи начин:

1) **Слање захтијева.** Сваки улазни порт i ако је $VOQ(i,j)$ непразан (гдје је j преферирани излазни порт) излазном порту j шаље X_t=1, а X_t=0 је послат осталим излазним портовима. У супротном, за сваки *VOQ* на улазном порту i поступак је сљедећи:
   a) Ако се статус *VOQ* реда промијени из “празан” (у слоту t-1) у “остали” или из “остали” у “најдужи” или из “празан” у “најдужи” X_t=1 је послат одговарајућем излазном порту.
   b) Ако се статус *VOQ* реда промијени из “најдужи” у “остали” или из “остали” у “празан” или из “најдужи” у “празан” X_t=0 је послат одговарајућем излазном порту.
   c) Ако статус остаје “остали” X_t=!X_{t-1} је послат.
   d) Ако статус остаје “најдужи” X_t=1 је послат.
   e) Ако статус остаје “празан” X_t =0 је послат.
2) **Слање потврда.** Ако излазни порт j прими X_t=1 са преферираног улазног порта њему се шаље потврда за слање. У супротном, од свих примљених захтијева шаље се потврда оном са максималном вриједношћу X_tX_{t-1} (ако их има више бира се једна на случајан начин).
3) **Прихватање потврда.** Сваки улазни порт i, ако прими потврду са свог преферираног излазног порта њу прихвата. У супротном од свих примљених потврда прихвата ону са највећим рангом (ако их има више бира се једна на случајан начин).

Дакле, ако улазни порт има пакет за свој преферирани излазни порт онда њему шаље захтјев X_t=1, а осталим шаље захтјеве X_t=0. У овом случају захтјеви нису кодирани у складу са табелом 3.1.1., јер они не означавају повећање или смањење ранга. Међутим, ако улазни порт нема пакет за свој преферирани излаз онда се захтијев шаље у складу са табелом 3.1.1. Ако излазни порт *j* прими захтјев X_t=1 са свог преферираног улаза он ће му послати потврду за слање. С друге стране, ако излазни порт прими X_t=0 он не зна да ли то треба да буде тумачено као кодирани захтијев (тј. смањење ранга) или као индикатор да не треба слати потврду. Разлог за ово је што излазни порт не зна да ли одговарајући улазни порт (који је послао X_t=0) има пакет за свој преферирани излаз или не. *CHRF* алгоритам сматра да ако је X_t примљен са непреферираног улазног порта излазни порт га увијек третира као кодирани захтијев.

Излазни порт шаље потврду свом преферираном улазном порту с највећим приоритетом. Ако је *VOQ* ред на одговарајућем улазном порту празан онда излазни порт шаље потрвду улазном порту који има највећи ранг. Примјећено је да начин рада *CHRF* алгоритма може ограничити перформансе у случају неуниформног и великог саобраћаја. Разлог је тај што у том случају слање захтијева и потврда на основу ранга који се користи у *HRF* постиже боље перформансе.

Приликом прихватања потврда улазни порт увијек прихвата потврду прво са свог преферираног излазног порта. Ако та потврда не постоји улазни порт прихвата захтијев са највећим рангом.

За разлику од *HRF* алгоритма који захтијева сортиртање *N VOQ* редова, код *CHRF* алгоритма постоје само три вредности ранга, што је једноставније за имплементацију. С друге стране, ово рјешење постиже нешто лошије перформансе у односу на *HRF*, али и даље боље у односу на друга рјешења са баферима на улазним портовима.

vi) Parallel Complex Coloring (PCC) алгоритам

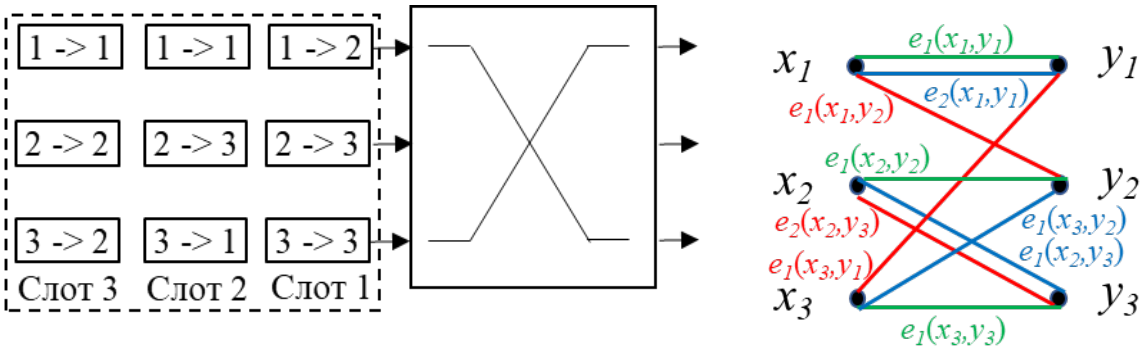
PCC алгоритам за конфигурисање комутатора са баферима на улазним портовима је предложен у [116]. Овај алгоритам користи *frame-based* приступ, што није типично за већину других алгоритама код комутатора са улазним баферима. Наиме, подразумијева се да је вријеме подијељено у низ фрејмова. Бипартитни граф $G_k=(X \cup Y, E)$ је конструисан за пакете који стижу у *k*-том фрејму. Чвор x_i у X и чвор y_j у Y представљају улазни порт *i* и излазни порт *j*, а $e_t(x_i,y_j)$ представља *t*-ти пакет у баферу на улазном порту *i* који треба послати на излазни порт *j*, гдје је $i,j = 1,2,\ldots,N$. На слици 3.1.15 а) је дат примјер стања у комутатору, а на слици 3.1.15 б) је представљен изглед бипартитног графа који служи за конфигурсање овог комутатора. Два пакета са улазног порта 1 за излазни порт 1 су означени са $e_1(x_1,y_1)$ и $e_2(x_1,y_1)$. У овом примјеру, максимални степен Δ бипартитног графа је 3 што је једнако максималном броју пакета у баферима на сваком улазном порту. Дакле, слање свих пакета ће захтијевати три слота у сљедећем фрејму, који су означени са три боје из скупа $C = \{c_1, c_2, c_3\} = \{c, z, p\}$, што је приказано на слици 3.1.15 б). У идеалном случају, сви пакети који су стигли у току текућег фрејма би требало да буду послати у току сљедећег фрејма, што значи да би *f* боја требало да буде довољно за одговарајуће бојење бипартитног графа G_k у сваком фрејму *k*. У [117] је показано да је за бојење довољно *f* боја ако је максимални степен графа $\Delta \leq f$. Овај услов је лако задовољен ако је саобраћај слабог интезитета, али при већим оптерећењима може се десити да Δ буде већ е од *f*. Ако је Δ већ е од *f*, немогуће је све пакете одабрати за слање у датом фрејму. Стога је неопходно правилно одабрати величину фрејма како би се са великом вјероватноћом гарантовало да је $\Delta \approx f$. У [116] је показано да ако

је величина фрејма довољно велика, максимални степен Δ ће бити врло близу f са великом вјероватноћом.

Принцип бојења бипартитног графа се заснива на сљедећем принципу. Саме ивице графа су подељене на две половине - једна половина је инцидентна одговарајућем чвору из скупа X, а друга половина је инцидентна одговарајућем чвору из скупа Y. Ове половине ивице се називају линковима. На почетку се изврши иницијализација графа тако што се сви линкови обоје. Потом се креће од чворова из скупа X, где се врши размена боја на оним (њиховим) линковима чија се боја не поклапа са линком на истој ивици. Размена се заснива да се линк разматраног чвора обоји бојом линка који представља другу половину ивице. Ако приликом размене боја, линк добије боју коју користи већ неки други линк истог чвора онда тај други линк добија првобитну боју линка који је урадио размену боја. Потом се исти поступак врши на линковима чворова из скупа Y. Описани поступак се понавља кроз више итерација, док се за све ивице не добије да им линкови имају поклапајућу боју или док се не испуни услов максималног броја итерација. Ивице чији линкови немају поклапајућу боју се сматрају необојеним ивицама и преносе се у сљедећи фрејм.

Конфигурисање комутатора се врши на сљедећи начин:

1) **<u>Иницијализација.</u>** Сваки улазни и излазни порт имплементира инцијализацију графа за тренутни фрејм у складу са пакетима примљеним у претходном фрејму и необојеним ивицама $M_{i,j}$ претходног фрејма.
2) **<u>Извршавање алгоритма.</u>** Извршава се претходно описани алгоритам бојења бипартитног графа (*PCC* алгоритам)и као резултат се добија обојени бипартитни граф који се користи за одређивање конфигурација комутационог модула у текућем фрејму.
3) **<u>Ажурирање информација.</u>** За сваки улазни порт i ажурира се скуп необојених ивица $M_{i,j}$ које се преносе у сљедећи фрејм.

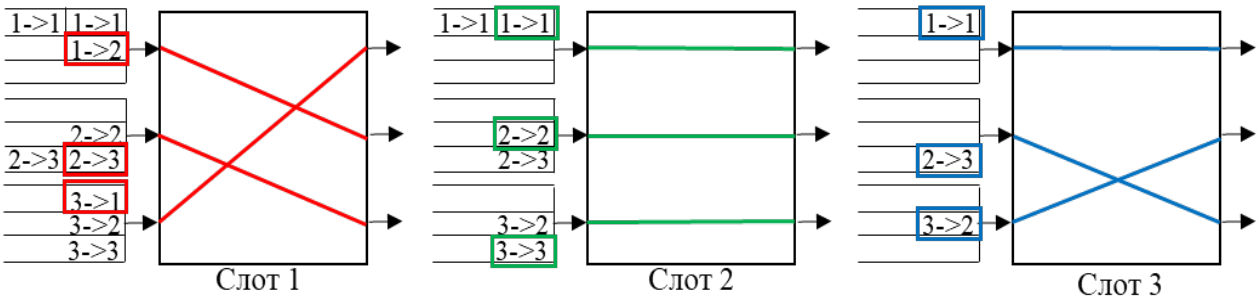


а) **б)**

Слика 3.1.15 а) Примјер стања у комутатору б) Одговарајући граф за примјер а)

У обојеном бипартитном графу, ивице исте боје одговарају конфигурацијама у истом слоту. Међутим, треба нагласити да проста примјена алгоритма за бојење графа може довести до премећаја редоследа пакета. Ово се дешава ако ивица одговара пакету који је раније стигао, али је обојена бојом која одговара каснијем слоту. Примјер је дат на слици 3.1.15. Пакети на улазним портовима су поређани на основу слота у коме су стигли на улаз, док боје из скупа $C = \{c_1, c_2, c_3\} = \{c, z, p\}$ представљају одговарајуће слотове у фрејму. Као

што се види са слике 3.1.15 б), ивице $e_1(x_2,y_3)$ и $e_2(x_2,y_3)$ између улазног порта x_2 и излазног порта y_3 су обојене са c_3 (плава) и c_1 (црвена) што значи да ће први пакет e_1 бити послат у трећем слоту док ће други пакет e_2 бити послат у првом слоту. Дакле, јасно је да ће доћи до поремећаја редоследа пакета што може изазвати велике проблеме у *TCP/IP* мрежама. Међутим, овај проблем је ријешен употребом *VOQ* редова, што је показано на слици 3.1.16. Када се изврши *PCC* алгоритам сваки улазни порт уствари добија токене за слање пакета на одређени излазни порт у одређеним слотовима сљедећег фрејма. Онда ће у сваком од тих слотова на излазни порт бити послат *HoL* пакет из одговарајућег *VOQ* реда, чиме се избјегава проблем поремећаја редоследа пакета. У примјеру са слике 3.1.16, улазни порт 2 има токен за слање пакета на излазни порт 3 у првом слоту. Умјесто слања другог пакета $e_2(x_2,y_3)$ улазни порт 2 ће послати *HoL* пакет из *VOQ* (2,3).



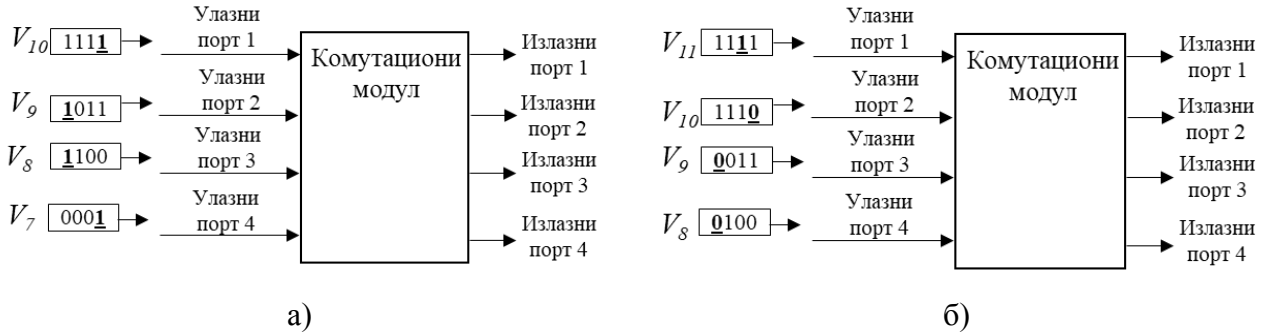
Слика 3.1.16. Редослед прослеђивања пакета у складу са графом са слике 3.1.15 б)

3.1.3. Sequential Greedy Scheduling (SGS) алгоритам

Сви претходно описани алгоритми захтијевају имплементацију централног распоређивача који врши конфигурисање комутатора. Главни недостатак оваквог рјешења је ограничена скалабилност како због имплементације линија за повезивање распоређивача са портовима тако и због потребе да се комплетан прорачун једне конфигурације комутационог модула обави у једном слоту. Како би превазишао овај проблем *SGS* алгоритам се заснива на коришћењу дистрибуираног распоређивача [118-120]. Код овог алгоритма улазни портови формирају тзв. ланац за међусобну размјену информација. Улазни портови користе вектор заузетости (који се састоји од N бита) који садржи информације о слободним излазним портовима. Улазни портови међусобно размењују овај вектор, и на основу њега бирају ком излазном порту ће послати пакет. Принцип рада алгоритма је сљедећи. Одабир пакета за слање почиње од улазног порта 1. На почетку сваког временског слота, сви бити у вектору заузетости су постављени на 1, што значи да ниједан излазни порт није заузет. Улазни порт може пакет изабрати коришћењем *round-robin* механизма, избором пакета из *VOQ* реда у ком има највише пакета итд. Услов је само да у избор улазе излазни портови који су означени као слободни у вектору заузетости. Када улазни порт i одабере ком излазном порту ће послати пакет, бит који одговара том излазном порту у вектору заузетости се поставља на 0. Затим улазни порт i прослеђује овај вектор сљедећем улазном порту у ланцу. Када последњи улазни порт у ланцу одабере излазни порт, један прорачун конфигурације комутационог модула је завршен. Очигледно прорачун једне конфигурације траје N слотова, али се користи пајплајн техника. Наиме, први улазни порт у ланцу у сваком временском слоту започиње прорачун нове конфигурације. Тиме се пајплајн техником омогућава да се у сваком тренутку обавља N прорачуна конфигурација, односно да проток ових прорачуна и даље буде једна конфигурација по једном временском слоту. Тиме се остварује иста фреквенција генерисања

конфигурација као и код алгоритама са централизованим распоређивачем, али са предношћу да је прорачун дистрибуиран и по улазним портовима и по временским слотовима што га чини мање сложеним за имплементацију.

Важно је нагласити да сваки улазни порт комуницира само са своја два сусједна порта, осим улазних портова 1 и N који комуницирају само са једним сусједним портом. Како у једном слоту, сви улазни портови обрађују различите векторе заузетости (који одговарају различитим слотовима сљедећег циклуса), то значи да се алгоритам на сваком улазном порту извршава у паралели. Овакво рјешење је рачунарски знатно мање захтијевно, али се уноси додатно кашњење јер је неопходно сачекати да прође један циклус да би се започело слање пакета. Међутим, ово кашњење није толико критично, имајући у виду врло кратко трајање једног временског слота. У наставку је дат један примјер *SGS* алгоритма.



а) б)

Слика 3.1.17. Примјер рада *SGS* алгоритма

На слици 3.1.17 а) је дато стање у временском слоту 10. Улазни порт 1, као први у ланцу, генерише вектор заузетости V_{10} са свим јединицима и бира пакет који ће бити послат у временском слоту 14. Улазни порт 1 бира пакет за излазни порт 4 и поставља одговарајући бит у вектору заузетости на 0. Улазни порт 2 обрађује вектор V_9 који је у претходном слоту обрадио улазни порт 1. Улазни порт 2 бира пакет за излазни порт 1 који ће бити послат у временском слоту 13. Улазни порт 3 обрађује вектор V_8 који је у претходном слоту обрадио улазни порт 2. Улазни порт 3 бира пакет за излазни порт 1 који ће бити послат у временском слоту 12. Улазни порт 4 обрађује вектор V_7 који је у претходном слоту обрадио улазни порт 3. Улазни порт 4 бира пакет за излазни порт 4 који ће бити послат у временском слоту 11. Тиме је завршен прорачун конфигурације комутационог модула за временски слот 11. На слици 3.1.17 б) приказује дешавања у временском слоту 11. Улазни порт 1 ће генерисати вектор заузетости V_{11} у коме су опет сви бити једнаки „1“ и бира пакет за излазни порт 3 који ће бити послат у слоту 15. Такође, одговарајућу позицију у вектору ресетује на 0 пре прослеђивања вектора заузетости улазном порту 2 који ће тај вектор процесирати у наредном временском слоту. Улазни порт 2 ће од улазног порта 1 добити ажурирани вектор V_{10} на основу кога види да је већ одабран пакет за излазни порт 4 и зато ће одабрати пакет за неки други излазни порт који ће бити послат у слоту 14. Улазни порт 3 од улазног порта 2 прима ажурирани вектор V_9 на основу кога види да су већ одабрани пакети за излазне портове 1 и 2 и зато ће одабрати пакет за неки други излазни порт који ће бити послат у временском слоту 13. Улазни порт 4 од улазног порта 3 прима ажурирани вектор V_8 на основу кога види да су већ одабрани пакети за излазне портове 1, 3 и 4 и зато ће одабрати пакет за излазни порт 2 који ће бити послат у временском слоту 12. На овај начин, улазни портови раде у паралели, тј. истовремено процесирају различите векторе заузетости, чиме је превазиђен проблем скалабилности централизованих распоређивача. Међутим, и код *SGS* алгоритма остаје проблем случајних конфигурација који постоји код свих комутатора са баферима на улазним

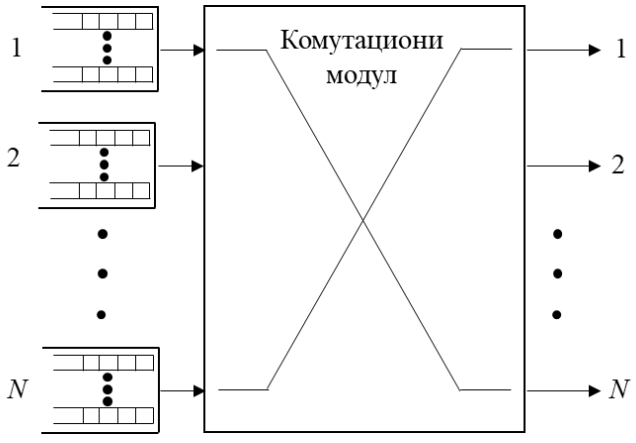
портовима. Наиме, и даље ове конфигурације морају да се проследе комутационом модулу и комутациони модул на основу њих мора да конфигурише проспајања којима повезује жељене парове улаз-излаз, а на великим брзинама то може бити ограничавајући фактор.

3.2. *Load-Balanced Birkhoff-von Neumann* класа пакетских комутатора

У овом потпоглављу ће бити представљена рјешења заснована на *Load-Balanaced Birkhoff-von Neumann* (*LB-BvN*) класи комутатора. У претходном потпоглављу су описани принципи рада најпопуларнијих пакетских комутатора са баферима на улазним портовима. Међутим, главни недостатак већине представника ове класе пакетских комутатора јесте неопходност постојања централног распоређивача који у сваком временском слоту мора да извршава алгоритам задужен за упаривање улазних и излазних портова. Такође, представљен је *SGS* алгоритам који не користи централни распоређивач, већ дистрибуирани. Али и код овог алгоритма остаје проблем конфигурисања комутационог модула по случајном принципу у сваком временском слоту. Такође, да би се постигла 100% пропусност најчешће се захтијева и убрзање, што може бити проблематично за веће вредности убрзања. Међутим, најквалитенија рјешења са баферима на улазу могу да користе мање вредности убрзања (типично убрзање 2) које могу релативно једноставно да се реализују данашњом технологијом што се тиче убрзања интерних линкова, али проблем је што се захтева и еквивалентно убрзање конфигурисања самог комутационог модула што може бити проблематично имајући у виду већ поменути проблем са "случајним конфигурацијама". Врло популарно рјешење за превазилажење ових проблема јесу управо пакетски комутатори који су засновани на *LB-BvN* архитектури. У наредним одељцима ће бити представљени принцип рада основног *Birkhoff-von Neumann* (*BvN*) комутатора и *LB-BvN* комутатора, а потом и најбоља рјешења заснована на *LB-BvN* архитектури.

3.2.1. Birkhoff-von Neumann (BvN) комутатор

На слици 3.2.1 је представљена *BvN* архитектура. И овдје се користе бафери са *VOQ* редовима за чекање. На сваком улазном порту се налази *N VOQ* редова, тако да су на улазним портовима пакети разврстани по *VOQ* редовима у складу са тиме на који излазни порт треба да буду прослијеђени.



Слика 3.2.1. *BvN* архитектура

Овај тип пакетског комутатора носи име по знаменитим математичарима Џорџу Дејвиду Биркофу и Џону вон Најману јер упаривање улазних и излазних портова користи принцип декомпозиције матрице који су они описали у својим радовима [121] и [122].

Матрица се може искористити да се представи дистрибуција саобраћаја између улазних и излазних портова. Наиме, сваки члан матрице $\lambda_{i,j}$ представља нормализовану вриједност саобраћаја (удио у укупном саобраћају који стиже на улазни порт *i*) који са улазног порта *i* треба да буде прослијеђен на излазни порт *j*. При томе, посматра се сценарио гдје важи да укупна количина саобраћаја на било ком улазном или излазном порту мора бити мања од 100%, то јест:

$$\sum_{j=1}^{N} \lambda_{i,j} \le 1 \text{, } i=1,2,\ldots, N \tag{3.2.1}$$

и

$$\sum_{i=1}^{N} \lambda_{i,j} \le 1 \text{, } j=1,2,\ldots, N \tag{3.2.2}$$

Биркоф и вон Најман су показали да постоји скуп позитивних бројева ϕ_k и матрица пермутација P_k, k=1,2,…,K, при чему важи $K \le N^2 - 2 \cdot N + 2$ тако да су испуњене сљедеће релације:

$$\lambda \le \sum_{k=1}^{K} \phi_k \cdot P_K \tag{3.2.3}$$

и

$$\sum_{k=1}^{K} \phi_k = 1 \tag{3.2.4}$$

гдје је λ матрица саобраћаја између улазних и излазних портова комутатора.

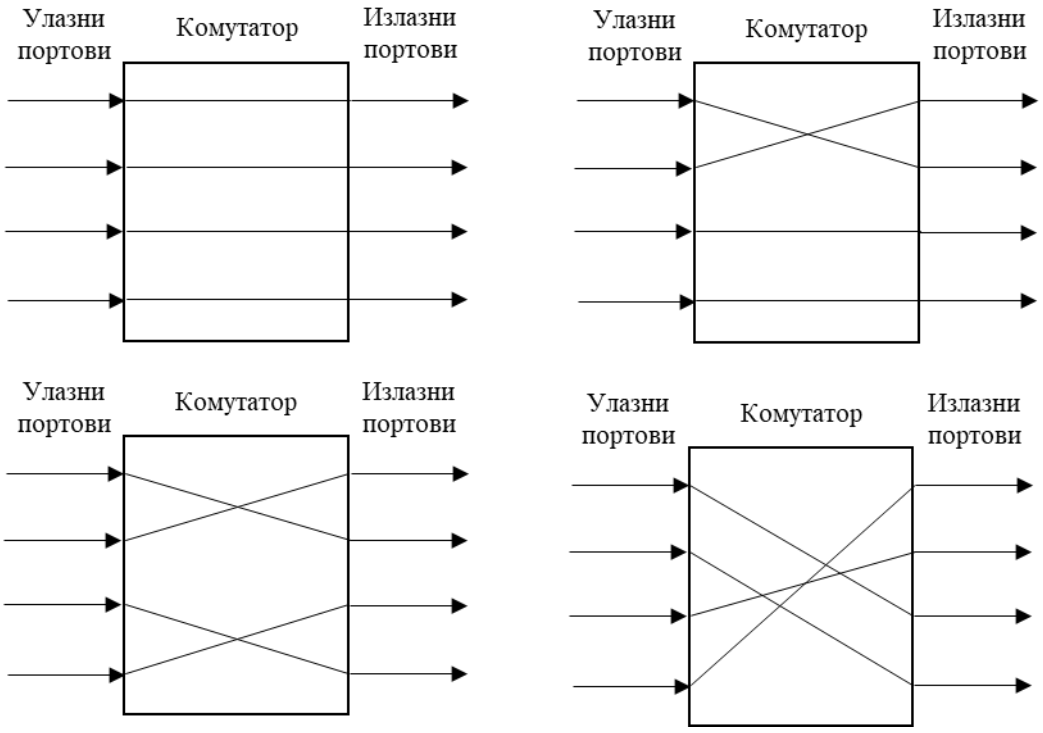
Отуда је на основу матрице саобраћаја могуће одредити, принципом декомпозиције матрице, скуп конфигурација комутационог модула који може да опслужи дотични саобраћај без губитака. Свака матрица пермутације представља једну конфигурацију, а коефицијент ϕ_k представља фреквентност појављивања *k*-те конфигурације. На примјер, ако је матрица саобраћаја:

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ \frac{1}{2} & \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{2} \end{pmatrix}$$

на основу Биркхоф-вон Најманове теореме, добијамо сљедеће матрице пермутација P_1, P_2, P_3 и P_4 са одговарајућим тежинским факторима ϕ_1, ϕ_2, ϕ_3 и ϕ_4:

$$\frac{1}{4} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \frac{1}{4} \cdot \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \frac{1}{4} \cdot \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \frac{1}{4} \cdot \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Очигледно за дати пример постоје 4 конфигурације које се подједнако често јављају. Стога се може направити детерминистички редослед од наведене 4 конфигурације које би се јављале по једном у 4 узастопна временска слота, и тај образац би се понављао у времену и опслуживао дати саобраћај без губитака. Предност овог приступа је што након иницијалног прорачуна, више нема потребе за новим прорачунима, односно конфигурисање комутационог модула више није случајно, већ детерминистичко. Изглед конфигурација за дати пример је дат на слици 3.2.2. Доказано је да *BvN* пакетски комутатор постиже 100% пропусност, и то без интерног убрзања за било коју задату матрицу саобраћаја која задовољава формуле 3.2.1 и 3.2.2 [123,124].



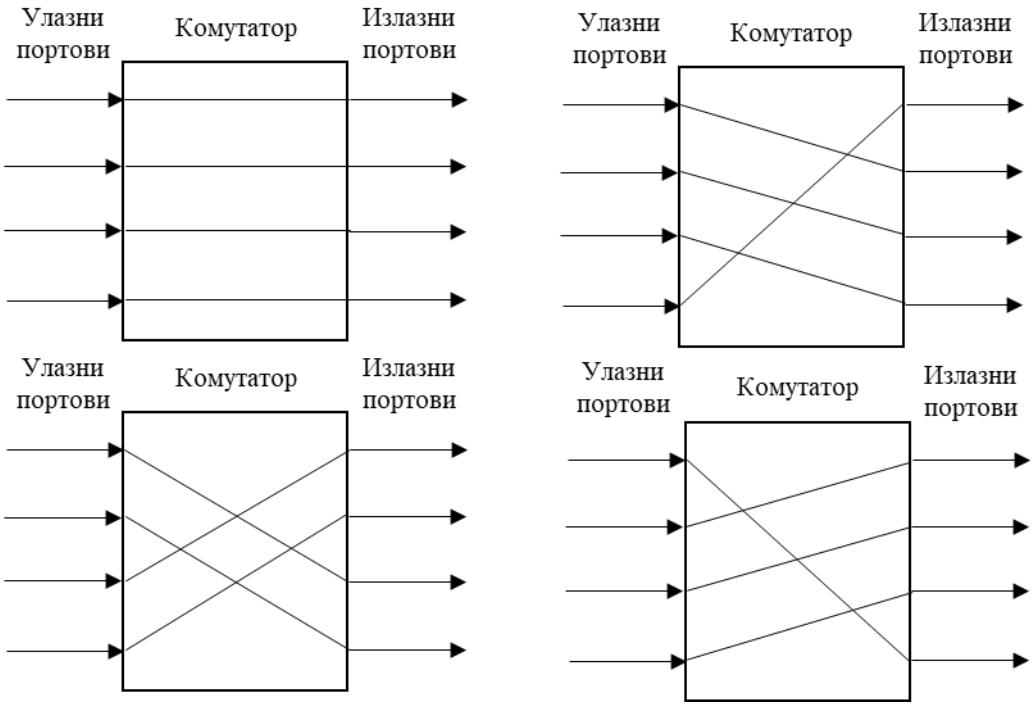
Слика 3.2.2. Шаблони конфигурисања *BvN* комутатора за дати примјер

Међутим, иако представља врло атрактивно рјешење, главни недостатак овог рјешења јесте комплексност израчунавања декомпозиције матрице које је $O(N^{4.5})$, док је број матрица пермутација које се могу добити на основу овог алгоритма $O(N^2)$ [123,124]. Сам проблем комплексности прорачуна конфигурација не би био критичан када би се извршавао само једном. Међутим, проблем настаје услед тога што матрица саобраћаја није константна већ се мења у времену. То значи да би било потребно континуално процењивати матрицу саобраћаја и у складу с њеним променама рачунати нове матрице пермутације тј. конфигурације. Ту до изражаја долази комплексност израчунавања матрица пермутације, а додатно постоји и проблем мерења саобраћаја, односно процене матрице саобраћаја у реалном времену јер та процена утиче на перформансе *BvN* пакетског комутатора.

3.2.2. Load-Balanaced Birkhoff-von Neumann (LB-BvN) комутатор

BvN пакетски комутатор би могао да оствари идеалне перформансе када би матрица саобраћаја била константна у времену. Такође, уочено је да када је долазни саобраћај униформан тј. равномјерно распоређен по свим улазним и излазним портовима, да би у периоду од N слотова сваки улазни порт требало да буде по једном повезан са сваким од

излазних портова. Отуда је лако креирати скуп од *N* конфигурација где би сваки улаз био спојен са сваким излазом тачно једном у скупу конфигурација. Један пример, који се типично и користи у пракси је приказан на слици 3.2.3. Отуда би идеално било ако би долазни саобраћај био униформан. Ово својство је искоришћено за конструкцију *Load-Balanaced Birkhoff-von Neumann* (*LB-BvN*) пакетских комутатора где се балансирањем саобраћаја покушава униформисати произвољан долазни саобраћај где би се онда на њега могао применити *BvN* принцип, али без потребе за константним прорачунавањем конфигурација или мерењем матрице саобраћаја.



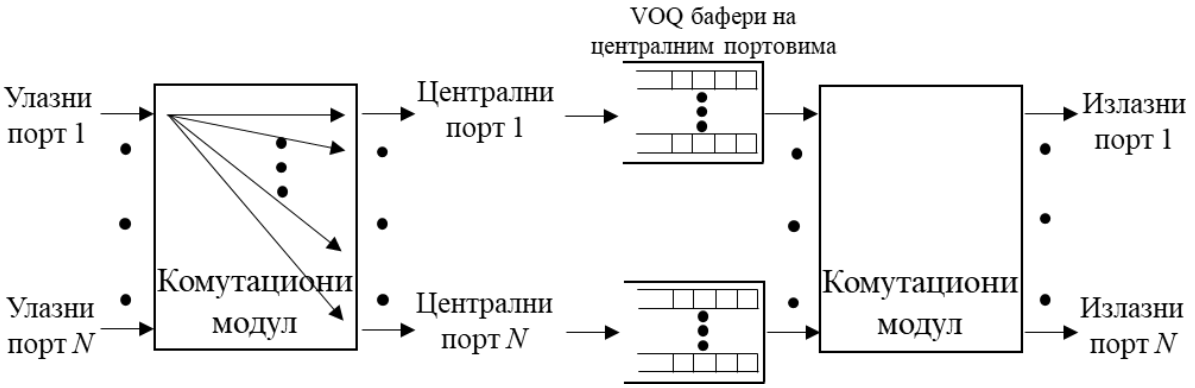
Слика 3.2.3. Примјер конфигурација *BvN* комутатора за униформни саобраћај

Load-Balanaced Birkhoff-von Neumann (*LB-BvN*) комутатор [125], као што се на основу имена може закључити, заснива се на *BvN* комутатору. Показано је, да би у случају када је долазни саобраћај равномјерно распоређен по свим улазним и излазним портовима, шаблон повезивања улазних и излазних портова у току једног циклуса био познат, без потребе за сталним прорачунавањем конфигурација. На тај начин би се превазишао основни проблем *BvN* комутатора, а то је управо извршавање комплексног *BvN* алгоритма за декомпозицију матрице саобраћаја. На слици 3.2.4 је дата архитектура *LB-BvN* пакетског комутатора. *LB-BvN* комутатор се састоји од два комутациона модула. Улазни портови првог комутационог модула представљају улазне портове *LB-BvN* комутатора, излазни портови другог комутационог модула представљају излазне портове *LB-BvN* комутатора док се између два комутациона модула налазе тзв. централни портови.

Сваки централни порт има *N VOQ* редова, по један за чување пакета које треба прослиједити на сваки од излазних портова, респективно. Задатак првог комутационог модула јесте да долазни саобраћај равномјерно распореди по централним портовима. Конфигурација овог комутационог модула је једноставна; у току једног циклуса (циклус садржи *N* временских слотова), сваки улазни порт је тачно једном повезан са сваким од централних портова. Други комутациони модул је класични *BvN* комутатор. Међутим, како је захваљујући првом комутатору извршено равномјерно распоређивање саобраћаја по

централним портовима, на основу *BvN* алгоритма је лако израчунати конфигурацију другог комутатора. Наиме, резултат *BvN* алгоритма је очигледан; у оквиру једног циклуса, сваки централни порт би требало да буде повезан тачно једном са сваким од излазних портова. На овај начин је превазиђен основни проблем оригиналног *BvN* комутатора, а то је комплексност извршавања *BvN* алгоритма. *LB-BvN* комутатор има неколико предности:

- **Скалабилност.** Пошто су конфигурације комутационих модула унапријед познате, нема потребе за извршавањем алгоритма за прорачун конфигурација, тј. комплексност је $O(1)$.
- **100% пропусност.** *LB-BvN* комутатор постиже 100% пропусност за дозвољене саобраћајне сценарије - под дозвољеним сценаријима подразумевамо да ниједан излазни порт није преоптерећен. Напоменимо да теоретски може да се креира сценарио у ком ће пропусност бити мала за случај када се балансирање саобраћаја не ради правилно, али у овом поглављу се разматрају рјешења која немају тај проблем, односно балансирање саобраћаја се ради правилно.
- **Добре перфромансе.** *LB-BvN* комутатор постиже добре перформансе, нарочито при високим оптерећењима.



Слика 3.2.4. Архитектура *LB-BvN* пакетског комутатора

Међутим, *LB-BvN* комутатор има један други недостатак, а то је поремећај у редоследу пакета што негативно утиче на *TCP* токове. *TCP* протокол осим контроле грешке и загушења има задатак да осигура правилан редослед примљених пакета на пријему. Може да се деси да услед поремећаја редоследа дође до погрешног закључка да је дошло до загушења у мрежи од стране *TCP* протокола што као последицу може да има смањење протока погођеног *TCP* тока иако загушење у мрежи не постоји. Ова ситуација је недопустива, па се зато од пакетских комутатора захтијева да се приликом прослеђивања пакета избјегне поремећај њиховог редоследа. Управо је ово кључни недостатак *LB-BvN* комутатора до кога долази због тога што пакети једног тока од улаза до излаза иду различитим путевима кроз сам пакетски комутатор (постоји N различитих путева што се може видети са слике 3.2.4) а кашњења на тим путевима могу бити различита јер долази до мешања са другим токовима који су намењени истом излазу. Услед разлике у кашњењима може доћи до поремећаја у редоследу пакета тока, јер каснији пакет у току може стићи раније до излаза у односу на неки ранији пакет зато што је имао мање кашњење у пролазу кроз комутатор. На кашњење утиче време проведено у реду за чекање у централном порту. Предложено је много рјешења за превазилажење овог проблема. Генерално, постоје три приступа решавању овог проблема:

- употреба ресеквенционих бафера на излазним портовима који исправљају редослед пакета на излазима.
- слање пакета у фрејмовима (*frame-based*) тако да сви пакети у истом фрејму имају једнако кашњење унутар комутатора.
- употреба одређених специфичних конфигурација комутационих модула које спречавају поремећај редоследа пакета.

У наставку ће бити представљена најпознатија рјешења за сваку од наведених категорија.

3.3. *LB-BvN* комутатори са ресеквенционим баферима на излазним портовима

У оквиру овог потпоглавља биће представљени најпознатији *LB-BvN* пакетски комутатори који користе ресеквенционе бафере на излазним портовима за решавање поремећаја редоследа пакета. Ако пакети на излазни порт стигну ван редоследа, они се смјештају у ресеквенциони бафер на том порту док не стигну пакети који недостају. Затим се пакети у правилном редоследу шаљу даље у мрежу.

3.3.1. First-Come-First-Serve (FCFS) комутатор

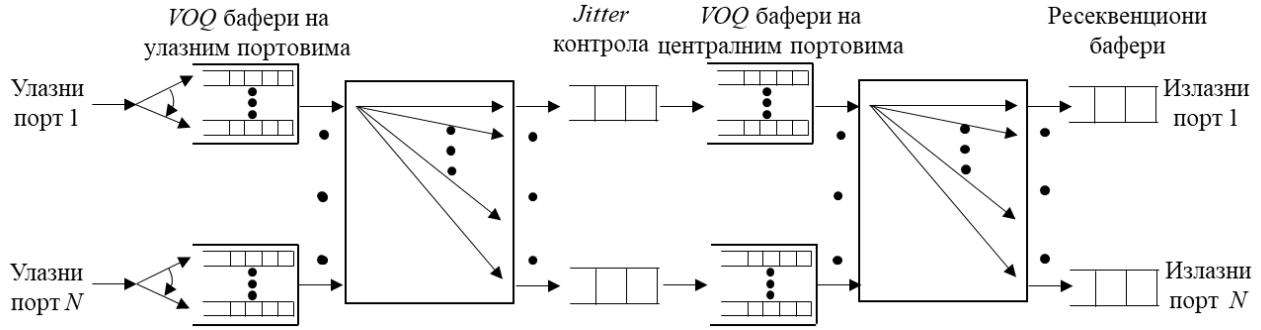
FCFS комутатор је предложен у [126]. На слици 3.3.1 је дата архитектура овог комутатора. У односу на *LB-BvN* комутатор код овог рјешења је на сваком улазном порту додато по *N VOQ1* редова за чекање и *N flow-splitter*-а, док су на сваком излазном порту додати ресеквенциони бафери. Као и код *LB-BvN* комутатора, између два комутациона модула се на сваком централном порту налази по *N VOQ2* редова за чекање. Принцип рада овог комутатора је сљедећи. На сваком улазном порту, за сваки ток пакета на том порту постоји *flow-splitter* који показује у који *VOQ1* ред треба смјестити сљедећи пакет тог тока. Овај *flow-splitter* се ажурира по *round-robin* принципу. На тај начин, пакети свих токова на неком улазном порту се равномјерно распоређују на свих *N VOQ1* редова на том порту. Конфигурације оба комутациона модула су детерминистичке, и одређене су сљедећим формулама:

$$i=(j+t) \bmod N \tag{3.3.1}$$

и

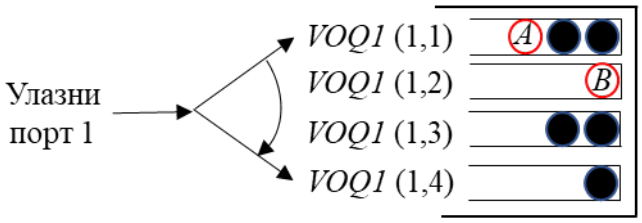
$$k=(j+t) \bmod N \tag{3.3.2}$$

гдје су са i означени улазни портови, са j централни портови и са k излазни портови.



Слика 3.3.1. Архитектура *FCFS* комутатора

Како пакети истог тока могу имати различита кашњења у пролазу са улазних на централне портове, *FCFS* комутатор имплементира *jitter* контролу на централним портовима, да би се исправио редослед пакета. Ово се постиже на начин што се свим пакетима уноси додатно кашњење које може бити до $N\cdot(N\text{-}1)$ временских слотова, прије него што ти пакети буду смјештени у одговарајуће *VOQ2* редове. На овај начин, сви пакети ће имати кашњење од $N\cdot(N\text{-}1)$ временских слотова од тренутка када су послати са улазних портова док не буду смјештени у *VOQ2* редове. Имплементација *jitter* контроле повећава хардверску комплексност и утиче на повећање кашњења до $N\cdot(N\text{-}1)$ временских слотова. Међутим, ни ово не гарантује да ће пакети на излазне портове стићи у правилном редоследу, јер количине пакета у *VOQ2* редовима могу бити различите. Да би се исправио редослед пакета на излазним портовима, имплементирани су ресеквенциони бафери, што такође повећава комплекност и кашњење пакета. На следећем примјеру је објашњен принцип рада *FCFS* комутатора.



Слика 3.3.2. Пример рада *FCFS* комутатора

Посматрају се два узастопна пакета тока $F(1,1)$, који су означени са A и B, при чему је пакет A први стигао на улазни порт 1. Ови пакети ће бити смјештени у $VOQ1(1, j)$ и $VOQ1(1, (j+1)$ **mod** $N)$, респективно. У конкретном примјеру, може се претпоставити да су смјештени у $VOQ1(1,1)$ и $VOQ1(1,2)$, као што је показано на слици 3.3.2. Како се у $VOQ1(1,1)$ испред пакета A налазе већ два пакета, јасно је да ће пакет A стићи на одговарајући централни порт $2\cdot N$ слотова након пакета B. Зато је на централним портовима имплементирана *jitter* контрола. Наиме, пакет B ће бити додатно закашњен за $2\cdot N$ временских слотова. Овим се постиже да пакети A и B у истом временском слоту буду смјештени у одговарајуће *VOQ2* редове на централним портовима. Међутим, број пакета у различитим *VOQ2* редовима се такође може разликовати, што ће опет узроковати поремећај редоследа пакета на излазном порту. Да би се ријешио овај проблем на излазним портовима су имплементирани ресеквенциони бафери. Након исправљања редоследа пакета, они се смјештају у одговарајуће бафере на излазним портовима и шаљу даље у мрежу. Величина ресеквенционих бафера мора бити бар N^2 пакета да не буде губитака, што је доста велико, нарочито за велико N.

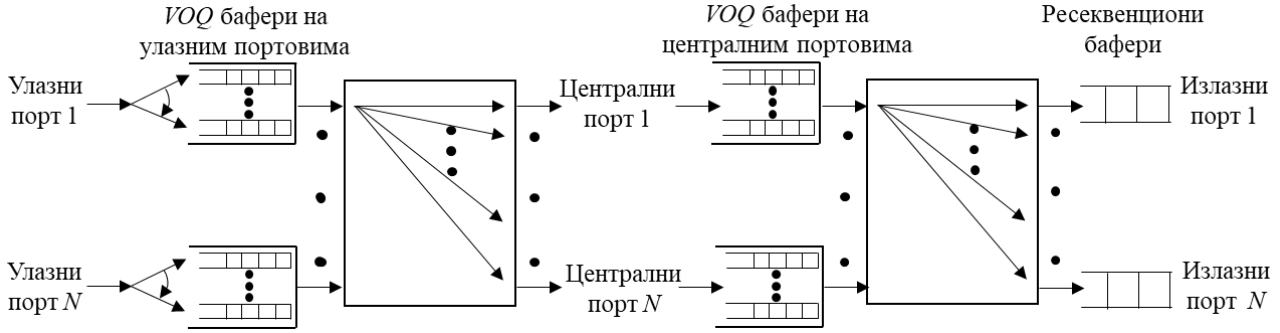
Дакле, *FCFS* комутатор решава проблем поремећаја редоследа пакета, али се притом захтијева сљедеће [36]:

- Имплементација бафера са *VOQ1* редовима на улазним портовима и ресеквенционих бафера на излазним портовима.
- Уноси се додатно кашњење од $N\cdot(N\text{-}1)$ временских слотова услед *jitter* контроле.

3.3.2. Earliest Deadline First (EDF) комутатор

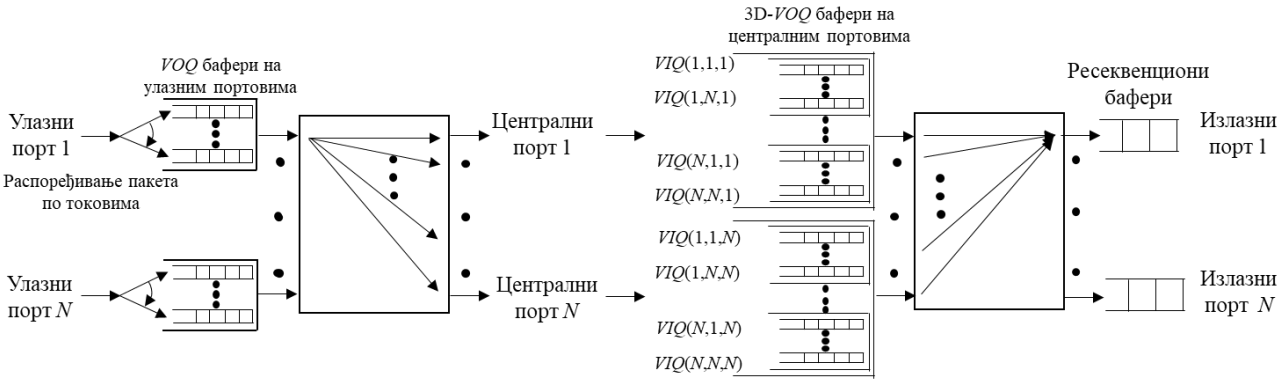
EDF комутатор [126] је још један примјер комутатора који користи ресеквенционе бафере на излазним портовима за решавање проблема поремећаја редоследа пакета. Архитектура овог комутатора (слика 3.3.3) је слична као код *FCFS* комутатора, с том разликом да није имплементирана *jitter* контрола на централним портовима. Умјесто тога,

код *EDF* комутатора се сваком пакету додјељује одређена временска ознака, на основу коју се одређује када пакет треба да напусти одговарајући *VOQ2* ред. Ова ознака на примјер, може бити вријеме доласка пакета у комутатор. Пакет који има мању временску ознаку ће раније бити прослијеђен из *VOQ2* реда на одговарајући излазни порт. Међутим, како пакети у *VOQ2* редове пристижу на некоординисан начин, *HoL* пакет не мора нужно имати најмању временску одредницу. Тражење пакета с најмањом временском ознаком може бити рачунарски врло захтијеван и комплексан задатак. Такође, како се пакети истог тока прослеђују преко различитих централних портова, доћи ће до поремећаја редоследа пакета. Зато су на излазним портовима имплементирани ресеквенциони бафери величине $2 \cdot N^2 - 2 \cdot N$ пакета како би се исправио њихов редослед.



Слика 3.3.3. Архитектура *EDF* комутатора

EDF-3DQ комутатор представља унапређење *EDF* комутатора, код кога су на централним портовима умјесто класичних *VOQ* редова имплементирани тродимензионални бафери (*3DQ*). На слици 3.3.4 је дата архитектура овог комутатора. Дакле, на сваком централном порту, умјесто *N VOQ2* редова се налази N^2 *3DQ* бафера. Код *EDF* комутатора, сваки *VOQ2* ред садржи пакете које треба прослиједити на исти излазни порт. С друге стране, код *EDF-3DQ* комутатора на сваком централном порту пакети су разврстани не само по излазном порту на који треба да буду прослијеђени већ и по улазном порту на који су стигли у комутатор. На примјер, *3DQ*(1,1,1) садржи само пакете тока *F*(1,1), док *3DQ*(2,1,1) садржи само пакете тока *F*(2,1). Код класичног *EDF* комутатора, пакети из ових токова би могли завршити у истом *VOQ2* реду. Дакле, идеја код *EDF-3DQ* комутатора је да се пакети разврстају по токовима, тако да *HoL* пакет увијек има најмању временску ознаку. Самим тим, код овог рјешења је довољно да се упореде временске ознаке *N HoL* пакета да би се пронашао пакет с најмањом временском ознаком. Поређења ради, код *EDF* комутатора је неопходно упоредити временске ознаке свих пакета у *VOQ2* реду.



Слика 3.3.4. Архитектура *EDF-3DQ* комутатора

На слици 3.3.5, је дато поређење начина рада *EDF* и *EDF-3DQ* комутатора. Са слике 3.3.5 б) се види да код *EDF-3DQ* комутатора *HoL* пакети имају најмању временску ознаку. Отуда је за проналажење пакета с најмањом временском ознаком довољно упоредити временске ознаке *N HoL* пакета. Међутим, иако је код *EDF-3DQ* комутатора дјелимично олакшано проналажење пакета с најмањом временском ознаком, и даље је неопходно поредити временске ознаке *N HoL* пакета у сваком временском слоту, што ограничава величину комутатора. Такође, имплементација N^3 *3DQ* бафера као и ресеквенционих бафера величине $2 \cdot N^2 - 2 \cdot N$ пакета значајно утиче на трошкове хардверске имплементације комутатора [36].

	$F(3,1)$ $t=15$	$F(2,1)$ $t=12$	$F(2,1)$ $t=8$	$F(1,1)$ $t=10$

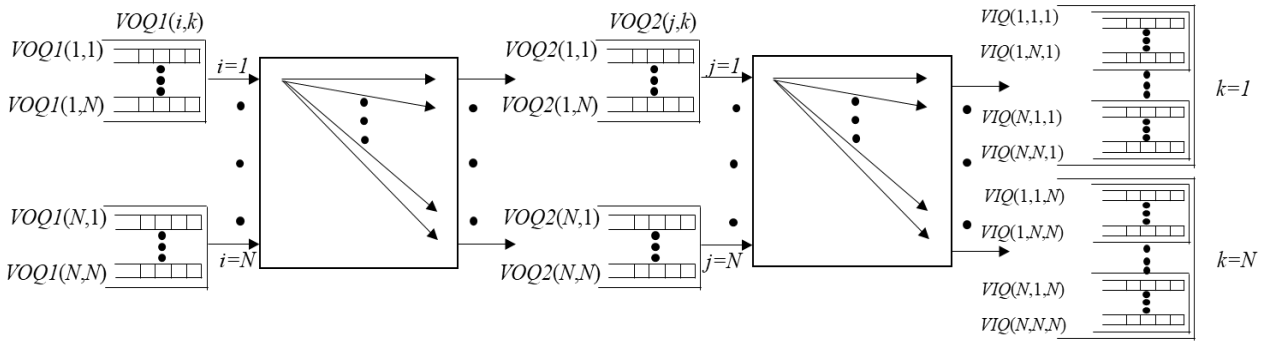
а)

		$F(1,1)$ $t=10$
	$F(2,1)$ $t=12$	$F(2,1)$ $t=8$
		$F(3,1)$ $t=15$

б)

Слика 3.3.5. Стање у баферима код а) *EDF* и б) *EDF-3DQ* комутатора

3.3.3. Byte-Focal (BF) комутатор



Слика 3.3.6. Архитектура *BF* комутатора

Byte-Focal (*BF*) комутатор [127] је врста *LB-BvN* комутатора који за решавање проблема поремећаја редоследа пакета такође користи ресеквенционе бафере (чија је величина N^2 пакета) на излазним портовима. На слици 3.3.6 је приказана архитектура овог рјешења. Састоји се од два комутациона модула, и бафера на улазним портовима *i*, бафера на централним портовима *j* и бафера на излазним портовима *k*. Оба ова комутациона модула имају унапријед одређене конфигурације које се периодично понављају. На сваком улазном порту постоји бафер са по *N VOQ1* редова за чекање, при чему се у сваком *VOQ1* реду смјештају пакети једног тока. На централним портовима се такође налазе бафери са по *N VOQ2* редова, гдје се у сваком *VOQ2* реду смјештају пакети које треба прослиједити на један излазни порт. Конфигурације оба комутациона модула су исте као и код *FCFS* комутатора.

При том, улазни портови за сваки свој ток имају показивач који показује на који централни порт треба прослиједити сљедећи пакет тог тока. На тај начин се гарантује да се број пакета истог тока који су прослеђени на сваки од централних портова разликује највише за један између било која два од њих.

Како се пакети истог тока прослеђују преко различитих централних портова, генерално ће доћи до поремећаја њиховог редоследа. Зато су на излазним портовима имплементирани ресеквенциони бафери који користе *Virtual Input Queue* (*VIQ*) редове. На сваком излазном порту се налази N скупова *VIQ* редова, по један за сваки улазни порт. У оквиру сваког скупа *VIQ* редова се налази и N логичких редова, по један за сваки од централних портова. То значи да су пакети на излазним портовима разврстани не само по улазним портовима са којих су дошли, већ и по централним портовима преко којих су прослијеђени на излазне портове. Ово гарантује да пакети у сваком *VIQ* имају правилан редослед. Како су пакети истог тока распоређени по различитим *VIQ*, сваки излазни порт има показивач који за сваки ток показује на *VIQ* у коме треба да се нађе сљедећи пакет тог тока тј. *Head of Flow* (*HoF*) пакет. Како су пакети сваког тока равномјерно распоређени по централним портовима, ажурирање показивача је једноставно. Када излазни порт пошаље пакет тока $F(i,k)$ из VIQ (i, j, k), показивач се ажурира да показује на VIQ $(i,(j+1) \bmod N, k)$. На овај начин се одржава правилан редослед пакета у оквиру истог тока. Како се може десити да излазни порт има више пакета који се могу прослиједити, сваки излазни порт има додатни ред за чекање тзв. *Department Queue* (*DQ*). *DQ* је у суштини *FIFO* бафер са N локација, и функционише по *round-robin* принципу. У њему су смјештени индекси *VIQ* редова, али највише један из скупа *VIQ* редова у којима се чувају пакети истог тока. Када на излазни порт дође пакет који се може прослиједити у мрежу, индекс тог *VIQ* реда се додаје на крај *DQ*. Када се неки пакет пошаље са излазног порта у мрежу, онда се индекс *VIQ* реда у ком је био смјештен тај пакет брише са чела листе, а на крај *DQ* се додаје индекс *VIQ* реда у ком је смјештен сљедећи пакет тог тока (ако постоји). Предност коришћења *VIQ* редова и *DQ* је то што је временска комплексност тражења и прослеђивања пакета $O(1)$. Дакле, у сваком временском слоту, излазни порт прослеђује пакет с врха *DQ*, док сваки скуп *VIQ* редова користи показиваче ради провјере да ли је *HoF* пакет стигао.

Такође, и на улазним портовима може постојати више пакета доступних за слање. Како би се постигле што боље перформансе, одабир пакета за слање на улазним портовима је врло важан. Како је већ речено, пакети тока $F(i,k)$ су циклично распоређени по централним портовима. То значи, да ако је последњи пакет тока $F(i,k)$ био послат на централни порт j, онда сљедећи пакет тог тока мора бити послат на централни порт $(j+1) \bmod N$. Тако, сваки $VOQ1(i,k)$ има показивач $P_{i,k}(t)$ који указује на централни порт коме треба послати сљедећи пакет из тог *VOQ1* реда. Са $S_j(t)=\{VOQ1(i,k) \mid P_{i,k}(t)=j\}$ се дефинише скуп свих *VOQ1* редова који могу послати пакет централном порту j. Када се улазни порт i, у неком временском слоту t, повеже са централним портом j, распоређивач бира из којег *VOQ1* реда из скупа $S_j(t)$ ће се послати пакет. У наставку су представљена четири начина одабира пакета.

- *Round-robin* (*RR*). Овај алгоритам ради на једноставан начин. Када неки *VOQ1* ред пошаље пакет, показивач се ажурира да показује на сљедећи *VOQ1* ред.
- *Longest Queue First* (*LQF*). Овај алгоритам бира пакет из *VOQ1* реда који има највише пакета.
- *Fixed Threshold Scheme* (*FTS*). Иако *LQF* постиже добре перформансе, проналажење најдужег *VOQ1* реда може бити временски захтијевно за комутаторе са већим бројем портова. *FTS* се заснива на томе да се посматрају само они *VOQ1*

редови чија дужина превазилази унапријед одређени праг (*TH*), који је једнак *N* пакета. Нека је $q_{i,k}(t)$ дужина *VOQ1*(i,k) у тренутном временском слоту *t*, и нека је $q_{i,s}(t)$ дужина *VOQ1* реда који тренутно шаље пакет. Дефинише се подскуп $S_j^{'}(t) = \{VOQ1(i,k) \mid VOQ1(i,k) \in S_j(t) \text{ and } q_{ik} \geq TH\}$ који представља скуп свих *VOQ1* редова који имају више од *TH* пакета и који могу послати пакет централном порту *j*. *FTS* алгоритам ради по сљедећем принципу:

- У сваком временском слоту, ако је $q_{i,s}(t) \geq TH$, наставља се слање пакета из овог $VOQ1_{i,s}$.
- Ако је $q_{i,s}(t)<TH$, распоређивач бира сљедећи *VOQ1* ред из подскупа $S_j^{'}(t)$ по *round-robin* принципу.
- Ако је $S_j^{'}(t)$ празан (тј. ако не постоји *VOQ1* ред који има више од *TH* пакета, и ако је $q_{i,s}(t) > 0$, наставља се слање пакета из тог *VOQ* реда.
- Ако је $q_{i,s}(t) = 0$, распоређивач бира сљедећи *VOQ1* ред из скупа $S_j(t)$ по *round-robin* принципу

- *Dynamic Threshold Scheme* (*DTS*). Када се ради о комутаторима са великим бројем портова, постављање прага да буде једнак броју портова комутатора, утиче на велико просјечно кашњење. Разлог је тај што број пакета у *VOQ1* мора да досегне велику вриједност *N*, прије него што тај *VOQ1* пређе праг. Прије досезања прага, овакав *VOQ1* ред мора да се такмичи са другим *VOQ1* редовима који имају мали број пакета. Како би се боље идентификовали *VOQ1* редови који имају већи број пакета, предложен је *DTS* алгоритам. Динамички праг (*TH*) је постављен на $\frac{Q(t)}{N}$, гдје је $Q(t)$ укупан број пакета на посматраном улазном порту у временском тренутку *t*, тј. то је просјечна дужина свих *VOQ1* редова на том улазном порту. Овај алгоритам даље ради на исти начин као и *FTS*, с том разликом да се праг одређује на другачији начин.

У [128] је показано да *LQF* постиже најбоље перформансе. Међутим, овај алгоритам је знатно комплекснији за имплементацију у односу на *FTS* и *DTS*. Показано је да *DTS* постиже перформансе упоредиве са *LQF* алгоритмом. Такође, за разлику од *FTS*, *DTS* алгоритам се прилагођава промењивом улазном оптерећењу, и зато постиже боље перформансе, док је комплексност имплементације и даље ниска.

3.4. *Frame-based LB-BvN* комутатори

У наредним потпоглављима ће бити представљени најпознатији *frame-based LB-BvN* комутатори. Код ових комутатора се пакети не посматрају појединачно, већ у оквиру фрејмова који се састоје од *N* пакета једног тока. Тек када се на неком улазном порту формира фрејм може се започети слање пакета из тога фрејма. Ови пакети се шаљу један за другим у *N* узастопних временских слотова. Идеја је да сви пакети приликом прослеђивања унутар комутатора имају једнако кашњење, па се на тај начин избјегава поремећај редоследа пакета и потреба за ресеквенционим баферима на излазним портовима. Наведеним начином слања се постиже да се пакети неког фрејма "не мијешају" са пакетима других токова, односно постиже се њихово исто кашњење у прослеђивању са улаза на излазе.

3.4.1. Full Frame First (FFF) комутатор

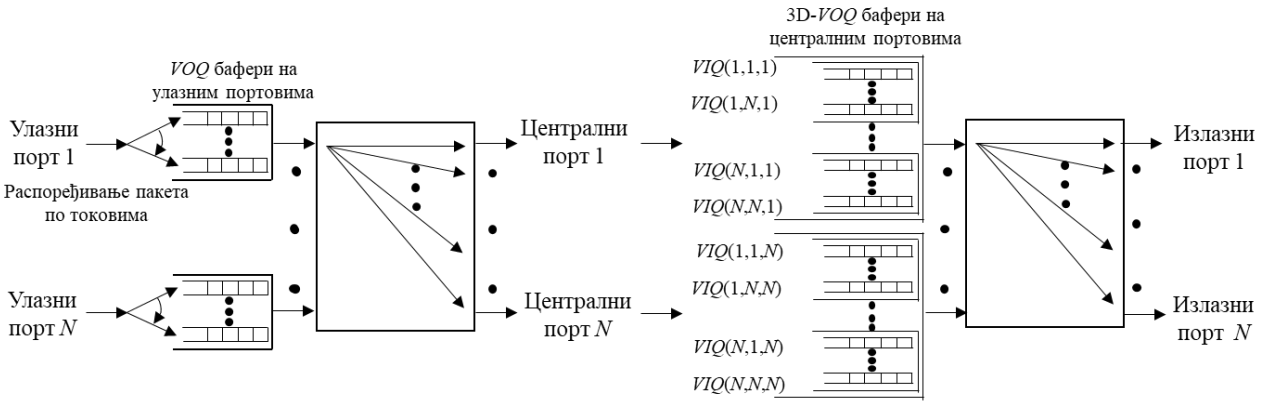
FFF комутатор је предложен у [128]. На слици 3.4.1 је дата архитектура овог пакетског комутатора. На сваком улазном порту се налази *flow splitter*, као и *N VOQ1* редова. У једном *VOQ1* реду се смјештају пакети које треба прослиједити на исти централни порт. Осим тога, на сваком централном порту се налазе *3DQ* бафери, који се састоје од N^2 *VOQ* редова. Принцип рада *FFF* комутатора је сљедећи. Пакети истог тока $F(i,k)$ су равномјерно распоређени по *N VOQ1* редова на улазном порту *i*. *Flow splitter* на сваком улазном порту чува показивач на *VOQ1* ред у који треба прослиједити сљедећи пакет сваког тока. Конфигурације оба комутациона модула су детерминистичке и одређене формулама:

$$i = (j+t) \bmod N \tag{3.4.1}$$

и

$$k = (j+t) \bmod N \tag{3.4.2}$$

гдје су са *i* означени улазни портови, са *j* централни портови и са *k* излазни портови.



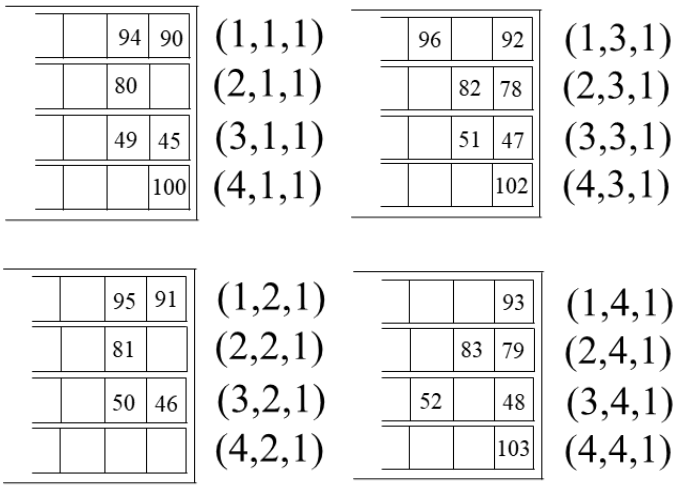
Слика 3.4.1. Архитектура *FFF* комутатора

У сваком временском слоту, пакет из једног *VOQ1* реда се прослеђује на одговарајући централни порт. Да би се избјегао проблем поремећаја редоследа пакета на излазним портовима, неопходно је пажљиво извршити одабир пакета за слање са централних портова. Код *FFF* комутатора, скуп кандидата *3DQ* бафера за ток $F(i,k)$ се састоји од пакета од *3DQ* бафера $(i,1,k)$, $(i,2,k)$, … , (i, N, k). При томе, треба имати на уму да сваки $3DQ(i,j,k)$ садржи пакете из само једног тока. Услед балансирања оптерећења пакети једног тока ће бити равномјерно распоређени на *N* различитих *3DQ* бафера. Такође, захваљујући *flow splitter*-у, ако је последњи пакет тока $F(i,k)$ смјештен у $3DQ(i,j,k)$, онда ће сљедећи пакет тог тока бити смјештен у $3DQ\ (i, (j+1) \bmod N, k)$.

Као што је већ речено на почетку, *FFF* припада класи комутатора који проблем поремећаја редоследа пакета решавају слањем пакета у фрејмовима. Код *FFF* комутатора, циклус се дефинише као низ од *N* узастопних временских слотова, а фрејм за ток $F(i,k)$ се састоји од пакета из $3DQ\ (i, p_{ik}, k)$, $(i, p_{ik}+1, k)$, … , (i,N,k), при чему је p_{ik} показивач на сљедећи пакет тока $F(i,k)$ који је на реду за слање. Фрејм је пун ако у сваком од ових *3DQ* бафера постоји бар по један пакет. Важно је примјетити, да ако је фрејм пун, онда се пакети из тог тока могу континуирано слати на излазни порт почев од пакета из бафера $3DQ(i, p_{ik}, k)$ до пакета из бафера $3DQ(i,N,k)$. На овај начин се спречава поремећај редоследа пакета. Алгоритам рада *FFF* комутатора се састоји од три корака:

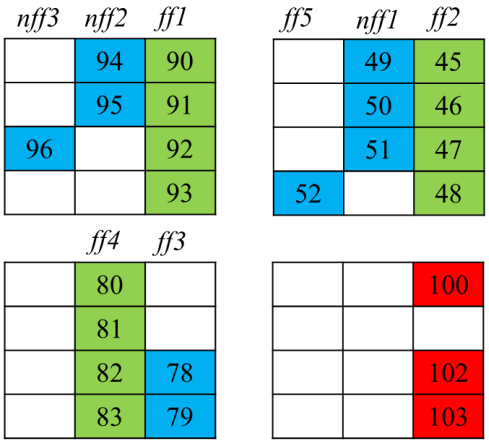
1) **Корак 1.** Сваки излазни порт k одређује који од фрејмова $f(i,k)$ који припадају том порту су пуни.
2) **Корак 2.** На сваком излазном порту постоји показивач на пуни фрејм, $p_{ff}(k)$. Излазни порт k тражи први пун фрејм, почињући од $p_{ff}(k)$. Када се нађе такав фрејм, $p_{ff}(k)$ се ажурира да показује на улазни порт ком припада тај фрејм.
3) **Корак 3.** На сваком излазном порту такође постоји и показивач на не-пуни фрејм, $p_{nff}(k)$. Уколико пуни фрејм не постоји, тражи се први не-пуни фрејм почев од показивача $p_{nff}(k)$. Када се пронађе такав фрејм, $p_{nff}(k)$ се ажурира по истом принципу као и p_{ff} показивач.

У наставку је дат примјер рада *FFF* комутатора. Посматра се 4x4 комутатор. На слици 3.4.2 је представљено стање *3DQ* бафера који одговарају излазном порту 1. Бројеви у баферима представљају редне бројеве пакета у одговарајућем току.



Слика 3.4.2. Примјер рада *FFF* комутатора

Такође, претпоставимо да је $p_{ff}(1)=4$, $p_{nff}(1)=2$, као и да је $p_{11}(1)=p_{31}(1)=p_{41}(1)=1$ и $p_{21}(1)=3$. Ради лакшег објашњења принципа рада алгоритма, на слици 3.4.3 је дат приказ бафера с тим да су пакети који су с истих улазних портова поређани један до другог. Прегледности ради, зеленом бојом су означени пакети који чине пуни фрејм, плавом бојом пакети који чине не-пуни фрејм, а црвеном они који не чине ни пуни ни не-пуни фрејм.



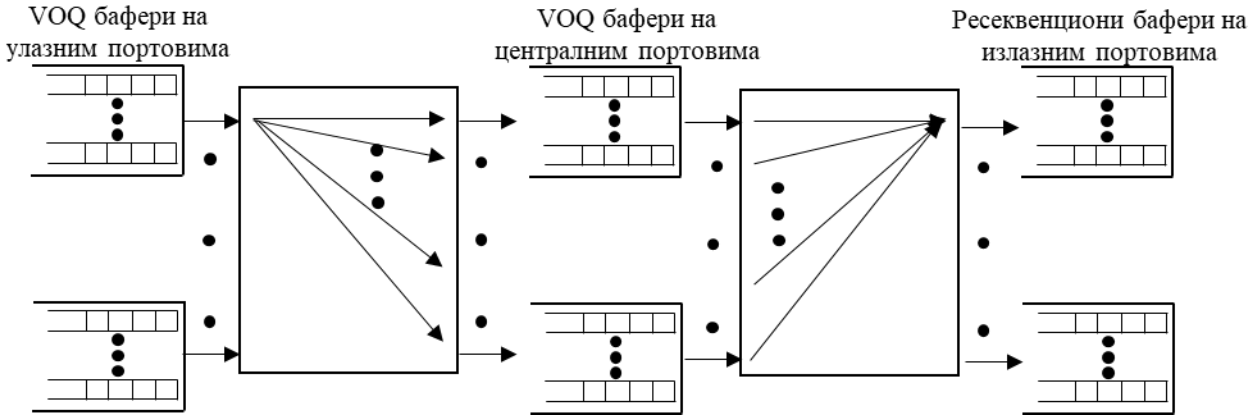
Слика 3.4.3. Стање у баферима *FFF* комутатора

На почетку првог циклуса излазни порт 1 провјерава да ли постоји пуни фрејм за тај порт. Како је $p_{ff}(1)=4$, за слање се бирају пакети из тока $F(1,1)$ с редним бројевима 90, 91, 92 и 93. Показивачи се ажурирају тако да је $p_{ff}(1)=1$ и $p_{11}(1)=1$. У другом циклусу такође постоји пуни фрејм за излазни порт 1. У овом циклусу се шаљу пакети који припадају току $F(3,1)$ с редним бројевима 45, 46, 47 и 48. Показивачи се ажурирају тако да је $p_{ff}(1)=3$ и $p_{31}(1)=1$. У трећем циклусу пуни фрејм за излазни порт 1 представљају пакети из тока $F(2,1)$ с редним бројевима 78 и 79 јер је $p_{21}(1)=3$ Показивачи се ажурирају тако да је $p_{ff}(1)=2$ и $p_{21}(1)=1$. У четвртом циклусу опет постоји пуни фрејм за слање. То су пакети с редним бројевима 80, 81, 82 и 83 који такође припадају току $F(2,1)$. Показивачи се ажурирају тако да је $p_{ff}(1)=2$ и $p_{21}(1)=1$. Пакети с редним бројевима 100, 102 и 103 не могу бити послати јер не чине ни пуни ни не-пуни фрејм јер фали пакет с редним бројем 101. У петом циклусу, не постоји пуни фрејм. Зато се за слање бирају пакети из тока $F(3,1)$ чији су редни бројеви 49, 50 и 51. Показивачи се ажурирају тако да је $p_{nff}(1)=3$ и $p_{31}(1)=4$. У шестом циклусу опет не постоји пуни фрејм, па се за слање бирају пакети из не-пуног фрејма који припадају току $F(1,1)$. То су пакети с редним броејвима 94 и 95. Показивачи се ажурирају тако да је $p_{nff}(1)=1$ и $p_{11}(1)=3$. У седмом циклусу пакет 52 представља пуни фрејм јер је $p_{31}(1)=4$, па је то пуни фрејм иако је у питању само један пакет. Показивачи се ажурирају тако да је $p_{ff}(1)=3$ и $p_{31}(1)=1$. У осмом циклусу не постоји пуни фрејм за слање, па се за слање бира не-пуни фрејм којег чини пакет 96 из тока $F(1,1)$. Показивачи се ажурирају тако да је $p_{nff}(1)=1$ и $p_{11}(1)=4$.

Иако *FFF* комутатор не захтијева имплементацију ресеквенционих бафера на излазним портовима, *3DQ* бафери утичу на повећање хардверске комплексности и трошкова имплементације. Осим тога, захтијева се и велика количина комуникације између портова ради проналажења пуних фрејмова [36].

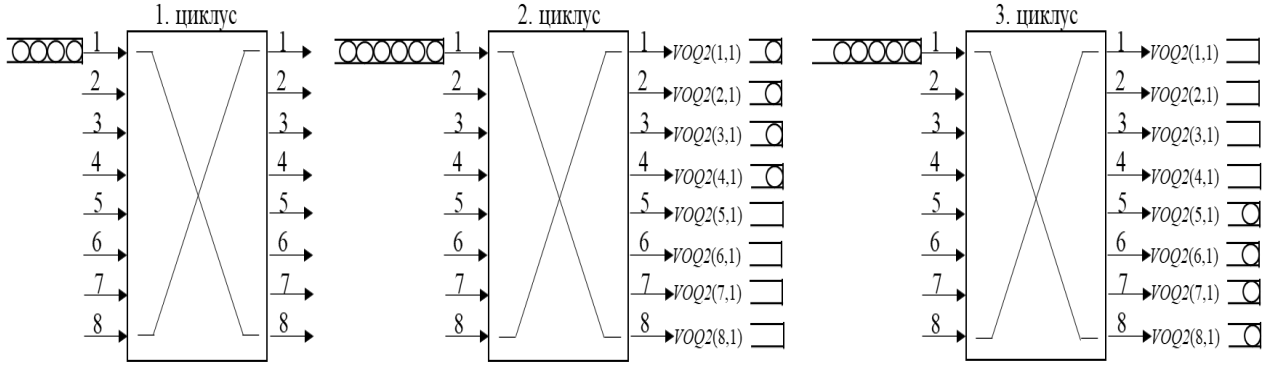
3.4.2. Full Ordered Frame First (FOFF) комутатор

FOFF комутатор је предложен у [129]. Шема *FOFF* комутатора је дата на слици 3.4.4. На сваком улазном порту постоји *N VOQ1* редова. Сваки од ових бафера служи за смјештање пакета који припадају истом току. На сваком централном порту, такође постоји *N VOQ2* редова. И на централним портовима се пакети истог тока смјештају у једном *VOQ2* реду. Осим тога, код *FOFF* комутатора на сваком излазном порту постоји *N* ресеквенционих бафера *RB VOQ*(*j*,*k*), по један за пакете са сваког од централних портова. Конфигурације оба комутациона модула су исте као и код *FFF* комутатора.



Слика 3.4.4. Архитектура *FOFF* комутатора

За разлику од *FFF* комутатора, *FOFF* комутатор дозвољава да дође поремећаја редоследа пакета. Код *FOFF* комутатора, циклус се дефинише као низ од *N* узастопних временских слотова, док пуни фрејм представља *N* пакета који припадају истом току. На почетку сваког циклуса, улазни порт бира из којег *VOQ1* реда ће слати пакете у току тог циклуса. Улазни порт бира пакете из оног *VOQ1* реда који садржи пуни фрејм. Уколико има више кандидата, бира се један у складу са *round-robin* принципом. Ако не постоји *VOQ1* ред који садржи пун фрејм, онда се бира неки *VOQ1* ред, опет у складу са *round-robin* принципом. Ако на сваком улазном порту постоји по један пуни фрејм, неће доћи до поремећаја редоследа пакета. Међутим, ако неки улазни порт шаље пакете из не-пуног фрејма, тада ће доћи до поремећаја редоследа пакета услед неједнаког оптерећења *VOQ2* редова. Међутим, у [129] је показано да је величина ресеквенционих бафера ограничена на N^2+1 пакета. Такође, слање пакета из не-пуног фрејма, осим поремећаја редоследа пакета може довести и до некоришћења ресурса првог комутационог модула. То даље утиче на повећање просјечног кашњења пакета унутар комутатора. Ово је објашњено на сљедећем примјеру, слика 3.4.5.



Слика 3.4.5. Примјер повећања просјечног кашњења код *FOFF* комутатора

Претпоставимо да улазни порт 1 одабере не-пуни фрејм од k пакета ($k<N$) за слање у сљедећем циклусу. Једноставности ради, претпоставља се да се први пакет овог фрејма шаље на централни порт j=1. Дакле, у току овог циклуса од N временских слотова улазни порт 1 ће послати свега k пакета, што значи да се у току N-k слотова неће слати пакети. У наредном циклусу, на улазном порту 1, фрејм са N-k пакета представља пуни фрејм, и тај фрејм се шаље у току тог циклуса. То значи да ће у овом циклусу, свега N-k пакета бити послато, иако је могуће да на улазном порту 1 постоји неки пуни фрејм. Последично, у току два циклуса (од $2 \cdot N$ временских слотова), биће послато укупно N пакета. Ово ће утицати на повећање просјечног кашњења приликом прослеђивања пакета унутар комутатора.

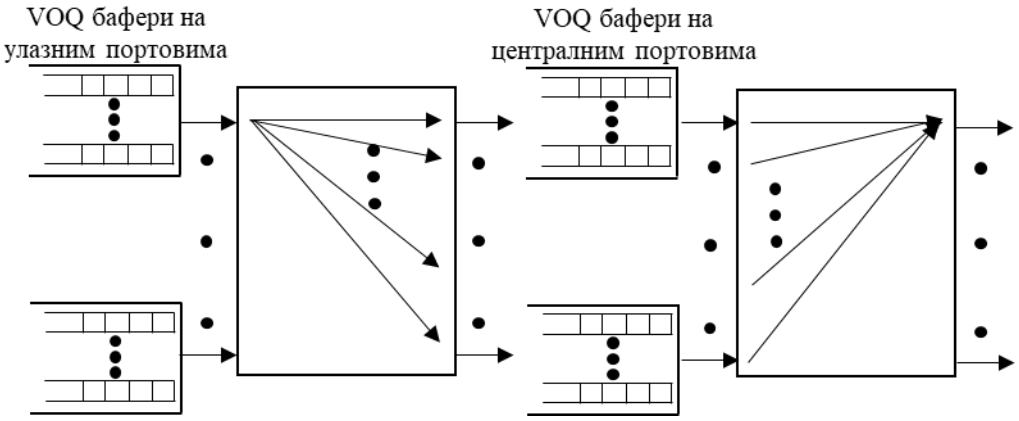
Принцип рада ресеквенционих бафера на излазним портовима се састоји од сљедећих корака:

1) **<u>Корак 1.</u>** Одредити ком току припада *HoL* пакет из одговарајућег *VOQ* реда ресеквенционог бафера.
2) **<u>Корак 2.</u>** Упоредити редни број *HoL* пакета са очекиваним редним бројем сљедећег пакета тока којем припада тај *HoL* пакет.
3) **<u>Корак 3.</u>** Ако се редни бројеви поклапају, *HoL* пакет се шаље, а редни број сљедећег пакета из тока којем припада тај пакет се ажурира.
4) **<u>Корак 4.</u>** Уколико се редни бројеви у кораку 2 не поклапају, процедура се понавља за *HoL* пакет из сљедећег *VOQ* реда ресеквенционог бафера.

Јасно је да ће у најгорем случају бити неопходно извршити N итерација овог алгоритма прије проналажења пакета за слање, што може да утиче на подржане брзине линкова од стране *FOFF* комутатора.

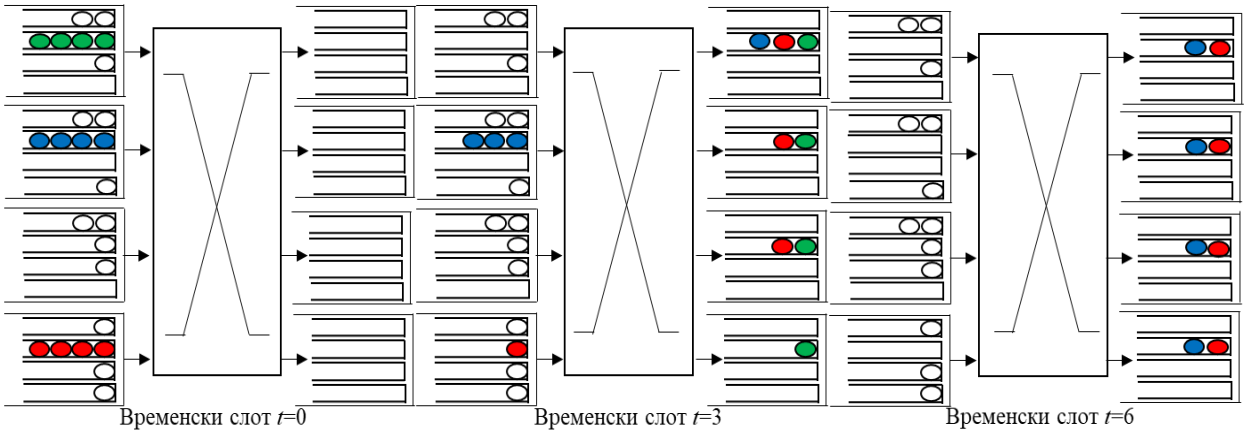
3.4.3. *Uniform Frame Spreading (UFS) комутатор*

UFS комутатор је предложен у [130]. Архитектура овог комутатора је дата на слици 3.4.6. На сваком улазном порту постоји N *VOQ1* редова за чекање. Сваки од ових бафера служи за смјештање пакета који припадају истом току. На сваком централном порту, такође постоји N *VOQ2* редова. И на централним портовима се пакети истог тока смјештају у један *VOQ2* ред. N пакета који припадају истом току чини пуни фрејм. Конфигурације оба комутациона модула су исте као и код *FFF* и *FOFF* комутатора.



Слика 3.4.6. Архитектура *UFS* комутатора

Слање пакета једног фрејма с неког улазног порта почиње када се тај порт повеже са првим централним портом, а завршава када се повеже са централним портом N. Дакле, пакети који припадају том фрејму су униформно распоређени на свих N централних портова. На овај начин ће сви пакети из истог тока, један за другим долазити на чело одговарајућих *VOQ2* редова. Уколико на неком улазном порту постоји више *VOQ1* редова са N или више пакета, бира се један у складу са *round-robin* принципом.



Слика 3.4.7. Примјер рада *UFS* комутатора

Примјер рада *UFS* комутатора је дат на слици 3.4.7. Једноставности ради, посматра се комутатор величине 4x4 и пакети који су намијењени за излазни порт 2. Пакети који стижу на улазне портове се смјештају у одговарајуће *VOQ1* редове, док се не формира бар један пун фрејм од *N* пакета. Пакети који чине пун фрејм напуштају улазне портове у хронолошком редоследу. Улазни порт ће започети слање пакета једино ако у тренутку повезивања с првим централним портом, у неком *VOQ1* реду на том порту постоји пуни фрејм. Јасно је да ће *j*-ти пакет једног фрејма бити послат на *j*-ти централни порт. Са слике се види да за излазни порт 2 постоје три пуна фрејма, на улазним портовима 1, 3 и 4. Ако претпоставимо да је у овом временском слоту (означимо га са t=0), улазни порт 1 повезан се централним портом 1, онда ће он тада послати први пакет из свог пуног фрејма. Улазни порт 4 ће први пакет из свог фрејма послати у временском слоту t=1, док ће улазни порт 3 први пакет из свог фрејма послати у временском слоту t=3. Јасно је да ће сви *VOQ2* редови на централним портовима који одговарају истим излазним портовима бити исте величине јер су пакети из једног пуног фрејма равномјерно распоређени по свим централним портовима. Такође, слање пакета једног фрејма на одређени излазни порт ће почети када се тај излазни порт повеже са централним портом 1. У конкретном примјеру, излазни порт 2 ће у временском слоту t=0, бити повезан са централним портом 2. На том централном порту нема пакета за излазни порт 2. У временском слоту t=1, излазни порт 2 ће бити повезан са централним портом 3, у временском слоту t=2, излазни порт ће бити повезан са централним портом 3 итд. Ниједан од ових централних портова нема пакет за слање на излазни порт 2. Тек у временском слоту t=3, када се излазни порт 2 повеже са централним портом 1, он ће примити први пакет из фрејма који припада току $F(1,2)$. У наредна три слота преостала 3 пакета овог фрејма ће у правилном редоследу стићи на излазни порт 2. У суштини *UFS* комутатор ради на исти начин као и *FOFF* комутатор када увијек постоји пуни фрејм.

Предности *UFS* комутатора су:

- *UFS* комутатор гарантује прослеђивање пакета у правилном редоследу без употребе ресеквенционих бафера на излазним портовима што утиче на смањење хардверске комплексности и трошкова имплементације.
- Линијски модули на којима су имплементирани портови могу да функционишу независно један од другог, те се не захтијева додатна размјена информација између њих.

Међутим, кључни недостатак овог рјешења је што је неопходно сачекати формирање пуног фрејма прије слања пакета. То ће довести до значајног повећања просјечног кашњења пакета унутар комутатора нарочито при мањим оптерећењима. У наредном потпоглављу је представљено рјешење које настоји да ријеши овај проблем.

3.4.4. Padded Frames (PF) комутатор

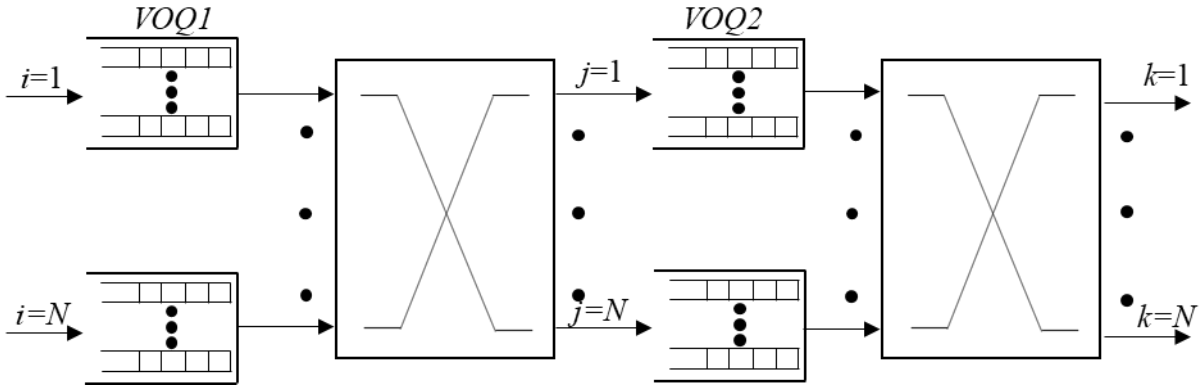
PF комутатор је предложен у [131]. Архитектура овог комутатора је дата на слици 3.4.8. На сваком улазном порту постоји *N VOQ1* редова за чекање. Сваки од ових бафера служи за смјештање пакета који припадају истом току. На сваком централном порту, такође постоји *N VOQ2* редова при чему се пакети истог тока смјештају у исти *VOQ2* ред. Конфигурације оба комутациона модула су детерминистичке и одређене формулама:

$$j = [(i+t\text{-}1) \ \mathbf{mod} \ N] + 1 \tag{3.4.3}$$

и

$$k = [(j+t\text{-}2) \ \mathbf{mod} \ N] + 1 \tag{3.4.4}$$

гдје су са *i* означени улазни портови, са *j* централни портови и са *k* излазни портови.



Слика 3.4.8. Архитектура *PF* комутатора

PF комутатор спречава поремећај редоследа пакета тако што осигурава да је број пакета у баферима на централним портовима једнак. И овај комутатор приликом одабира пакета за слање даје предност оним *VOQ1* редовима који имају пуни фрејм. Уколико има више кандидата, бира се један у складу са *round-robin* принципом. Ако не постоји пуни фрејм, *PF* комутатор бира најдужи не-пуни фрејм. Међутим, уколико се шаље не-пуни фрејм *PF* комутатор користи лажне пакете да надомјести број пакета који фали до пуног фрејма. Овакав фрејм се зове допуњени фрејм. При томе, допуњени фрејм се бира за слање једино ако је број пакета у централним баферима који одговарају излазном порту *k* мањи од *T*, гдје је *T* одговарајући праг. Наиме, број пакета у *VOQ2* реду је индикатор загушења, па је циљ да се избјегне слање лажних пакета централним баферима у којима има одређени број пакета. С друге стране, ако је број пакета у централним баферима мали, слање лажних пакета неће нарушити стабилност комутатора. На овај начин се спречава поремећај редоследа пакета, тако да није потребно имплементирати ресеквенционе бафере на излазне портове. Такође, идеја са увођењем лажних пакета је да се смање кашњења при малим оптерећењима. За улазни порт *i*, циклус започиње онда када се тај порт повеже са централним портом 1.

У наставку је дат принцип рада *PF* комутатора:

1) **Корак 1.** Када се улазни порт *i* повеже са централним портом 1, тражи се *VOQ1* ред који има бар *N* пакета. Уколико има више оваквих *VOQ1* редова, бира се један у складу са *round-robin* принципом.
2) **Корак 2.** Уколико не постоји такав *VOQ1* ред, бира се онај који садржи највише пакета. Рецимо да је то *VOQ1*(*i*,*k*) и нека је L_k дужина *VOQ2*(1,*k*). Ако је $L_k<T$, онда ће пакети из *VOQ1*(*i*,*k*) бити послати у току сљедећег циклуса.
3) **Корак 3.** Као што је већ речено слање пакета почиње кад се улазни порт *i* повеже са централним портом 1. Пакети из одабраног *VOQ1* реда ће се слати у наредних *N* слотова. Уколико више нема пакета у том *VOQ1* реду, улазни порт ће слати лажне пакете до краја циклуса.

Како се пакети равномјерно распоређују по свим централним портовима, тако ће број пакета у њима бити једнак. Међутим, претпоставимо да је на почетку циклуса улазни порт *i* одабрао да шаље пакете из *VOQ1*(*i*,*k*) који има мање од *N* пакета. Јасно је да ће ово захтијевати и слање лажних пакета. Међутим, у току циклуса се може десити да на улазни порт *i* дође пакет за излазни порт *k*. У том случају ће се тај пакет послати умјесто лажног

пакета. На тај начин се поправља ефикасност комутатора. У суштини, када је T=0, *PF* комутатор се понаша као *UFS* комутатор.

Увођење лажних пакета доводи до повећања просјечног кашњења пакета, тако да је пожељно на неки начин ограничити број лажних пакета. Зато је предложен *PF+* комутатор. Идеја је да се користи бројач који показује колико допуњених фрејмова постоји на централним портовима за сваки од излазних портова. Ови бројачи се ажурирају на сваких N слотова. Такође, како је број допуњених фрејмова ограничен, број лажних пакета на централним портовима за велика оптерећења је смањен, јер је смањена вјероватноћа слања допуњених фрејмова који имају велики број лажних пакета. Наиме, код *PF+* комутатора, пакети из одређеног $VOQ1(i,k)$ ће бити одабрани за слање, ако је број пакета на централним портовима за излазни порт k мањи од T и ако је број допуњених фрејмова на централним портовима за излазни порт k мањи од W, гдје је W одговарајући праг. Параметар T служи да гарантује стабилност комутатора, док је задатак параметра W да смањи кашњење пакета при малим оптерећењима. Дакле, разлика у односу на *PF* комутатор је у сљедећим корацима:

2) **Корак 2.** Уколико не постоји *VOQ1* ред који садржи N или више пакета, бира се онај који садржи највише пакета. Рецимо да је то $VOQ1(i,k)$ и нека је L_k дужина $VOQ2(1,k)$, а нека је PF_k бројач допуњених фрејмова на централним портовима које треба прослиједити на излазни порт k. Ако је $L_k<T$ и ако је $PF_k<W$, онда ће пакети из $VOQ1(i,k)$ бити послати у току сљедећег циклуса. Такође, и PF_k се ажурира тако да је $PF_k=PF_k+1$. У супротном, $VOQ1(i,k)$ неће бити изабран.
3) **Корак 3.** Као што је већ речено слање пакета почиње кад се улазни порт i повеже са централним портом 1. Пакети из одабраног *VOQ1* реда ће се слати у наредних N слотова. Уколико више нема пакета у том *VOQ1* реду, улазни порт ће слати лажне пакете до краја циклуса.
4) **Корак 4.** Када излазни порт k прими комплетан допуњени фрејм, PF_k се ажурира, $PF_k=PF_k-1$.

3.4.5. Contention and Resolution (CR) комутатор

CR комутатор [132] се састоји од два комутациона модула, који имају детерминистичке конфигурације у сваком временском слоту. На слици 3.4.9 је дата архитектура овог комутатора. *CR* комутатор комбинује особине *frame-based* комутатора и комутатора за *feedback* механизмом. Наиме, за овај тип комутатора је карактеристично да ради у два режима: *contention* начин рада и *reservation* начин рада. На сваком од улазних портова се налазе бафери са по N *VOQ* редова за чекање. И код овог комутатора се у једном *VOQ* реду чувају пакети које треба прослиједити на исти излазни порт. Бафери који се налазе између првог и другог комутационог модула се називају централни бафери и они се састоје од тзв. *I-VOQ* редова. Пакети који се прослеђују у току *contention* режима рада се називају *contention* пакети, док се пакети који се прослеђују у току *reservation* режима рада се називају *reservation* пакети. Конфигурације оба комутатора су одређена по формули:

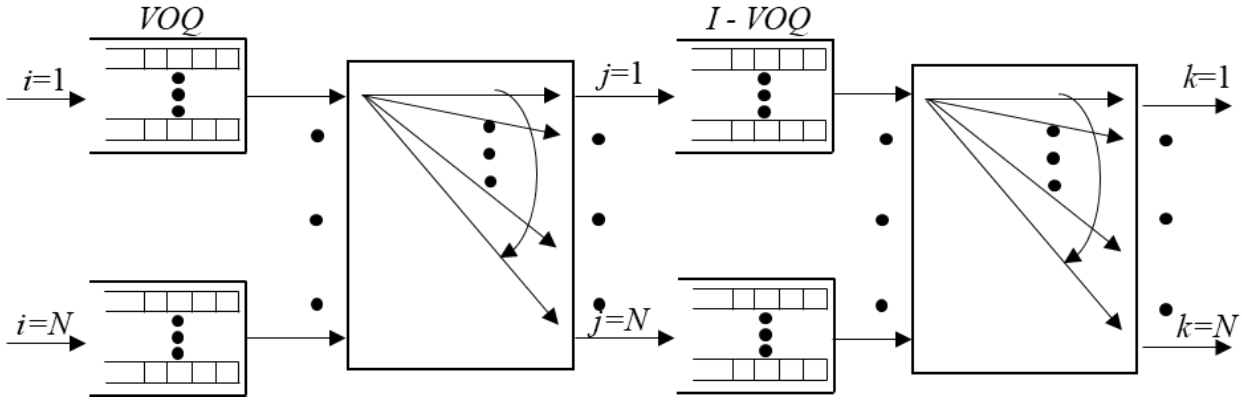
$$(i+j) \bmod N = (t+1) \bmod N \tag{3.4.5}$$

односно

$$(j+k) \bmod N = (t+1) \bmod N \tag{3.4.6}$$

гдје су i улазни портови, j централни портови, k излазни портови, а t тренутни временски слот. На основу овога се види да ће, за неки позитивни цијели број g, улазни порт i бити

повезан са излазним портом 1 у временском слоту $i+(g-1)\cdot N$, са излазним портом 2 у временском слоту $i+1+(g-1)\cdot N$,…, са излазним портом N у временском слоту $i-1+g\cdot N$.



Слика 3.4.9. Архитектура *CR* комутатора

Такође, може се закључити да су ове конфигурације симетричне, у смислу да ако је улазни порт *i* повезан са централним портом *j*, онда то значи и да је централни порт *j* повезан са излазним портом *i*. Како су улазни порт *i* и излазни порт *i* имплементирани на истом линијском модулу, симетрична природа конфигурација омогућава да сваки централни бафер може да пошаље повратну информацију улазном порту с којим је повезан преко излазног порта на који је повезан.

Како би се задржао исправан редослед пакета (и *contention* и *reservation* пакета) у овом рјешењу се у централним портовима користе бафери са *I-VOQ* редовима. Слично класичним *VOQ* редовима, и у овим *I-VOQ* редовима се чувају пакети које треба прослиједити на исти излазни порт. Разлика је што *I-VOQ* редови дозвољавају да новопридошли пакет замијени постојећи *HoL* пакет. Постоје три врсте пакета у *I-VOQ* редовима: лажни пакети, *contention* пакети и *reservation* пакети. Лажни пакет је генерисан од стране *I-VOQ* реда онда када нема других пакета у њему. На тај начин, лажни пакет је увијек *HoL* пакет и то гарантује да у сваком *I-VOQ* реду постоји бар један пакет. Када *contention* пакет стигне, а *HoL* пакет у том *I-VOQ* реду је лажни пакет, онда је тај лажни пакет замијењен тим *contention* пакетом који постаје *HoL* пакет. У супротном, *contention* пакет је одбачен. С друге стране, кад *I-VOQ* ред прими *reservation* пакет, онда се тај пакет додаје на његов крај. Како у *I-VOQ* реду увијек постоји бар један пакет, *reservation* пакет никад неће бити *HoL* у тренутку доласка. Када је централни порт повезан с одређеним излазним портом, тада се *HoL* пакет (без обзира да ли је лажни или не) из одговарајућег *I-VOQ* реда прослеђује на тај излаз. Пошто је могуће да централни порт одбаци *contention* пакет који му је послат, сваки улазни порт захтијева повратну информацију, да ли је пакет примљен или не. Ова информација се прослеђује заједно с пакетом који се с централног бафера прослеђује на излазни порт. На основу особине симетричности, централни порт је повезан са улазним и излазним портовима с истим индексом, тако да кад та информација стигне на излазни порт, она ће бити доступна и улазном порту. Ако је пакет успјешно примљен, он ће бити избрисан из бафера на улазном порту. У супротном, и даље остаје смјештен у том баферу.

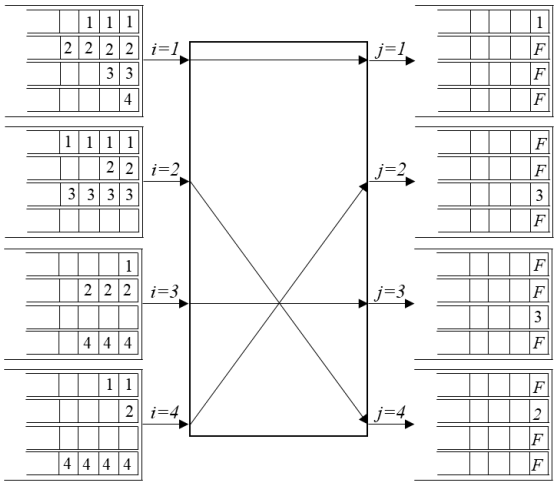
Што се тиче режима рада, он зависи од количине саобраћаја коју *CR* комутатор треба да прослиједи. *CR* комутатор функционише на *frame-based* начин, при чему се фрејм састоји од N узастопних временских слотова. Међутим, за различите улазне/излазне портове временски слот у коме почиње фрејм је различит. Наиме, *f*-ти фрејм за улазни порт *i* почиње у временском слоту када се улазни порт *i* по *f*-ти пут повеже са првим централним портом.

Тако ће се *f*-ти фрејм за улазни порт *i* састојати од временских слотова $i+(f-1)\cdot N,\ldots, i-1+f\cdot N$. Ако је број пакета у *VOQ* реду на улазном порту већи од или једнак *N*, онда се тај *VOQ* ред зове *full-framed VOQ*. Ако на почетку фрејма, улазни порт има пун фрејм, онда комутатор функционише у *reservation* начину рада и тај фрејм се назива *reservation* фрејм. У супротном, комутатор функционише у *contention* режиму, и тај фрејм се зове *contention* фрејм.

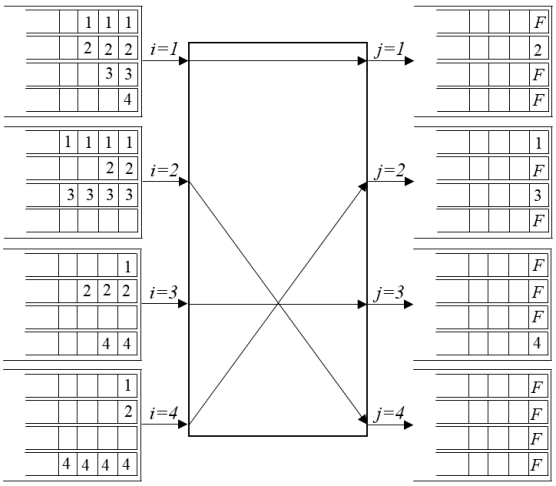
Сваки улазни порт има свој *reservation* показивач за одабир *full-framed VOQ*. На почетку сваког фрејма се бира онај *VOQ* ред који је најближи *reservation* показивачу, у складу са *round-robin* принципом. Када се *VOQ* ред одабере, показивач се ажурира да показује на први сљедећи *VOQ* ред у складу са *round-robin* принципом. У сваком слоту посматраног фрејма, централним баферима се шаљу *HoL* пакети из одабраног *VOQ* реда. Такође, уз сваки пакет се шаље додатни бит који означава да је у питању *reservation* пакет. Ти пакети су смјештени на крају *I-VOQ* реда. Осим *reservation* показивача, сваки улазни порт има и свој *contention* показивач, који служи за одабир *contention* пакета за слање. Ови показивачи функционишу по истом принципу као и *reservation* показивачи. Када се ови пакети шаљу централним баферима, и они садрже додатни бит који означава да је у питању *contention* пакет. Када централни порт прими овај пакет, ако је на челу одговарајућег *I-VOQ* реда лажни пакет, тај пакет се замјењује *contention* пакетом, и шаље се бит који потврђује успјешну трансмисију. Када улазни порт прими ову информацију, пакет који је послат се брише са улазног порта. Ако је на челу *I-VOQ* реда прави пакет, онда се *contention* пакет одбацује и о томе се шаље одговарајућа информација улазном порту.

На слици 3.4.10 је дат примјер рада *CR* комутатора. Претпоставимо да је $t=4\cdot n+1$, што значи да у том слоту почиње фрејм за улазни порт 1. На слици 3.4.10 а) је показано стање комутатора у слоту *t* прије слања пакета, док је на слици 3.4.10 б) показано стање у комутатору након слања пакета у слоту *t*. На сликама 3.4.10 в), г) и д) је показано стање у комутатору након слања пакета у одговарајућим слотовима. Како је већ речено, у слоту *t* почиње фрејм за улазни порт 1. Пакет *F* на слици представља лажни пакет. Пошто за ток пакета *F*(1,2) постоји пуни фрејм, улазни порт 1 улази у *reservation* начин рада и у току овог фрејма ће се слати пакети овог тока. Улазни порт 2 је у *contention* режиму, и шаље пакет из тока *F*(2,2). Међутим, пошто у централном порту 4 већ постоји пакет у одговарајућем *VOQ* реду, овај пакет ће и даље остати на улазном порту 2. Улазни порт 3 је такође у *contention* режиму рада. Шаље се пакет из тока *F*(3,4). Пошто је у одговарајућем *VOQ* реду на централном порту лажни пакет, овај пакет ће бити успјешно послат, а *contention* показивач се ажурира да показује на излазни порт 1. Улазни порт 4 се такође налази у *contention* режиму рада. Шаље се пакет из тока *F*(4,1). Пошто је у одговарајућем *VOQ* реду на централном порту лажни пакет, овај пакет ће бити успјешно послат, а *contention* показивач се ажурира да показује на излазни порт 2. Такође, у току овог слота, централни порт 1 шаље пакет на излазни порт 1, централни порт 3 шаље пакет на излазни порт 3, а централни порт 4 шаље пакет на излазни порт 2.

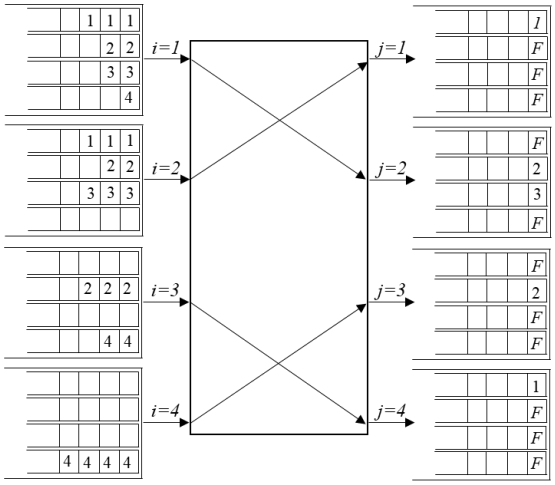
У току временског слота *t*+1, пошто је улазни порт 1 у *reservation* режиму рада, наставља се слање пакета из тока *F*(1,2). У овом временском слоту почиње фрејм за улазни порт 2. Пошто за ток пакета *F*(2,1) постоји пуни фрејм, овај улазни порт улази у *reservation* режим рада, и шаље се пакет из овог тока. Улазни порт 3 је и даље у *contention* режиму рада, и шаље се пакет из тока *F*(3,1). Након успјешног слања, показивач се ажурира да показује на излазни порт 2. Улазни порт 4 је такође у *contention* режиму рада, и шаље се пакет из тока *F*(4,2). Након успјешног слања, показивач се ажурира да показује на излазни порт 3. Такође, у току овог слота, централни порт 1 шаље пакет на излазни порт 3, централни порт 3 шаље пакет на излазни порт 1, а централни порт 3 шаље пакет на излазни порт 4.



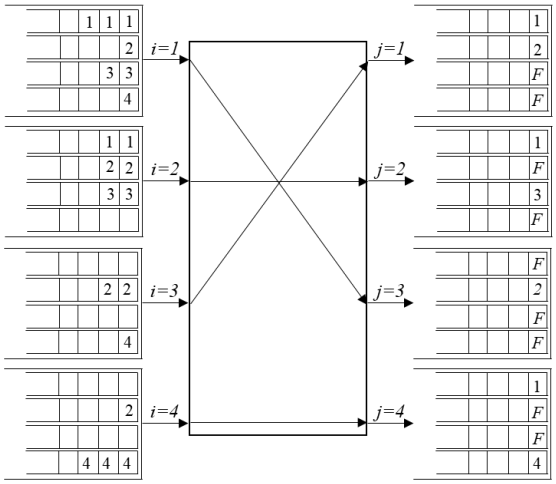
а) Временски слот *t-*



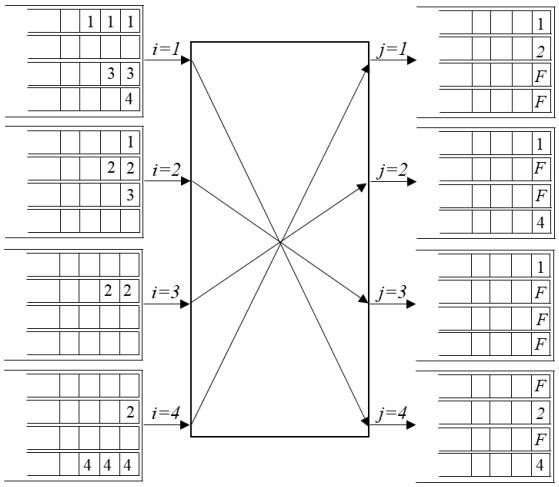
б) Временски слот *t+*



в) Временски слот *t+1*



г) Временски слот *t+2*



д) Временски слот *t+3*

Слика 3.4.10. Примјер рада CR комутатора

У току временског слота *t*+2, улазни порт 1 наставља да ради у *reservation* режиму рада, шаље пакет из тока *F*(1,2). Како је и улазни порт 2 у *reservation* режиму, шаље се пакет из тока *F*(2,1). У овом временском слоту почиње фрејм за улазни порт 3. Пошто на овом порту не постоји ниједан пуни фрејм, овај улазни порт наставља да ради у *contention* режиму рада. Улазни порт 3 је и даље у *contention* режиму рада, и шаље се пакет из тока *F*(3,2). Након успјешног слања, показивач се ажурира да показује на излазни порт 3. Улазни порт 4 наставља да ради у *contention* режиму, и шаље се пакет из тока *F*(4,4). Након успјешног слања, показивач се ажурира да показује на излазни порт 1. Такође, у току овог слота, централни порт 2 шаље пакет на излазни порт 2.

У току временског слота *t*+3, улазни порт 1 наставља да ради у *reservation* режиму рада, шаље пакет из тока *F*(1,2). Како је и улазни порт 2 у *reservation* режиму, шаље се пакет из тока *F*(2,1). Улазни порт 3 је у *contention* режиму, и шаље се пакет из тока *F*(3,4). Након успјешног слања, показивач се ажурира да показује на излазни порт 1. У овом временском слоту почиње фрејм за улазни порт 4. Пошто на овом порту не постоји ниједан пуни фрејм, овај улазни порт наставља да ради у *contention* режиму рада, и шаље се пакет из тока *F*(4,2). Међутим, пошто одговарајући централни порт има пакте намијењен за излазни порт 2, тако да ће овај пакет остати на улазном порту 4. Такође, у току овог слота, централни порт 2 шаље пакет на излазни порт 3, централни порт 3 шаље пакет на излазни порт 2, а централни порт 4 на излазни порт 1.

Дакле, из датог примјера је јасно да ће *CR* комутатор при великим оптерећењима (када на улазним портовима постоје пуни фрејмови) радити на сличан начин као и *frame-based*комутатори. Међутим, *frame-based* комутатори имају проблем када је оптерећење мало јер је потребно вријеме док се не попуни фрејм што лоше утиче на перформансе комутатора. *CR* комутатор решава тај проблем преласком у *contention* режим рада када на улазном порту не постоји пун фрејм, тако да се не губи вријеме чекајући на формирање пуног фрејма. На овај начин се побољшавају перформансе.

Постоје различите стратегије како се бирају пакети за слање у *contention* режиму рада:

- *SAFA* (*Success: Advance / Failure: Advance*): ако је одабрани пакет успјешно послат, *contention* показивач се ажурира да показује на први сљедећи непразни *VOQ* ред; Ако је слање било неуспјешно *contention* показивач се ажурира да показује на први сљедећи непразни *VOQ* ред.
- *SAFP* (*Success: Advance / Failure: Persist*): ако је одабрани пакет успјешно послат, *contention* показивач се ажурира да показује на први сљедећи непразни *VOQ* ред; Ако је слање било неуспјешно *contention* показивач показује на исти *VOQ* ред.
- *SPFA* (*Success: Persist / Failure: Advance*): ако је одабрани пакет успјешно послат, *contention* показивач показује на исти *VOQ* ред; Ако је слање било неуспјешно *contention* показивач се ажурира да показује на први сљедећи непразни *VOQ* ред.
- *SPFP* (*Success: Persist / Failure: Persist*): ако је одабрани пакет успјешно послат, *contention* показивач показује на исти *VOQ* ред; Ако је слање било неуспјешно *contention* показивач показује на исти *VOQ* ред.
- *SPFP-Longest* (*Success: Persist / Failure: Persist*): ако је одабрани пакет успјешно послат, *contention* показивач показује на исти *VOQ* ред; Ако је слање било неуспјешно *contention* показивач се ажурира да показује на најдужи *VOQ*ред.
- *SPFA-LMQ* (*Success: Persist / Failure: Advance – Longer than or equal to the Medium Queue*): ако је одабрани пакет успјешно послат, *contention* показивач показује на

исти *VOQ* ред; Ако је слање било неуспјешно *contention* показивач се ажурира да показује на *VOQ* ред који је једнаке или веће дужине него што је просјечна дужина *VOQ* редова на том улазном порту.

Симулације показују да се најбоље перформансе постижу коришћењем *SPFA-LMQ* алгоритма [132].

3.5. Употреба специфичне конфигурације комутационих модула

Идеја ове класе рјешења је употреба специфичних конфигурација оба комутациона модула које обезбјеђују ефикасну размјену контролних информација између портова. Такође, постиже се да сви пакети једног тока имају једнако кашњење чиме се гарантује да ће на излаз доћи у правилном редоследу.

3.5.1. Feedback Staggered Symmetry (FBSS) комутатор

Ово рјешење [133,134], као и остала рјешења заснована на *LB-BvN* архитектури, се састоји од два комутациона модула, као и бафера са *VOQ* редовима на улазним (*VOQ1*) и централним (*VOQ2*) портовима. Бафери се састоје од *N VOQ* редова, при чему се у сваком од њих смјештају пакети које треба прослиједити на исти излазни порт. За ово рјешење је карактеристично да у сваком *VOQ2* реду на централним портовима може да се смјести само по један пакет, тј. у сваком централном порту може да се смјести највише по један пакет за сваки од излазних портова. Архитектура овог комутатора ја дата на слици 3.5.1. Проблем поремећаја редоследа пакета истог тока је ријешен специфичним, детерминистичким и синхронизованим (тзв. *staggered symmetry and in order*) конфигурацијама у оба комутациона модула. Наиме, улазни и централни портови су повезани по сљедећем шаблону:

$$j = (i+t) \bmod N \tag{3.5.1}$$

гдје је i редни број улазног порта, а j редни број централног порта. Истовремено, повезивање централних и излазних портова је одређено по сљедећем принципу:

$$k = (j+N-1-t) \bmod N \tag{3.5.2}$$

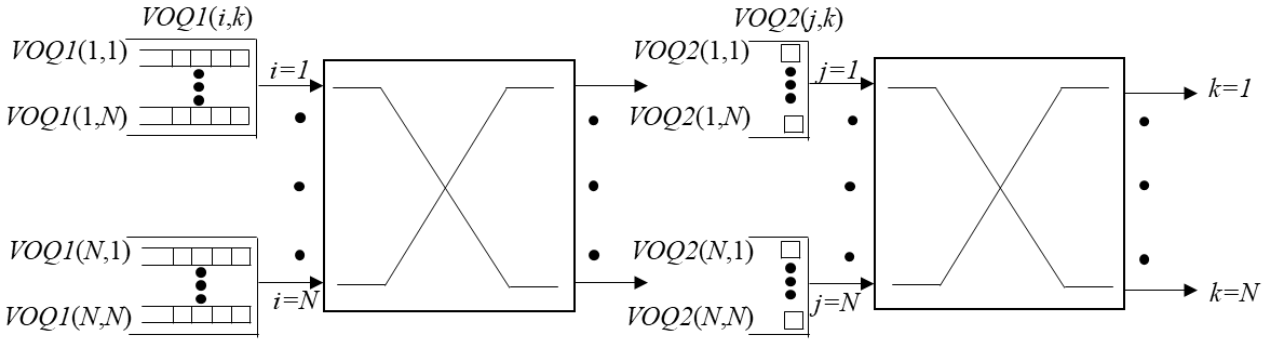
гдје је j редни број централног порта, а k редни број излазног порта.

На слици 3.5.2 је дат један примјер ових конфигурација. Овакав начин конфигурисања оба комутациона модула гарантује сљедеће [134]:

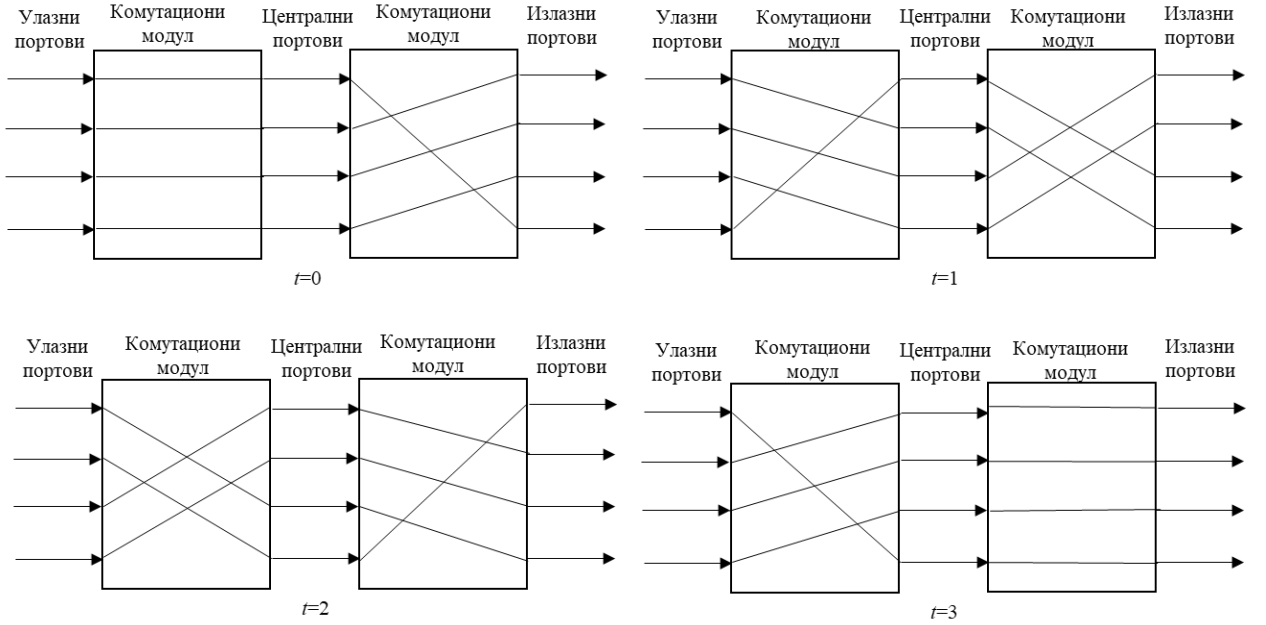
- Улазни порт i је увијек повезан са излазним портом $k = (i+N-1) \bmod N$, преко неког од централних портова.
- Нека је K излазни порт на који је повезан улазни порт i. Сваки пакет тока $F(i,k)$ ће на централним портовима имати исто кашњење од d слотова, гдје је d одређено на сљедећи начин:

$$d = \begin{cases} N, & \text{ако } K = k \\ K - k, & \text{ако } K > k \\ K + N - k, & \text{ако } K < k \end{cases} \tag{3.5.3}$$

- Гарантује се да ће пакети бити испоручени на излазне портове у правилном редоследу користећи горе поменуте шаблоне конфигурисања комутатора.



Слика 3.5.1. Архитектура ФБСС комутатора



Слика 3.5.2. Примјер *staggered symmetry* конфигурација

Сваком централном порту је придружен вектор у коме се налазе информације о заузетости *VOQ2* реда у том централном порту. Овај вектор се састоји од N бита, при чему сваки бит одговара једном излазном порту. Уколико централни порт није примио пакет за неки излазни порт k, онда је k-ти бит у његовом вектору заузетости постављен на 1. У супротном, ако је централни порт примио пакет за неки излазни порт k, онда је k-ти бит у његовом вектору заузетости постављен на 0. Подразумијева се да су на почетку сви бити у векторима заузетости свих централних портова постављени на 1. Даље, како централни портови примају пакете, врши се ажурирање ових вектора. Улазни портови на основу ових вектора могу да одреде из којих *VOQ1* редова пакети могу бити послати. Наиме, када се улазни порт повеже с неким централним портом j, он процесира одговарајући вектор V_j. На основу овог вектора, улазни порт може да одреди за које излазне портове, централни порт може да прими пакет. Међу свим *VOQ1* редовима из којих се може послати пакет, бира се онај у коме има највише пакета. Централни портови информацију о заузетости својих бафера прослеђују као додатак пакету који се прослеђује на излазни порт. Пошто су улазни, централни и излазни портови са истим индексом имплементирани на истој линијској картици, када излазни порт прими пакет, информација о заузетости централног бафера с ког је послат тај пакет се лако прослиједи до улазног порта са исте линијске картице. Претходно

објашњени принципи конфигурисања комутатора гарантују да ако је централни порт *j* у временском слоту *t* повезан са излазним портом *k*, онда ће улазни порт *k* у сљедећем временском слоту бити повезан са централним портом *j*. На тај начин, улазни портови на ефикасан и једноставан начин добијају информације о заузетости бафера на централном порту с којим су повезани. На основу ове информације, они утврђују из којих *VOQ1* редова могу послати пакет. Овај принцип може да се уочи када се погледају конфигурације на слици 3.5.2. Ако се погледа слика 3.5.2, може се уочити да је у тренутку t=0 централни порт 1 повезан са излазним портом 4. Тада централни порт прослеђује информацију о заузетости *VOQ2* реда излазу 4. Излазни порт 4 одмах преослеђује ту информацију улазном порту 4 пошто су на истој линијској картици. Отуда у тренутку t=1 кад је улаз 4 повезан на централни порт 1, улаз зна на који излаз смије да пошаље пакет јер зна која мјеста су слободна у *VOQ2* реду.

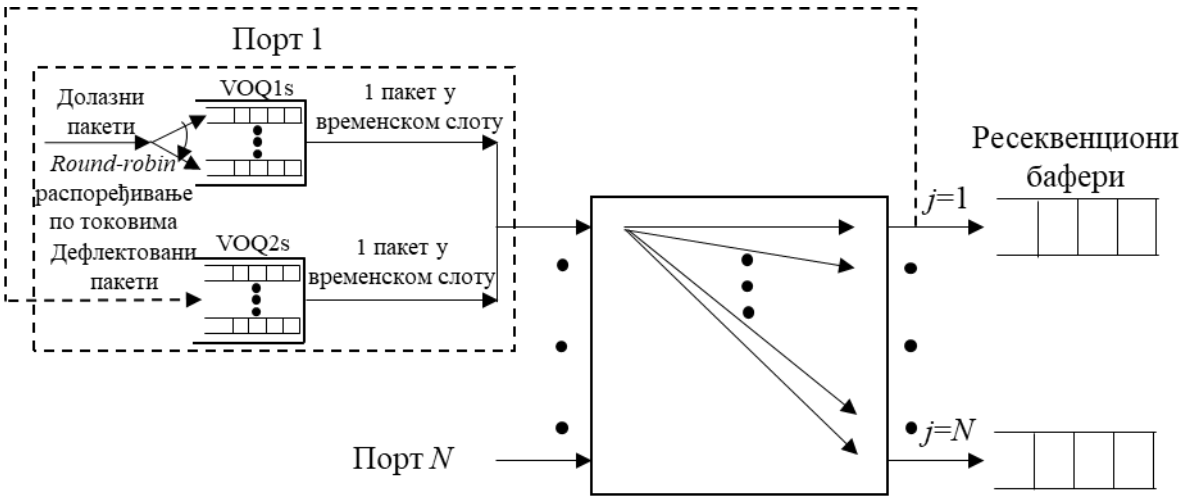
4. ПРЕДЛОГ РЈЕШЕЊА ЗАСНОВАНИХ НА LB-BVN КОМУТАТОРУ

У наставку су представљена рјешења која су резултат истраживања и рада у оквиру ове дисертације. Основна идеја приликом развоја ових рјешења је била да се комбинују најбоље особина комутатора са баферима на улазним портовима и *LB-BvN* комутатора, а да се истовремено превазиђу ограничења која имају ова рјешења. Наиме, комутатори са баферима на улазним портовима користе меморије које у току једног слота треба да изврше највише један упис и једно читање када се не користи убрзање. Ово је предност ових комутатора, јер технолошки није захтијевно за реализацију. Међутим, велики недостатак ових комутатора је што су конфигурације комутационог модула случајне, што је тешко реализовати за велике брзине линкова. С друге стране, *LB-BvN* комутатори су врло популарно рјешење баш зато што су њихове конфигурације детерминистичке, тј. није неопходно у сваком слоту вршити прорачун конфигурација комутационих модула. Из ових разлога, рјешења која ће бити представљена у наставку имају архитектуру као *LB-BvN* комутатори, са додатим баферима на улазним портовима. Такође, иако се *LB-BvN* комутатори састоје од два комутациона модула, код рјешења, која ће у овом поглављу бити представљена, њихове конфигурације су исте, што значи да имају тзв. „преклопљену“ архитектуру. Ово значи да за разлику од већине осталих комутатора на бази *LB-BvN* комутатора, ова рјешења не морају да користе два физичка комутациона модула, већ само један. Међутим, код *LB-BvN* комутатора постоји проблем поремећаја редоследа пакета, што је у *TCP*/*IP* мрежама недопустиво. Приликом рада на развоју ових комутатора, предложена су рјешења за овај проблем. Један приступ је да се на излазним портовима користе ресеквенциони бафери, који врше исправљање редоследа пакета. Величина ових бафера утиче на комплексност рјешења и трошкове имплементације, па је циљ да они буду што је могуће мањи. Други приступ решавању проблема поремећаја пакета јесте да се на улазним портовима изврши распоређивање пакета прије слања, тако да они на излазе стигну у правилном распореду. Ово је могуће јер су конфигурације оба комутатора детерминистичке, па можемо одредити када ће који пакет стићи на одговарајући излаз. Циљ при развоју ових рјешења јесте да добијено рјешење постиже исте или боље перформансе у односу на досад предложена рјешења, а да се при том задржи низак ниво хардверске комплексности. Осим тога, предложена рјешења би требало да имају подршку за фер сервис, тј. да се сви токови приближно равноправно опслужују у случајевима када су неки излази преоптерећени. Коначно, оно рјешење које се покаже као најбоље ће бити унапријеђено да подржи и мултикаст саобраћај.

4.1. *Birkhoff-von Neumann* комутатор са *Deflection-Based Load Balancing* (*BvN-DLB*)

У овом поглављу ће бити представљен *Birkhoff-von Neumann* комутатор са *Deflection-Based Load Balancing* (*BvN-DLB*) који представља један од доприноса ове дисертације [135]. *BvN-DLB* комутатор врши балансирање долазног саобраћаја дефлектовањем пакета кроз све портове. Балансирање се врши по токовима како би се елиминисао сценарио гдје би се

пакети које треба прослиједити на исти излазни порт балансирали кроз мали број истих портова што може оборити пропусност комутатора. Како су у овом случају конфигурације оба комутациона модула идентичне, могуће је користити један физички уређај умјесто два, што смањује хардверску комплексност *BvN-DLB* комутатора. Да би се користио један уређај потребно је користити интерно убрзање 2, с тим што се то односи само на прослеђивање пакета, али не и на конфигурацију комутационог модула која остаје идентична у току једног временског слота.



Слика 4.1.1. Архитектура *BvN-DLB* комутатора

На слици 4.1.1 је дата шема овог рјешења. На улазним портовима се налазе бафери са два типа *VOQ* редова, *VOQ1* и *VOQ2*. *VOQ1* редови служе за балансирање оптерећења и смјештање долазних пакета. На сваком улазном порту постоји *N VOQ1* редова. У *VOQ1*(*i,k*) су смјештени пакети који долазе на улазни порт *i* и који се дефлектују на порт *k*. Када стигне нови пакет који припада току *F*(*i,j*), *round-robin* показивач који одговара том току показује у који *VOQ1* ред треба уписати тај пакет. Након тога, тај показивач се инкрементира. На сваком улазном порту постоји *N* показивача, по један за сваки ток који пролази кроз тај порт. Показивач садржи $\log_2 N$ бита како би могли да показују на неки од *N VOQ1* редова. Функција ових показивача је да спријече ситуацију гдје би се пакети истог тока прослеђивали преко једног или неколико портова. Ако се пакети из више токова прослеђују преко истог порта (или истог скупа портова) пропусност комутатора може бити значајно смањена. Такође, на сваком улазном порту постоји *N VOQ2* редова, у којима се смјештају дефлектовани пакети. Пакети из ових редова се прослеђују на своје крајње дестинације (тј. одговарајуће излазне портове). У *VOQ2*(*i,j*) су смјештени пакети који су дефлектовани на порт *i* и које треба прослиједити на излазни порт *j*. У току циклуса од *N* временских слотова, сваки порт ће по једном бити повезан са сваким од портова. Када се порт *i* повеже са портом *j*, тада се један пакет из *VOQ2*(*i,j*) и један пакет из *VOQ2*(*i,j*) шаљу на порт *j*. То значи да је неопходно интерно убрзање 2, да би се у току једног временског слота послала два пакета. Међутим, конфигурација комутационог модула у току тог слота остаје иста, односно за конфигурисање није потребно интерно убрзање. Пошто ће услед балансирања оптерећења доћи до поремећаја редоследа пакета, на сваком порту су имплементирани ресеквенциони бафери чија је величина N^2, слично као и код других рјешења која користе ресеквенционе бафере. Уједно, ово представља и кључни недостатак овог рјешења, јер употреба ових бафера повећава трошкове имплементације. Предност је креирање математичког модела заснованог

на детерминистичкој теорији сервисних система [136] који описује перформансе овог рјешења, као што ће бити изложено у наставку овог потпоглавља.

Ово рјешење је неблокирајуће за сваки дозвољени шаблон саобраћаја. Доказ је дат у наставку текста. Са C је означен максимални улазни и излазни капацитет једног порта, и он је једнак протоку једног пакета у једном временском слоту. Са R_{ij} је означена брзина тока $F(i,j)$. Претпоставка је да су бафери довољно великог капацитета да се избјегне губитак пакета. Касније ће бити дат и прорачун горњих лимита бафера. Очигледно је да релација:

$$\sum_{j=1}^{N} R_{ij} \le C, \;\; i = 1,...,N \tag{4.1.1}$$

мора да важи, јер капацитет свих токова који долазе на исти порт не може превазићи капацитет улазног линка. Како је претпоставка да се ради о дозвољеном сценарију, онда ниједан излазни порт није преоптерећен, па важи:

$$\sum_{i=1}^{N} R_{ij} \le C, \;\; j = 1,...,N \tag{4.1.2}$$

Пошто се сви токови пакета дистрибуирају једнако преко N портова, брзина тока који улази у $VOQ1(i,k)$ (i=1,…,N, k=1,…,N) је једнака:

$$\sum_{j=1}^{N} R_{ij} / N \le C / N \tag{4.1.3}$$

Како је порт i у сваком N-том временском слоту повезан са портом k, то значи да се $VOQ1(i,k)$ опслужује брзином C/N, и на основу (4.1.3) $VOQ1$ не може бити преоптерећен. Брзина тока дефлектованих пакета који улазе у $VOQ2(k,j)$ (k=1,…,N,j=1,…,N) може се лако израчунати зато што су сви токови пакета које треба прослиједити на исти излазни порт балансирани преко свих портова:

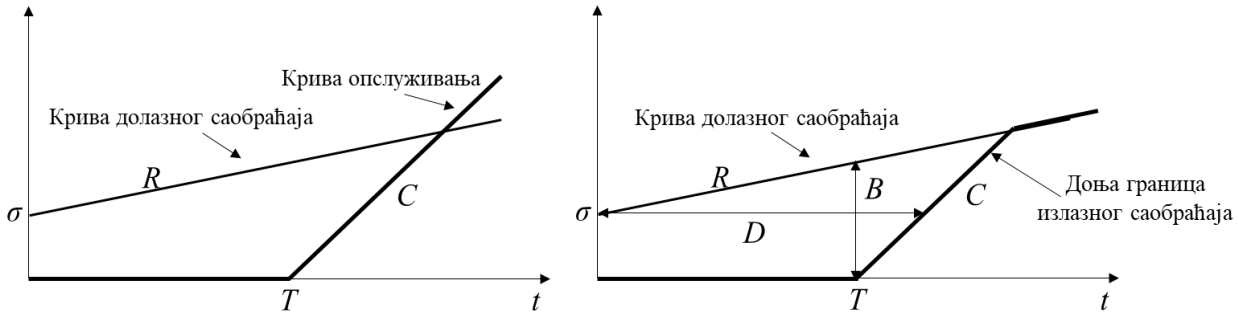
$$\sum_{i=1}^{N} R_{ij} / N \le C / N \tag{4.1.4}$$

Пошто је порт k, у сваком N-том временском слоту повезан са портом j, онда је $VOQ2(k,j)$ опслужен брзином C/N и у складу са (4.1.4) бафери са $VOQ2$ редовима не могу бити преоптерећени. Како ниједан бафер са VOQ редовима не може бити преоптерећен, комутатор је стабилан (тј. неблокирајући) за било који дозвољени сценарио саобраћаја.

Ово рјешење има предност у односу на већину осталих јер омогућава да се математички прорачуна граница кашњења и величина бафера. У наставку је дат овај прорачун који је изведен коришћењем *network calculus* теорије, односно детерминистичке теорије сервисних система [136]. *Network calculus* пружа аналитички алат за прорачун граница кашњења и бафера које никад неће бити прекршене тј. могу да се гарантују. Криве долазног саобраћаја се користе да одреде горње границе долазног саобраћаја, док се криве опслуживања користе да репрезентују опслуживање долазног саобраћаја. *Min-plus* конволуција криве долазног саобраћаја и криве опслуживања представља границу најгорег случаја излазног саобраћаја мрежног чвора (тј. комутатора) у временском домену. Границе кашњења и бафера (D и B) могу бити израчунате користећи ову границу излазног саобраћаја и криве долазног саобраћаја као што је приказано на слици 4.1.2. Природа саобраћаја у пакетским мрежама је спорадична. Крива долазног саобраћаја дефинисана као:

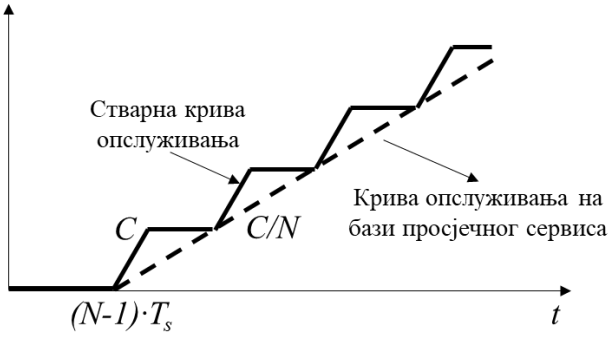
$$\alpha(t) = \begin{cases} 0, & t = 0 \\ \sigma + R \cdot t, & t > 0 \end{cases} \tag{4.1.5}$$

и може се користити за описивање спорадичног саобраћаја, гдје σ представља спорадичност а *R* представља просјечну брзину саобраћаја.



а) **б)**

Слика 4.1.2. Границе кашњења и величине бафера



Слика 4.1.3. Крива опслуживања

Крива долазног саобраћаја је дата на сликама 4.12 а) і б). За сваки ток *F*(*i,j*) се користи дефиниција (4.1.5) са параметрима σ_{ij} и R_{ij}. Сваки *VOQ* ред у *BvN-DLB* комутатору је опслужен у сваком *N*-том слоту брзином *C*. Крива опслуживања је дата на слици 4.1.3. Међутим, у овој анализи се узима у обзир кашњење криве опслуживања која је једнака просјечној брзини опслуживања *C/N* сваког *VOQ* реда. Ово ће резултирати нешто песимистичнијим границама кашњења и величина бафера, али је прорачун много једноставнији. У обијема кривама опслуживања, кашњење је постављено на (*N*-1)·T_s, гдје је T_s трајање једног временског слота. Ово кашњење представља најгори случај када је пакет који је смјештен у празан *VOQ* ред, управо пропустио конекцију тог *VOQ* реда и мора чекати (*N*-1) временских слотова да буде опслужен. Пошто се у овом рјешењу користи *round-robin* балансирање оптерећења, ток *F*(*i,j*) је подијељен у *N* подтокова, од којих сваки припада по једном *VOQ1* реду на порту *i*. Крива долазног саобраћаја једног подтока је (показује се само вриједност за *t*>0):

$$\alpha_{ij}(t) = 1 + \sigma_{ij} / N + R_{ij} \cdot t / N \tag{4.1.6}$$

Спорадичност није дефинисана са σ_{ij}/N, него са $1+\sigma_{ij}/N$ услед чињенице да се ток дијели на нивоу пакета што значи да сваки *VOQ1* ред на порту *i* прима цјелобројан број пакета. Међутим, σ_{ij}/N не мора да буде цијели број, тако да ће можда неки *VOQ1* ред да прими један пакет више у односу на границу σ_{ij}/N. Сума свих подтокова који улазе у *VOQ1*(*i,k*) (*k*=1,…,*N*) је:

$$\alpha_{VOQ1ik}(t) = \sum_{j=1}^{N}(1+\sigma_{ij}/N + R_{ij}\cdot t/N) \tag{4.1.7}$$

На основу кашњења криве опслуживања лако је израчунати границе кашњења (*D*) и бафера (*B*) за *VOQ1*(*i,k*) дат на слици 4.1.2 б):

$$D_{VOQ1ik} = (N-1)\cdot T_s + \sum_{j=1}^{N}(N+\sigma_{ij})/C \tag{4.1.8}$$

$$B_{VOQ1ik} = \sum_{j=1}^{N}(1+\sigma_{ij}/\mathrm{N}+\mathrm{R}_{ij}\cdot(\mathrm{N}-1)\cdot T_s/\mathrm{N}) \tag{4.1.9}$$

Крива долазног саобраћаја подтокова на излазу *VOQ1*(*i,k*) има исту просјечну брзину, али повећану спорадичност:

$$\sigma_{ij}^{'} = 1+\sigma_{ij}/N + R_{ij}\cdot\left((\mathrm{N}-1)\cdot \mathrm{T}_s + \sum_{\substack{j=1\\ j\neq i}}^{N}(N+\sigma_{ij})/C\right)/N \tag{4.1.10}$$

Међутим, сви *burst*-ови токова на улазном линку *i* не могу стићи симултано у истом моменту. Отуда, може се користити оптимистичнија једначина за повећану спорадичност гдје је утицај других токова одређен на основу:

$$\sigma_{ij}^{'} = 1+\sigma_{ij}/N + R_{ij}\cdot(N-1)\cdot T_s/\mathrm{N} \tag{4.1.11}$$

Токови, чије пакете треба прослиједити на исти излаз *j*, се смјештају у *VOQ2*(*k,j*) на порту *k* (*k*=1,…,*N*). Ови токови имају повећану спорадичност. Можемо израчунати границе кашњења и бафера са *VOQ2* редовима на исти начин као што је то урађено за бафере са *VOQ1* редовима (крива опслуживања је иста као и случају бафера са *VOQ1* редовима):

$$D_{VOQ2kj} = (N-1)\cdot T_s + \sum_{i=1}^{N}(N+\sigma_{ij}^{'})/C \tag{4.1.12}$$

$$B_{VOQ2kj} = \sum_{i=1}^{N}(1+\sigma_{ij}^{'}/N + R_{ij}\cdot(N-1)\cdot T_s/N) \tag{4.1.13}$$

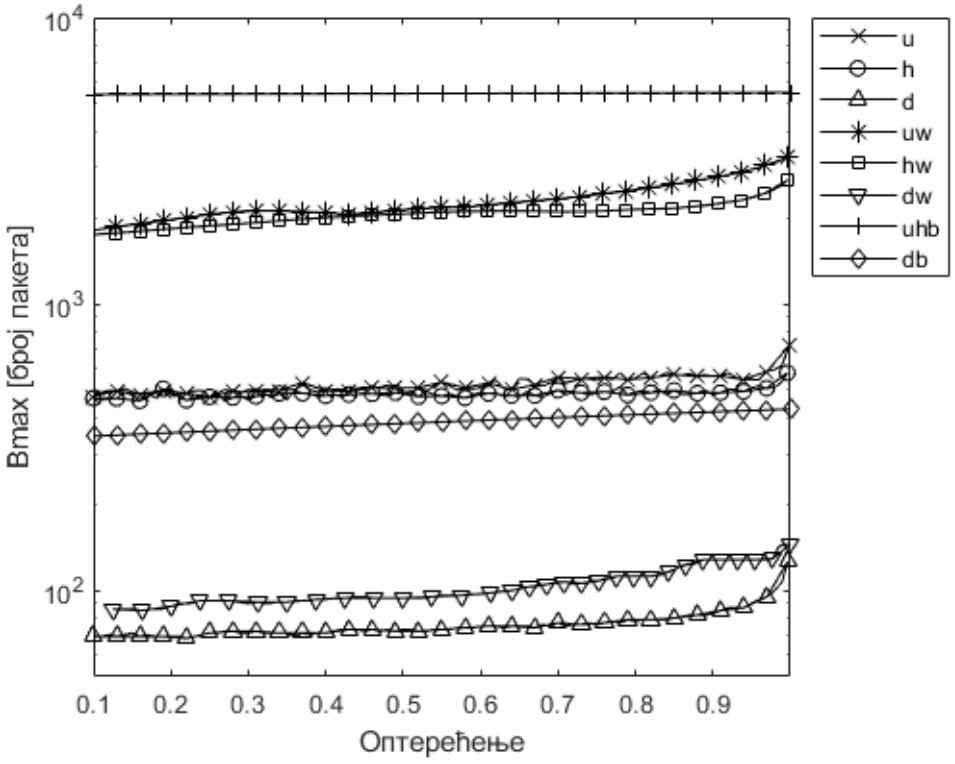
Максимално кашњење кроз *BvN-DLB* комутатор D_{max} и максимална захтијевана величина бафера по порту B_{max} могу се израчунати комбинујући границе бафера са *VOQ1* и *VOQ2* редовима:

$$D_{\max} = \max_{i,j,k=1,\ldots,N}(D_{VOQ1ik} + D_{VOQ2kj}) \tag{4.1.14}$$

$$B_{\max} = \max_{i=1,\ldots,N} \left(\sum_{k=1}^{N} (B_{VOQ1ik} + B_{VOQ2ik}) \right) \tag{4.1.15}$$

На основу формула 4.1.8, 4.1.9 и 4.1.11 види се да границе линеарно расту са порашћу спорадичности. Ова особина представља веома важну и добру особину предложеног рјешења.

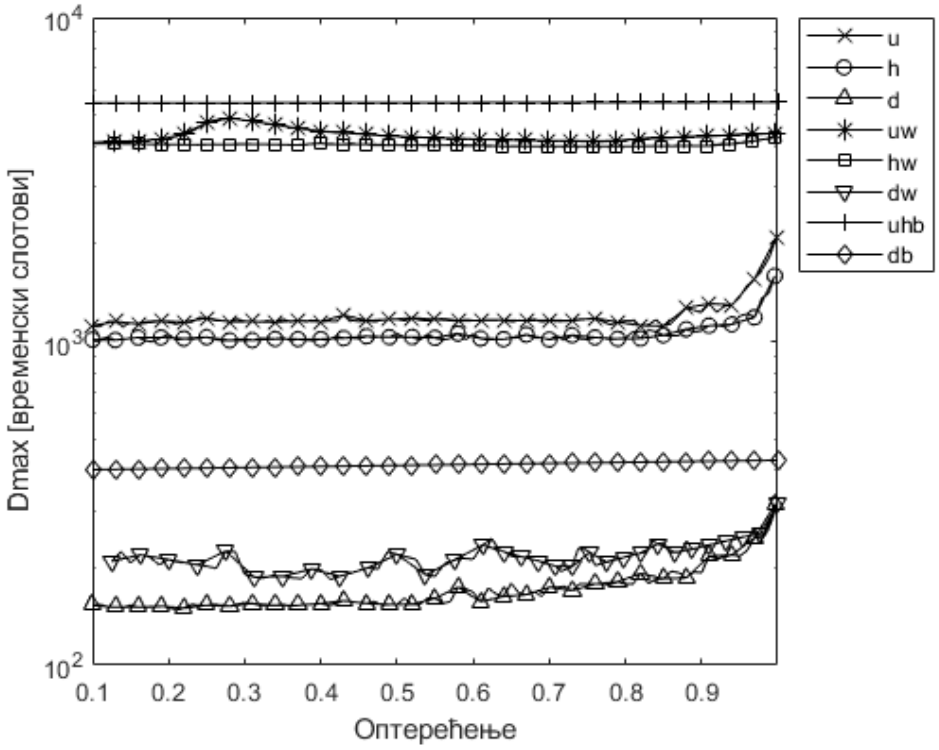
Како би се ови резултати провјерили, урађена је симулација предложеног рјешења. Симулирани су Бернулијев униформни (*u*), Бернулијев *hot-spot* (*h*) и дијагонални (*d*) саобраћајни сценарији. Бернулијев униформни модел саобраћаја одговара изразито симетричном саобраћају. У овом моделу важи да је оптерећење сваког улазног и сваког излазног порта једнако, као и да се са сваког улазног порта једнака количина саобраћаја шаље ка сваком од излазних портова. С друге стране, дијагонални модел саобраћаја представља изразито асиметричан модел саобраћаја, гдје сваки улазни порт шаље пакете само на два излазна порта. Тачније, 50% пакета са улазног порт *i* је намијењено за излазни порт *i*, а преосталих 50% пакета треба прослиједити на излазни порт (*i*+1) **mod** *N*. Бернулијев *hot-spot* модел саобраћаја представља својеврсну средину два претходна модела саобраћаја у смислу симетричности. У овом моделу се 50% саобраћаја са улазног порта *i* се прослеђује излазном порту *i*, док се осталих 50% равномјерно шаље према осталим излазним портовима.



Слика 4.1.4. Зависност максималног кашњења пакета *BvN-DLB* комутатора за различите саобраћајне сценарије

За сваки од ових сценарија је симулиран посебан случај (најнеповољнији сценарио - *worst case scenario*) гдје *burst*-ови стижу један за другим на свим улазним портовима, и *burst*-ови су синхронизовани у складу са њиховим дестинацијама. На примјер, када се посматра униформни или *hot-spot* сценарио, на свим улазним портовима: *burst* намијењен за излазни

порт 0 стиже први, затим стиже *burst* намијењен излазном порту 1 и тако даље. Овај посматрани случај би требало да представља најнеповољнији сценарио. У наставку су приказани резултати за 32x32 комутатор. Максимална величина *burst*-а за све токове је постављена на 100 пакета у свим сценаријима. Слике 4.1.4 и 4.1.5 показују D_{max} и B_{max} за симулирани комутатор. На графицима, *w* означава претходно поменути посебан случај, а *b* означава границе кашњења, односно бафера. За *y* осу је коришћена логаритамска скала. Границе које су добијене за униформни и *hot-spot* сценарије су исте. D_{max} и B_{max} су мањи у нормалном него у најгорем сценарију, зато што није много вјероватно да се најгори сценарио догоди. Из истог разлога се разлика између нормалног и најгорег сценарија повећава како се повећава спорадичност саобраћаја. Границе кашњења и бафера које су претходно прорачунате нису прекршене чак ни и у најгорем случају. Дакле, ове границе могу бити искоришћене за гарантовање кашњења и димензионисање бафера комутатора. D_{max} и B_{max} не расту значајно са порастом оптерећења зато што спорадичност саобраћаја има већи утицај на понашање комутатора него саобраћајно оптерећење.



Слика 4.1.5. Зависност захтијеване величине бафера *BvN-DLB* комутатора за различите саобраћајне сценарије

4.2. *Load-Balanced Birkhoff-von Neumann* комутатор са *Greedy Scheduling* алгоритмом

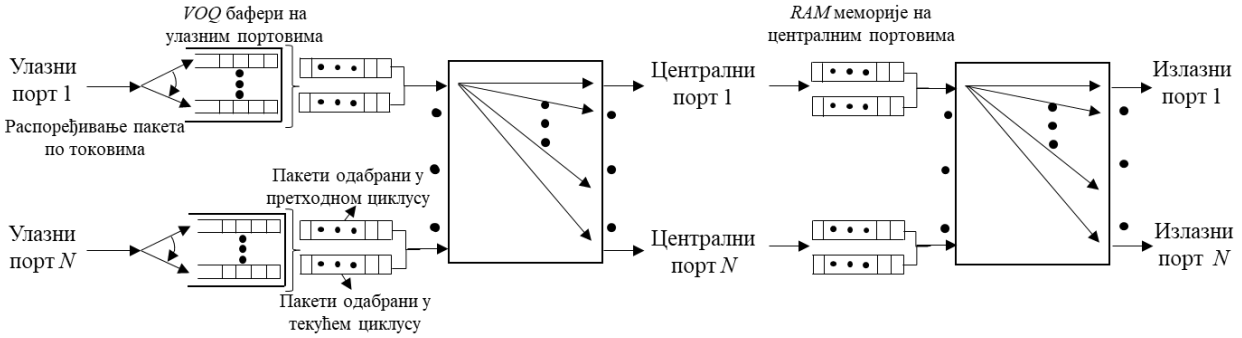
У овом потпоглављу ће бити представљен предлог рјешења *Load-Balanaced Birkhoff-von Neumann* комутатора са *Greedy Scheduling* алгоритмом (*LB-BvN-GS*) који је један од главних доприноса ове дисертације [137]. Рјешење *BvN-DLB* представљено у претходном поглављу се такође заснива на *LB-BvN* архитектури. Међутим, код овог рјешења постоји потреба за имплементацијом ресеквенционих бафера димензије N^2 на излазним портовима, што утиче на хардверску комплексност рјешења као и трошкове имплементације. Да би се

ријешио овај проблем, *LB-BvN-GS* комутатор комбинује најбоље особине комутатора са баферима на улазним портовима као и *LB-BvN* комутатора. Наиме, конфигурација *LB-BvN-GS* комутатора је унапријед позната у сваком временском слоту, чиме се избјегава главни проблем код комутатора са баферима на улазним портовима, а то је потреба за прорачунавањем конфигурације комутатора у сваком временском слоту. Такође, пошто су конфигурације оба комутациона модула исте и временски синхронизоване, *LB-BvN-GS* комутатор, за разлику од већине осталих рјешења на бази *LB-BvN* комутатора, не мора да користи два физичка комутациона модула, већ само један. Архитектура где су два комутациона модула практично спојена у један се назива преклопљена архитектура. Преклопљена архитектура је могућа за имплементацију јер су у пракси улазни порт *i*, централни порт *i* и излазни порт *i* имплементирани на истој линијској картици. Преклопљена архитектура захтева убрзање интерних линкова 2 пошто се у истом слоту кроз исту конфигурацију прослеђује један пакет са улазног порта ка централном порту, као и један пакет са централног порта на излазни порт. Али, убрзање конфигурације комутационог модула није потребна. Код других рјешења заснованих на *LB-BvN* комутатору конфигурације оба комутациона модула углавном или нису исте или временски синхронизоване услед чега не могу да користе преклопљену архитектуру.

LB-BvN-GS комутатор на улазним портовима има механизам за распоређивање пакета који решава проблем поремећаја редоследа пакета. Предложено је и алтернативно рјешење, *mLB-BvN-GS* комутатор, који умјесто овог механизма користи ресеквенционе бафере који су смјештени на излазним портовима, при чему је њихова величина ограничена на *N* пакета што је прихватљиво са становишта хардверске комплексности и знатно боље од свих осталих рјешења која користе ресеквенционе бафере код којих је минимална величина N^2. Овај комутатор има мање кашњење и једноставније улазне портове у односу на *LB-BvN-GS*. У наставку овог потпоглавља ће бити представљена архитектура *LB-BvN-GS* и *mLB-BvN-GS* комутатора, објашњен принцип рада и биће доказано да је комутатор неблокирајући уз унутрашње убрзање 2. У наредном поглављу ће се показати да овај комутатор постиже боље перформансе у односу на конкурентска рјешења при великим оптерећењима, уз прихватљиве захтјеве у смислу хардверске комплексности.

На слици 4.2.1 је дата архитектура *LB-BvN-GS* комутатора. Она се састоји од два комутациона модула, између којих се налазе централни портови. Конфигурације оба комутациона модула су у сваком временском слоту унапријед познате, и исте су у оба комутациона модула. Самим тим нема потребе за прорачунавањем конфигурација комутационих модула у сваком слоту. Слика 4.3.1 показује комутационе модуле раздвојене ради лакшег разумевања, али, као што је речено, подржана је преклопљена архитектура где се користи само један комутациони модул. На сваком улазном порту се налази бафер са *N VOQ1* редова, по један за пакете које треба прослиједити на сваки од излазних портова. Такође, на сваком улазном порту, за сваки ток постоје двије додатне *VOQ* меморије које означавамо као *sVOQ* меморије. Свака од ових меморија има капацитет за чување *N* пакета. Једна од ових меморија служи за смештање одабраних пакета који ће бити послати у наредном циклусу. У другом *sVOQ* се чувају пакети који су одабрани у претходном циклусу и који се шаљу у току текућег циклуса. Улоге ових меморија се међусобно мијењају на почетку сваког циклуса. Осим тога, на сваком улазном порту постоје и двије *RAM* меморије, у којима се чувају информације о највише *N* пакета који ће бити прослијеђени на централне портове. И овдје једна *RAM* меморија садржи информације о пакетима који ће бити послати у наредном циклусу док се у другој *RAM* меморији чувају информације о пакетима који су одабрани у претходном циклусу и који се шаљу у току текућег циклуса. И ове двије

меморије међусобно мијењају своје улоге на почетку сваког циклуса. На сваком централном порту се налазе по двије *RAM* меморије димензије *N* локација. Свака локација одговара једном излазном порту. У једној меморији се смјештају пакети који су пристигли са улазних портова у току текућег циклуса, и који ће бити прослијеђени на излазне портове у току сљедећег циклуса. У другој меморији се налазе пакети који су примљени у току претходног циклуса и који се шаљу на излазне портове у текућем циклусу. Када се централни порт повеже са излазом *k* тада се шаље пакет са локације *k*. Улоге ове двије *RAM* меморије на централном порту се такође међусобно мијењају на почетку сваком циклуса.



Слика 4.2.1. Архитектура *LB-BvN-GS* комутатора

Приликом селекције пакета који ће бити послати са улазних портова користи се sekvencijalni *greedy scheduling* (*GS*) алгоритам. Наиме, сваком централном порту је придружен одговарајући вектор заузетости тог порта *V*, који има *N* бита. Сваки бит показује да ли је за дати централни порт већ одабран пакет којег треба прослиједити на излазни порт чији је редни број исти као и редни број бита у вектору заузетости тог порта. Подсећамо да *RAM* меморије на централним портовима чувају по један пакет за сваки излаз. Ако је вриједност неког бита 1, то значи да за посматрани централни порт није одабран пакет за одговарајући излазни порт. У супротном, ако је вриједност бита 0, то значи да је за посматрани централни порт већ одабран пакет за одговарајући излаз. На почетку сваког циклуса сви бити у векторима заузетости сваког централног порта имају вриједност 1. Како је већ речено, конфигурације оба комутациона модула су исте и одређене су формулама:

$$j = (i+t) \bmod N \tag{4.2.1}$$

односно

$$k = (j+t) \bmod N \tag{4.2.2}$$

гдје *i* представља улазне портове, *j* централне портове, а *k* излазне портове.

Дакле, на почетку циклуса, улазни порт 1 је спојен са централним портом 1, улазни порт 2 са централним портом 2,…, улазни порт *N* са централним портом *N*. Такође, централни порт 1 је спојен са излазним портом 1, централни порт 2 са излазним портом 2,…, централни порт *N* са излазним портом *N*. Тако, на почетку сваког циклуса, улазни порт 1 процесира вектор заузетости централног порта 1 (V_1), улазни порт 2 процесира вектор заузетости централног порта 2 (V_2),…, улазни порт *N* процесира вектор заузетости централног порта *N* (V_N). Сваки улазни порт бира пакет који се може послати преко централног порта с којим је тренутно повезан. У првом слоту сваког циклуса, улазни портови могу да одаберу пакет из било ког тока, јер за централне портове није одабран пакет ни за један излазни порт. Ако улазни портови могу послати пакете из више *VOQ1* редова, онда се бира пакет из оног *VOQ1* реда у коме има највише пакета. Када се одабере пакет који ће бити

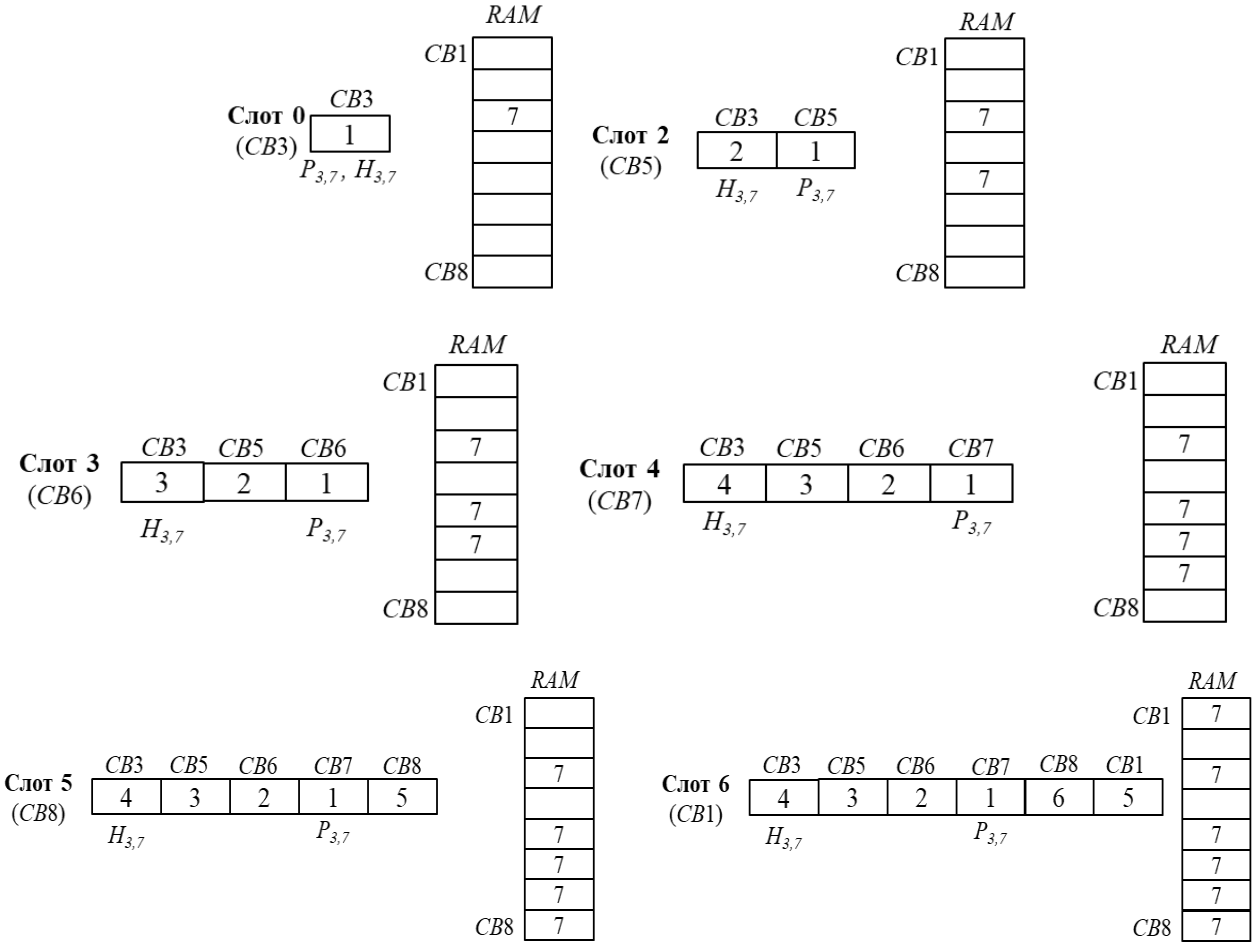
послат на дати централни порт, улазни порт поставља на 0 вриједност бита који одговара излазном порту на који ће бити прослијеђен тај пакет. Након ажурирања, вектор заузетости датог централног порта се прослеђује улазном порту који ће у сљедећем слоту бити повезан са тим централним портом. На примјер, улазни порт i који на почетку циклуса процесира V_i (јер је повезан са централним портом i), у сљедећем слоту ће процесиратиV$_{(i+1)\mathbf{mod}N}$ који ће му бити прослијеђен од улазног порта (i+1) **mod** N. Такође, улазни порт i ће вектор V_i који је процесирао у претходном слоту прослиједити улазном порту (i-1) **mod** N, јер ће овај улазни порт у сљедећем слоту бити повезан са централним портом i. На тај начин, сваки улазни порт комуницира само са два сусједна улазна порта, једним коме прослеђује вектор заузетости централног порта с којим је био повезан у претходном временском слоту, и другим од кога прима вектор заузетости централног порта с којим је повезан у датом временском слоту. На овај начин се осигурава да сваки централни порт у току једног циклуса може примити највише по један пакет за сваки од излазних портова. Последично, сви пакети које неки централни порт прими у току једног циклуса ће сигурно бити прослијеђени на излазне портове у току сљедећег циклуса.

Да би се осигурало да пакети једног тока хронолошки стижу на излазне портове, на улазним портовима је имплементиран механизам за распоређивање пакета. Како је већ речено на почетку овог потпоглавља, на сваком улазном порту, осим N *VOQ1* редова који служе за чување пакета N токова на том порту, за сваки ток постоје и двије додатне *sVOQ* меморије. Један *sVOQ* служи за чување пакета који су одабрани у текућем циклусу, и који ће бити послати у сљедећем циклусу. У другом *sVOQ* су смјештени пакети који су одабрани у претходном циклусу и који се шаљу на централне бафере у текућем циклусу. Улоге ова два *sVOQ* се мијењају у сваком циклусу. Такође, на сваком улазном порту постоје двије *RAM* меморије у којима се чувају информације о пакетима који се шаљу. Свака меморија има N локација, при чему свака локација одговара једном централном порту. Тако, када се изабере пакет који ће бити послат централном баферу j, онда се у меморијској локацији j чува идентификација излазног порта на који треба прослиједити одабрани пакет. У сљедећем циклусу, када се посматрани улазни порт повеже са централним портом j, онда улазни порт из меморијске локације j ишчита идентификацију *sVOQ*-а из кога пакет треба да буде прослијеђен. У једној *RAM* меморији се чувају информације о пакетима изабраним у текућем циклусу који ће бити послати централним портовима у сљедећем циклусу. У другој *RAM* меморији се чувају информације о пакетима изабраним у претходном циклусу који се шаљу централним портовима у текућем циклусу. Улоге ове двије меморије се мијењају на почетку сваког циклуса.

Пошто су конфигурације оба комутациона модула познате, може се знати и којим редоследом ће пакети једног тока стићи на одговарајући излаз. Њиховим пажљивим распоређивањем, може се осигурати да они на одговарајући излаз стигну у правилном редоследу. Принцип рада овог механизма је објашњен у наставку. Наиме, на сваком улазном порту, за сваки ток пакета који пролази кроз тај порт постоје два показивача, *head* и *pivot*. Посматрајмо неки ток пакета $F(i,k)$. У првом слоту сваког циклуса, улазни порт i процесира V_i, затим у другом слоту $V_{(i+1)\ \mathbf{mod}\ N}$ итд. Показивачи *head* и *pivot* показују на први одабрани пакет из неког тока. Сви пакети који су одабрани за слање, процесирањем вектораV_i, $V_{(i+1)\ \mathbf{mod}\ N}$,…, V_k, се додају на почетак реда, и *head* показивач се ажурира да показује на последњи додати пакет. На овај начин, пакети који су раније дошли на улазни порт ће бити послати на централне портове који ће се раније повезати са одговарајућим излазним портом. Након што улазни порт почне са обрађивањем вектора $V_{(k+1)\ \mathbf{mod}\ N}$ и наредних вектора V, сви одабрани пакети ће бити додати иза *pivot* показивача. При томе, *pivot* показивач се не ажурира већ све

вријеме показује на исти пакет. Ово гарантује да ће пакети који су касније одабрани бити послати на централне портове који ће се касније повезати на одговарајући излаз.

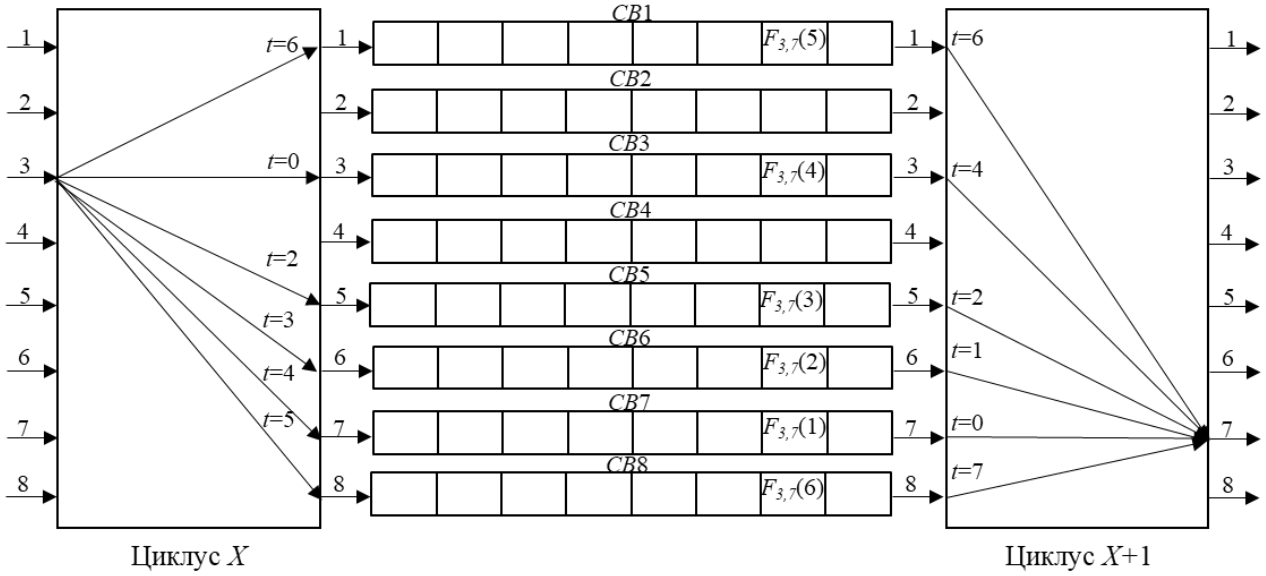
На примјеру са слике 4.2.2 је приказан конкретан пример везан за распоређивање одабраних пакета, а да се очува њихов редослед приликом њиховог пристизања на излазни порт. Овај пример би требало да појасни објашњења дата у претходном делу текста. Једноставности ради, посматра се комутатор величине 8x8 (исти принципи наравно важе за комутаторе било које величине). Рецимо да се посматра ток $F(3,7)$. У првом слоту (t=0, јер нумерација почиње од 0), улазни порт 3 бира пакет који ће бити прослијеђен централном порту 3. Бира се први пакет тока и на њега ће показивати и одговарајући *head* и *pivot* показивачи. Истовремено, у *RAM* меморији, у меморијску локацију под редним бројем 3, се уписује редни број излазног порта на који треба прослиједити одабрани пакет.



Слика 4.2.2. Примјер распоређивања пакета на улазним портовима *BvN-GS* комутатора

У овом конкретном примјеру, то је излазни порт 7. Нека се у другом слоту изабере пакет из неког другог тока, и нека се у трећем слоту опет бира пакет из тока $F(3,7)$. У трећем слоту циклуса, улазни порт 3 ће бити повезан са централним портом 5. Пошто се V_5 налази између V_3 и V_7, сљедећи пакет тока ће бити додат испред пакета на који тренутно показује *head* показивач. Напомена, у листи на слици 4.2.2 се наводе редни бројеви пакета из тока $F(3,7)$. Такође, *head* показивач се ажурира да показује на последњи додати пакет, док *pivot* показивач и даље показује на исти пакет. Опет се у *RAM* меморију у меморијску локацију под редним бројем 5 уписује излазни порт 7. У четвртом слоту циклуса улазни порт се

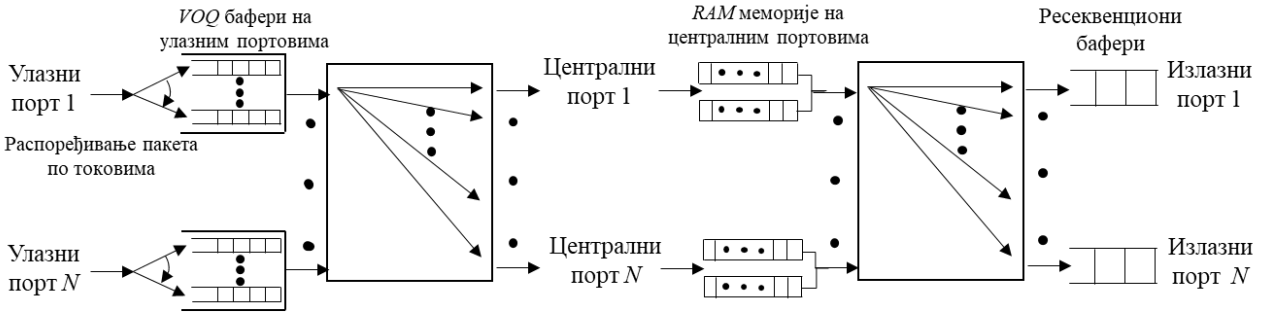
повезује са централним портом 6. Пошто се V_6 налази између V_3 и V_7 пакет тока $F(3,7)$ изабран у овом слоту ће бити додат испред пакета на који тренутно показује *head* показивач, а затим се овај показивач ажурира да показује на последњи додати пакет. У петом слоту се бира пакет који ће бити послат централном порту 7. Опет узмимо да се бира пакет тока тока $F(3,7)$. Пошто се V_7 налази између V_3 и V_7 и овај пакет ће бити додат испред пакета на који тренутно показује *head* показивач, а затим се овај показивач ажурира да показује на последњи додати пакет. У шестом слоту се бира пакет који ће бити послат централном порту 8. Опет узмимо да се бира пакет тока тока $F(3,7)$. Пошто се V_8 налази изван опсега између V_3 и V_7, пакет који је изабран у овом слоту се додаје иза пакета на који показује *pivot* показивач. За разлику од *head* показивача који се ажурира, *pivot* показивач све вријеме показује на исти пакет (који је одабран још у првом временском слоту). У седмом слоту се бира пакет који ће бити послат централном порту 1. Опет узмимо да се бира пакет тока тока $F(3,7)$. Пошто се V_1 налази изван опсега између V_3 и V_7, пакет који је изабран у овом слоту се додаје иза пакета на који показује *pivot* показивач. За разлику од *head* показивача који се ажурира, *pivot* показивач све вријеме показује на исти пакет (који је одабран још у првом временском слоту). Претпоставља се да је у осмом слоту изабран пакет који припада неком другом току. Користећи *head* и *pivot* показиваче направљен је распоред пакета који гарантује да ће сви пакети тока $F(3,7)$ на излазни порт 7 доћи у правилном редоследу. Наравно, и *RAM* меморија је ажурирана тако садржи информације из којих токова пакете треба прослиједити на одговарајуће централне портове.



Слика 4.2.3. Примјер слања пакета у *LB-BvN-GS* комутатору

На слици 4.2.3 је показано којим редоследом ће се пакети тока $F(3,7)$ из датог примера слати са улазних на централне портове, као и са централних на излазне портове. Дакле, у циклусу X-1 је извршен одабир и распоређивање пакета за слање, што је објашњено у претходном примјеру. У циклусу X се врши прослеђивање пакета са улазних на централне портове. У првом слоту се четврти пакет тока $F(3,7)$ шаље на централни порт 3. У слоту t=2 се трећи пакет тока $F(3,7)$ шаље на централни порт 3. У слоту t=3 се други пакет тока $F(3,7)$ шаље на централни порт 3.У слоту t=4 се први пакет тока $F(3,7)$ шаље на централни порт 7.У слоту t=5 се шести пакет тока $F(3,7)$ шаље на централни порт 8. Коначно у слоту t=6 се пети пакет тока $F(3,7)$ шаље на централни порт 1. У циклусу X+1 се врши прослеђивање пакета са

централних на излазне портове. Излазни порт 7 ће у слоту t=0 бити повезан на централни порт 7 који ће му послати први пакет тока $F(3,7)$. У слоту t=1 ће излазни порт 7 бити повезан на централни порт 6 који ће му послати други пакет тока $F(3,7)$. У слоту t=2 ће излазни порт 7 бити повезан на централни порт 5 који ће му послати трећи пакет тока $F(3,7)$. У слоту t=4 ће излазни порт 7 бити повезан на централни порт 3 који ће му послати четврти пакет тока $F(3,7)$. У слоту t=6 ће излазни порт 7 бити повезан на централни порт 1 који ће му послати пети пакет тока $F(3,7)$. Коначно, у слоту t=7 ће излазни порт 7 бити повезан на централни порт 8 који ће му послати шести пакет тока $F(3,7)$. Дакле, јасно је да ће сви пакети тока $F(3,7)$ на излазни порт стићи у правилном редоследу.



Слика 4.2.4. Архитектура *mLB-BvN-GS* комутатора

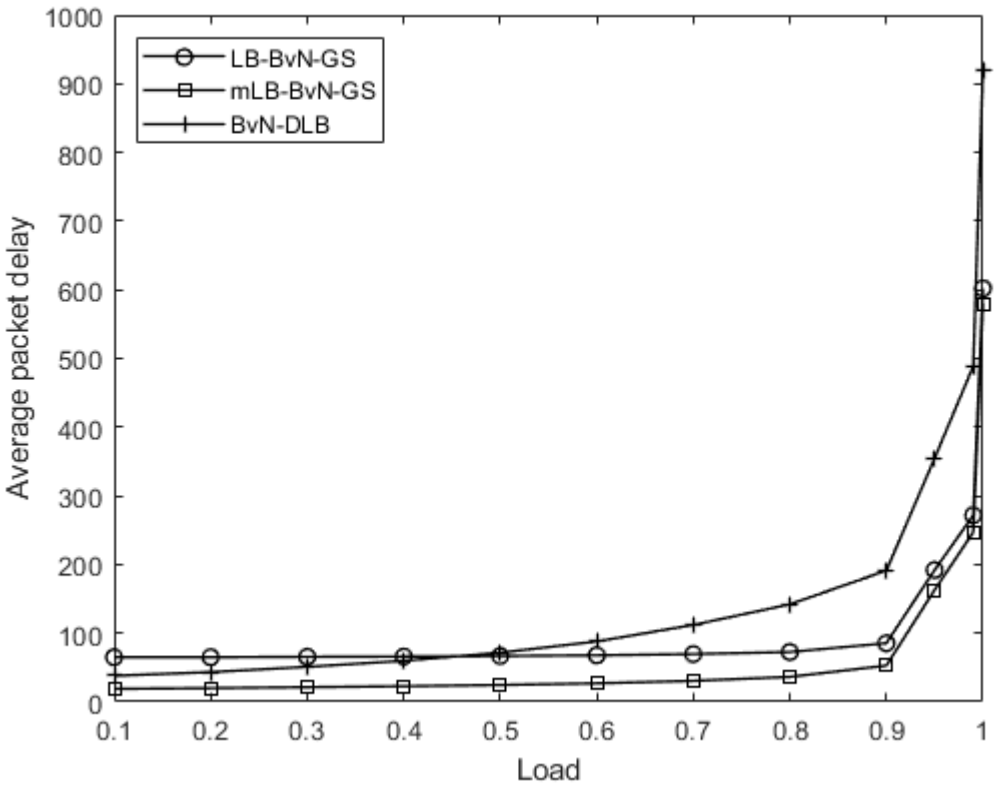
Такође, у овој дисертацији је предложено и алтернативно рјешење које не користи овај механизам за распоређивање пакета. Ово модификовано рјешење се у даљем тексту означава као *mLB-BvN-GS*. Ово рјешење има сличну архитектуру као *LB-BvN-GS* комутатор (слика 4.2.4). Дакле, састоји се од два комутациона модула, који имају идентичне конфигурације које су одређене формулама (4.2.1) и (4.2.2). На сваком улазном порту постоји по N *VOQ1* редова, по један за сваки ток пакета који пролази кроз одређени улазни порт. Између два комутациона модула се налазе централни портови, и даље са по 2 *RAM* меморије, при чему се у свакој меморији смјешта највише по један пакет за сваки од излазних портова - локација k одговара излазном порту k. Основне разлике су у томе што је поједностављен улазни порт пошто нема механизма за дефинисање редоследа пакета тока који не ремети редослед пакета на излазу и додавање ресеквенционих бафера на излазним портовима, али уз важну напомену да је димензија ресеквенционих бафера свега N. Код овог рјешења, када улазни порт у датом временском слоту одабере пакет за слање, тај пакет се одмах прослеђује централном баферу с којим је повезан. На овај начин се не чека цијели један циклус прије него су одабрани пакети послати на централне портове, што утиче на смањење кашњења. Такође, пошто се на улазним портовима не врши распоређивање пакета, нема потребе за *sVOQ* редовима нити за *RAM* меморијама на улазним портовима. С друге стране, пакети ће у оваквом режиму рада на излазне портове стизати у поремећеном редоследу. Да би се ријешио овај проблем, *mLB-BvN-GS* на сваком излазном порту има ресеквенциони бафер, у коме се врши исправљање редоследа пакета. Међутим, величина ових бафера је ограничена на N пакета, пошто је то максималан број пакета које један излазни порт може примити у току једног циклуса. То је значајно мање него код осталих рјешења, код којих је величина ресеквенционог бафера N^2 или већа.

LB-BvN-GS комутатор је неблокирајући за произвољни дозвољен саобраћајни сценарио када се користи убрзање 2. Посматрајмо довољно дуг временски период од T слотова такав да ниједан излаз није преоптерећен тј. ни за један излаз нема више од T пакета. Узмимо неки произвољан пакет и анализирајмо најгори случај за његово распоређивање. У најгорем случају, посматрани пакет мора да чека све пакете који су намењени жељеном

излазу, а таквих пакета сигурно нема више од T-1. С друге стране, у најгорем случају посматрани пакет мора да чека и све друге пакете са истог улаза, а ни њих нема више од T-1 јер на један улаз је могло да стигне максимално T пакета у посматраном периоду. Самим тим, максимално чекање у најгорем случају је збир претходно два наведена времена чекања, односно 2T-2. У складу с тим, ако постоји убрзање 2, *LB-BvN-GS* комутатор би био неблокирајући. Наведено важи и за *mLB-BvN-GS* комутатор. Суштински овај доказ је идентичан ономе за *SGS* комутатор, пошто се користи веома сличан принцип. У *SGS* случају се пакети распоређују за будуће временске слотове, а у *LB-BvN-GS* случају централни портови играју улогу будућих временских слотова. Важно је нагласити да се ово интерно убрзање се односи на брзине линкова, али не и на конфигурације комутационих модула што ово рјешење чини атрактивнијим. Међутим, како ће бити показано у сљедећем поглављу, и без интерног убрзања се постижу веома добре перформансе.

Рјешење представљено у овом потпоглављу је одабрано и као основа за даља унапређења у смислу подршке за мултикаст саобраћај и фер сервис, јер нема проблем поремећаја редоследа пакета. Такође, и алтернативно рјешење (*mLB-BvN-GS*) код којег долази до поремећаја редоследа пакета, величина ресеквенционих бафера на излазним портовима којима се тај проблем решава је прихватљива и скалабилна.

4.3. Поређење перформанси рјешења предложених у дисертацији

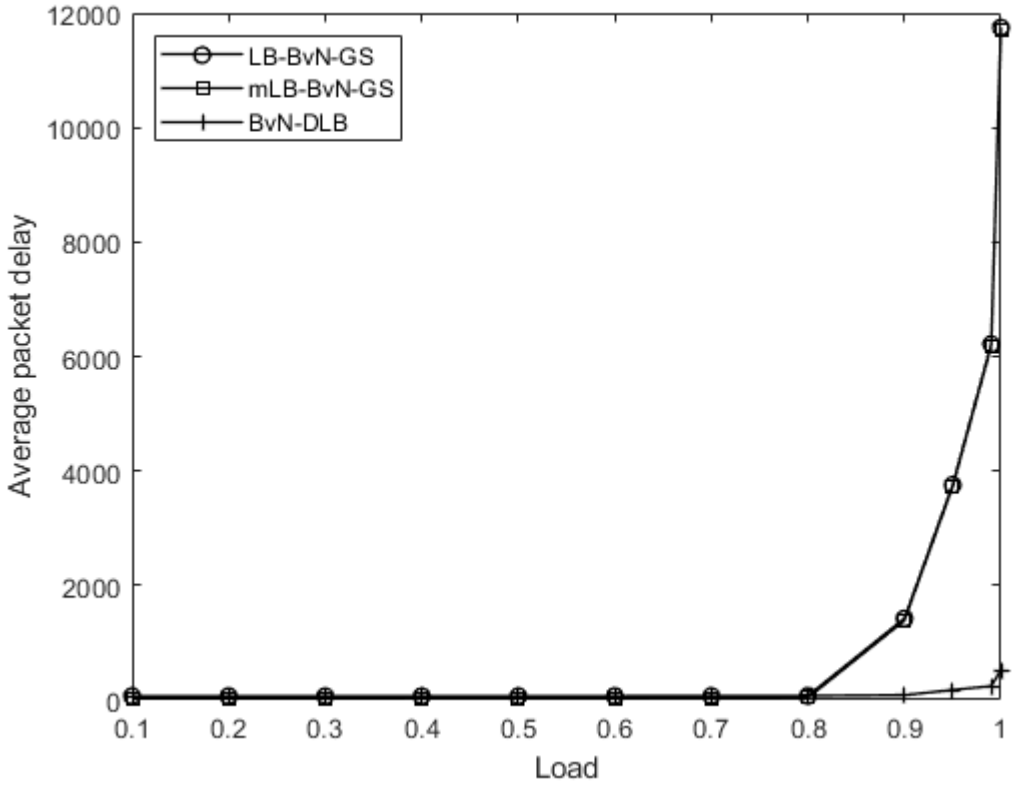


Слика 4.3.1. Резултати за Бернулијев униформни саобраћај

У овом потпоглављу су упоређене перформансе рјешења која су предложена у дисертацији, односно у претходна два потпоглавља. Перформансе комутатора се мјере на основу просјечног кашњења пакета од тренутка кад дође у комутатор, па док не буде

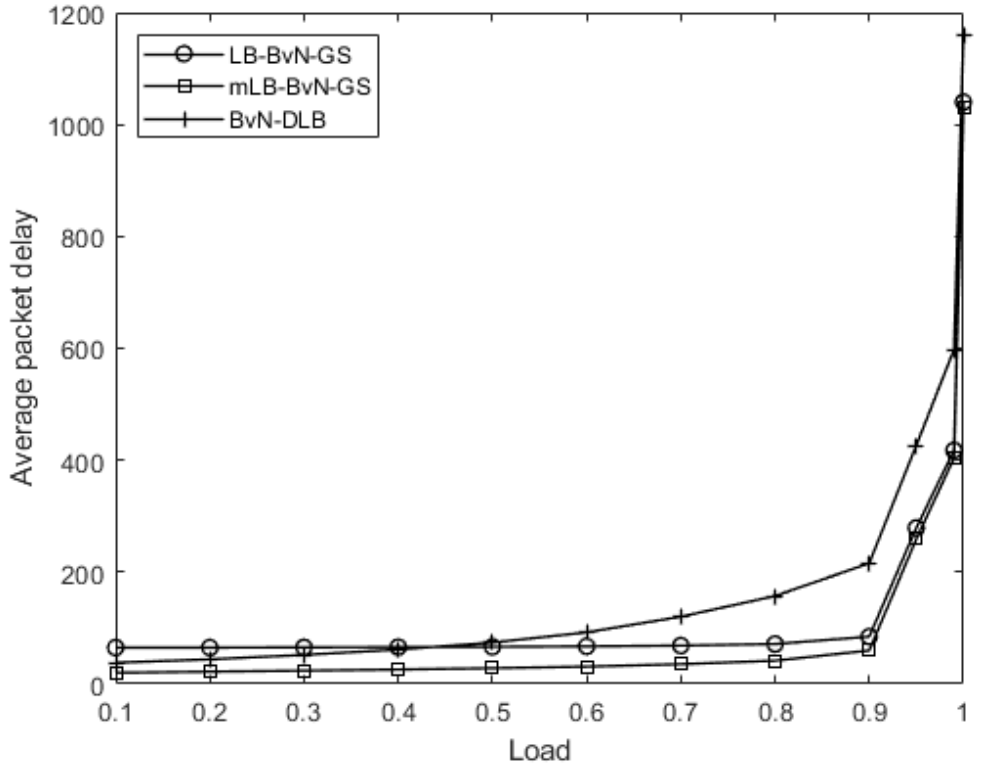
прослијеђен сљедећем мрежном чвору. Посматрају се резултати за различита оптерећења портова. Представљени су резултати за 32x32 комутаторе. Рјешења су тестирана за различите сценарије саобраћаја: Бернулијев униформни, дијагонални, Бернулијев *hot-spot* и униформни *bursty* саобраћај. Прва три модела саобраћаја су описани у потпоглављу 4.1. Униформни *bursty* модел саобраћаја се користи за моделовање саобраћаја на интернету који је по природи спорадичан. У симулацији је претпостављено да је просјечна величина једног *burst*-а података *s*=30 пакета [133].

На слици 4.3.1 су дати резултати за Бернулијев униформни саобраћај. Са слике се уочава да *BvN-DLB* комутатор има веће кашњење при средњим и већим оптерећењима, Такође, *LB-BvN-GS* има нешто веће кашњење у односу на *mLB-BvN-GS*, због додатног циклуса током којег се врши одабир и распоређивање пакета за слање. Међутим, при највећим оптерећењима разлика се смањује. На слици 4.3.2 су дати резултати за дијагонални саобраћај. Овдје се уочава да сва рјешења имају сличне перформансе за мања и средња оптерећења, док при највећим оптерећењима *BvN-DLB* постиже најбоље перформансе.

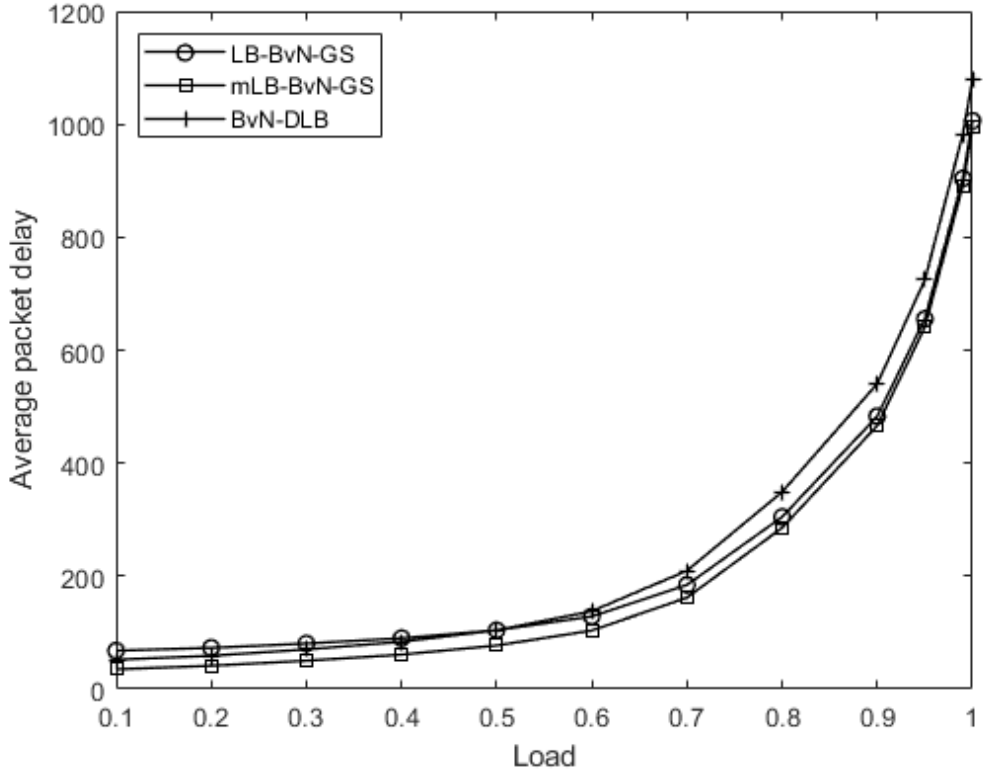


Слика 4.3.2. Резултати за дијагонални саобраћај

На слици 4.3.3 су приказани резултати за Бернулијев *hot-spot* саобраћај. За овај модел саобраћаја *BvN-DLB* комутатор постиже веће кашњење при средњим и већим оптерећењима, док *LB-BvN-GS* и *mLB-BvN-GS* имају врло сличне перфромансе. И овдје *mLB-BvN-GS* постиже нешто боље перформансе, нарочито при нижим оптерећењима. На слици 4.3.4 су дати резултати за униформни *bursty* саобраћај. Са слике се уочава да сва рјешења постижу сличне перформансе - нешто слабије перформансе при средњим и већим оптерећењима има *BvN-DLB* комутатор.



Слика 4.3.3. Резултати за Бернулијев *hot-spot* саобраћај



Слика 4.3.4. Резултати за униформни *bursty* саобраћај

На основу претходно изложених резултата се може закључити да *LB-BvN-GS* и *mLB-BvN-GS* комутатори постижу сличне или боље перформансе у односу на *BvN-DLB* за скоро све сценарије и оптерећења. Анализа је показала да су *LB-BvN-GS* и *mLB-BvN-GS* најбоље избалансирани по питању различитих типова саобраћаја са становишта симетричности и спорадичности (*burstiness*). *BvN-DLB* није адекватан за симетрични саобраћај, а управо овај сценарио је најчешће анализиран у литератури. Осим тога, *LB-BvN-GS* и *mLB-BvN-GS* су једноставнија за имплементацију јер се не захтијева имплементација ресеквенционих бафера (*LB-BvN-GS*) или је њихова величина ограничена на само *N* пакета (*mLB-BvN-GS*). Из тог разлога су ова два рјешења одабрана за поређење са другим рјешењима. Такође, ова два рјешења су искоришћена као основа за даље унапређење како би се ефикасно управљало и мултикаст саобраћајем.

5. ПОРЕЂЕЊЕ СА ПОСТОЈЕЋИМ УНИКАСТ РЈЕШЕЊИМА

У овом поглављу ће бити представљено поређење перформанси најпопуларнијих постојећих рјешења уникаст комутатора са *LB-BvN-GS* и *mLB-BvN-GS* рјешењима. Као метрика се користи просјечно кашњење пакета, при чему се посматра кашњење од тренутка када пакет стигне на улазни порт комутатора, па док са излазног порта не буде послат даље у мрежу. Такође, подразумијева се да пакети напуштају комутатор у истом редоследу у ком су и дошли у комутатор. У предстојећим анализама се посматрају комутатори са 32 улазна и 32 излазна порта, осим ако није другачије назначено. Анализе су урађене и за друге величине комутатора, али су закључци исти као и они који ће бити представљени у наставку. Посматрају се перформансе за различита оптерећења, при чему се подразумијева да важи:

$$\sum_{j=1}^{N} \lambda_{i,j} \leq 1,\ i=1,2,\ldots,N \tag{5.1}$$

и

$$\sum_{i=1}^{N} \lambda_{i,j} \leq 1,\ j=1,2,\ldots,N \tag{5.2}$$

гдје је $\lambda_{i,j}$ нормализована количина саобраћаја између улазног порта i и излазног порта j. Ово значи да у случају највећег оптерећења, у сваком временском слоту на сваки улазни порт може да дође највише један пакет. Такође, ниједан излазни порт није преоптерећен - формула (5.2).

Перформансе свих комутатора су посматране за различите моделе саобраћаја, и то: Бернулијев униформни саобраћај, Бернулијев *hot-spot* саобраћај, Бернулијев дијагонални саобраћај и *bursty* униформни саобраћај. Бернулијев униформни модел саобраћаја одговара изразито симетричном саобраћају. У овом моделу важи да је оптерећење сваког улазног и сваког излазног порта једнако, као и да се са сваког улазног порта једнака количина саобраћаја шаље ка сваком од излазних портова. На слици 5.1 је дата матрица овог модела саобраћаја.

$$\begin{pmatrix} \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \frac{1}{N} & \cdots & \frac{1}{N} \end{pmatrix}$$

Слика 5.1. Матрица Бернулијевог униформног саобраћаја

С друге стране, дијагонални модел саобраћаја представља изразито асиметричан модел саобраћаја, гдје сваки улазни порт шаље пакете само на два излазна порта. Тачније, 50% пакета са улазног порт i је намијењено за излазни порт i, а преосталих 50% пакета треба прослиједити на излазни порт (i+1) **mod** N. На слици 5.2 је дата матрица оваквог модела саобраћаја.

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdots & 0 \\ \vdots & & & \vdots & & \vdots \\ \vdots & & & \vdots & & \vdots \\ \frac{1}{2} & 0 & \cdots & \cdots & 0 & \frac{1}{2} \end{pmatrix}$$

Слика 5.2. Матрица дијагоналног саобраћаја

Бернулијев *hot-spot* модел саобраћаја представља комбинацију својеврсну средину два претходна модела саобраћаја са становишта симетричности. У овом моделу се 50% саобраћаја са улазног порта i се прослеђује излазном порту i, док се осталих 50% равномјерно шаље према осталим излазним портовима. На слици 5.3 је дата матрица оваквог модела саобраћаја.

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2\cdot(N-1)} & \frac{1}{2\cdot(N-1)} & \cdots & \frac{1}{2\cdot(N-1)} \\ \frac{1}{2\cdot(N-1)} & \frac{1}{2} & \frac{1}{2\cdot(N-1)} & \cdots & \frac{1}{2\cdot(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{1}{2\cdot(N-1)} & \cdots & \cdots & \frac{1}{2\cdot(N-1)} & \frac{1}{2} \end{pmatrix}$$

Слика 5.3. Матрица Бернулијевог *hot-spot* саобраћаја

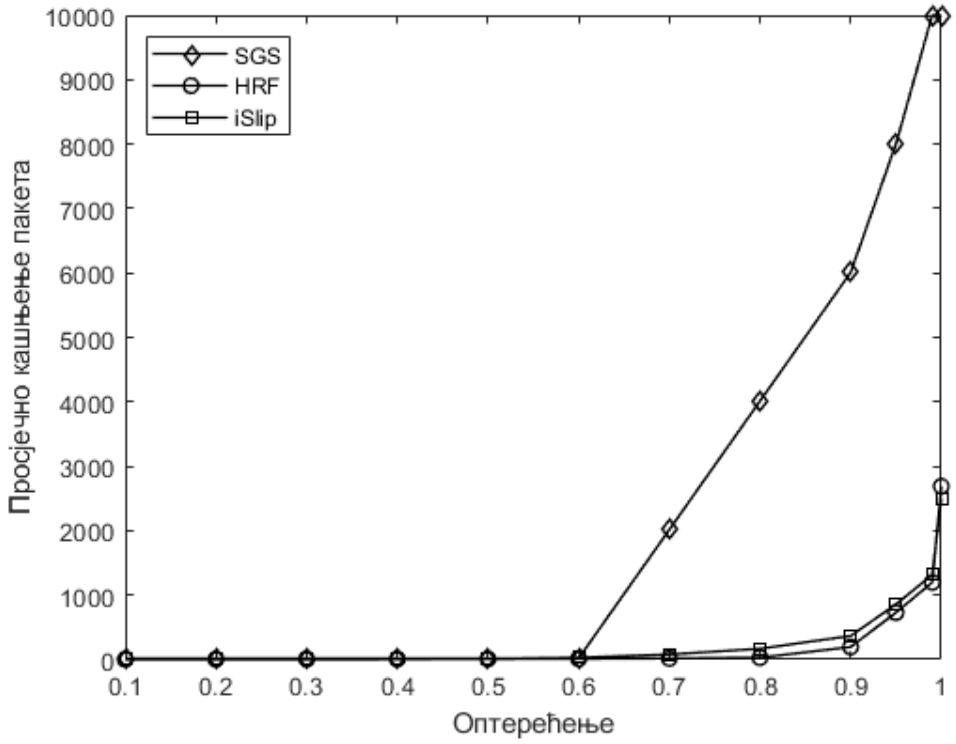
Коначно, *bursty* униформни модел саобраћаја можда најбоље одговара реалном саобраћају у мрежи, нарочито када се зна да генерално саобраћај на интернету има *bursty* карактер. За моделовање овог типа саобраћаја коришћен је *ON-OFF* модел, и претпостављено је да је просјечна величина једног *burst*-а података s=30 пакета [133]. То значи да ако у претходном временском слоту улазни порт није примио пакет, вјероватноћа да ће у текућем временском слоту примити пакет је:

$$P = \frac{\lambda}{s \cdot (1 - \lambda)} \tag{5.1}$$

гдје је λ просјечно оптерећење улазних портова. У супротном, ако је у претходном временском слоту улазни порт примио пакет, вјероватноћа да у текућем временском слоту неће примити пакет је:

$$P = \frac{1}{s} \tag{5.2}$$

Када су у питању комутатори са баферима на улазним портовима анализирани су *iSlip*, *SGS* и *HRF* комутатори. *iSlip* је одабран као представник комутатора са централним распоређивачем, *SGS* као представник комутатора са дистрибурианим распоређивачем и *HRF* као представник најновијих рјешења комутатора са баферима на улазним портовима. Представници су изабрани по томе што остварују најбоље перформансе у поређењу са другим рјешењима исте класе.

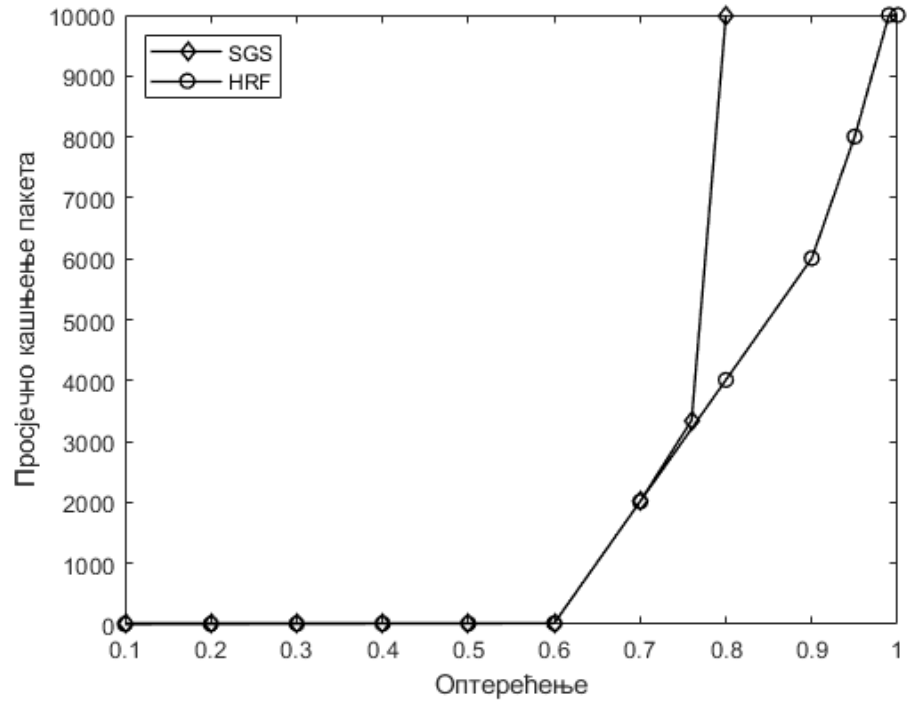


Слика 5.4. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај за комутаторе са баферима на улазним портовима

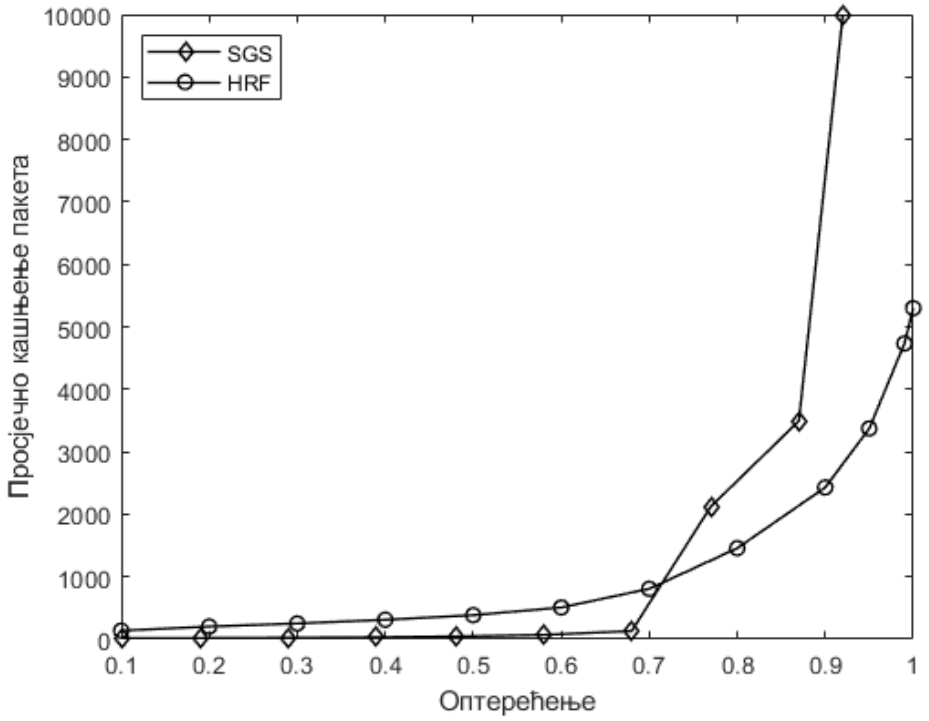
На слици 5.4 су дати резултати за Бернулијев униформни саобраћај. Са слике се види да сва три комутатора постижу сличне перформансе при малим и средњим оптерећењима. Међутим, већ при оптерећењу од 70% долази до наглог погоршања перформанси код *SGS* комутатора. С друге стране, *iSlip* и *HRF* комутатори се понашају врло стабилно и тек при врло великим оптерећењима долази до наглог повећања средњег кашњења.

На слици 5.5 су дати резултати за *hot-spot* саобраћај. Овдје нису приказани резултати за *iSlip* комутатор, јер за неуниформни саобраћај постаје брзо нестабилан [36]. И у случају *hot-spot* саобраћаја, *SGS* и *HRF* постижу добре перформансе за мања и средња оптерећења.

Међутим, кад оптерећење порасте преко 70%, оба комутатора постају врло нестабилни и долази до наглог пораста средњег кашњења.



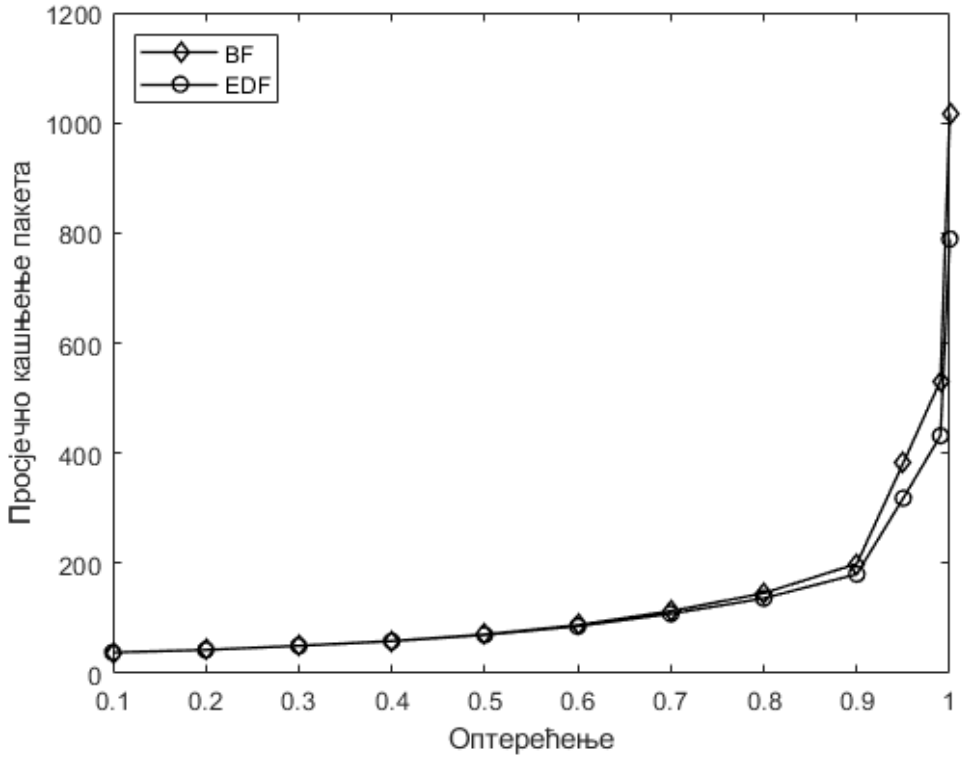
Слика 5.5. Зависност средњег кашњења пакета од оптерећења за *hot spot* саобраћај за комутаторе са баферима на улазним портовима



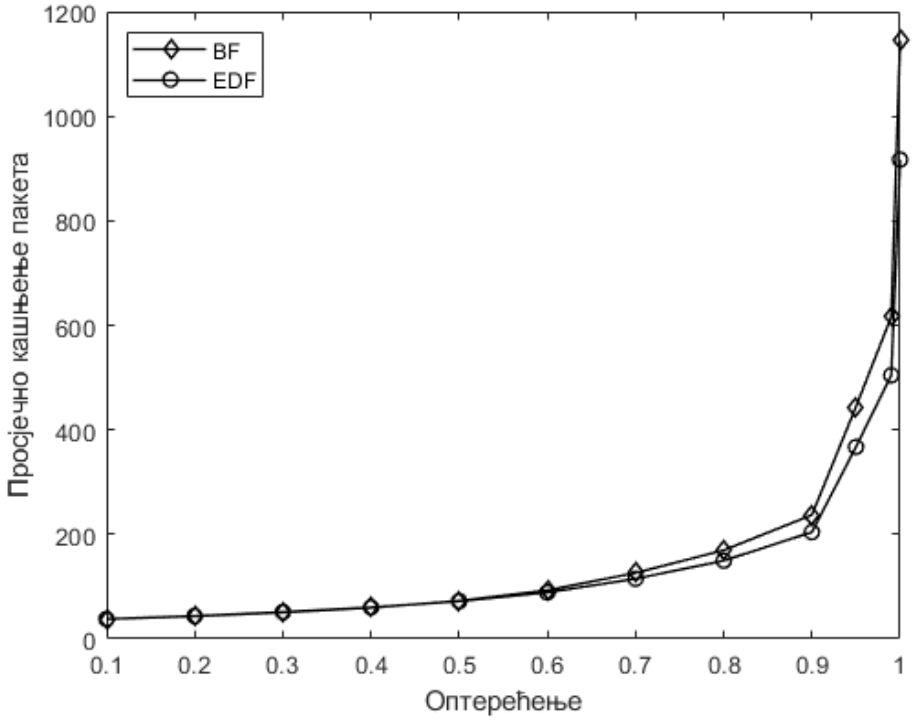
Слика 5.6. Зависност средњег кашњења пакета од оптерећења за униформни *bursty* саобраћај за комутаторе са баферима на улазним портовима

На слици 5.6 су дати резултати за униформни *bursty* саобраћај. Ни овдје нису приказани резултати за *iSlip* комутатор, јер за неуниформни саобраћај постаје брзо нестабилан [36]. *SGS* комутатор постиже боље перформансе при малим и средњим оптерећњима, док на високим оптерећењима средње кашњење нагло почиње да расте тако да *HRF* има боље перформансе.

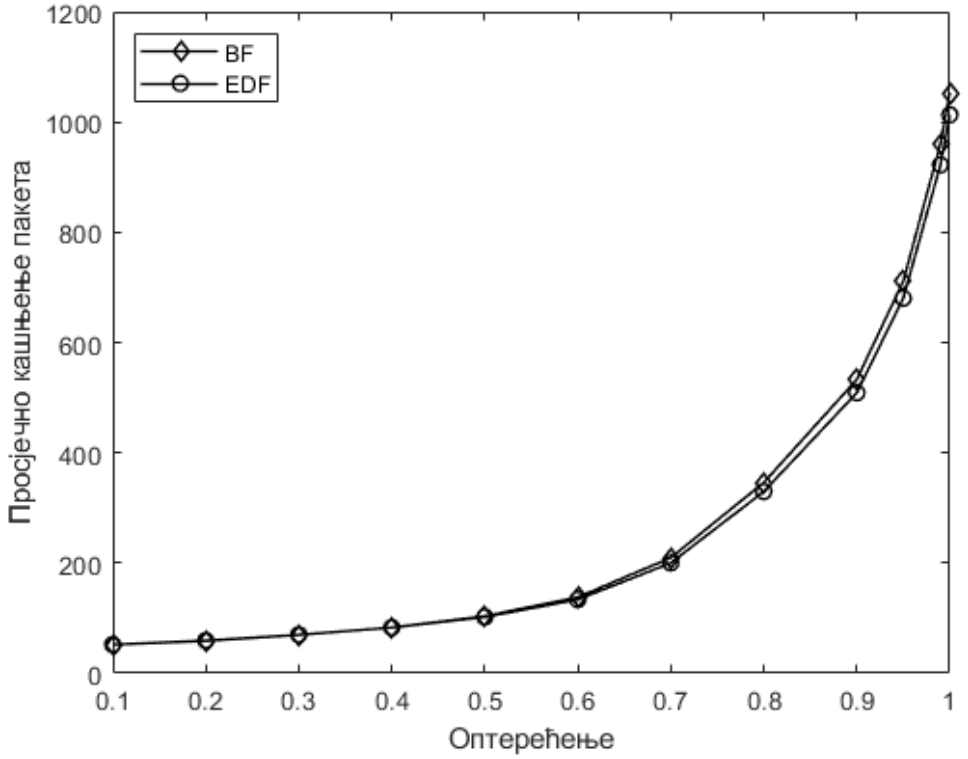
На сликама 5.7, 5.8 и 5.9 су дати резултати за комутаторе са ресеквенционим баферима на излазима за униформни, *hot-spot* и униформни *bursty* саобраћај, респективно. У анализи су дати резултати за *BF* и *EDF* комутаторе. *FCFS* комутатор није укључен у анализу јер *jitter* контрола уноси додатно кашњење од $N\cdot(N\text{-}1)$ слотова што негативно утиче на перформансе. Оба комутатора постижу сличне перформансе за различите типове саобраћаја. На основу резултата се закључује да *EDF* комутатор постиже нешто боље перформансе мада је разлика практично занемарљива. С друге стране, *BF* комутатор користи бафере величине N^2 за разлику од *EDF* комутатора који користи бафере величине $2\cdot N^2\text{-}2\cdot N$, па *BF* комутатор представља боље рјешење када се узме у обзир и имплементациона комплексност.



Слика 5.7. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај за *BvN* комутаторе са ресеквенционим баферима

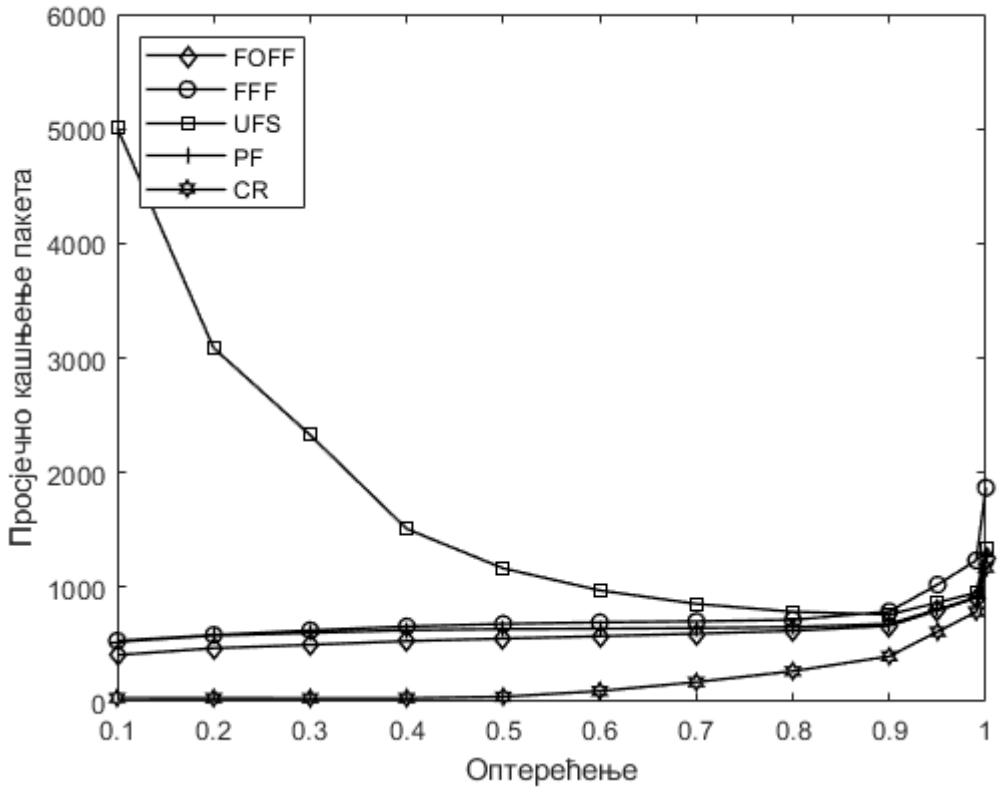


Слика 5.8. Зависност средњег кашњења пакета од оптерећења за *hot spot* саобраћај за *BvN* комутаторе са ресеквенционим баферима



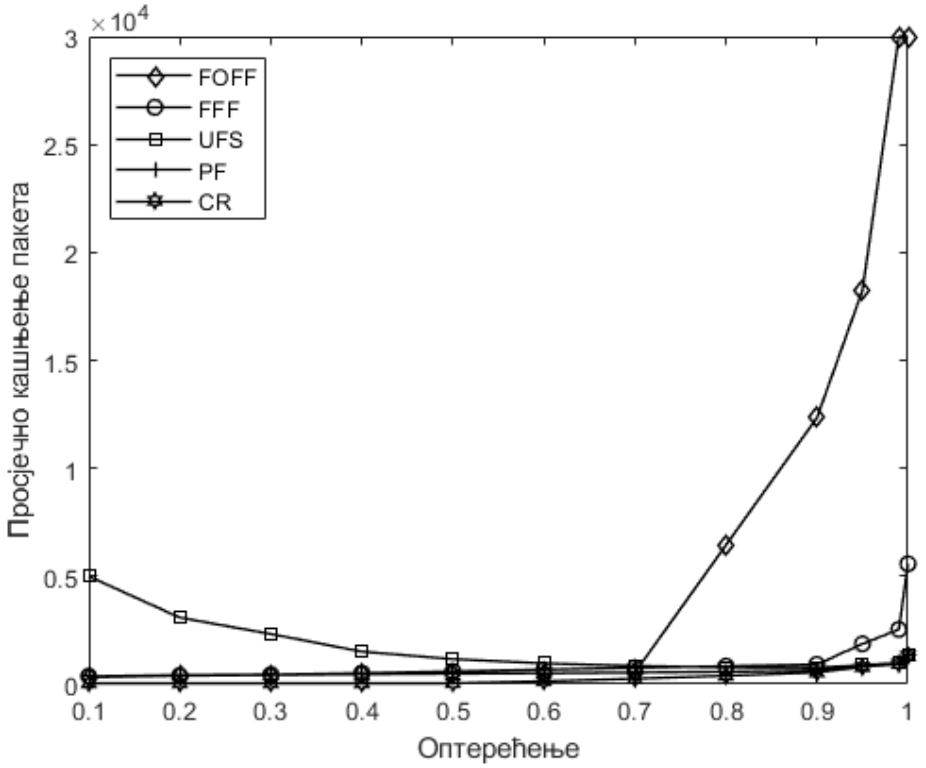
Слика 5.9. Зависност средњег кашњења пакета од оптерећења за униформни *bursty* саобраћај за *BvN* комутаторе са ресеквенционим баферима

На слици 5.10 су дати резултати за *frame-based* комутаторе за униформни саобраћај. У анализи су дати резултати за *FFF, FOFF, UFS, PF* и *CR* комутаторе, иако *CR* комутатор није класични *frame-based* комутатор већ више представља хибридно рјешење. Са слике се види да *UFS* комутатор постиже најлошије перформансе, што је посебно изражено при малим оптерећењима. Разлог за ово је што је при малим оптерећењима потребно да се сачека одређени број слотова док се не формира пуни фрејм. Како оптерећење расте, средње кашњење овог комутатора се приближава већини осталих рјешења која имају врло сличне перформансе за различита оптерећења. Са графика је јасно да *CR* комутатор генерално постиже најбоље резултате јер при малом оптерећењу не користи *frame-based* приступ чиме се избјегава дуго чекање на формирање фрејма што лоше утиче на перформансе осталих комутатора из ове категорије.

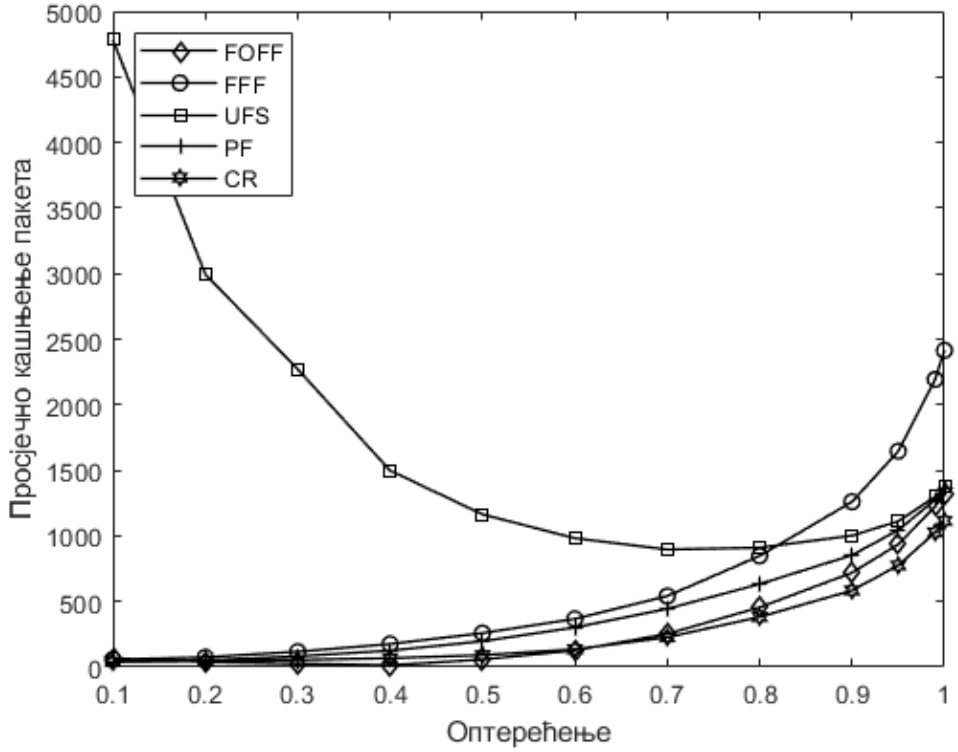


Слика 5.10. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај за *frame-based BvN* комутаторе

На слици 5.11 су дати резултати за *frame-based* комутаторе за *hot-spot* саобраћај. И у овом сценарију *UFS* постиже слабије перформансе у односу на остала рјешења при мањим оптерећењима. С друге стране, са графика се види да кашњење код *FOFF* комутатора при оптерећењима већим од 70% почиње нагло да расте. *FFF* комутатор при врло великим оптерећењима такође постиже нешто слабије перформансе у односу на *UFS* и *PF* комутаторе. И у овом сценарију, *CR* комутатор постиже боље перформансе у односу на остала рјешења. Генерално, овај комутатор представља најбоље рјешење међу *frame-based* комутаторима.



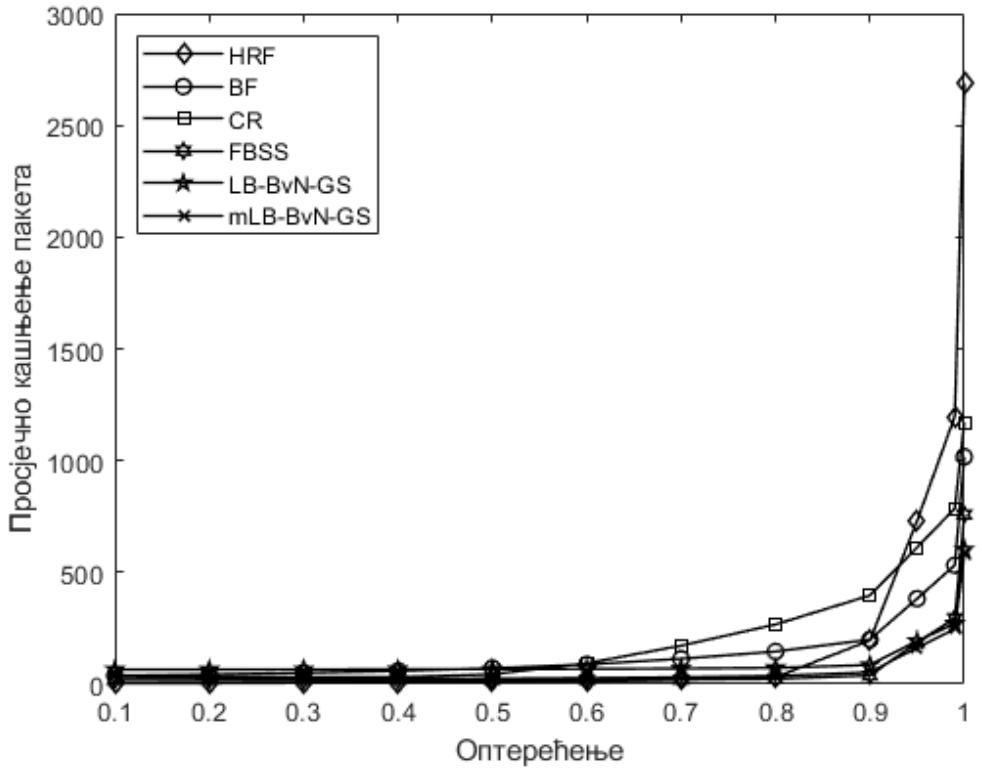
Слика 5.11. Зависност средњег кашњења пакета од оптерећења за *hot spot* саобраћај за *frame-based BvN* комутаторе



Слика 5.12. Зависност средњег кашњења пакета од оптерећења за униформни *bursty* саобраћај за *frame-based BvN* комутаторе

На слици 5.12 су дати резултати за *frame-based* комутаторе за униформни *bursty* саобраћај. И у овом сценарију *UFS* комутатор постиже најлошије перформансе при малим и средњим оптерећењима. Како оптерећење расте, средње кашњење овог комутатора се приближава већини осталих рјешења која имају врло сличне перформансе за различита оптерећења. С друге стране, *FFF* комутатор постиже најгоре перформансе при великим оптерећењима. И овдје се уочава да *CR* комутатор генерално постиже најбоље резултате јер при малом оптерећењу не користи *frame-based* приступ чиме се избјегава дуго чекање на формирање фрејма што лоше утиче на перформансе осталих комутатора из ове категорије.

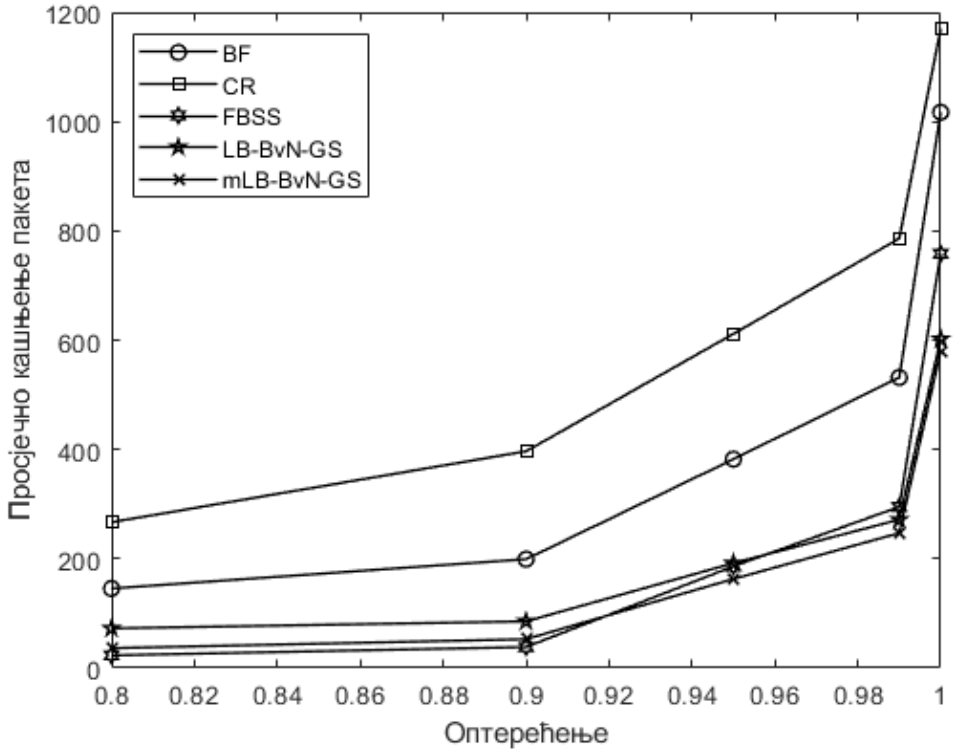
У наставку су рјешења предложена у овој дисертацији (*LB-BvN-GS* и *mLB-BvN-GS*) упоређена са најбољим постојећим рјешењима. На основу претходно изложених анализа изабрани су најбољи представници сваке категорије (комутатори са баферима на улазним портовима, *LB-BvN* комутатори са ресеквенционим баферима, *LB-BvN* комутатори који користе слање пакета у фрејмовима, *LB-BvN* комутатори са повратном спрегом гдје постоји само један представник). *HRF* комутатор је изабран као најбоље рјешење међу комутаторима са баферима на улазним портовима. *BF* комутатор је изабран као најбоље рјешење међу *LB-BvN* комутаторима који користе ресеквенционе бафере. *CR* комутатор је одабран као најбоље рјешење на бази *LB-BvN* комутатора које користи слање пакета у фрејмовима. Коначно, *FBSS* комутатор је одабран као рјешење које користи специфичне, предефинисане конфигурације ради спречавања поремећаја редоследа пакета.



Слика 5.13. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај

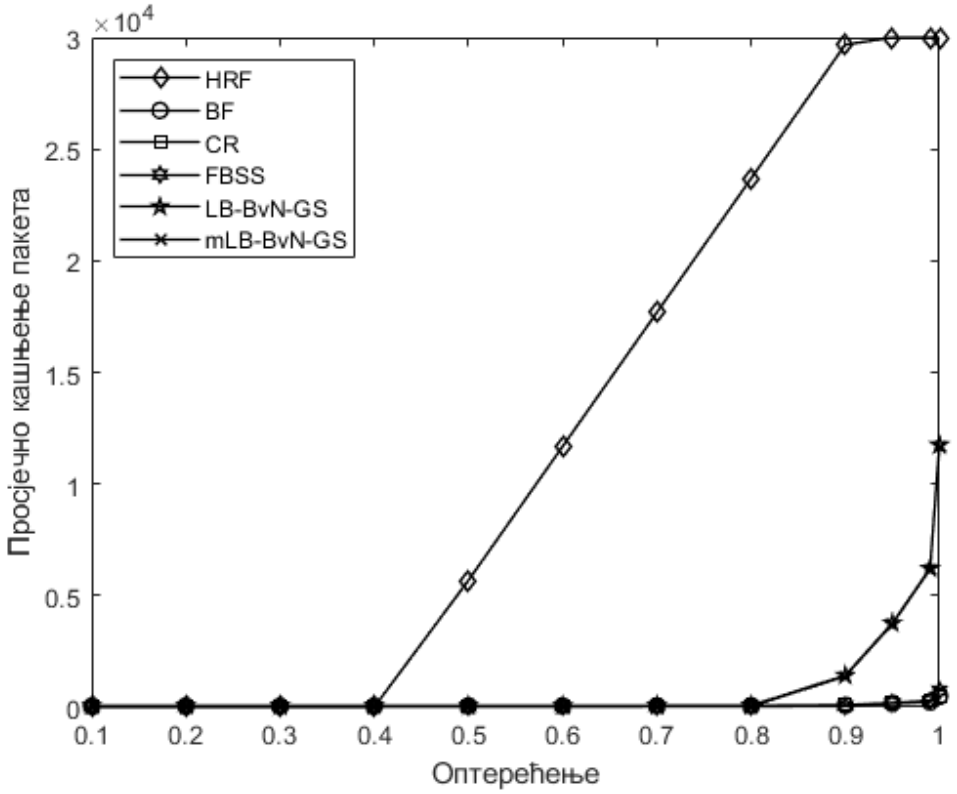
На слици 5.13 су дати резултати за униформни саобраћај. При малим и средњим оптерећењима *CR* комутатор има веће кашњење него други комутатори. *BF* комутатор такође има веће кашњење у односу на *FBSS*, *LB-BvN-GS* и *mLB-BvN-GS* при средњим и већим оптерећењима. При великим оптерећењима *HRF* комутатор има највеће кашњење. *mLB-BvN-*

GS постиже сличне перформансе као *FBSS*. Средње кашњење пакета у *LB-BvN-GS* комутатору је нешто веће због додатног циклуса у коме се врши распоређивање пакета на улазним портовима, али разлика у односу на *FBSS* и *mLB-BvN-GS* није много значајна. На слици 5.14 је дат увећани приказ графика 5.13 за велика оптерећења. Овдје је *HRF* комутатор искључен из поређења због значајно лошијих перформанси за тај опсег оптерећења. Са слике се јасно види да *CR* и *BF* комутатори имају већа кашњења при великим оптерећењима, док *FBSS*, *LB-BvN-GS* и *mLB-BvN-GS* постижу врло сличне перформансе.

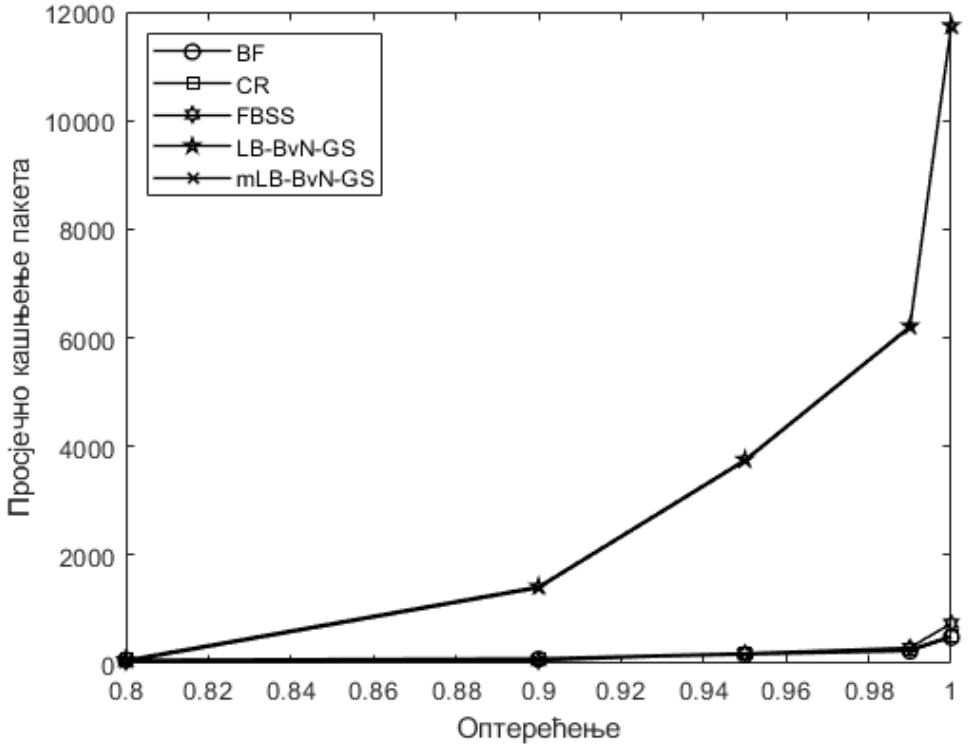


Слика 5.14. Зависност средњег кашњења пакета при великим оптерећењима за Бернулијев униформни саобраћај

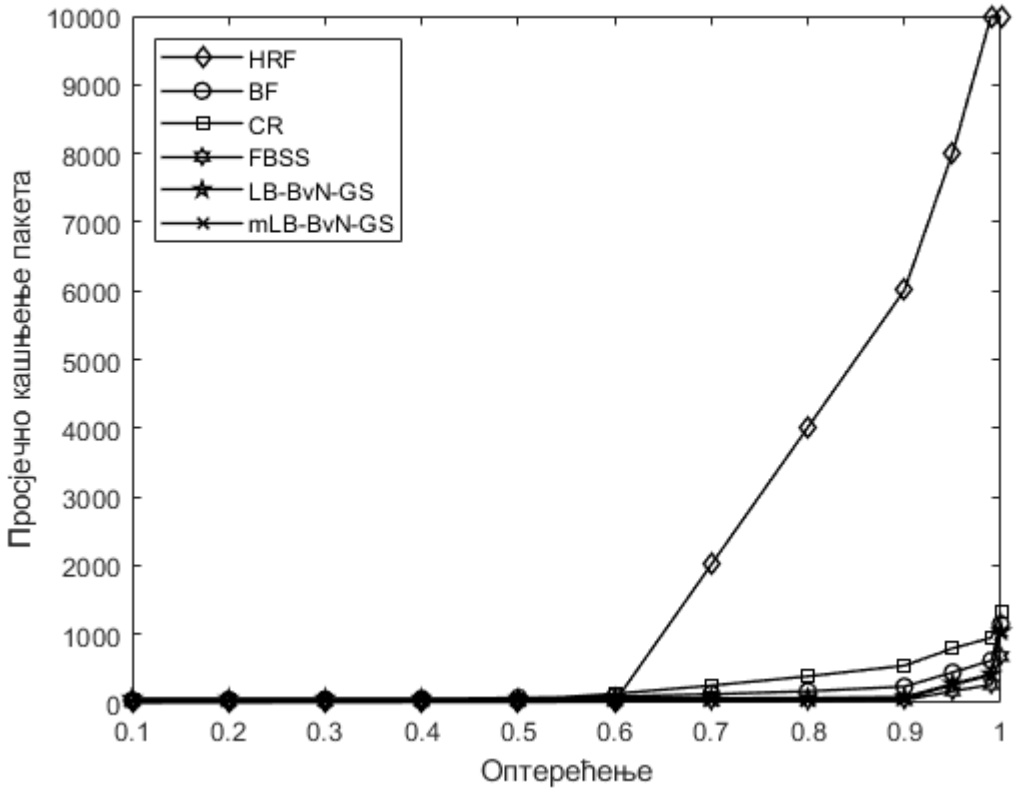
На слици 5.15 су дати резултати за дијагонални саобраћај. Јасно се уочава да *HRF* комутатор има значајно горе перформансе у односу на остала рјешења. На слици 5.16 је дат увећани приказ графика 5.15 за велика оптерећења, при чему је *HRF* искључен из поређења због знатно горих перформанси. Овдје видимо да *LB-BvN-GS* комутатор има нешто веће просјечно кашњење при високим оптерећењима, док *FBSS* и *mLB-BvN-GS* постижу сличне перформансе.



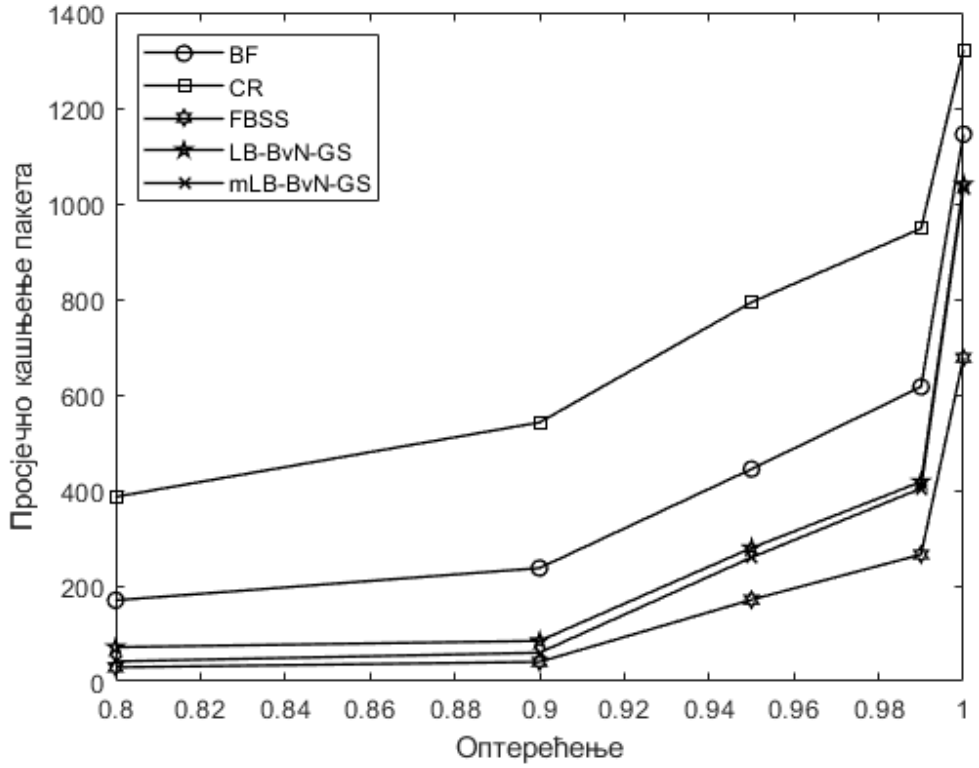
Слика 5.15. Зависност средњег кашњења пакета од оптерећења за дијагонални саобраћај



Слика 5.16. Зависност средњег кашњења пакета при великим оптерећењима за дијагонални саобраћај



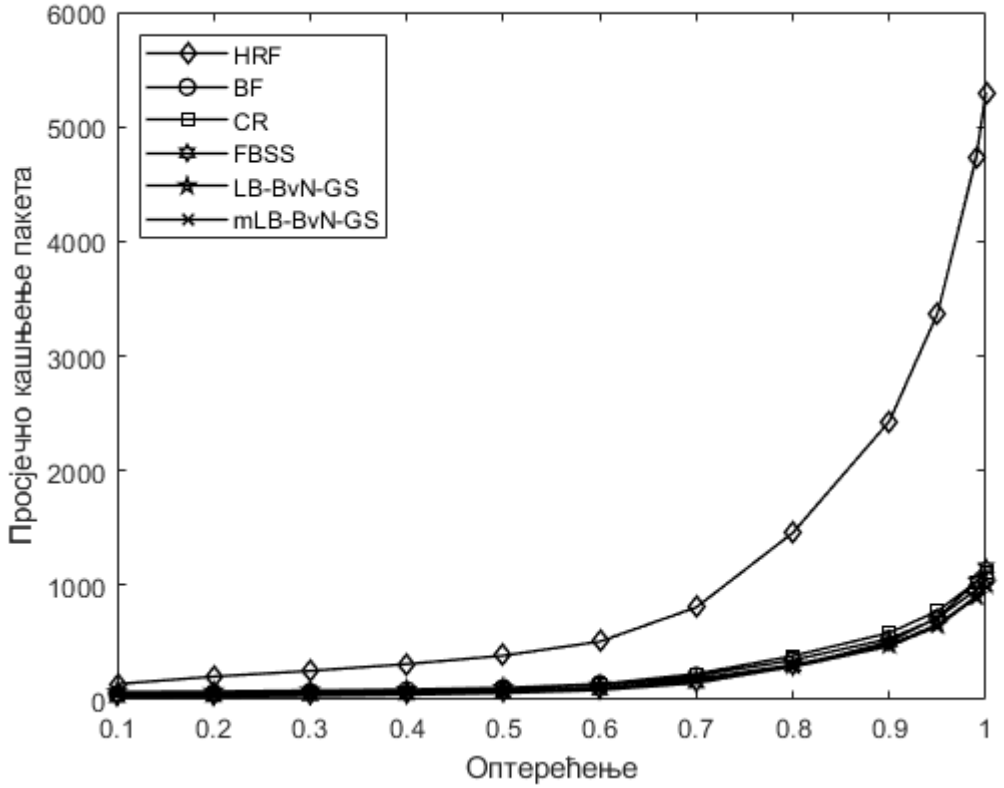
Слика 5.17. Зависност средњег кашњења пакета од оптерећења за *hot-spot* саобраћај рјешења



Слика 5.18. Зависност средњег кашњења пакета при великим оптерећењима за *hot-spot* саобраћај

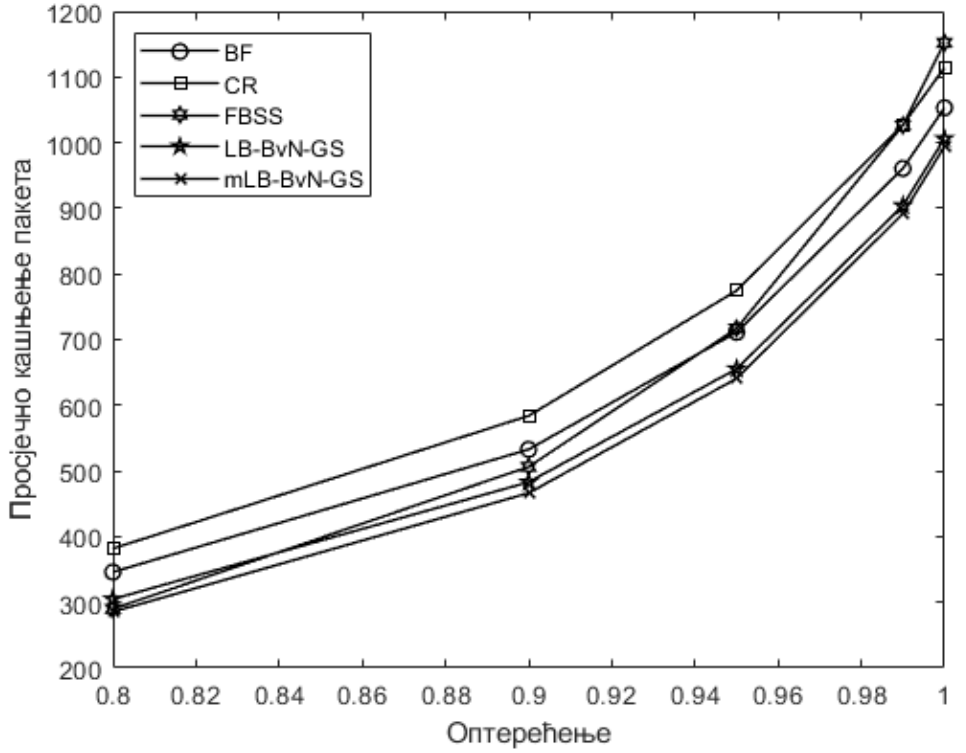
На слици 5.17 су дати резултати за *hot-spot* саобраћај. Закључци су готово исти као и за униформни саобраћај. *HRF* комутатор и у овом сценарију има доста лошије перформансе. На слици 5.18 је дат увећани приказ графика 5.17 за велика оптерећења, при чему је *HRF* комутатор искључен из поређења због знатно горих перформанси. Овдје се уочава да *LB-BvN-GS* и *mLB-BvN-GS* комутатори имају боље перформансе у односу на *CR* и *BF* комутаторе, али и нешто слабије перформансе у односу на *FBSS* комутатор.

На слици 5.19 су дати резултати за униформни *bursty* саобраћај. И у овом сценарију се види да *HRF* комутатор има знатно веће кашњење нарочито при великим оптерећењима. На слици 5.20 је дат увећани приказ графика 5.19 за велика оптерећења, при чему је *HRF* комутатор искључен из поређења због знатно горих перформанси. Са графика се види да *LB-BvN-GS* и *mLB-BvN-GS* комутатори постижу боље перформансе у односу на друга рјешења.

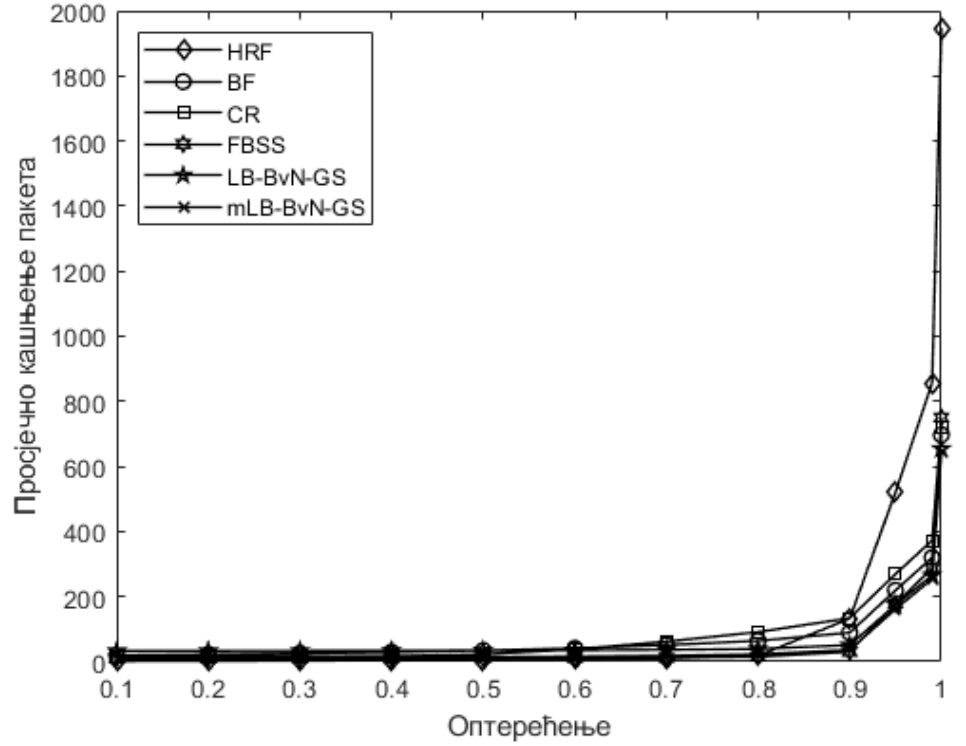


Слика 5.19. Зависност средњег кашњења пакета од оптерећења за униформни *bursty* саобраћај

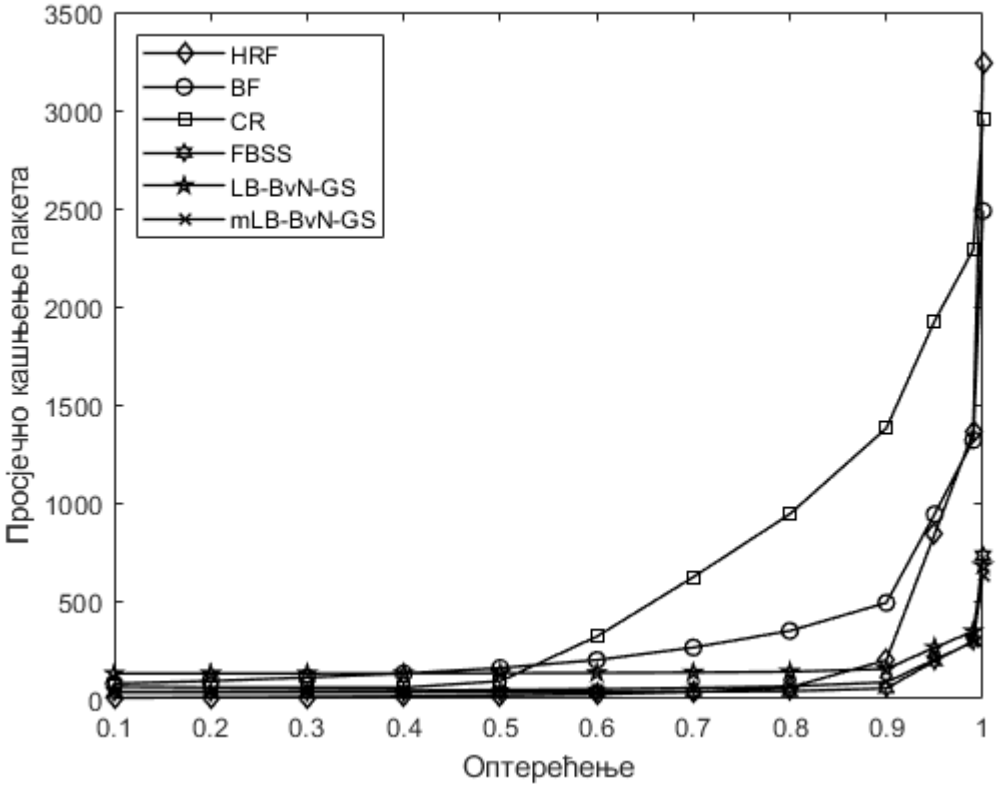
На сликама 5.21 и 5.22 су приказани резултати за Бернулијев униформни саобраћај за комутаторе са 16 односно 64 порта, респективно. У принципу, закључци остају исти као и у случају комутатора са 32 порта са становишта рјешења предложених у овој дисертацији (*LB-BvN-GS* и *mLB-BvN-GS*).



Слика 5.20. Зависност средњег кашњења пакета при великим оптерећењима за униформни *bursty* саобраћај

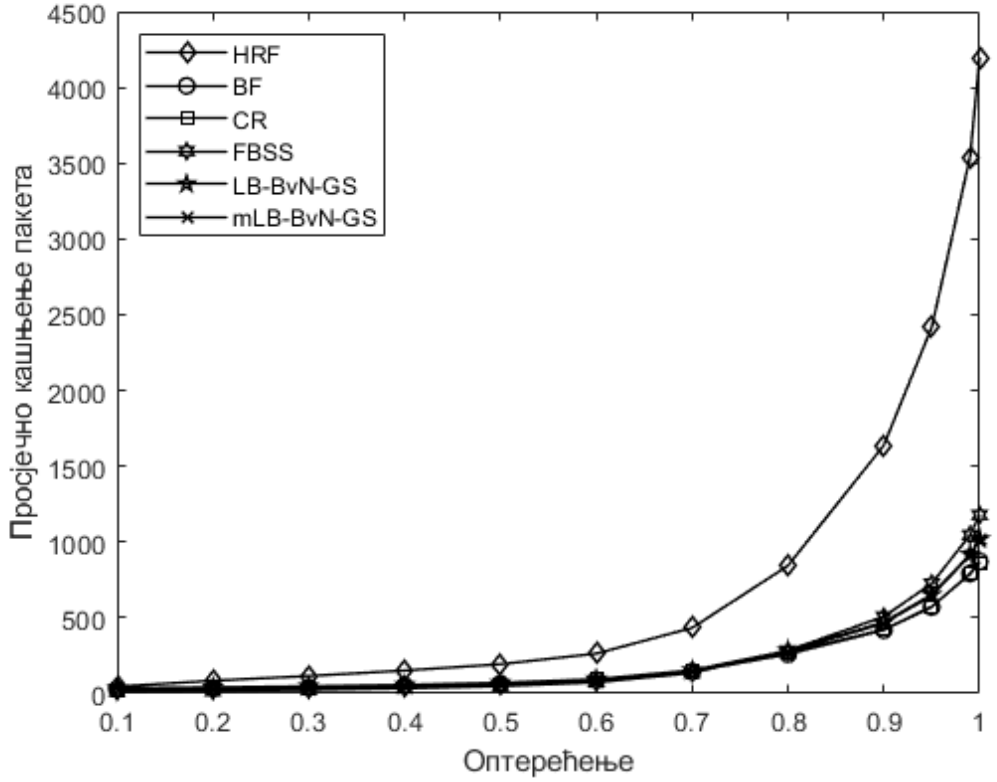


Слика 5.21. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај за 16x16 комутаторе

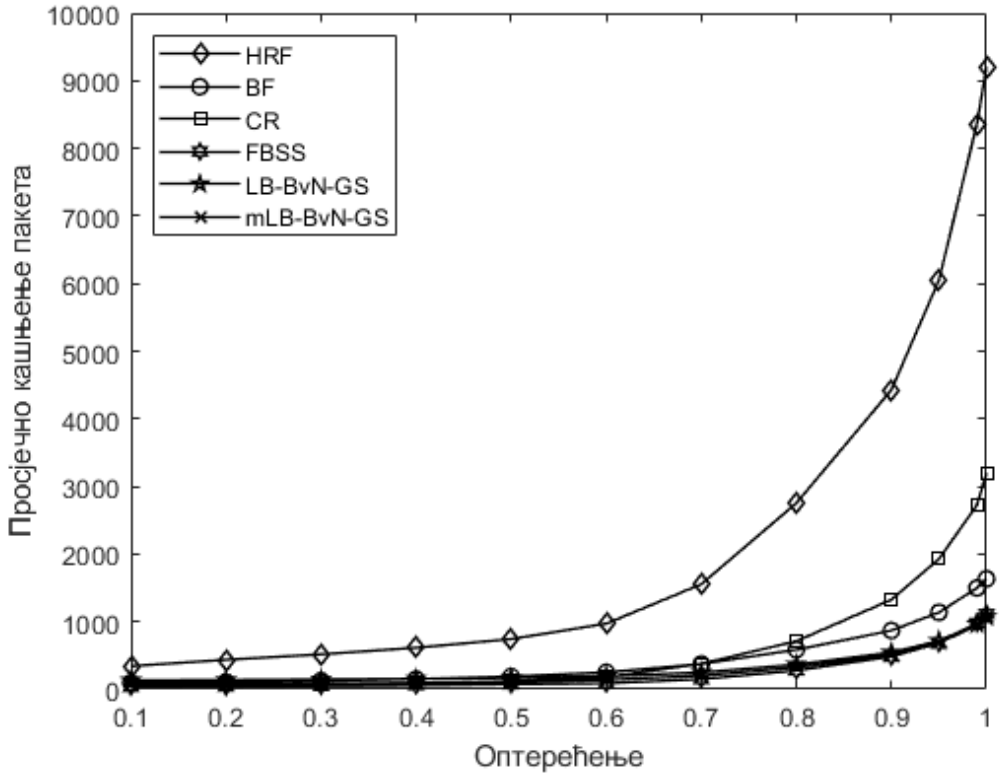


Слика 5.22. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај за 64x64 комутаторе

На сликама 5.23 и 5.24 су приказани резултати за униформни *bursty* саобраћај за комутаторе са 16 односно 64 порта, респективно. И у овом саобраћајном сценарију закључци остају исти као и у случају комутатора са 32 порта са становишта рјешења предложених у овој дисертацији (*LB-BvN-GS* и *mLB-BvN-GS*). Такође, може се уочити да *BF* и *CR* рјешења нешто лошије скалирају у односу на друга рјешења са повећањем димензије комутатора ако се пореде резултати на сликама 5.21 и 5.22, односно 5.23 и 5.24. Поредећи резултате за димензије комутатора 16x16 и 64x64 види се да *BF* и *CR* постају лошија у односу на друга рјешења са повећањем димензије комутатора. Додатно, резултати приказани на сликама 5.21 и 5.22, као и на сликама 5.23 и 5.24, потврђују да су предложена рјешења *LB-BvN-GS* и *mLB-BvN-GS* скалабилна.



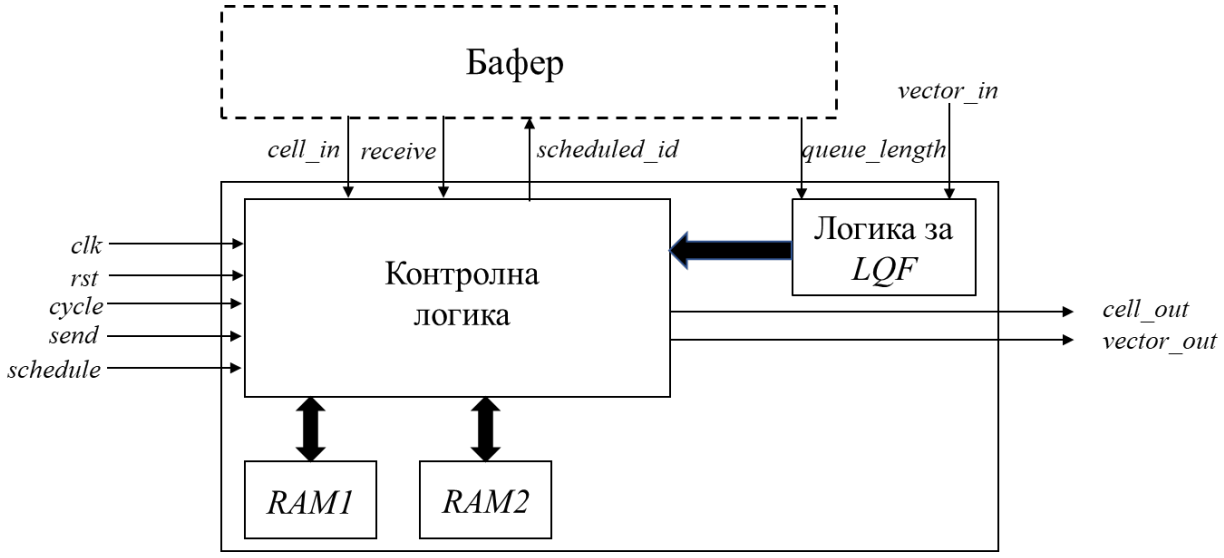
Слика 5.23. Зависност средњег кашњења пакета од оптерећења за *bursty* саобраћај за 16x16 комутаторе



Слика 5.24. Зависност средњег кашњења пакета од оптерећења за *bursty* саобраћај за 64x64 комутаторе

6. ПРОЦЈЕНА ХАРДВЕРСКЕ КОМПЛЕКСНОСТИ LB-BvN-GS

У оквиру овог поглавља ће бити дата анализа хардверске комплексности изабраног рјешења *LB-BvN-GS* и његове модификације *mLB-BvN-GS*. Циљ анализе је да се потврди претпоставка из четвртог поглавља о једноставности предложеног рјешења комутатора. Такође, циљ је и да се утврди који дијелови предложеног *LB-BvN-GS* комутатора су критични са становишта перформанси (брзине рада или количине заузетих ресурса). У ту сврху је развијена хардверска имплементација дијелова *LB-BvN-GS* комутатора који се налазе на улазном и централном порту, односно у случају *mLB-BvN-GS* и на излазном порту пошто се у овој варијанти комутатора мора користити ресеквенциони бафер. Важно је напоменути да хардверска имплементација приказана у овом поглављу има за циљ прије свега процјену потребних ресурса, односно комплексности и да није рађена потпуна оптимизација имплементације пошто она није главни предмет ове тезе. Међутим, сами интерфејси су осмишљени тако да се лако могу прилагодити произвољној системској архитектури рутера. Сама имплементација је урађена у *VHDL* (*VHSIC Hardware Description Language*) језику, и коришћено је *ISE* развојно окружење компаније *Xilinx*.



Слика 6.1. Системска архитектура улазног порта *LB-BvN-GS* комутатора

LB-BvN-GS архитектура за један порт рутера се састоји из два дијела - распоређивач на улазном порту и централни бафер на централном порту. У случају *mLB-BvN-GS*, постоји још и ресеквенциони бафер на излазном порту. Као што се може видјети и из описа *LB-BvN-GS* рјешења датог у четвртом поглављу, главно и најсложеније процесирање се врши на улазном порту. Стога је очекивано да је најкомплекснији дио распоређивач на улазном порту. На слици 6.1 је приказана системска архитектура дијела *LB-BvN-GS* комутатора који се имплементира на улазном порту и који има улогу *LB-BvN-GS* распоређивача. Улази овог модула су сљедећи:

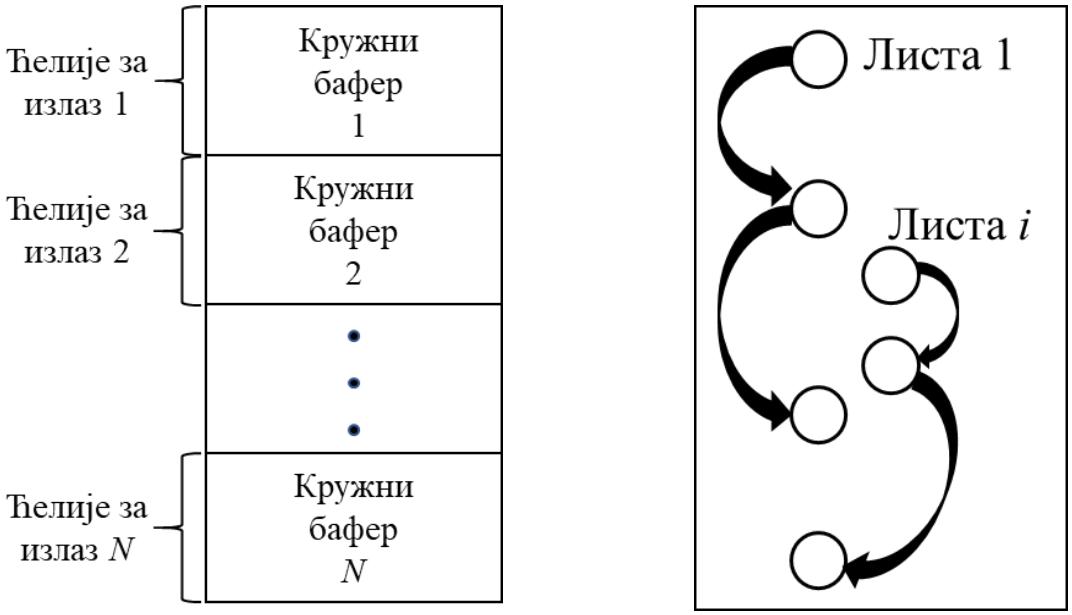
- *clk* - сигнал такта

- *rst* - ресет сигнал
- *cycle* - контролни сигнал који сигнализира почетак новог циклуса (активна вредност 1)
- *schedule* - контролни сигнал који сигнализира да треба распоредити ћелију (активна вредност 1)
- *vector_in* - улазни вектор заузетости излазних портова
- *queue_length* - дужине редова за чекање (низ од *N* вектора, гдје сваки члан низа представља дужину одговарајућег реда за чекање)
- *receive* - контролни сигнал који сигнализира да треба примити распоређену ћелију и смјестити је у меморију (активна вредност 1)
- *cell_in* - распоређена ћелија која се прима
- *send* - контролни сигнал који сигнализира да треба послати ћелију распоређену у претходном циклусу (активна вредност 1)

Излази овог модула су сљедећи:

- *scheduled_id* - изабрани ред за чекање (састоји се из бита који сигнализира да ли је нека ћелија распоређена, а остали бити указују из ког реда за чекање је распоређена ћелија)
- *vector_out* - ажурирани вектор заузетости излазних портова
- *cell_out* - ћелија која се шаље

Битно је напоменути да су улази и излази осмишљени тако да буду лако прилагодљиви произвољној архитектури остатка улазног порта рутера. Као подсјетник, неке од функционалности које улазни порт рутера обавља, а наведене у поглављу 2, су испитивање исправности заглавља, сегментација пакета у ћелије, баферовање пакета, мултигигабитски интерфејси и др. Многе од ових функционалности имају различита рјешења тј. приступе попут баферовања пакета, при чему и одабир технологије утиче на коначну системску архитектуру рјешења.

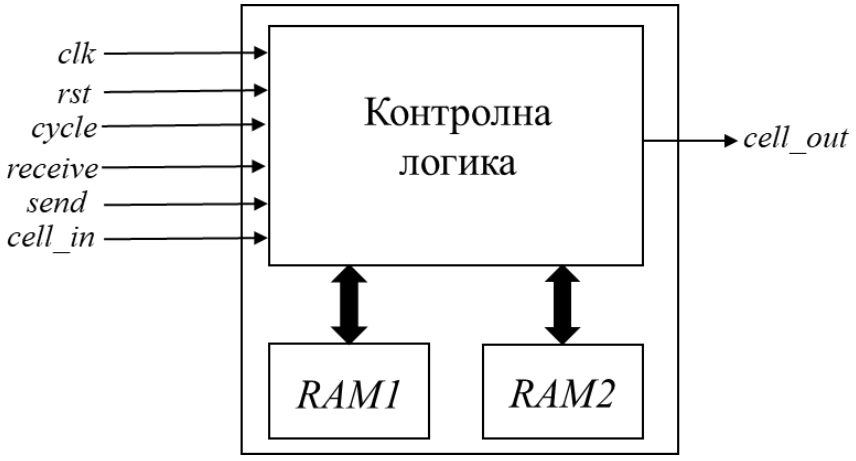


Слика 6.2. а) Меморија која користи принцип кружног бафера б) Меморија која користи принцип листи

На примјер, системи на чипу или *FPGA* чипови могу имати огромне ресурсе у зависности од изабране фамилије и чипа, али и даље постоје ограничења у меморијским ресурсима за смјештање пакета због чега се типично користе и екстерне меморије попут *RLDRAM* које су развијене управо у такве сврхе. Неки од *FPGA* чипова најновије генерације посједују и *HBM* (*High Bandwidth Memory*) меморију која нуди велики капацитет и проток, при чему је меморија дио читавог чипа [138]. У наставку текста ћемо и *HBM* меморију сматрати екстерном због сличног принципа у раду са њом. Додатно, за екстерне меморије великог капацитета и брзине је карактеристично да је пожељно уписивати *burst*-ове података на сукцесивне локације ради постизања максималне брзине рада меморије тј. протока података. У складу с тим постоје разни приступи у реализацији бафера. Два таква приступа су илустрована на сликама 6.2 а) и б). Оба приступа користе принцип да се почетни (*head*) и завршни (*tail*) дио бафера налазе на чипу, а остатак бафера се смјешта у екстерну меморију. Када се завршни дио бафера довољно напуни, онда се формира *burst* података који се уписује у екстерну меморију чиме се постиже велики проток тј. брзина у раду са меморијом. При томе се ослобађа завршни дио бафера за пријем нових ћелија. Међутим, сама организација почетног и завршног дијела бафера може да се разликује. Једна варијанта (слика 6.2 а)) се заснива на томе да се сваком току додијели одређени број сукцесивних локација у меморији на чипу чиме се формира кружни бафер за тај ток. Контролна логика у овом случају је веома једноставна јер је потребан само један показивач који кружно броји тј. показује. Мана оваквог приступа је што су меморијски ресурси чипа лошије искоришћени јер је мала вјероватноћа да већина меморије буде заузета. Друга варијанта (слика 6.2 б)) је да се користе листе за формирање почетног и завршног дијела бафера, по једна листа за сваки ток. У овој варијанти је могуће да се користе мањи укупни меморијски ресурси чипа, али је контролна логика комплекснија јер је потребно користити показиваче на почетак и крај листе, а сви чланови листе морају да садрже показивач на сљедећег члана. Сличне варијанте приступа могу да се користе и за смештање и организацију централних дијелова бафера у екстерној меморији. Отуда сами бафери нису дио архитектуре приказане на слици 6.1, већ су креирани интерфејси преко којих се размјењују ћелије и контролни сигнали попут задавања дохватања (читања) ћелија са почетка бафера.

Као што је приказано на слици 6.1 основни дијелови распоређивача на улазном порту су: логика за избор најдужег реда за чекање (логика за *LQF*), меморије за смјештање ћелија (*RAM*1 и *RAM*2), контролна логика. Логика за избор најдужег реда за чекање на основу вектора заузетости излазних портова и вредности дужина редова за чекање бира најдужи ред за чекање а који има право да се изабере јер је одговарајући излазни порт означен као слободан у вектору заузетости. Као резултат рада ове логике добија се бит индикатор који сигнализира да ли је неки ред изабран, као и редни број реда за чекање (тј. тока) који је изабран. Сама логика за избор најдужег реда за чекање је организована у виду нивоа где се налази одговарајући број блокова који раде компарацију два суседна члана (дужине реда за чекање). Побједници компарације иду у сљедећи ниво који ради по истом принципу и слична структура се понавља док се не изабере побједник тј. најдужи ред за чекање, а који има право да се изабере. На саму дужину се додаје на почетак инвертовани бит заузетости дотичног реда за чекање чиме се стимулише да предност у избору имају редови за чекање који имају право избора. Уколико на крају изабрани ред за чекање на врху има вредност бита 0, то значи да није изабран ниједан ред за чекање - формално је изабран ред али тај ред нема право да шаље ћелију пошто је дотични излаз већ заузет у процесу распоређивања од неког ранијег улазног порта у току текућег циклуса.

Као што је већ објашњено у четвртом поглављу, постоје двије меморије које мијењају улоге током циклуса. Једна служи за слање ћелија које су распоређене у претходном циклусу, а друга служи за смјештање распоређених ћелија у текућем циклусу. Контролна логика управља слањем и распоређивањем ћелија тј. уписом и читањем ћелија из наведених меморија. У току распоређивања ћелија, на основу резултата рада логике за избор најдужег реда за чекање ажурира вектор заузетости, а такође шаље захтјев баферу да се пошаље распоређена ћелија коју ће по пријему контролна логика да смести у одговарајућу меморијску локацију. Приликом смјештања распоређене ћелије у меморију такође се смјешта информација о редном броју тока којем распоређена ћелија припада, и ажурирају се вриједности показивача тј. листи по принципу објашњеном у поглављу 4. Меморија из које се шаљу ћелије се чита на основу вриједности показивача на почетак одговарајуће листе, а избор показивача тј. листе се врши на основу редног броја тока коме припада ћелија смјештена у локацију меморије која одговара тренутном централном порту с којим се повезује дотични улазни порт.



Слика 6.3. Системска архитектура централног порта

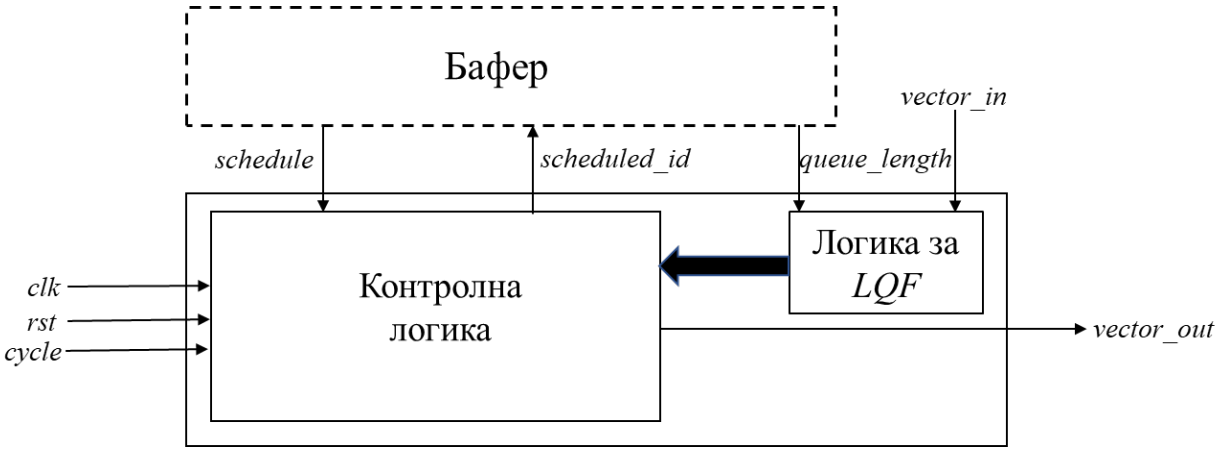
Као што је приказано на слици 6.3 сама структура централног бафера је веома једноставна и састоји се из двије меморије које смјештају пристигле ћелије, односно из којих се ишчитавају ћелије ка излазним портовима преко комутационог модула. Једина логичка функција која се извршава је замјена вриједности у интерном заглављу ћелије. Ћелија која пристиже са улазног порта носи информацију ком излазу је намијењена дотична ћелија. Ћелија која се шаље излазном порту треба да носи информацију са ког улазног порта је потекла ћелија. Стога се у централном порту пре уписа у бафер врши замјена вриједности у заглављу и ставља се вриједност улазног порта са ког је ћелија пристигла. Системска архитектура централног бафера је дата на слици 6.4. Улази овог модула су сљедећи:

- *clk* - сигнал такта
- *rst* - ресет сигнал
- *cycle* - контролни сигнал који сигнализира почетак новог циклуса (активна вредност 1)
- *receive* - контролни сигнал који сигнализира да треба примити ћелију пристиглу са улазног порта и смјестити је у меморију (активна вредност 1)
- *cell_in* - ћелија која се прима
- *send* - контролни сигнал који сигнализира да треба послати ћелију ка излазном порту (активна вредност 1)

Излази овог модула су сљедећи:

- *cell_out* - ћелија која се шаље

Принцип рада је у складу са објашњењем датим у поглављу 4. Постоје двије меморије (*RAM*1 и *RAM*2) чије се улоге обрћу на нивоу циклуса. Једна меморија служи за смјештање ћелија које пристижу са улазних портова, а друга меморија служи за слање ћелија примљених током претходног циклуса. Контролна логика је задужена за одређивање у коју локацију треба да се упише пристигла ћелија, што се одређује на основу редног броја излаза коме је ћелија намијењена а који се налази у заглављу ћелије. Приликом уписа ћелије, у заглављу се редни број излаза коме је ћелија намијењена, замјењује вриједношћу редног броја улаза са ког је ћелија пристигла а што је познато на основу детерминистичке конфигурације комутационог модула па се зна редослед улазних портова са којима се централни порт повезује током циклуса. Слање ћелије се врши ишчитавањем ћелије са локације која одговара излазу са којим се тренутно повезује централни порт.



Слика 6.4. Системска архитектура улазног порта mLB-BvN-GS комутатора

У случају модификоване верзије тј. *mLB-BvN-GS* централни бафер има исту реализацију. Разлика је у реализацији улазног порта која је знатно једноставнија него код основне верзије тј. *LB-BvN-GS*, као и у постојању ресеквенционог бафера на излазном порту. Структура улазног порта код *mLB-BvN-GS* комутатора је дата на слици 6.4.

У случају модула за распоређивање ћелија на улазном порту, улазни портови су:

- *clk* - сигнал такта
- *rst* - ресет сигнал
- *cycle* - контролни сигнал који сигнализира почетак новог циклуса (активна вредност 1)
- *schedule* - контролни сигнал који сигнализира да треба распоредити ћелију (активна вредност 1)
- *vector_in* - улазни вектор заузетости излазних портова
- *queue_length* - дужине редова за чекање (низ од *N* вектора, где сваки члан низа представља дужину одговарајућег реда за чекање)

Излазни портови модула за распоређивање ћелија су:

- *scheduled_id* - изабрани ред за чекање (састоји се из бита који сигнализира да ли је нека ћелија распоређена, а остали бити указују из ког реда за чекање је распоређена ћелија)

- *vector_out* - ажурирани вектор заузетости излазних портова

На улазном порту у модулу распоређивача сада практично постоји само логика за избор најдужег реда за чекање који има право да буде изабран, и контролна логика која је много једноставнија него код *LB-BvN-GS* варијанте. Контролна логика сада само шаље информацију који ред за чекање је изабран тј. за који је ћелија распоређена. Пошто се распоређена ћелија одмах шаље, сада нема потребе да се она смјешта у модул распоређивача већ се може директно из бафера проследити даље (ка комутационом модулу). Отуда нема потребе за реализацијом меморија у модулу распоређивача на улазном порту.

Што се тиче имплементације ресеквенционог бафера на излазном порту, потребно је узети у обзир чињеницу да је због реконструкције пакета из ћелија неопходно постојање малих бафера на излазу који складиште ћелије док не пристигну све ћелије пакета да би он могао бити реконструисан. Отуда је најлогичније да се ти бафери искористе и за потребе ресеквенционог бафера. Сами бафери за чување ћелија пре њиховог спајања у оригинални пакет могу бити организовани по једном од два принципа приказаним на сликама 6.2 а) и б). Долазак ћелија једног тока на излазни порт у оквиру једног циклуса се може посматрати у два дела, прије и после пивот конекције. Ако ћелије долазе са улазног порта *i*, тада је пивот конекција слот у ком се спаја централни бафер *i* са посматраним излазним портом. Пристигле ћелије се могу посматрати као двије листе, листа до пивот конекције која садржи први дио ћелија (раније ћелије тока), и послије пивот конекције која садржи други дио ћелија (касније ћелије тока). У обије листе ћелије долазе у обрнутом редоследу (касније ћелије долазе прве). Имајући то у виду могуће је примљене ћелије уписивати у бафере за реконструкцију пакета у уређеном редоследу. У случају варијанте кружног бафера, потребно је прослиједити редни број ћелије да би се знало на коју локацију уписати пристиглу ћелију јер је немогуће знати унапријед колико ћелија неког тока има у циклусу. У случају листи потребно је само прослиједити да ли ћелија припада дијелу прије или послије пивот конекције ради адекватног уписа у листу који се ради по сличном принципу као на улазном порту. Ако ћелија припада дијелу до пивот конекције тада се ћелија умеће на крај листе ћелија које су примљене у претходним циклусима. Ако ћелија припада дијелу послије пивот конекције тада се ћелија умеће на крај листе ћелија која је формирана до пивот конекције (принцип уметања као на улазном порту). У складу са овим претпоставкама, креиран је модул на излазном порту који прослиједјује одговарајуће информације баферима за реконструкцију пакета, а који истовремено служе и као ресеквенциони бафери. Улази овог модула су:

- *clk* - сигнал такта
- *rst* - ресет сигнал
- *cycle* - контролни сигнал који сигнализира почетак новог циклуса (активна вредност 1)
- *receive* - контролни сигнал који сигнализира да треба примити ћелију пристиглу са улазног порта (активна вредност 1)
- *cell_in* - ћелија која се прима

Излази овог модула су сљедећи:

- *cell_out* - ћелија која се шаље
- *send* - контролни сигнал који сигнализира да се шаље ћелија ка баферу за реконструкцију пакета (активна вредност 1)

- *pivot_flag* - сигнализација да ли је ток ком припада послата ћелија прошао пивот конекцију (вредност 1) или не (вредност 0)

Реализовани модул подржава истовремено обије варијанте реализације бафера (кружни бафери или листе). У случају кружних бафера, сама информација о редном броју ћелије се налази у ћелији тј. њеном заглављу. Информација о проласку пивот конекције за неки ток се ажурира током циклуса постављањем активне вредности 1 на одговарајућу позицију вектора дужине *N* бита. Приликом слања ћелије се на излаз *pivot_flag* поставља одговарајући бит тог вектора који одговара току ком припада ћелија која се шаље. На тај начин варијанта са листама зна како треба уметнути ћелију у одговарајућу листу. У обе варијанте информација ком току припада ћелија (са ког улазног порта је послата) се налази у заглављу ћелије.

У табелама 6.1-6.4 су дати процијењени хардверски ресурси добијени анализом и синтезом у *ISE* развојном окружењу претходно описаних модула. *N* се односи на број портова комутатора, *CL* се односи на дужину ћелије, док је *M* број бита потребан за кодирање дужине *VOQ* реда. Када су у питању хардверски ресурси, посматра се број коришћених *LUT* (*LookUp Table*) елемената, регистара као и величина меморија.

Табела 6.1 Процијењени хардверски ресурси за дио на улазном порту *LB-BvN-GS* комутатора

	LUT	**Регистри [*b*]**	**Меморија [*b*]**
N=8, *CL*=128, *M*=12	771	323	2096
N=8, *CL*=128, *M*=16	823	323	2096
N=8, *CL*=256, *M*=12	1075	451	4144
N=8, *CL*=256, *M*=16	1127	451	4144
N=8, *CL*=512, *M*=12	1561	707	8240
N=8, *CL*=512, *M*=16	1713	707	8240
N=16, *CL*=128, *M*=12	1295	606	4224
N=16, *CL*=128, *M*=16	1411	606	4224
N=16, *CL*=256, *M*=12	1599	734	8320
N=16, *CL*=256, *M*=16	1715	734	8320
N=16, *CL*=512, *M*=12	2191	990	16512
N=16, *CL*=512, *M*=16	2307	990	16512
N=32, *CL*=128, *M*=12	2603	1265	8512
N=32, *CL*=128, *M*=16	2847	1265	8512
N=32, *CL*=256, *M*=12	2907	1393	16684
N=32, *CL*=256, *M*=16	3151	1393	16684
N=32, *CL*=512, *M*=12	3499	1649	33088
N=32, *CL*=512, *M*=16	3743	1649	33088

На основу резултата из табеле 6.1 може се закључити да број коришћених *LUT*-ова расте са порастом броја портова, дужине ћелија и броја бита који се користе за кодирање дужине *VOQ* редова, односно постоји зависност од сва три параметра. Што се тиче броја коришћених регистара, он зависи само од броја портова и дужине ћелија, при чему је зависност приближно линеарна по наведеним параметрима. Такође, величина меморија зависи од броја портова и дужине ћелија - од оба параметра појединачно има линеарну зависност што је и очекивано.

Табела 6.2 Процијењени хардверски ресурси за дио на улазном порту *mLB-BvN-GS* комутатора

	LUT	**Регистри [*b*]**	**Меморија [*b*]**
N=8, *CL*=128, *M*=12	180	-	-
N=8, *CL*=128, *M*=16	231	-	-
N=8, *CL*=256, *M*=12	180	-	-
N=8, *CL*=256, *M*=16	231	-	-
N=8, *CL*=512, *M*=12	180	-	-
N=8, *CL*=512, *M*=16	231	-	-
N=16, *CL*=128, *M*=12	402	-	-
N=16, *CL*=128, *M*=16	515	-	-
N=16, *CL*=256, *M*=12	402	-	-
N=16, *CL*=256, *M*=16	515	-	-
N=16, *CL*=512, *M*=12	402	-	-
N=16, *CL*=512, *M*=16	515	-	-
N=32, *CL*=128, *M*=12	855	-	-
N=32, *CL*=128, *M*=16	1099	-	-
N=32, *CL*=256, *M*=12	855	-	-
N=32, *CL*=256, *M*=16	1099	-	-
N=32, *CL*=512, *M*=12	855	-	-
N=32, *CL*=512, *M*=16	1099	-	-

У табели 6.2 је дат преглед коришћених хардверских ресурса за дио на улазном порту *mLB-BvN-GS* комутатора. С обзиром на то да се овдје пакети одмах по одабиру прослеђују централним портовима, не постоји потреба за регистрима или меморијама. Што се тиче броја коришћених *LUT*-ова не зависи од дужине ћелија већ само од броја портова и броја бита који се користе за кодирање дужине *VOQ* редова што је очекивано јер ћелије не пролазе кроз сам модул. Такође, ресурси на улазном порту су знатно мањи него код *LB-BvN-GS* што је у складу са описом *mLB-BvN-GS* датим у пглављу 4.2.

Када је у питању централни порт, његова архитектура је иста и за *LB-BvN-GS* и за *mLB-BvN-GS* комутатор. На основу резултата се уочава да број коришћених *LUT*-ова

првенствено зависи од дужине ћелије, а мање од броја портова. Што се тиче броја коришћених регистара и величине меморије, они расту са порастом броја портова или дужине ћелије. При томе, меморија очекивано линеарно зависи од димензије ћелије, односно броја портова, док регистри доминантно и приближно линеарно зависе од дужине ћелије.

Табела 6.3 Процијењени хардверски ресурси за дио на централном порту за *LB-BvN-GS* , односно *mLB-BvN-GS* комутатор

	LUT	**Регистри [*b*]**	**Меморија [*b*]**
N=8, *CL*=128	471	155	2048
N=8, *CL*=256	901	283	4096
N=8, *CL*=512	1751	540	8192
N=16, *CL*=128	411	174	4096
N=16, *CL*=256	757	302	8192
N=16, *CL*=512	1435	559	16384
N=32, *CL*=128	421	209	8192
N=32, *CL*=256	723	337	16384
N=32, *CL*=512	1315	594	32768

Табела 6.4 Процијењени хардверски ресурси за дио на излазном порту *mLB-BvN-GS* комутатора

	LUT	**Регистри [*b*]**	**Меморија [*b*]**
N=8, *CL*=128	18	12	-
N=8, *CL*=256	18	12	-
N=8, *CL*=512	18	12	-
N=16, *CL*=128	29	21	-
N=16, *CL*=256	29	21	-
N=16, *CL*=512	29	21	-
N=32, *CL*=128	55	38	-
N=32, *CL*=256	55	38	-
N=32, *CL*=512	55	38	-

У табели 6.4 су дати резултати за дио на излазном порту *mLB-BvN-GS* комутатора. Јасно се уочава да број коришћених *LUT*-ова или регистара не зависи од дужине ћелије већ само од броја портова, а сами ресурси су веома скромни јер се практично право ресеквенцирање врши у самим баферима који се користе и за чување ћелија док се пакет кога сачињавају не комплетира.

На основу датих резултата из табела 6.1-6.4, може се закључити да је потреба за ресурсима скромна за све модуле, као што је и претпостављено у поглављу 4. Такође, може

се закључити да потреба за ресурсима код улазних и излазних портова расте скоро па линеарно са бројем портова N и да дизајн нема проблема са скалабилношћу. Увидом у архитектуру добијену процесом анализе и синтезе за сваки од реализованих модула, закључено је да је најкритичнији дио одабир најдужег реда за чекање, због више степени комбинационе логике што може да лимитира радну фреквенцију такта због кашњења кроз ту комбинациону логику.

7. ПОДРШКА ЗА ФЕР СЕРВИС

У петом поглављу је извршена компарација предложеног рјешења са већ постојећим комутаторима при чему су разматрани регуларни саобраћајни сценарији. Под регуларним саобраћајним сценаријима се подразумијевају они сценарији који не доводе до преоптерећења неког излазног порта, односно максимални саобраћај намјењен неком излазном порту не превазилази његов капацитет. Међутим, веома је битно и понашање комутатора под нерегуларним условима, односно кад су неки од излазних портова преоптерећени. Наиме, веома често се дешава да у случају преоптерећења излазног порта неки од токова добију знатно већи удио у капацитету него што је фер према другим токовима.

Приликом анализа нерегуларних саобраћајних сценарија показало се да постојећа рјешења заснована на *LB-BvN* архитектури имају лоше перформансе са становишта фер опслуживања [139,140]. Приликом почетних анализа *LB-BvN-GS* рјешења се показало да он има боље перформансе у погледу фер опслуживања, али не и идеалне, нарочито у одређеним испитиваним сценаријима. Стога је у овом поглављу предложено унапријеђење *LB-BvN-GS* које омогућава бољу подршку за фер сервис. У наставку овог поглавља ће прво бити изложено унапријеђење *LB-BvN-GS* које даје бољу подршку за фер сервис, а потом ће бити дати резултати компарације са постојећим рјешењима, као и *LB-BvN-GS* рјешењем без унапријеђења. Напоменимо да су по питању фер сервиса *LB-BvN-GS* и *mLB-BvN-GS* исти јер се избор пакета на улазном порту врши по истом принципу. Из тог разлога, предложено унапријеђење је примјенљиво и на *mLB-BvN-GS*.

7.1. Фер сервис подршка за LB-BvN-GS

Принцип одабира пакета за слање у *LB-BvN-GS* се заснива на размјени вектора заузетости између улазних портова, при чему су улазни портови повезани у кружни ланац. При томе критеријум за избор тока из ког ће бити послат пакет је дужина реда за чекање (наравно, уз услов да вектор заузетости означава да ток може да учествује у избору). Један од проблема који се јавља приликом одабира пакета приликом загушења неког од излазних портова је да ће токови ка том излазном порту имати дуже редове за чекање па ће бити чешће бирани ако вектор заузетости дозвољава њихов избор. Ако посматрамо два сусједна порта (на пример, i и i+1) због кружног поретка ланца, порт i+1 ће практично све векторе заузетости, сем вектора који креће од порта i, да процесира раније. Отуда ако оба порта имају загушене токове ка излазу j постоји опасност да већину слободних мјеста за излаз j у векторима заузетости заузме улазни порт i+1. Да се то не би дешавало, осмишљена је допуна принципа заузимања излаза у векторима заузетости описаног у поглављу 4.2.

Поред вектора заузетости, уводи се вектор оптерећености токова. Сваки улазни порт генерише вектор оптерећености токова за које је тај улазни порт изворишни. При томе, за сваки ток се додјељују два бита сигнализације. Један бит означава загушеност тог тока, а други бит представља индикацију да ли је ред за чекање дотичног тока празан или не. Стога један вектор оптерећености садржи 2N бита. Вектори оптерећености се шаљу по истом принципу као и вектор заузетости - кроз кружни ланац улазних портова. На тај начин, у току

једног циклуса сваки улазни порт добија бинарну информацију о стањима свих токова. У случају загушења потребно је да за загушене излазе постоји арбитар који расподјељује ресурсе загушеног излаза на фер начин. Предложено унапријеђење предлаже да се арбитар имплементира на једном од улазних портова у виду додатног модула пошто сви улазни портови имају исту информацију о загушености токова. Алтернатива је да се арбитар имплементира као централни контролер који би информацију (векторе оптерећености) добијао од улазних портова. Арбитар би у сваком циклусу послао улазним портовима информацију на које централне портове у наредном циклусу треба да пошаљу пакете за загушене излазе. Улазни портови и даље бирају ток из ког ће послати пакет ка неком централном порту по начину описаном у поглављу 4.2, али уз додатни услов да тај централни порт није већ оглашен од стране арбитра за тај улаз и за тај циклус. Очигледно ако нема загушених токова, принцип рада остаје потпуно идентичан ономе описаном у поглављу 4.2.

Битан фактор претходно описаног приступа за постизање фер сервиса је детектовање загушених токова. Дефинишу се горњи и доњи праг детекције. Када број пакета у реду за чекање превазиђе горњи праг активира се ознака да је ток загушен. Тек кад се спусти број пакета испод доњег прага у реду за чекање који је означен као загушен, тај ред ће бити означен као незагушен. Разлог за постојање доњег прага је да не дође до честог осциловања ознаке за загушење неког тока.

Потреба за централним арбитром (реализован или на једном од улазних портова као додатни модул или као посебан централни контролер) проистиче из потребе да се фер сервис омогући и у комплекснијим сценаријима где улазни портови могу имати више преоптерећених токова тј. редова за чекање, при чему долази до преклапања по излазним портовима са другим преоптерећеним токовима других улазних портова. У таквим ситуацијама је најбоље да се све информације налазе на истом мјесту ради распоређивања преоптерећених капацитета на правичан начин.

Принцип рада арбитра је сљедећи. У сваком циклусу се врши распоређивање токова по преоптерећеним излазним портовима тако што се преоптерећени излази обрађују редом. При томе редослијед обрађивања излазних портова се одређује *round-robin* принципом кроз циклусе, тако да се постигне фер распоређивање са становишта преоптерећених излазних портова, тј. одговарајућих токова. Са становишта преоптерећеног излазног порта се разликују четири класе токова, празни, непразни, једноструки загушени и вишеструки загушени токови. Празни токови су они који немају пакете за дотични излазни порт. Непразни токови су они који имају пакете за дотични излазни порт, али нису означени као преоптерећени токови. Једноструки загушени токови су они који имају пакете за дотични излазни порт и означени су као преоптерећени токови, али на том улазном порту не постоје други преоптерећени токови. Вишеструки загушени токови су они који имају пакете за дотични излазни порт и означени су као преоптерећени токови, и на том улазном порту постоје и други преоптерећени токови. Класификација токова се ради на основу вриједности вектора оптерећености токова.

Постоје две фазе распоређивања пакета за преоптерећени излазни порт. У првој фази се за све токове који нису непразни (а чије одредиште је дотични преоптерећени порт) распоређује по један пакет тако што се за сваки тај ток бира по један централни порт. При томе, разликују се две класе централних портова. Једну чине они централни портови које су већ раније (приликом процесирања преоптерећених излазних портова који су раније обрађивани) заузели сви улазни портови са вишеструким загушеним токовима. А другу чине

сви остали централни портови. Приоритет при одабиру има прва класа. Сам одабир централних портова се врши по принципу вектора заузетости. Такође, уведени су и улазни вектори заузетости који дефинишу за сваки улазни порт који централни порт су већ заузели за неки од преоптерећених излаза (током циклуса улазни порт може послати максимално један пакет ка неком централном порту).

У другој фази учествују централни портови који још нису заузети за пренос пакета ка тренутно обрађиваном преоптерећеном излазном порту, као и улазни портови који имају загушени (једноструки или вишеструки) ток ка том тренутно обрађиваном излазном порту. По *round-robin* принципу улазни портови се смењују и бирају централни порт из скупа преосталих централних портова. Избор се врши или док се не заузму сви централни портови или док се не исцрпи избор.

Након што су обрађени сви преоптерећени излази, информације се шаљу улазним портовима о распоређеним пакетима токова за преоптерећене излазне портове. Ове информације се могу слати и сукцесивно након завршетка сваког обрађеног преоптерећеног излазног порта да би се релаксирао интезитет слања. Битно је напоменути да претходно описани рад арбитра (укључујући и слање резултата ка улазним портовима) може трајати нешто краће од трајања циклуса.

7.2. Испитивање перформанси предложеног унапријеђења

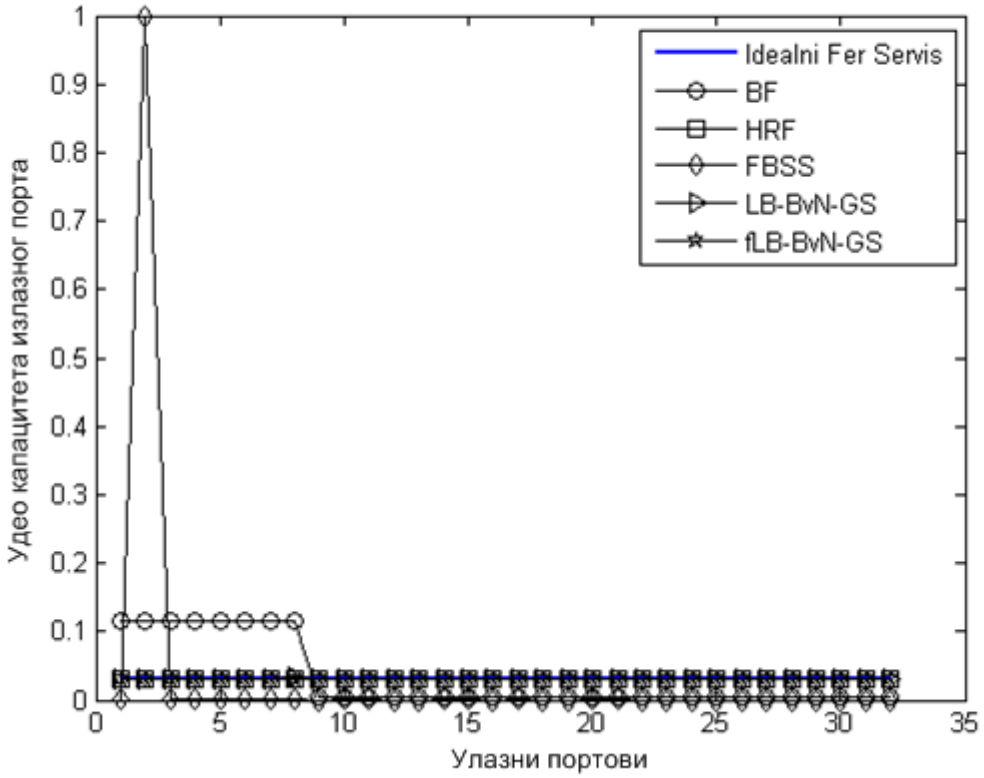
У оквиру овог потпоглавља су приказани резултати испитивања перформанси са становишта фер сервиса предложеног унапријеђења. У наставку ће бити коришћена скраћеница *fLB-BvN-GS* (*fair LB-BvN-GS*) за предложено унапријеђење. При томе је урађена компарација са другим алгоритмима са којима је рађено поређење у поглављу 5 - алгортими *BF*, *CR*, *FBSS*, *HRF*. Такође су приказани резултати и за *LB-BvN-GS* (основна варијанта описана у поглављу 4.2, тј. без унапријеђења). Напомена је да *mLB-BvN-GS* има исте перформансе као и *LB-BvN-GS* са становишта фер сервиса због истог принципа одабира пакета за слање. Приликом анализе перформанси се испоставило да *CR* и *BF* у свим испитиваним сценаријима имају практично идентичне перформансе (разлике су занемарљиве) тако да је на графицима у овом потпоглављу приказан само *BF*, али исте криве важе и за *CR*.

Компарација је дата за димензије комутатора 32x32, али исти закључци се добијају и за друге димензије комутатора. У свим сценаријима оптерећење улазних портова је максимално тј. 100%. Индекси портова се нумеришу почев од 1 у описима сценарија и приказу резултата. За компарацију приказану у овом потпоглављу изабрани су сљедећи сценарији који загушују један или више излазних портова, а дају јасан увид у перформансе поређених алгоритама са становишта фер сервиса:

- **Сценарио 1** - У оквиру овог сценарија првих *X* портова шаље све пакете на излаз 1, док остали униформно распоређују свој саобраћај по свим излазним портовима. У овом сценарију је преоптерећен излазни порт 1.
- **Сценарио 2** - У оквиру овог сценарија првих *X* портова шаље све пакете на излаз 1, док остали распоређују свој саобраћај по свим излазним портовима по *hot-spot* расподјели. У овом сценарију је преоптерећен излазни порт 1.
- **Сценарио 3** - У оквиру овог сценарија првих *X* портова шаље све пакете на излаз 32, док остали распоређују свој саобраћај по свим излазним портовима по *hot-spot* расподјели. У овом сценарију је преоптерећен излазни порт 32. Разлика у односу на

сценарио 2 је што сада у преоптерећеном излазном порту учествује и *hot-spot* ток са улазног порта који има исти индекс као преоптерећени излазни порт.

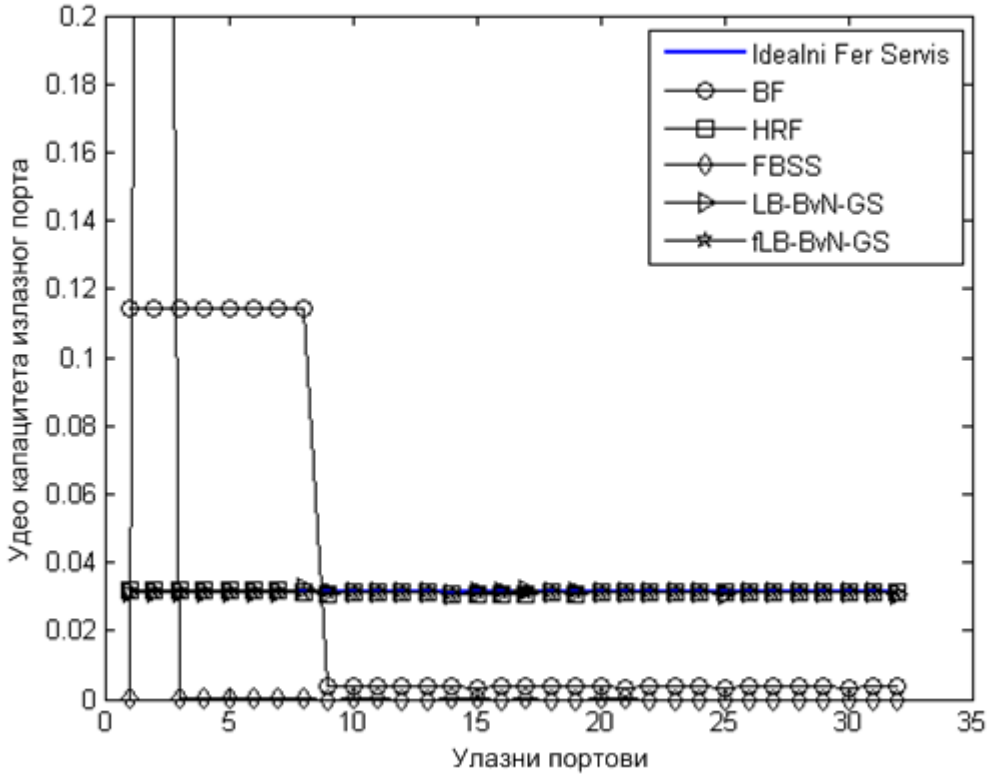
- **Сценарио 4** - У оквиру овог сценарија постоје четири тока са улазног порта 6. У питању су токови 6->1, 6->11, 6->21, 6->31, при чему сва четири тока имају исто улазно оптерећење тј. 25%. Остали улазни портови распоређују свој саобраћај по свим излазним портовима по *hot-spot* расподјели. У овом сценарију су преоптерећени излазни портови 1, 11, 21 и 31.
- **Сценарио 5** - Овај сценарио је сличан сценарију 4. Разлика је што сада улазни порт 16 има исто понашање као улазни порт 6, односно на том улазном порту постоје четири тока 16->1, 16->11, 16->21, 16->31, при чему сва четири тока имају исто улазно оптерећење тј. 25%. И у овом сценарију су преоптерећени излазни портови 1, 11, 21 и 31.
- **Сценарио 6** - Овај сценарио представља модификацију сценарија 5, тако што се мијењају токови на улазном порту 16. Сада су токови на том порту: 16->3, 16->13, 16->23, 16->32. У овом сценарију су преоптерећени излазни портови 1, 3, 11, 13, 21, 23, 31 и 32. Такође, може се рећи и да сценарио 6 суштински представља сценарио 4 са два дисјунктна скупа преоптерећених излазних портова.



Слика 7.2.1 Сценарио 1 - *X*=8

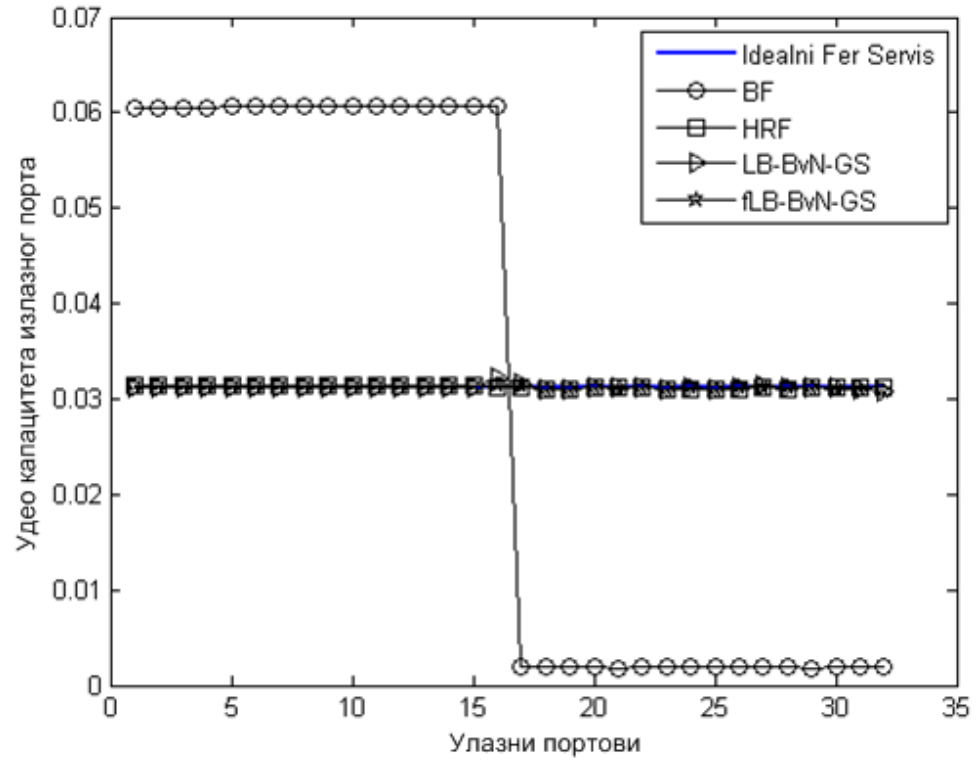
На сликама 7.2.1 и 7.2.2 су приказани остварени резултати за сценарио 1 када је *X* постављено на вриједност 8. Приказан је удио капацитета преоптерећеног излазног порта 1 по токовима. Може се уочити да је *FBSS* изразито нефер и да један од улазних портова са ког долази агресивни ток ка преоптерећеном излазу практично заузима комплетан капацитет преоптерећеног излазног порта. Разлог за ово понашање је у природи заузимања мјеста на

централном порту јер један од агресивних токова почиње да заузима мјесто намијењено преоптерећеном излазу и тиме блокира све остале токове чије је одредиште преоптерећени излазни порт. Ово је у складу са анализама приказаним у [139,140]. Ради прегледности, у осталим сценаријима неће бити приказиван *FBSS* да би се могао јасније видјети однос у перформансама осталих испитиваних алгоритама. Слика 7.2.2 даје приказ за исти сценарио, али са ограничењем у приказу по *y* оси тако да се може боље видјети однос између осталих тестираних алгоритама. Може се уочити да *BF* има значајна одступања од идеалног фер сервиса, док остали алгоритми веома добро прате криву идеалног фер сервиса. Исте релације важе и за друге вриједности *X*, а на слици 7.2.3 је приказан резултат за *X*=16 који демонстрира ову тврдњу.

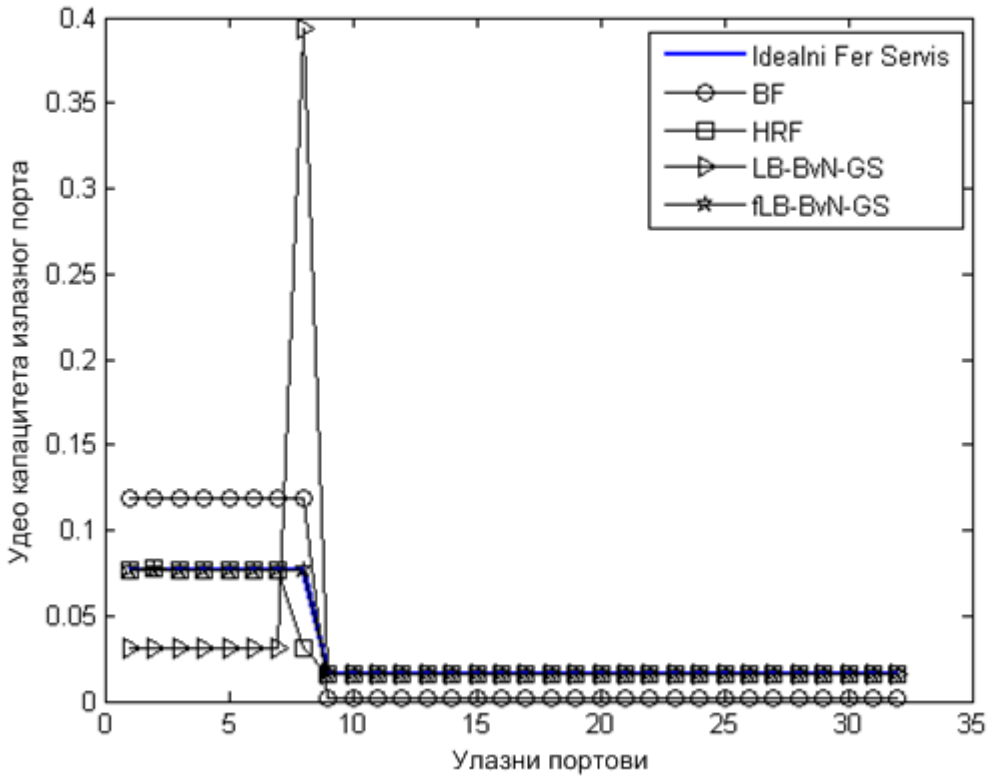


Слика 7.2.2. Сценарио 1 - *X*=8, приказ са ограничењем по *y* оси

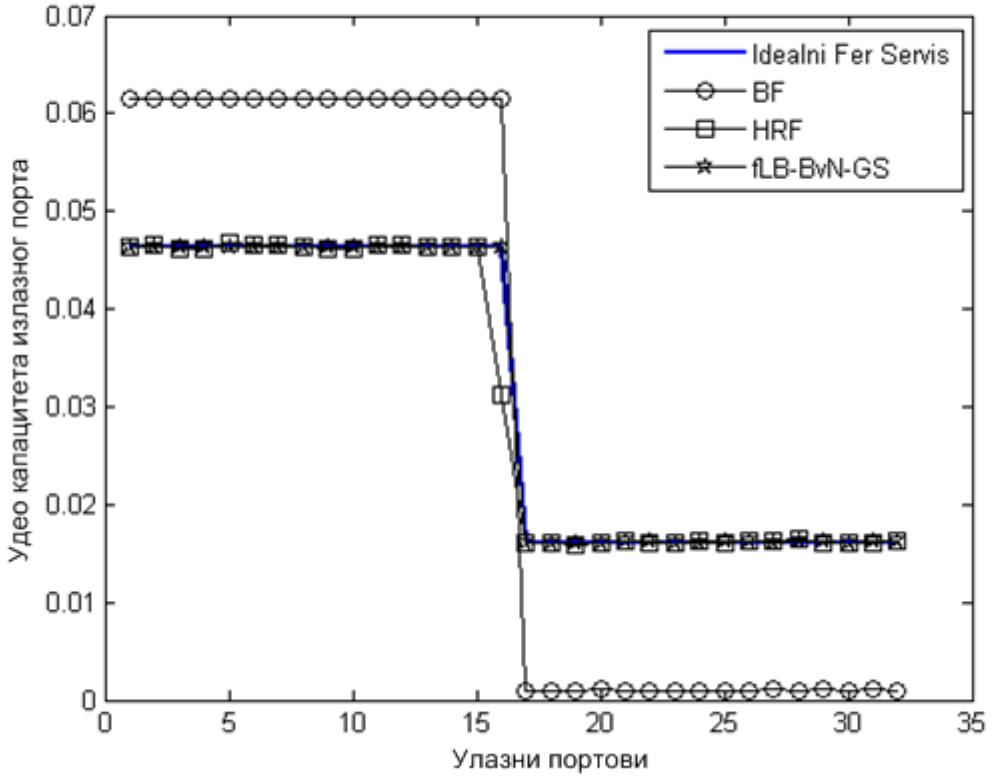
На слици 7.2.4 је приказан резултат компарације за сценарио 2 за *X*=8 при чему се посматра преоптерећени излазни порт 1. *BF* има слично понашање као и у сценарију 1, агресивнији токови добијају више удјела од онога што би требало да добију по фер расподјели. *HRF* прилично добро прати криву идеалног фер сервиса, али има једно значајније одступање и то за ток са улазног порта 8. С друге стране, сада *LB-BvN-GS* има значајнија одступања за агресивне токове, где један од агресивних токова покупи готово све централне портове (за излаз 1) у једном циклусу које нису заузели токови који су мање агресивни. Може се уочити да *fLB-BvN-GS* одлично прати криву идеалног фер сервиса, односно предложено унапријеђење успешно обавља своју намјену.



Слика 7.2.3. Сценарио 1 - *X*=16



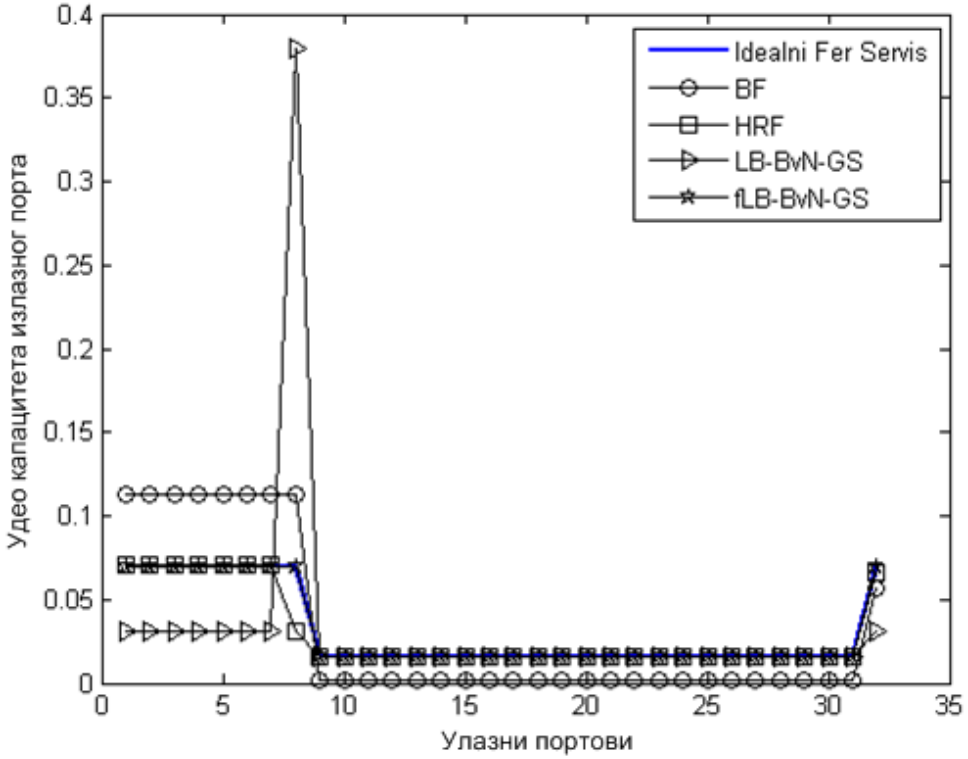
Слика 7.2.4. Сценарио 2 - *X*=8



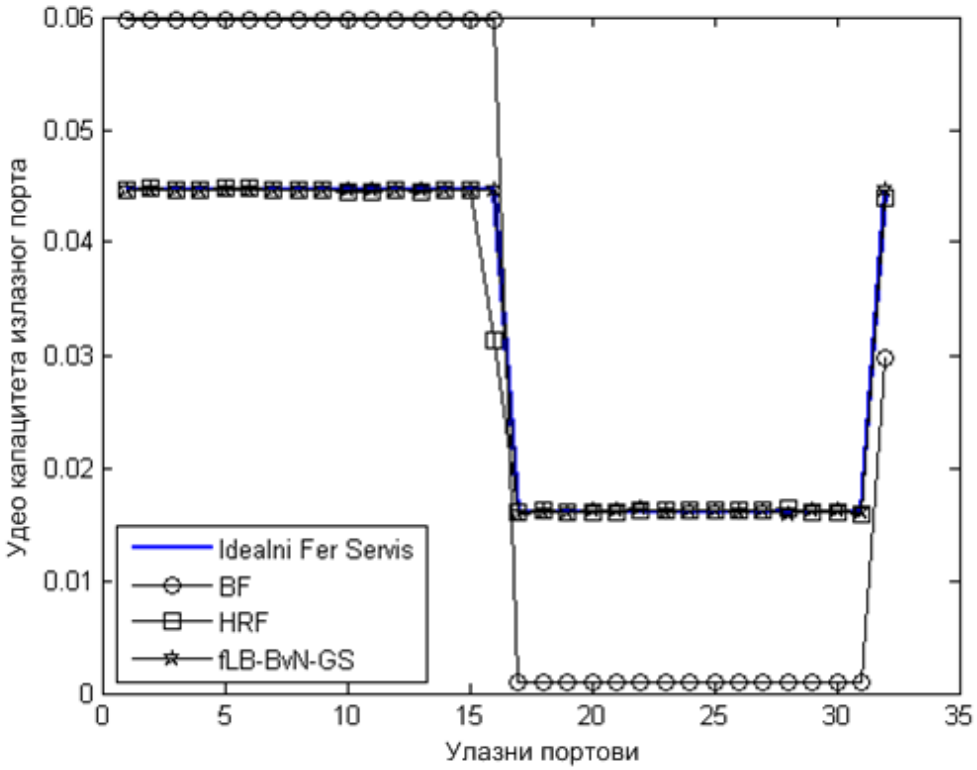
Слика 7.2.5. Сценарио 2 - *X*=16

На слици 7.2.5 је приказан резултат компарације за сценарио 2 за *X*=16 при чему се посматра преоптерећени излазни порт 1. Релације између свих алгоритама су исте као и за случај *X*=16. Сада је изостављен приказ за *LB-BvN-GS* да би се боље видјела релација између остала три алгоритма. Сада се још боље види одступање на једном току од идеалног фер сервиса за *HRF*, док *fLB-BvN-GS* готово идеално прати криву идеалног фер сервиса.

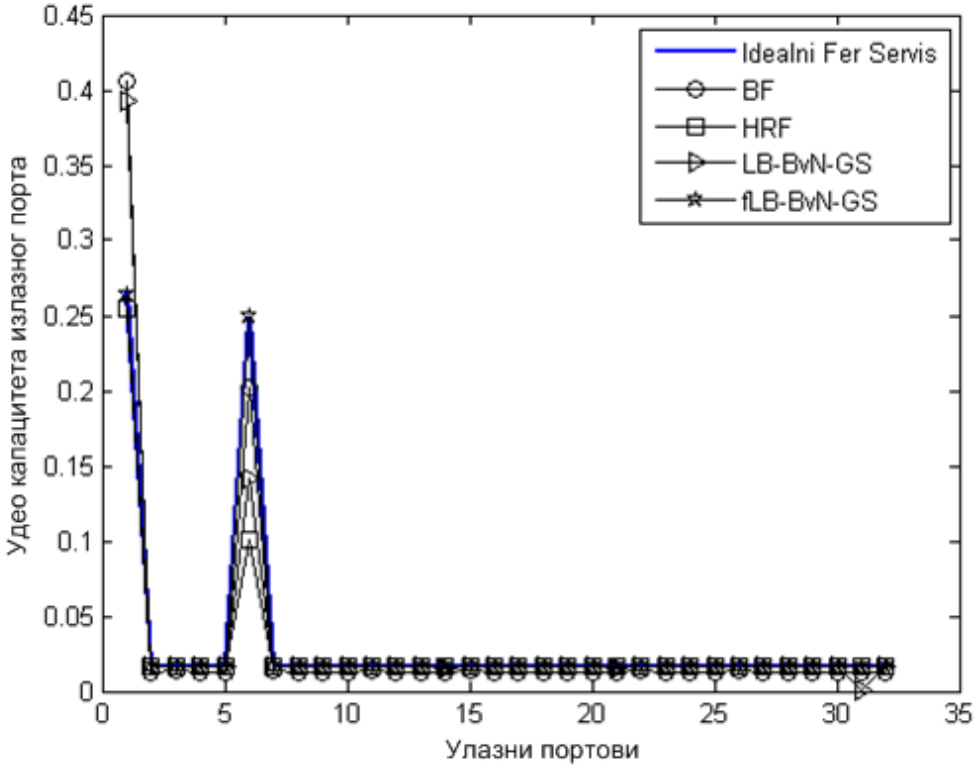
На слици 7.2.6 је приказан резултат компарације за сценарио 3 за *X*=8 при чему се посматра преоптерећени излазни порт 32. Сада поред *X* агресивних токова постоји још један ток (са улазног порта 32) који је такође загушен (има већи улазни проток од онога што му припада по фер сервису), али се може сматрати полуагресивним током јер је његов проток два пута мањи од осталих агресивних токова у сценарију. Добијено понашање за све испитиване алгоритме је готово идентично ономе из сценарија 2. Опет код *LB-BvN-GS* један агресивни ток добија знатно више удјела од предвиђеног фер удјела у односу на друге агресивне токове (у које спада и ток 32->32) а који добијају мање од предвиђеног фер удјела. *HRF* и даље има само један ток који одступа од криве идеалног фер сервиса, док *fLB-BvN-GS* и даље одлично прати криву идеалног фер сервиса. Код *BF*, ток 32->32 иако спада у агресивне токове добија нешто мање од предвиђеног фер удјела јер постоје други агресивнији токови. Ово је једина уочљива разлика посматрано са становишта свих алгоритама и резултата сценарија 2 и 3. На слици 7.2.7 је приказан резултат сценарија 3 за *X*=16 при чему је изостављен приказ за *LB-BvN-GS* да би се добио бољи увид у релације преостала три алгоритма. Понашање се поклапа са резултатима добијеним за *X*=8, односно релације између алгоритама и њихово понашање је остало исто.



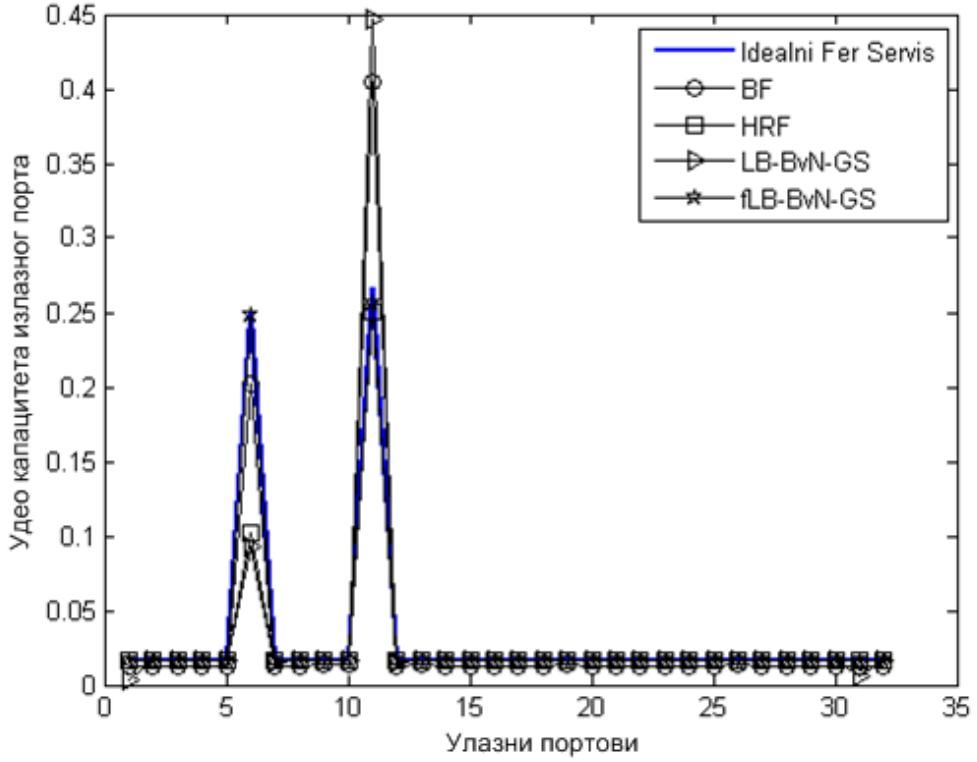
Слика 7.2.6. Сценарио 3 - *X*=8



Слика 7.2.7. Сценарио 3 - *X*=16

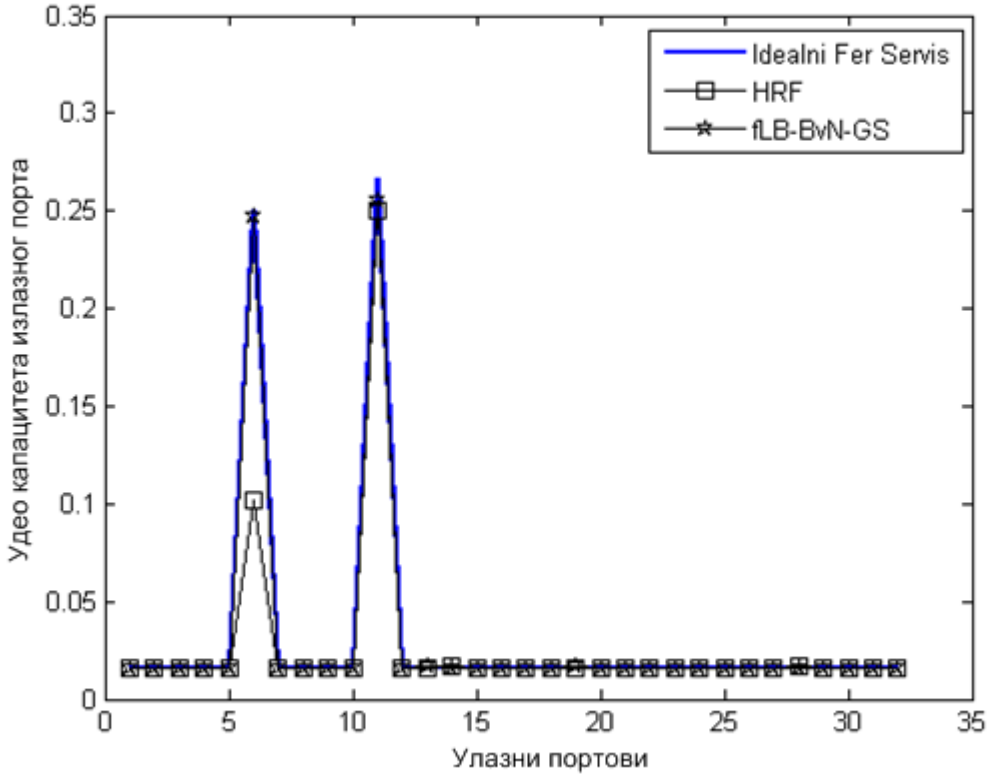


Слика 7.2.8. Сценарио 4 - излазни порт 1



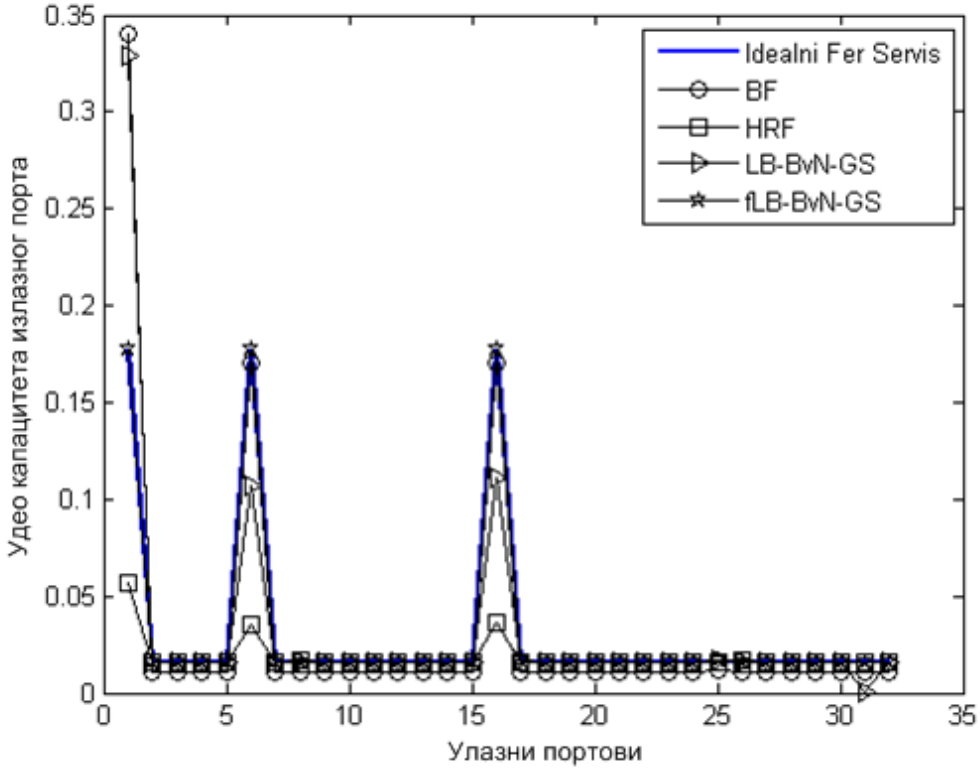
Слика 7.2.9. Сценарио 4 - излазни порт 11

На сликама 7.2.8 и 7.2.9 је дат приказ резултата за сценарио 4 за излазне портове 1 и 11, респективно. Резултати за излазне портове 21 и 31 нису приказани јер је понашање алгоритама идентично ономе приказано за излазне портове 1 и 11. Лако је уочити да дјелује да сви алгоритми углавном добро прате криву идеалног фер сервиса сем у случају агресивних токова гдје постоје највећа одступања. Пре свега највећа одступања имају *LB-BvN-GS* и *BF* гдје агресивнији (*hot-spot*) ток добија већи проценат капацитета него што треба. У случају *HRF*, опет један ток (у овом случају ток са улаза 6) добија нешто мањи удио. *fLB-BvN-GS* и даље готово идеално прати криву идеалног фер сервиса што се може видјети и из детаљнијег приказа за излаз 11 на слици 7.2.10 гдје је издвојен приказ два најбоља алгоритма (*HRF* и *fLB-BvN-GS*).

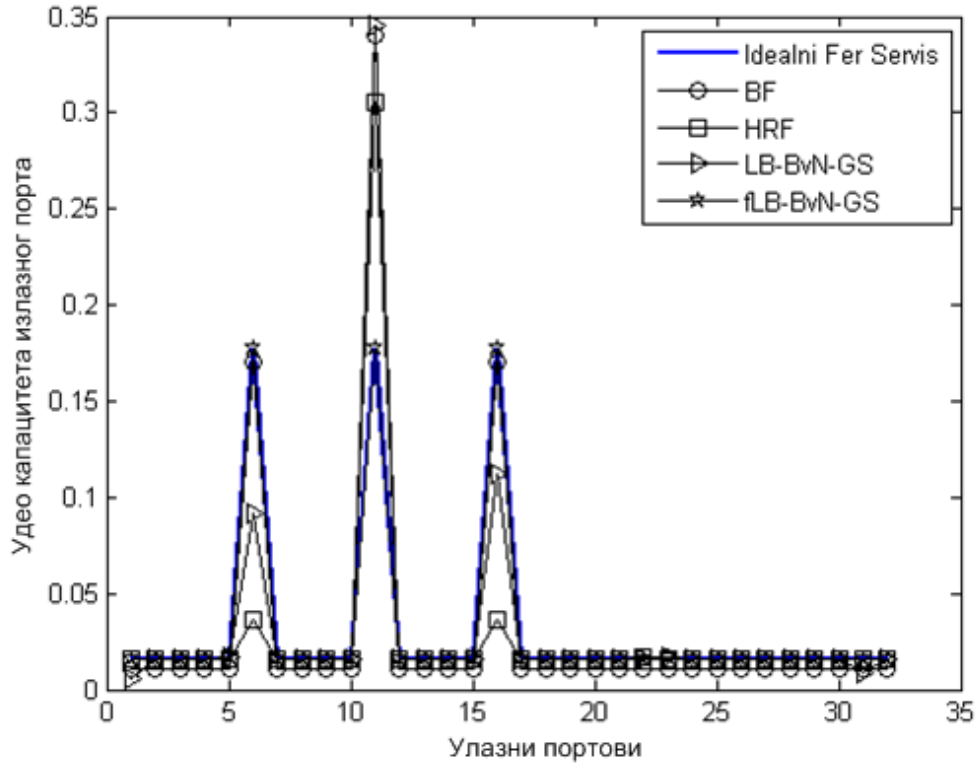


Слика 7.2.10. Сценарио 4 - излазни порт 11 - приказ два најбоља алгоритма

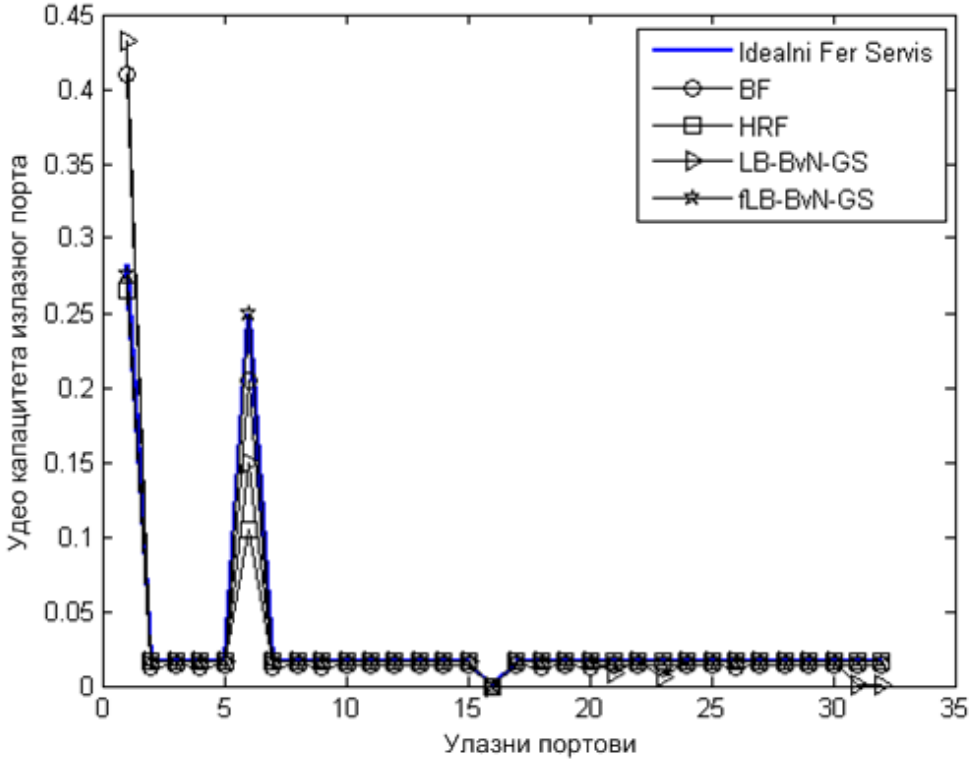
На сликама 7.2.11 и 7.2.12 је дат приказ резултата за сценарио 5 за излазне портове 1 и 11, респективно. Резултати за излазне портове 21 и 31 нису приказани јер је понашање алгоритама идентично ономе приказано за излазни порт 11. У суштини, понашање свих алгоритама је доста слично сценарију 4. *fLB-BvN-GS* и даље готово идеално прати криву идеалног фер сервиса и остварује најбоље перформансе са становишта фер опслуживања. Остали алгоритми и даље имају највећа одступања код агресивних токова по сличном принципу као у сценарију 4. Једино одступање од понашања из сценарија 4 је уочено код *HRF* алгоритма и излазног порта 1 гдје сви агресивни токови добијају мање удјела од предвиђене идеалне вриједности.



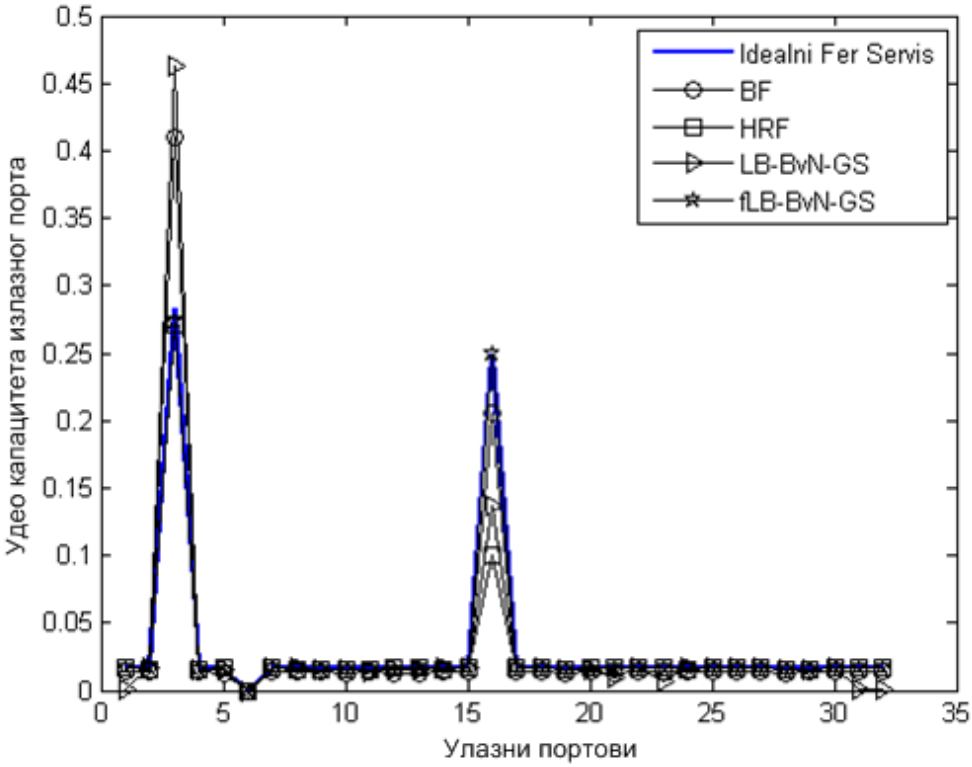
Слика 7.2.11. Сценарио 5 - излазни порт 1



Слика 7.2.12. Сценарио 5 - излазни порт 11



Слика 7.2.13. Сценарио 6 - излазни порт 1



Слика 7.2.14. Сценарио 6 - излазни порт 3

На сликама 7.2.13 и 7.2.14 је дат приказ резултата за сценарио 6 за излазне портове 1 и 3, респективно. Као што је речено у опису сценарија 6, овај сценарио представља сценарио 4 гдје постоје два раздвојена (међусобно независна) скупа преоптерећених излазних портова. У једном скупу учествују токови са улазног порта 6 и одговарајући *hot-spot* токови који учествују у тим излазним портовима, а у другом учествују токови са улазног порта 16 и одговарајући *hot-spot* токови за излазне портове тог другог скупа. Добијено понашање одговара ономе из сценарија 4, па је отуда дат приказ за по само један излазни порт из оба скупа. Лако је уочити да је понашање свих алгоритама у складу са оним из сценарија 4.

На основу добијених резултата сценарија, очигледно је да предложено унапријеђење (*fLB-BvN-GS*) остварује боље перформансе од других алгоритама, а такође постиже резултате који одговарају идеалном фер сервису.

8. МУЛТИКАСТ ПАКЕТСКИ КОМУТАТОРИ

У овом поглављу ће бити представљена постојећа рјешења мултикаст комутатора. У првим деценијама развоја интернет мреже, корисници су рачунаре већином користили за претраживање веб садржаја (попут вијести, музике и слично), и размјену електронских порука. Међутим, драстично повећање капацитета интернета и повећање доступности рачунара, таблета, паметних телефона већем броју људи, утицало је и на развој нових сервиса који су стекли изузетну популарност. За неке од тих сервиса попут гледања видео садржаја на захтјев, телевизије преко интернета (*IPTV*), учења на даљину, онлајн играња игара, је карактеристично да се исти садржај дијели истовремено са великим бројем корисника. Како би се мрежни ресурси што ефикасније користили, логично је да се не шаље много копија једног те истог садржаја, већ само једна, а онда се та копија умножава у појединим чворовима мреже да би стигла на сва одредишта. Такав тип саобраћаја се зове мултикаст саобраћај. Развијени су и протоколи, попут *PIM* (*Protocol Independent Multicast*) протокола, који служе за ефикасно управљање мултикаст саобраћајем у мрежи. Када мултикаст пакети дођу до одређеног мрежног чвора они се прослеђују на више излазних портова. Међутим, присуство мултикаст саобраћаја знатно отежава процес комутације. Један начин управљања мултикаст саобраћајем у мрежним чворовима је употреба тзв. мултикаст комутационих модула који омогућују да се мултикаст пакет симултано прослеђује на више излаза. Пошто у општем случају мултикаст пакете треба прослиједити на произвољан број излаза, уколико постоји више активних мултикаст токова у паралели врло је изазовно ефикасно конфигурисати пакетски комутатор тако да се оствари висока пропусност пошто су могућа преклапања између мултикаст токова по питању излазних портова на која треба да се проследе њихови пакети. Други приступ јесте да се на улазном порту направе копије пакета за сваки од излазних портова на које мултикаст пакет треба да се прослиједи и да се такви пакети третирају као уникаст пакети. На овај начин се значајно повећава проток пакета јер имамо ситуацију као да је на један улазни порт у истом слоту стигло више пакета. Иако је ово рјешење релативно једноставно за имплементацију јер користи једноставније уникаст комутационе модуле постоји проблем са ефикасношћу због вештачког повећања улазног протока. Како с развојем нових сервиса количина мултикаст саобраћаја више није занемарљива јавила се потреба за развојем нових рјешења која ће на ефикасан начин, истовремено управљати и уникаст и мултикаст саобраћајем.

Како је на почетку развоја интернета доминирао уникаст саобраћај, то је и фокус истраживања био на развоју комутатора који ће на најефикаснији начин управљати овим саобраћајем. С друге стране, како с развојем нових сервиса количина мултикаст саобраћаја више није занемарљива, дошло је до проблема ефикасног прослеђивања оваквог саобраћаја. Постојећи пакетски комутатори који остварују добре перформансе за уникаст саобраћај, углавном не успевају исто за мултикаст саобраћај због сложеније природе мултикаст саобраћаја која се огледа у потреби да мрежни чворови морају да праве додатне копије пакета. Зато се и јавила потреба за развојем нових рјешења која ће на ефикасан начин, истовремено управљати и уникаст и мултикаст саобраћајем.

Генерално, велики број мултикаст комутатора представља одређену модификацију постојећих уникаст комутатора. Постоји неколико приступа у управљању мултикаст саобраћајем. Прва идеја је да се умножавање пакета врши на улазним портовима. Наиме, када мултикаст пакет стигне на улазни порт, утврђује се на које излазне портове треба прослиједити тај пакет. Затим се прави одговарајући број копија пакета и они се даље третирају као уникаст пакети. Проблем код оваквог приступа је што може доћи до преоптерећења улазних портова пошто долази до вештачког повећања улазног протока. Други приступ јесте употреба мултикаст комутационог модула (*multicast capable switch fabric*). Код овог рјешења, умножавање пакета се врши унутар комутационог модула. Проблем овог приступа је употреба компликованијег и скупљег (мултикаст) комутационог модула, али се типично постижу боље перформансе.

Као што је већ напоменуто, већина предложених рјешења се заснива на проширењу или унапређењу постојећих комутатора који су првобитно били развијени искључиво за уникаст саобраћај. На примјер, предложено је унапређење комутатора са баферима на улазним портовима гдје су осим бафера у којим су смјештени уникаст пакети додати бафери у којима су смјештени мултикаст пакети [141]. Такође, алгоритми за конфигурисање комутатора су иновирани да би ефикасно управљали и са уникаст и са мултикаст саобраћајем [142].

Када су у питању *LB-BvN* комутатори, ова тема није пуно разматрана. Једино је за *FBSS* комутатор предложена модификација која омогућава ефикасно управљање мултикаст саобраћајем [143].

У наредним потпоглављима ће бити објашњен принцип рада новијих рјешења која су прилагођена за ефикасан рад и са уникаст и са мултикаст саобраћајем. Биће представљена рјешења која се заснивају на комутаторима са баферима на улазним портовима као и рјешење које је засновано на *FBSS* комутатору. Такође, у сљедећем поглављу биће представљено рјешење које представља један од главних доприноса ове дисертације, а које ће бити поређено са рјешењима представљеним у овом поглављу за различите моделе саобраћаја.

8.1. Итеративни алгоритам за управљање мултикаст саобраћајем за комутатор са баферима на улазним портовима

У [141] је предложено рјешење за управљање мултикаст саобраћајем у комутатору са баферима на улазним портовима. Како је већ речено у уводу овог поглавља, умножавање пакета код ових комутатора се може радити на улазним портовима. Након тога они се даље процесирају као уникаст пакети. У том случају, не постоји потреба за посебним распоређивачем за мултикаст саобраћај, већ је довољан распоређивач за уникаст саобраћај. Међутим, проблем код оваквог приступа је што може доћи до преоптерећења улазних портова, у случају када имамо велики прилив мултикаст пакета које треба прослиједити на већи број излазних портова. Наиме, када мултикаст пакет стигне на улазни порт, врши се његово умножавање у више примјерака зависно од тога на колико излазних портова треба да буде прослијеђен. Тако се на улазним портовима ствара исти ефекат као да је дошло више пакета, а не један. Самим тим, више не важи правило да у сваком временском слоту на један улазни порт може да дође највише један пакет, већ то зависи од броја мултикаст пакета и броја портова на који треба прослиједити те пакете.

Алтернатива је да се користи мултикаст комутациони модул. У том случају се умножавање пакета врши у самом комутационом модулу приликом комутације пакет тј.

његовог прослеђивања са улаза на одговарајуће излазе. То значи, да када улазни порт шаље мултикаст пакет, он шаље само један пакет, а онда се унутар комутационог модула прави одговарајући број копија које стижу на излазне портове. Као што је већ речено, за комутаторе са баферима на улазним портовима кључни проблем је *HoL* блокирање. У случају уникаст саобраћаја, овај проблем се решава са *N VOQ* редова на сваком улазном порту. Ако би хтјели да сличан принцип примјенимо на мултикаст саобраћај, требало би да постоји посебан *VOQ* ред за све мултикаст пакете које треба прослиједити на исте излазе. Ово рјешење није практично за мултикаст саобраћај, јер би захтијевало да на сваком улазном порту буде 2^N-1 *VOQ* редова. Наиме, јасно је да је за мултикаст пакете које треба прослиједити на један излаз потребно *N VOQ* редова, по један за сваки од излаза. Даље, за мултикаст пакете које треба прослиједити на 2 излаза потребан број *VOQ* редова је:

$$\binom{N}{2} = \frac{N!}{2! \cdot (N-2)!} = \frac{N \cdot (N-1)}{2} \tag{8.1.1}$$

За мултикаст пакете које треба прослиједити на 3 излаза потребан број *VOQ* редова је:

$$\binom{N}{3} = \frac{N!}{3! \cdot (N-3)!} = \frac{N \cdot (N-1) \cdot (N-2)}{3 \cdot 2} \tag{8.1.2}$$

За мултикаст пакете које треба прослиједити на 31 излаз потребан број *VOQ* редова је:

$$\binom{N}{31} = \frac{N!}{31! \cdot (N-31)!} = N \tag{8.1.3}$$

Коначно, за мултикаст пакете које треба прослиједити на 32 излаза потребан је један *VOQ* ред. Дакле, укупан број *VOQ* редова који је потребан за пријем свих мултикаст пакета је:

$$\sum_{k=1}^{32} \binom{N}{k} \tag{8.1.4}$$

што је даље једнако:

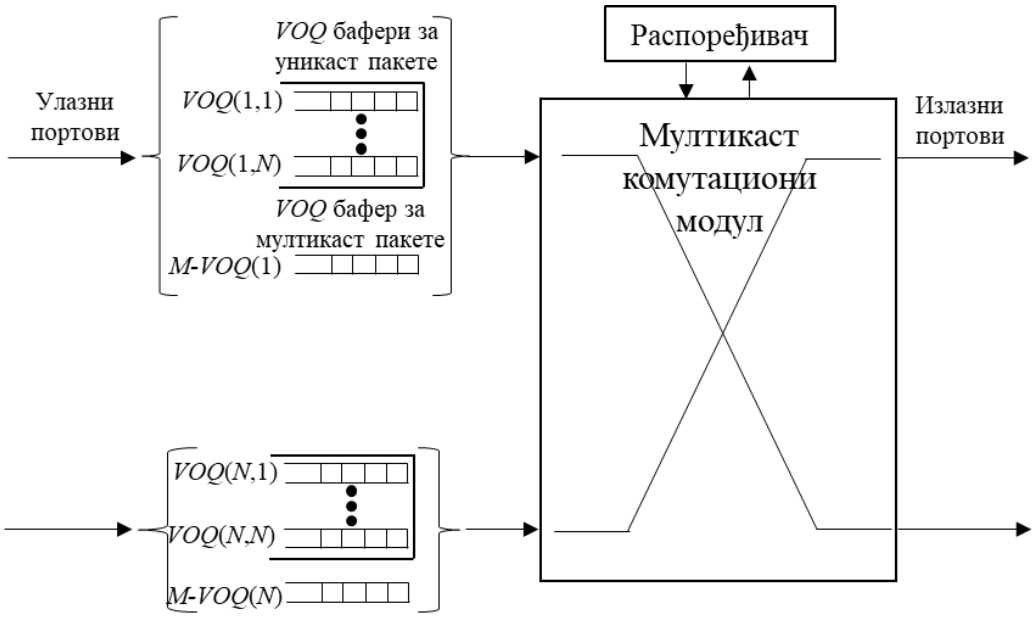
$$\sum_{k=1}^{32} \binom{N}{k} = \sum_{k=0}^{32} \binom{N}{k} - \binom{N}{0} = 2^N - 1 \tag{8.1.5}$$

Зато се у пракси, за смјештање мултикаст пакета, често користи *k VOQ* редова, гдје сваки *VOQ* ред одговара одређеној групи излазних портова. Када мултикаст пакет стигне на улазни порт он се смјешта у одговарајући мултикаст *VOQ* ред, зависно од тога на које излазне портове би требало да буду прослијеђен. Ако излазни портови на које треба прослиједити одређени мултикаст пакет одговарају различитим *VOQ* редовима, онда се по једна копија тог пакета смјешта у сваки од тих *VOQ* редова. На овај начин се и даље потенцијално увећава проток на улазним портовима, али је сада то повећање ограничено.

У [141] је предложено рјешење комутатора са баферима на улазним портовима прилагођено за управљање мултикаст саобраћајем. Архитектура овог комутатора је дата на слици 8.1.1. У овом комутатору се сви пакети смјештају у један *VOQ* ред. Приликом одабира пакета за слање користи се итеративни алгоритам. Алгоритам ради на сљедећи начин. Приликом слања дозвола, сви излазни портови у једном временском слоту у оквиру циклуса, дају предност једном, тачно одређеном улазном порту. Претпоставка је да улазни порт t **mod** N има предност у временском слоту t. Даље се одабир пакета за слање врши у три корака:

1) **Слање захтјева**. Сваки улазни порт шаље захтјев сваком излазном порту коме треба прослиједити *HoL* пакет.
2) **Слање дозвола**. Ако излазни порт прими више захтјева, бира онај који је први у *round-robin* редоследу, почињући од улазног порта с индексом t **mod** N.
3) **Прихватање дозвола**. Сваки улазни порт прихвата све примљене дозволе. Ако улазни порт не прими дозволе за све излазне портове на које треба прослиједити *HoL* пакет, онда се пакет шаље на оне излазне портове од којих је примљена дозвола (користи се мултикаст комутациони модул), а заглавље овог *HoL* пакета се ажурира тако да се ови портови бришу из листе излазних портова на које треба прослиједити овај пакет.

У временском слоту t+1, комутациони модул ће бити конфигурисан у складу са конфигурацијом која је добијена у временском слоту t. Главни недостатак овог комутатора, као и код осталих овог типа, јесте потреба да централна јединица, у сваком временском слоту, мора да прорачуна конфигурацију комутационог модула што ограничава скалабилност овог рјешења.

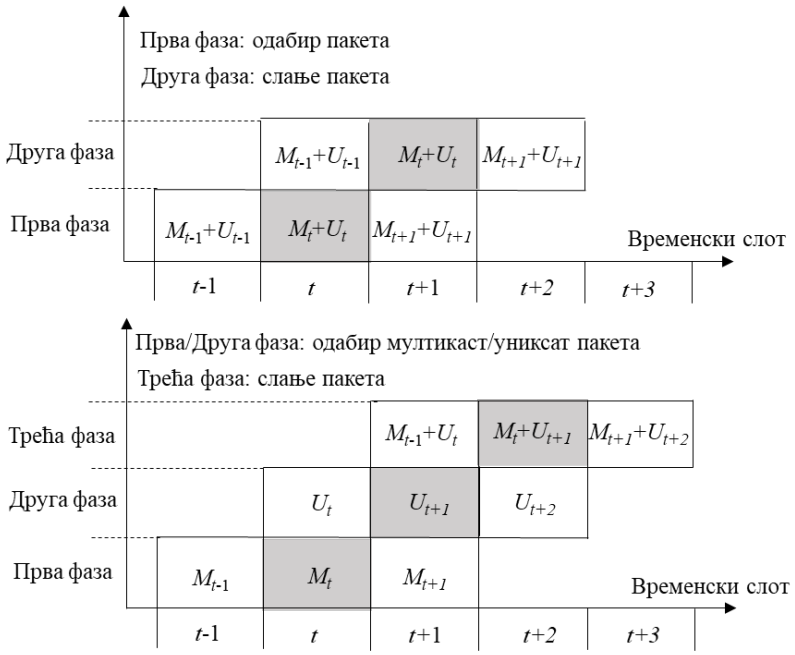


Слика 8.1.1. Архитектура комутатора са баферима на улазним портовима и итеративним алгоритмом за управљање мултикаст саобраћајем

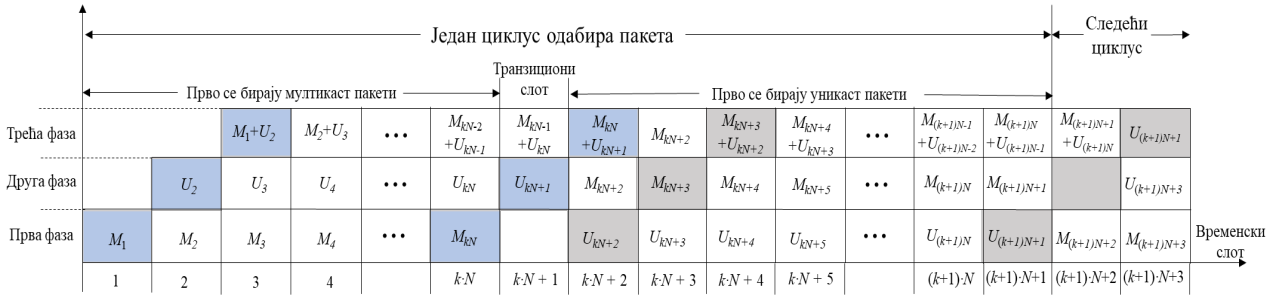
8.2. Пајплајн распоређивач за уникаст и мултикаст саобраћај за комутаторе са баферима на улазним портовима

У [142] је предложено рјешење за ефикасно управљање комбинованим уникаст и мултикаст саобраћајем у комутаторима са баферима на улазним портовима. Код овог рјешења, уникаст и мултикаст саобраћај су раздвојени на улазним портовима. Архитектура овог комутатора је иста као и код комутатора који је представљен у претходном потпоглављу, слика 8.1.1. Наиме, уникаст пакети се смјештају у једном од N *VOQ* редова колико их има на сваком од улазних портова. И овдје важи да су пакети које треба прослиједити на један излазни порт смјештени у истом *VOQ* реду. Такође, на сваком улазном порту постоји још један додатни, мултикаст *VOQ* ред, у који се смјештају мултикаст пакети.

Да би се ријешио проблем скалабилности, односно конфигурисања комутационог модула у сваком временском слоту, код овог рјешења се користи пајплајн техника. Зато овај комутатор користи и посебан тзв. пајплајн распоређивач, који врши адекватно упаривање улазних и излазних портова. Рад овог распоређивача се састоји од три корака, слика 8.2.1. У временском слоту *t*, овај распоређивач врши прорачун конфигурација на основу мултикаст захтјева. У сљедећем слоту, *t+1*, на ове, већ прорачунате конфигурације се додају нове конфигурације које су резултат прорачуна на основу уникаст захтјева, и које нису у конфликту са претходно добијеним конфигурацијама. Коначно, у трећем кораку, у временском слоту *t+2*, комутатор је конфигурисан у складу са претходно добијеним конфигурацијама, и врши се слање пакета. У оваквом режиму рада, минимално кашњење које уникаст пакет може имати је један слот, док је у случају мултикаст пакета два слота.



Слика 8.2.1. Примјер рада пајплајн распоређивача



Слика 8.2.2. Примјер рада распоређивача за мултикаст пакете

Јасно је да овакав начин рада даје предност мултикаст саобраћају. Да би се постигло фер опслуживање и уникаст и мултикаст саобраћаја, распоређивач наизмјенично даје предност уникаст па мултикаст саобраћају. Наиме, првих $k \cdot N$ слотова, распоређивач ради на горе описани начин, гдје се прво врши обрада мултикаст, па тек онда уникаст саобраћаја. Затим, у временском слоту $k \cdot N + 1$, распоређивач не разматра мултикаст пакете, већ само уникаст. Од временског слота $k \cdot N + 2$ распоређивач ради у нормалном режиму, с тим да се у

првом кораку врши обрада уникаст захтјева, док се у другом кораку врши обрада мултикаст захтјева. Тако ће у овом режиму рада уникаст саобраћај имати предност. Након што опет прође $k \cdot N$ слотова, распоређивач ће опет у првом кораку процесирати мултикаст саобраћај, а у другом уникаст саобраћај. Принцип рада распоређивача је дат на слици 8.2.2. На овај начин се обезбјеђује једнак третман и уникаст и мултикаст пакета.

У наставку ће бити објашњено како се врши упаривање улазних и излазних портова приликом обраде мултикаст и уникаст пакета, респективно. Како је већ речено, на сваком улазном порту постоји један мултикаст *VOQ* ред гдје се смјештају мултикаст пакети. Сваки улазни порт има предност при слању дозвола код свих излазних портова у једном, тачно одређеном, временском слоту у оквиру једног циклуса. Претпоставка је да улазни порт с индексом t **mod** N има предност у временском слоту t. Даље се одабир пакета за слање врши у три корака:

1) **Слање захтјева**. Сваки улазни порт шаље захтјев сваком излазном порту коме треба прослиједити *HoL* пакет.
2) **Слање дозвола**. Ако излазни порт прими више захтјева, бира онај који је први у *round-robin* редоследу, почињући од улазног порта с индексом t **mod** N.
3) **Прихватање дозвола**. Сваки улазни порт прихвата све примљене дозволе. Ако улазни порт не прими дозволе за све излазне портове на које треба прослиједити *HoL* пакет, онда се пакет шаље на оне излазне портове од којих су примљене дозволе, а заглавље овог *HoL* пакет се ажурира тако да се ови портови бришу из листе излазних портова на које треба прослиједити овај пакет.

Када се у временском слоту t, заврши са процесирањем мултикаст захтјева, у слоту $t+1$ се на сличан начин врши процесирање уникаст захтјева. У овом прорачуну су укључени само они улазни и излазни портови који нису упарени у претходном временском слоту. Приликом уникаст упаривања важи правило да у сваком временском слоту t, сваки улазни порт i при одабиру пакета за слање даје предност излазном порту j (важи и обратно, тј. да излазни порт j даје предност улазном порту i), при чему важи:

$$j = (i+t) \textbf{ mod } N \tag{8.2.1}$$

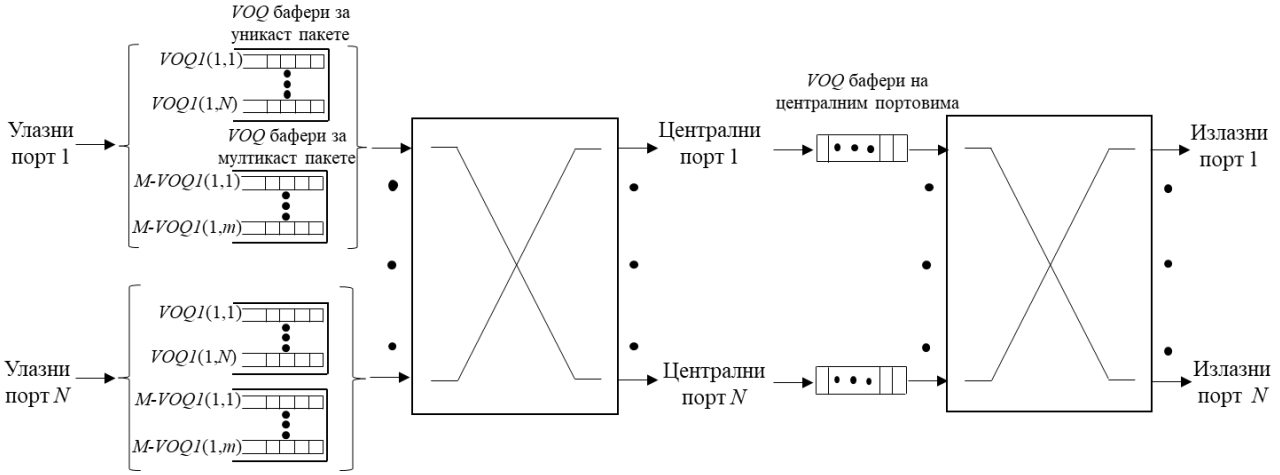
На основу (8.2.1) је јасно да сваки улазни порт даје предност сваком излазном порту у тачно једном временском слоту у току једног циклуса. Даље се упаривање улазних и излазних портова који су остали неупарени приликом обраде мултикаст пакета врши на сљедећи начин:

1) **Слање захтјева**. Ако неупарени улазни порт има уникаст пакет за слање на излазни порт који има предност у том временском слоту, онда њему шаље захтјев. У супротном, шаље захтјев излазном порту за кога има највише уникаст пакета.
2) **Слање дозвола**. Ако неупарени излазни порт прими захтјев од улазног порта који има предност у датом временском слоту, њему се шаље дозвола. У супротном, шаље се дозвола неком од улазних портова који су послали захтјев, и то случајним одабиром.

Иако овај комутатор користи пајплајн распоређивач, и даље је кључна мана, слично као и код претходног рјешења, потреба да се у сваком временском слоту врши прорачун конфигурација мултикасткомутационог модула. Као што је већ речено раније у поглављу, комутатори који користе мултикаст комутациони модул су комплекснији за хардверску имплементацију.

8.3. *Feedback Staggered Symmetry* (*FBSS*) комутатор за мултикаст саобраћај

Овај комутатор се заснива на *FBSS* комутатору који је представљен у одељку 3.5.1. Како овај комутатор постиже врло добре перформансе кад је у питању уникаст саобраћај, у [143] је предложено унапређење које омогућава ефикасно управљање и мултикаст саобраћајем. На слици 8.3.1 је дата архитектура овог комутатора. Прво ће бити дат кратак подсјетник на принцип рада *FBSS* комутатора.



Слика 8.3.1. Архитектура *FBSS* комутатора за мултикаст саобраћај

Конфигурације оба комутациона модула су детерминистичке и задовољавају тзв. *staggered symmetry and in order* својство. Наиме, улазни и централни портови су повезани по сљедећем шаблону:

$$j = (i+t) \bmod N \tag{8.3.1}$$

гдје је i редни број улазног порта, j редни број централног порта. Истовремено, повезивање централних и излазних портова комутатора је одређено по сљедећем принципу:

$$k = (j+N\text{-}1) \bmod N \tag{8.3.2}$$

гдје је j редни број централног порта, а k редни број излазног порта. Примјери приказа ових конфигурација су дати у потпоглављу 3.5 које описује *FBSS* комутатор за уникаст саобраћај (слика 3.5.2). У одељку 3.5.1 је показано да овакав шаблон повезивања улазних и централних, односно централних и излазних портова, омогућава ефикасан механизам за размјену информација о заузетости централних портова, између тих портова и одговарајућих улазних портова с којима су повезани. Наиме, ако је у временском слоту t, неки централни порт j повезан на излазни порт k, онда ће у слоту t+1, улазни порт k бити повезан са централним портом j. Тако, централни порт j може излазном порту k да пошаље свој вектор заузетости, који га даље прослеђује улазном порту k. Ово је могуће урадити на ефикасан начин, јер су улазни, централни и излазни порт с истим индексом реализовани на истој линијској картици. Такође, у одељку 3.5.1 је доказано да претходно описани шаблон конфигурисања комутационих модула гарантује да ће пакети истог тока на излазне портове стизати у правилном редоследу.

Кључно унапређење овог комутатора за подршку мултикаст саобраћају, у односу на *FBSS* комутатор, је у баферима на улазним портовима. Наиме, осим N *VOQ1* редова који служе за пријем уникаст пакета, на сваком улазном порту постоји још m мултикаст *VOQ1* редова који служе за пријем мултикаст пакета. Скуп од N излазних портова је подијељен у m,

једнаких подскупова који садрже по *N/m* излаза и који се међусобно не преклапају. Тако први подскуп садржи излазне портове 1,.., *N/m*-1, други подскуп садржи излазне портове *N/m,...,* $2 \cdot N / m - 1$ итд. Сваки од *m* додатних мултикаст *VOQ1* редова одговара једном од ових подскупова. Када мултикаст пакет стигне на улазни порт, прво се одређује на које излазне портове треба проследити тај пакет. Затим се тај пакет додаје у сваки мултикаст *VOQ1* ред који је задужен за бар један излазни порт на који треба проследити тај пакет. На примјер, претпоставимо 32x32 комутатор, који на сваком улазном порту има 4 мултикаст *VOQ1* реда. Ови мултикаст *VOQ1* одговарају сљедећим излазним портовима: *VOQ1*(1)={1, 2, 3, 4, 5, 6, 7, 8}, *VOQ1*(2)={9, 10, 11, 12, 13, 14, 15, 16}, *VOQ1*(3)={17, 18, 19 , 20, 21, 22, 23, 24}, *VOQ1*(4)={25, 26, 27, 28, 29, 30, 31, 32}. Рецимо да на неки улазни порт овог комутатора стигне мултикаст пакет који треба проследити на излазне портове 2, 5, 7, 20, 23. Пошто се овај скуп излазних портова преклапа са скуповима излазних портова за које су задужени редови *VOQ1*(1) и *VOQ1*(3), онда ће се једна копија овог пакета коју треба проследити на излазне портове 2, 5 и 7 смјестити у *VOQ1*(1), док ће се друга копија овог пакета коју треба проследити на излазне портове 20 и 23 смјестити у *VOQ1*(3). Број додатних мултикаст *VOQ1* редова је ствар компромиса. У случају кад би се користио само један мултикаст *VOQ1* ред, сви мултикаст пакети би се смјештали у исти мултикаст *VOQ1* ред. У том случају се сво умножавање пакета врши на централним портовима, али с друге стране у случају високог саобраћајног оптерећења до изражаја ће доћи проблем *HoL* блокаде. С друге стране, ако би се користило *N* мултикаст *VOQ1* редова, тада се комплетно умножавање пакета врши на улазним портовима. У овом случају, нема потребе за мултикаст *VOQ1* редовима јер су у њима сада смјештени уникаст пакети. Очигледно, зависно од броја мултикаст *VOQ1* редова умножавање пакета ће се доминантно вршити на улазним или централним портовима. Дакле, одабир броја мултикаст *VOQ1*редова представља компромис између *HoL* блокаде и повећања улазног протока.

Одабир пакета за слање се врши на сљедећи начин. Предност при одабиру имају пакети из мултикаст *VOQ1* редова. *HoL* пакети сваког мултикаст *VOQ1* реда садрже скуп излазних портова на које треба да буде прослијеђени. Када се улазни порт повеже с неким централним портом, бира се онај *HoL* пакет који има највеће преклапање са листом слободних излаза за које дати централни порт може да прими пакет. Када се пакет одабере, утврђује се на које све излазне портове дати пакет може да буде прослијеђен преко посматраног централног порта, и та информација се шаље централном порту. На основу ове информације, централни порт ажурира свој вектор заузетости. Такође, ти портови се затим бришу из скупа излазних портова на које тај пакет треба да буде прослијеђен. Ако више не постоји ни један излазни порт на који треба проследити дати *HoL* пакет, тај пакет се брише из одговарајућег мултикаст *VOQ1* реда. Када централни порт прими мултикаст пакет, он прави одговарајући број копија и смјешта их у оне *VOQ2* редове који одговарају излазним портовима на које треба да буду прослијеђени. Ако на датом улазном порту не постоји ниједан мултикаст пакет који се у датом временском слоту може послати централном порту с којим је тај улазни порт повезан, онда се бира неки уникаст пакет који може бити прослијеђен преко тог централног порта. Ако постоји више уникаст пакета који се могу послати, бира се пакет из оног *VOQ1* реда који има највише пакета.

9. МУЛТИКАСТ LB-BvN-GS

У овом поглављу ће бити представљен *Multicast Load-Balanced Birkhoff-von Neumann* комутатор са *Greedy Scheduling* алгоритмом (*M-LB-BvN-GS*) који је резултат рада на овој дисертацији. Ово рјешење представља унапређење *LB-BvN-GS* комутатора, које је представљено у поглављу 4, у циљу постизања постизања добрих перформанси при раду са комбинованим уникаст и мултикаст саобраћајем. Пошто је *M-LB-BvN-GS* комутатор врло сличан *LB-BvN-GS* комутатору, у наставку ће бити дат кратак опис архитектуре комутатора, а затим ће бити објашњена додата функционалност за ефикасно управљање мултикаст саобраћајем. Такође ће бити доказано да предложено рјешење *M-LB-BvN-GS* захтијева унутрашње убрзање 5 да би био неблокирајући за дозвољени мултикаст саобраћај. Такође, резултати представљени у сљедећем поглављу ће показати да се и без тог убрзања постижу одличне перформансе.

M-LB-BvN-GS комутатор се као и *LB-BvN-GS* комутатор састоји од два комутациона модула између којих се налазе централни портови. Шаблон конфигурисања је унапријед познат, и исти је за оба комутациона модула:

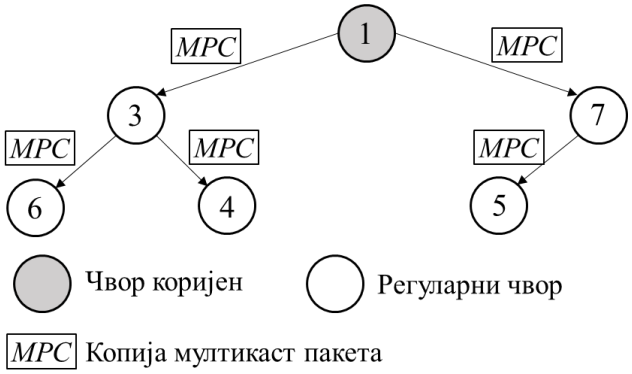
$$j = (i+1) \bmod N \tag{9.1}$$

односно

$$k = (j+1) \bmod N \tag{9.2}$$

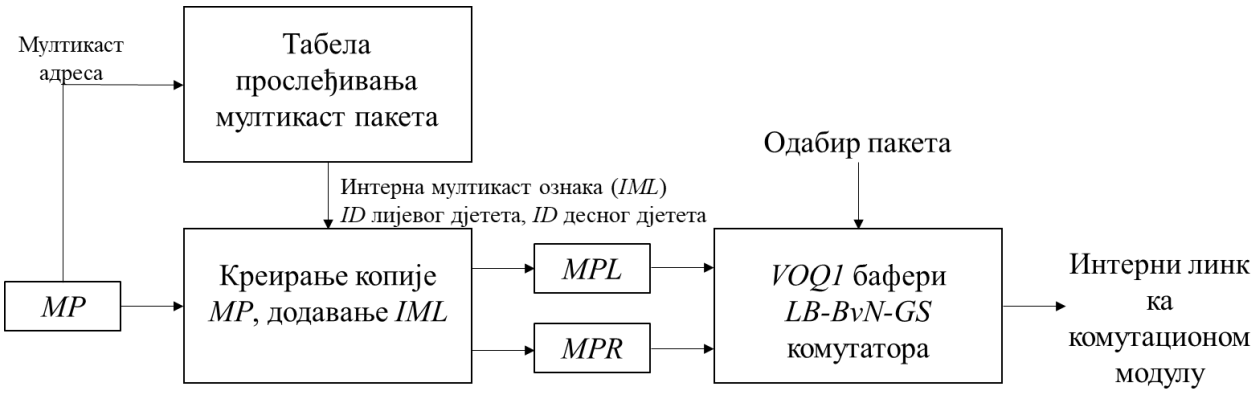
гдје су i улазни, j централни а k излазни портови комутатора. Како су шаблони конфигурисања оба комутациона модула исти и пошто се улазни, централни и излазни портови с истим индексом имплементирају на истој линијској картици, могуће је умјесто два користити само један физички комутациони модул, односно подржана је преклопљена архитектура. На саму *LB-BvN-GS* архитектуру, која је описана у поглављу 4.2, додаје се мултикаст контролер чиме се добија подршка за мултикаст саобраћај, односно *M-LB-BvN-GS* комутатор.

У наставку ће бити објашњен принцип рада мултикаст контролера који је додат *LB-BvN-GS* комутатору, како би на ефикасан начин управљао и мултикаст саобраћајем. Задатак мултикаст контролера је да за сваки мултикаст ток формира мултикаст стабло. Прије него буде објашњено како мултикаст контролер формира мултикаст стабло биће објашњена његова структура и разлог зашто се оно користи приликом прослеђивања мултикаст пакета. Под једним током мултикаст пакета или мултикаст током се подразумијевају пакети који стижу на један улазни порт и које треба прослиједити на исти скуп излазних портова. Да би се избјегло преоптерећење улазних портова на које долазе мултикаст пакети услед прављења копија пакета на улазним портовима, идеја је да се ово оптерећење распореди на све портове који припадају датом мултикаст току. За то се користи мултикаст стабло. Наиме, ово стабло се састоји од чвора коријена и регуларних чворова, при чему сваки чвор може да има највише два чвора потомка. Коријен стабла представља улазни порт на који стижу пакети неког мултикаст тока. Регуларни чворови су излазни портови на које треба прослиједити пакете посматраног мултикаст тока.



Слика 9.1. Примјер мултикаст стабла

На слици 9.1 је дат примјер мултикаст стабла једног мултикаст тока *F*. Индекс чвора одговара индексу порта комутатора. Мултикаст ток *F* представља пакете који стижу на улазни порт 1 и које треба прослиједити на излазне портове 3, 4, 5, 6 и 7. Улазни порт 1 представља коријен стабла, док излазни портови представљају регуларне чворове стабла. Са слике се види да ће улазни порт 1 прослиједити копије мултикаст пакета на излазне портове 3 и 7. Такође, порт 3 ће прослиједити копије пакета на портове 4 и 6, док ће порт 7 прослиједити копију мултикаст пакета на порт 5. На овај начин је оптерећење са улазног порта 1 дистрибуирано на више портова који припадају овом мултикаст току.

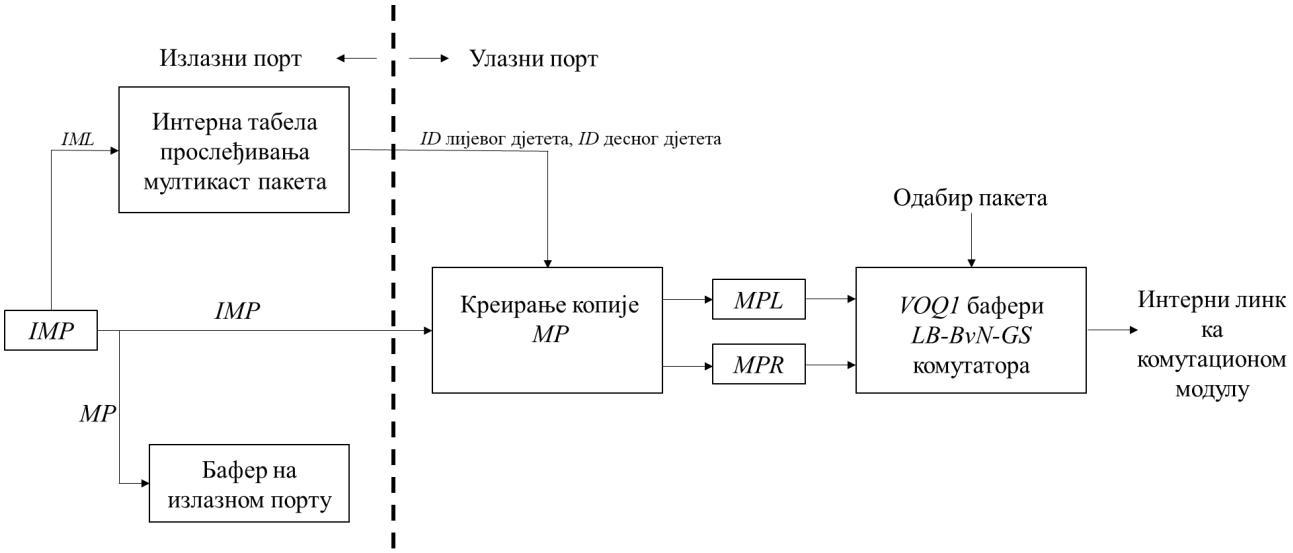


MP – Мултикаст пакет тока *F*, *MPL* – Мултикаст пакет лијевог дјетета (порт 3),
MPR – Мултикаст пакет десног дјетета (порт 7)

Слика 9.2. Прослеђивање мултикаст пакета на порту који је чвор коријен

На слици 9.2 је дата блок шема која објашњава дио који је додат на портовима који представљају коријен чвор у мултикаст стаблима. Наиме, када улазни порт прими неки мултикаст пакет, на основу његове мултикаст адресе се утврђује ком току тај пакет припада. Мултикаст адреса може бити *IP* адреса, *MPLS* ознака итд. Порт користи мултикаст адресу да изврши претраживање своје табеле прослеђивања мултикаст пакета. Резултат ове претраге су излазни портови на које треба прослиједити дати пакет, као и интерна мултикаст ознака коју треба додати овим пакетима. Ова ознака служи да би се направила разлика између мултикаст и уникаст пакета, као и да би се знало ком мултикаст току припада одређени мултикаст пакет. Излазни портови користе ове ознаке за даље прослеђивање пакета кроз мултикаст стабло. Ако чвор потомак не постоји, онда се из табеле прослеђивања мултикаст пакета враћа вриједност *NULL*. Дакле, у претходном примјеру са слике 9.1, када улазни порт 1 прими пакет тока *F* праве се двије копије овог пакета за излазне портове 3 и 7 којима се

додају одговарајуће интерне мултикаст ознаке и ови пакети се додају у одговарајуће *VOQ1* редове, у којима су смјештени и уникаст пакети које треба прослиједити на излазне портове 3 и 7. Даље се прослеђивање пакета врши на исти начин као код *LB-BvN-GS* комутатора, односно од тог момента се користи *LB-BvN-GS* комутатор.



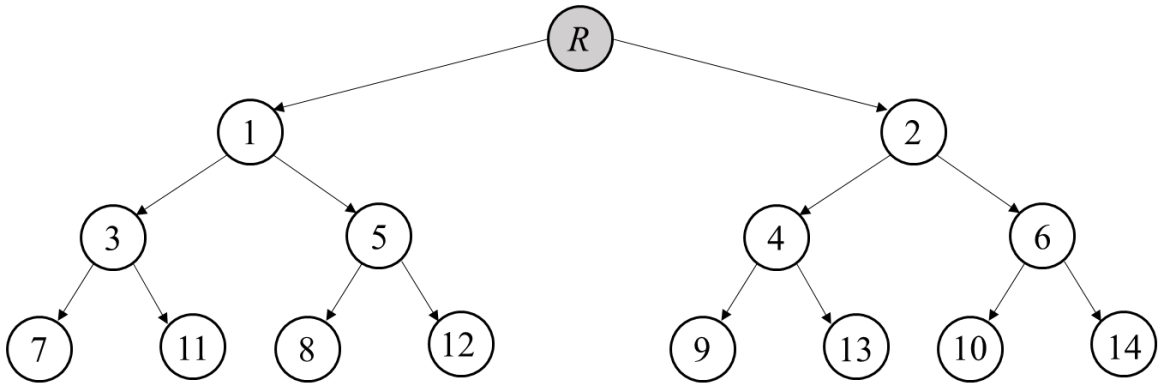
IMP – Мултикаст пакет тока *F* са додатом *IML*, *MP* – Мултикаст пакет тока *F* без додатог *IML*
MPL – Мултикаст пакет лијевог дјетета (порт 3), *MPR* – Мултикаст пакет десног дјетета (порт 7)

Слика 9.3. Прослеђивање мултикаст пакета на порту који је регуларан чвор

На слици 9.3 је објашњено како се врши обрада мултикаст пакета када дођу на неки порт који представља регуларни чвор у мултикаст стаблу. Наиме, када мултикаст пакет дође на порт који представља регуларни чвор у мултикаст стаблу, његова копија се шаље у бафер на излазном порту и даље у мрежу. С друге стране, издваја се његова интерна мултикаст ознака која се користи, као што је већ напоменуто, за прослеђивање кроз мултикаст стабло. Ова ознака се користи за претраживање интерне табеле прослеђивања мултикаст пакета. У овој табели, за сваку интерну мултикаст ознаку постоји информација о чворовима потомцима у датом мултикаст стаблу. Као резултат ове претраге добијају се портови на које треба прослиједити овај пакет. Ако потомци постоје, праве се копије датог пакета и оне се даље смјештају у одговарајуће *VOQ1* редове и од тог момента се користи *LB-BvN-GS* комутатор. Како је већ раније поменуто, улазни, централни и излазни порт с истим индексом су имплементирани на истој линијској картици, тако да прослеђивање пакета са излазног на улазни порт с истим индексом је релативно једноставна операција. У овом конкретном примјеру, пакети тока *F* са порта 3 ће бити даље прослијеђени на излазне портове 4 и 6. На порту 7 ће бити направљена само једна копија која ће бити прослијеђена на излазни порт 5, јер чвор 7 има само једног потомка. Пошто чворови 4, 5 и 6 немају потомке, они ће примљене пакете прослиједити у мрежу и неће правити додатне копије. На овом примјеру се види како се уникаст *LB-BvN-GS* комутатор, уз помоћ додатка у виду мултикаст контролера може ефикасно користити и за прослеђивање мултикаст саобраћаја.

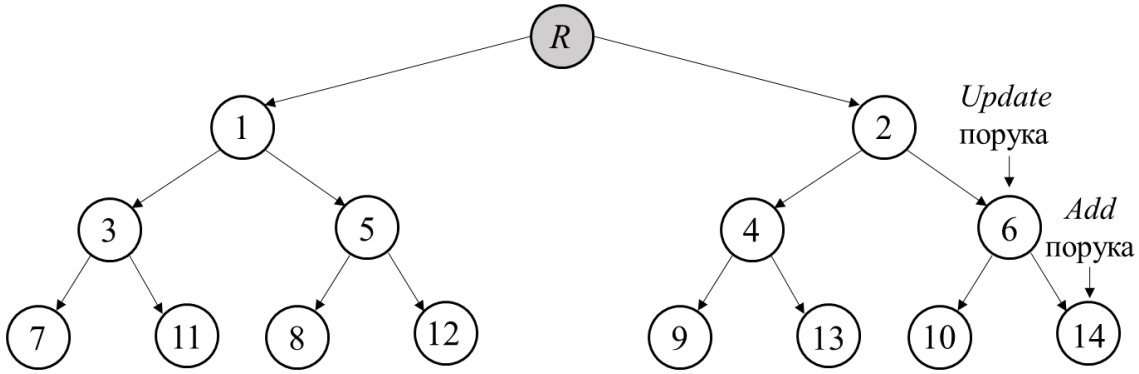
За формирање мултикаст стабала у мрежи се обично користе протоколи попут *PIM* протокола, а на основу тих мултикаст стабала у мрежи се креирају и мултикаст стабла у рутерима. Прецизније говорећи, на основу рада ових протокола, рутери могу, за сваки мултикаст ток *F*, да одреде који улазни порт прима пакете тог тока, и на које излазне портове треба прослиједити те пакете. Поруке протокола се обрађују у контролној равни, па се зато и

управљачки део мултикаст контролера имплементира у контролној равни. Ово својство омогућава да се предложено рјешење може прилагодити и за концепт софтверски дефинисаних мрежа (*SDN*). Задатак управљачког дела мултикаст контролера је креирање мултикаст стабла, и слање одговарајућих конфигурационих порука улазним и излазним портовима рутера. Мултикаст стабло се инкрементално ажурира, при чему се сваки нови чвор додаје у складу са *лијеви потомак први* правилом. Ово значи да се чворови у сљедећем нивоу стабла прво додају на мјеста лијевих потомака чворова у претходном нивоу. На слици 9.4 је дат редослед додавања нових чворова у мултикаст стабло. Напомињемо, да број чвора на слици 9.4 не представља индекс излазног порта, већ само одређује редослед додавања неког излазног порта мултикаст тока у мултикаст стабло. Тако се, са слике 9.4, може видјети да се прво додаје излазни порт на мјесто лијевог потомка улазног порта, а онда се сљедећи излазни порт тог мултикаст тока додаје на мјесто десног потомка улазног порта. У сљедећој итерацији, трећи излазни порт који припада посматраном мултикаст току се додаје на мјесто лијевог потомка, првог излазног порта који је додат. Четврти излазни порт се додаје на мјесто лијевог потомка другог излазног порта мултикаст тока. Онда се сљедећа два излазна порта додају на мјеста десних потомка првог и другог додатог порта. Исти принцип важи код даљег додавања портова у мултикаст стабло. На овај начин се добија избалансирано мултикаст стабло најмање могуће дубине. Ово је важно јер се на тај начин минимизује разлика кашњења између копија истог пакета који су прослијеђени на различите излазне портове.



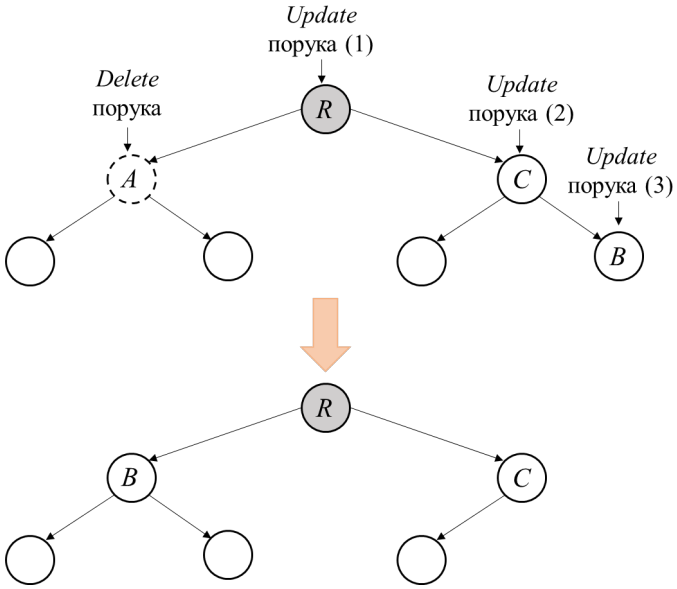
Слика 9.4. Примјер мултикаст стабла

Када се неки чвор избрише из стабла, онда се последњи додати чвор додаје на његово мјесто, осим ако није дошло до брисања управо последњег додатог чвора. На овај начин се осигурава да стабло у сваком моменту задржава компактну структуру. Слике 9.5 и 9.6 приказују конфигурационе поруке које се шаљу од стране управљачког дела мултикаст контролера у најтипичнијим сценаријима. Конфигурационе поруке које се шаљу улазним портовима (чворови коријени) служе за додавање, ажурирање и брисање чворова у табели прослеђивања на тим улазним портовима. Ове поруке садрже информације о интерним мултикаст ознакама, као и идентификације излазних портова који представљају лијевог и десног потомка чвора коријена. Конфигурационе поруке које се шаљу излазним портовима служе за додавање, ажурирање и брисање чворова у интерним табелама прослеђивања на тим излазним портовима. Ове поруке садрже информације о интерној мултикаст ознаци, као и идентификације излазних портова који представљају лијевог и десног потомка одговарајућег чвора у стаблу.



Слика 9.5. Примјер додавања новог чвора у стабло

На слици 9.5 је дат примјер конфигурационих порука које се шаљу када се додаје нови чвор у стаблу. У овом конкретном примјеру се шаљу двије поруке. Једна је *update* порука која се шаље родитељском чвору, која садржи идентификацију додатог чвора који је сад потомак тога чвора. Те информације се чувају у интерној табели прослеђивања мултикаст пакета тог чвора. Ако је родитељски чвор истовремено и коријен стабла, онда се те информације чувају у мултикаст табели прослеђивања. У конкретном примјеру, интерна табела прослеђивања мултикаст пакета на излазном порту који одговара позицији 6 у мултикаст стаблу, се ажурира тако да се на мјесту десног потомка тог чвора додаје идентификација излазног порта који одговара позицији 14 у мултикаст стаблу. Друга конфигурациона порука се шаље интерној табели прослеђивања мултикаст пакета на излазном порту који одговара позицији 14 у мултикаст стаблу. Пошто овај чвор нема синова, вриједности оба показивања су постављене на *NULL*.



Слика 9.6. Примјер ажурирања мултикаст стабла кад се брише чвор

Слика 9.6 показује најкомплекснији случај, када се брише неки постојећи чвор, који није последњи додати чвор. У овом случају је потребно слање четири конфигурационе поруке. На излазни порт који се брише из стабла шаље се *delete* порука, тако да се одговарајуће информације избришу из интерне табеле прослеђивања мултикаст пакета тог излазног порта. Последњи додати чвор у овом примјеру је чвор *B*. Овај чвор треба да се

премјести на мјесто избрисаног чвора (чвор *A* у датом примјеру). Ово захтијева да се и чвору *B* пошаље *update* порука којом се ажурира његова позиција у мултикаст стаблу. Такође, неопходно је послати *update* поруке и родитељским чворовима чвора *B* (чвор *C* у датом примјеру), односно избрисаног чвора *A* (чвор *R* у датом примјеру) како би се обезбиједило правилно прослеђивање мултикаст пакета кроз бинарно мултикаст стабло. Ова два родитељска чвора (чворови *R* и *C*) морају да ажурирају своје интерне табеле прослеђивања мултикаст пакета услед брисања чвора *A* и промјене позиције чвора *B*. Стога мултикаст контролер шаље *update* поруку чвору *B* како би његови чворови потомци сада постали потомци чворови избрисаног чвора *A*. Такође, *update* порука се шаље чвору *R*, тако да његов лијеви потомак буде чвор *B* (умјесто избрисаног чвора *A*), као и чвору *C* да би се његов десни потомак ажурирао да буде *NULL*, јер је чвор *B* премјештен на другу позицију.

Дакле, управљачки део мултикаст контролера води рачуна о свим бинарним мултикаст стаблима и ажурира структуре постојећих стабала на бази порука мултикаст протокола (нпр. *PIM* протокол) које се размењују с другим рутерима. На бази ових информација, мултикаст контролер креира одговарајуће конфигурационе поруке и шаље их равни података рутера како би се адекватно ажурирале табеле прослеђивања мултикаст пакета на одговарајућим портовима рутера.

Иако предложени мултикаст комутатор *M-LB-BvN-GS* има веома добре перформансе када се не користи унутрашње убрзање, у наставку је изведен доказ да је *M-LB-BvN-GS* комутатор неблокирајући за сваки дозвољени модел саобраћаја када се користи унутрашње убрзање 5. У наставку је дат доказ. Посматра се комутатор величине *N*x*N*, гдје је *N* број улазних/излазних портова. Пакети су фиксне величине која је једнака трајању једног слота. Пошто се посматра дозвољени шаблон саобраћаја, постоји $m \in Z^+$ тако да је саобраћај у интервалу од $m \cdot N$ слотова дозвољен, при чему Z^+ представља скуп позитивних цијелих бројева (≥ 1). То значи да у току овог интервала за сваки излазни порт може стићи највише $m \cdot N$ пакета. Такође, на сваки улазни порт у току овог интервала може доћи највише $m \cdot N$ пакета. Посматра се неки улазни порт *i* и пакет који треба прослиједити са тог улазног порта на неки излазни порт *k*. Најгори случај је када:

3) сви пакети који стижу на улазни порт *i* су мултикаст пакети, које треба прослиједити на бар два излазна порта
4) сви пакети који стижу на излазни порт *i* су мултикаст пакети за које је порт *i* регуларни чвор у мултикаст стаблу који има оба чвора потомка.

У најгорем случају, посматрани пакет мора да сачека да сви мултикаст пакети пристигли на улазни порт и излазни порт *i* буду прослијеђени. На основу 1) се закључује да у најгорем случају постоји $2 \cdot m \cdot N$ пакета за које је улазни порт *i,* чвор коријен. Такође, постоји $2 \cdot m \cdot N$ пакета које порт *i* као регуларни чвор у мултикаст стаблу треба да прослиједи. У најгорем случају, постоји и $m \cdot N$ пакета које треба прослиједити на излазни порт *k*. Тако да посматрани пакет мора да сачека прослеђивање $4 \cdot m \cdot N-1$ пакета који се шаљу са улазног порта *i,* и прослеђивање $m \cdot N-1$ пакета на излазни порт *k* са други улазних портова прије него што може да буде послат. На основу претходног се закључује да посматрани пакет мора да сачека прослеђивање $5 \cdot m \cdot N-2$ пакета, тако да је неопходно бар убрзање 5 да би се прослиједили сви пакети у периоду од $m \cdot N$ слотова.

10. ПОРЕЂЕЊЕ СА ПОСТОЈЕЋИМ МУЛТИКАСТ РЈЕШЕЊИМА

У овом поглављу се анализирају перформансе комутатора који су представљени у поглављима 8 и 9, приликом процесирања комбинованог уникаст и мултикаст саобраћаја. У наредним анализама се посматра средње кашњење пакета, при чему се кашњење мери од момента када пакет дође на улазни порт до момента када напусти комутатор. Подразумијева се да сви пакети напуштају комутатор у правилном распореду. Приказани су резултати за 32x32 и 64x64 комутаторе, а однос перформанси остаје сличан и за комутаторе других величина [144]. Такође, посматра се дозвољени модел саобраћаја, тј. важи сљедеће:

$$\sum_{j=1}^{N} \lambda_{i,j} \leq 1 \quad , \qquad j=1,2,\ldots, N \tag{10.1}$$

и

$$\sum_{j=1}^{N} \lambda_{i,j} \leq 1, \qquad i=1,2,\ldots, N \tag{10.2}$$

гдје је $\lambda_{i,j}$ нормализована количина саобраћаја између улазног порта i и излазног порта j. Ово значи да у случају највећег оптерећења, у току једног циклуса од N слотова (гдје је N број улазних/излазних портова) на сваки улазни порт може да дође највише N пакета. Исто важи и за излазне портове; у току једног циклуса од N слотова највише N пакета који су стигли у току овог циклуса треба прослиједити на један излазни порт. Овдје треба нагласити, да када на неки улазни порт стигне мултикаст пакет којег треба прослиједити на x излазних портова, рачунамо као да је за наведене излазне портове стигао пакет. У преводу, то значи да посматрано са становишта свих излаза, укупан број пакета може бити већи од N у једном слоту због мултикаст принципа. Али, принцип дозвољеног модела саобраћаја гарантује и даље да ниједан излаз није преоптерећен тј. за разлику од уникаст модела, овде постоје празнине на улазним портовима када се не прима пакет. То ће бити јасно у наставку из параметара симулације који се односе на вероватноћу појаве пакета на улазу која ће бити мања него код уникаст симулација приказаним раније у тези.

M-LB-BvN-GS комутатор је поређен са: комутатором са баферима на улазним портовима и итеративним алгоритмом за управљање мултикаст саобраћајем (*M-IQ-1*), комутатором са баферима на улазним портовима и *pipeline* распоређивачем за уникаст и мултикаст саобраћај (*M-IQ-2*), *Feedback Staggered Symmetry* (*M-FBSS*) комутатором за мултикаст саобраћај. Наведена два комутатора са баферима на улазним портовима су одабрани јер постижу најбоље перфромансе у тој категорији комутатора. *M-FBSS* комутатор је одабран јер постиже најбоље перформансе међу комутаторима који су развијени на основу *LB-BvN* комутатора. У анализама су одабрана две варијанте овог комутатора: са 2 *VOQ1* реда за смјештање мултикаст пакета (*M-FBSS2*) и са 16 *VOQ1* редова за смјештање мултикаст пакета (*M-FBSS16*). У поглављу 8.3 је већ речено да је број *VOQ1* редова за мултикаст пакете ствар компромиса, и од тога зависи да ли ће се умножавање пакета доминантно вршити на

улазним или централним портовима. Већи број *VOQ1* редова повећава хардверску комплексност, али и перформансе комутатора.

Комутатори су тестирани за три различита модела саобраћаја: Бернулијев униформни комбиновани саобраћај (*BUMT*), Бернулијев униформни мултикаст саобраћај (*BUMuT*) и *bursty* комбиновани саобраћај (*BMT*). *BUMT* саобраћај се симулира на сљедећи начин. λ је вјероватноћа да у посматраном временском слоту пакет стиже на улазни порт, P_m је вјероватноћа да је пакет мултикаст, P_u је вероватноћа да је пакет уникаст. Јасно је да важи:

$$P_u+P_m=1 \tag{10.3}$$

Такође, F_{min} представља минимални број излазних портова на које мултикаст пакет треба да буде прослијеђен, док је F_{max} максималан број излазних портова на које мултикаст пакет треба да буде прослијеђен. На почетку сваког временског слота се одређује да ли на посматрани улазни порт стиже пакет. Ако пакет стиже, онда се одређује да ли је пакет уникаст или мултикаст. Ако је у питању уникаст пакет, онда се на случајан начин одабира један од N излазних портова на који треба да буде прослијеђен. У супротном, ако се ради о мултикаст пакету, онда се на случајан начин бира неки цијели број x између F_{min} и F_{max} који представља број излазних портова на које мултикаст пакет треба да буде прослијеђен. Затим се, на случајан начин, бира x различитих излазних портова на које тај пакет треба да буде прослијеђен. На основу овога се закључује да је просјечно оптерећење излазних портова:

$$\lambda \cdot (P_u + P_m \cdot \frac{(F_{\min} + F_{\max})}{2}) \tag{10.4}$$

BUMuT модел саобраћаја заправо представља посебан тип *BUMT* саобраћаја када је сав саобраћај мултикаст, тј. P_m=1 и P_u=0. Коначно, *BMT* модел саобраћаја користи *ON/OFF* модел за генерисање *bursty* саобраћаја. Сваки улазни порт може да буде у *ON* или *OFF* стању. Ако је улазни порт у *ON* стању, он прима пакет. Обратно, ако је улазни порт у *OFF* стању, тада пакети не стижу на тај улазни порт. Овдје се дефинишу сљедећи параметри: s – просјечна величина *burst*-а, P_u – вјероватноћа да је пакет уникаст, P_m – вјероватноћа да је пакет мултикаст, F_{min} - минималан број излазних портова на које мултикаст пакет треба да буде прослијеђен, F_{max} - максималан број излазних портова на које мултикаст пакет треба да буде прослијеђен. Вјероватноћа да улазни порт прећи из *OFF* у *ON* стање је:

$$\frac{P}{s \cdot (1-P)} \tag{10.5}$$

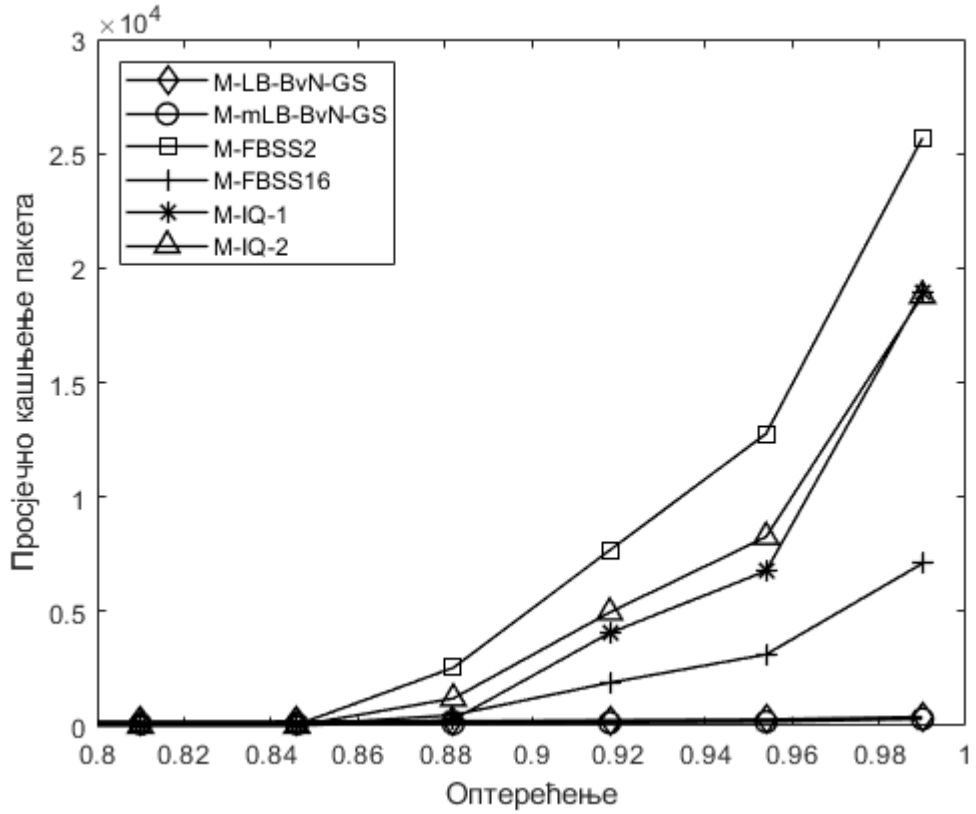
док је вјероватноћа да улазни порт прећи из *ON* у *OFF* стање:

$$\frac{1}{s} \tag{10.6}$$

На почетку су сви улазни портови у *OFF* стању. Када улазни порт прећи у *ON* стање, тада се прво одређује да ли су пакети у оквиру тог *burst*-а уникаст или мултикаст типа. Ако је пакет уникаст, на случајан начин се одређује на које излазни порт треба прослиједити тај пакет. Такође, сви пакети из тог *burst*-а се прослеђују на исти излазни порт. Уколико је у питању мултикаст пакет, онда је процес генерисања пакета исти као што је то описано за *BUMT* модел саобраћаја. Такође, и овдје важи да све пакете у оквиру једног *burst*-а треба прослиједити на исти скуп излазних портова. У симулацијама чији су резултати представљени у наставку текста претпостављено је да је величина *burst*-а s=30 [142,143], као и да је број излазних портова на који се шаље неки мултикаст пакет између F_{min}=2 и F_{max}=32.

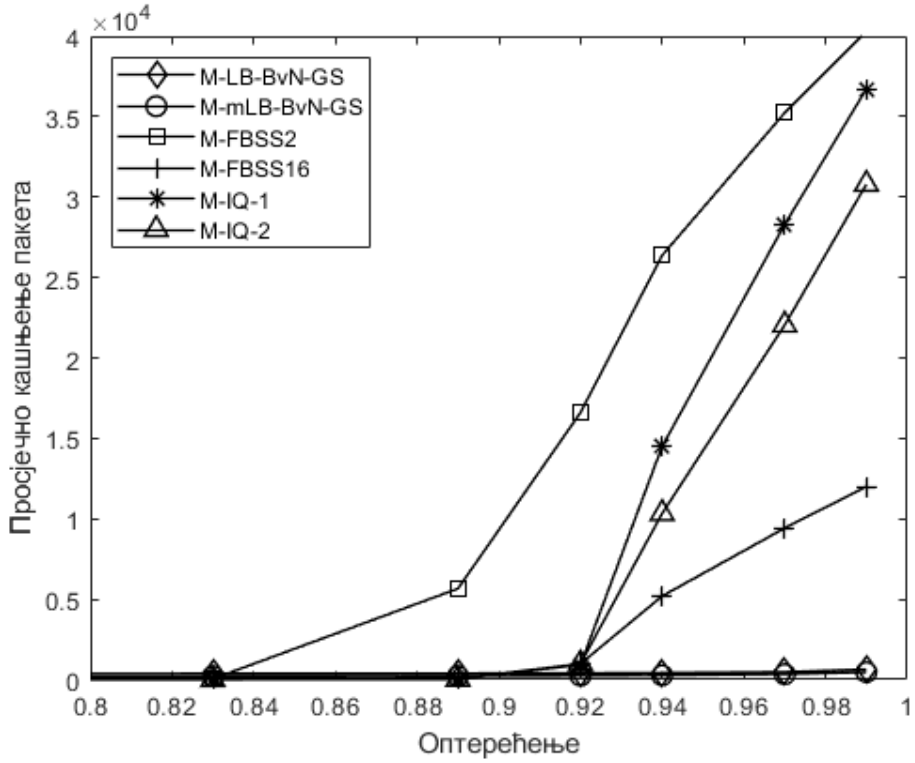
Разматрана су два сценарија: један у коме је количина уникаст и мултикаст саобраћаја иста (P_u=50% и P_m=50%) и други у коме је претпостављено да је сав саобраћај мултикаст (P_m=100%).

На слици 10.1 су представљени резултати за 32x32 комутаторе за Бернулијев униформни саобраћај при чему је 50% саобраћаја уникаст, а 50% мултикаст. На слици су дати резултати за велика оптерећења јер се разлике између комутатора примјећују тек при оптерећењима већим од 90%. У овом сценарију су сва рјешења стабилна при чему *M-LB-BvN-GS* и *M-mLB-BvN-GS* постижу најмање средње кашњење при највећем оптерећењу. Такође, уочава се да *M-FBSS2* има лошије перформансе у односу на *M-FBSS16* због мањег броја мултикаст *VOQ* редова. Услед тога *HoL* блокада има значајан утицај на перформансе. С друге стране, *M-FBSS16* креира већи број копија пакета и захтијева или брже меморије ради чувања свих копија у току једног слота или више физичких меморија. *M-IQ-1* и *M-IQ-2* комутатори постижу добре перформансе осим при највећем оптерећењу.

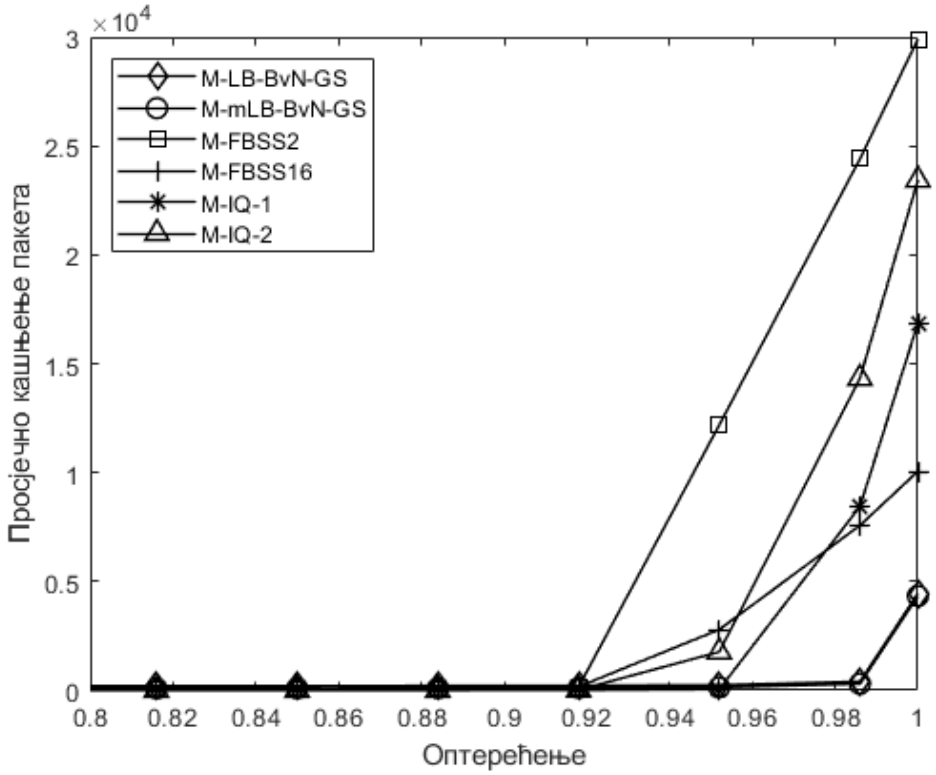


Слика 10.1. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај са 50% уникаст и 50% мултикаст саобраћаја за 32x32 комутаторе

На слици 9.2 су дати резултати за исти саобраћајни сценарио за 64x64 комутаторе. У принципу, закључци остају исти као и за 32x32 комутаторе, с тиме што је уочљивија разлика између *M-IQ-1* и *M-IQ-2* комутатора при највећим оптерећењима. Дакле, рјешења предложена у овој дисератцији постижу најбоље перформансе за различите величине комутатора.

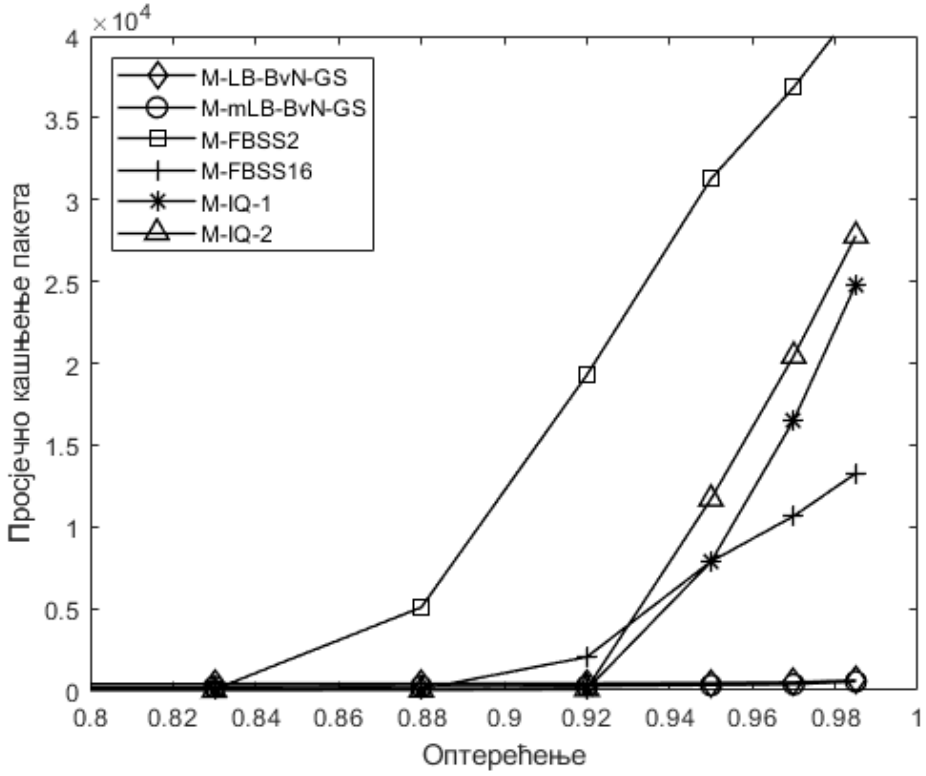


Слика 10.2. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај са 50% уникаст и 50% мултикаст саобраћаја за 64x64 комутаторе



Слика 10.3. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај са 100% мултикаст саобраћаја за 32x32 комутаторе

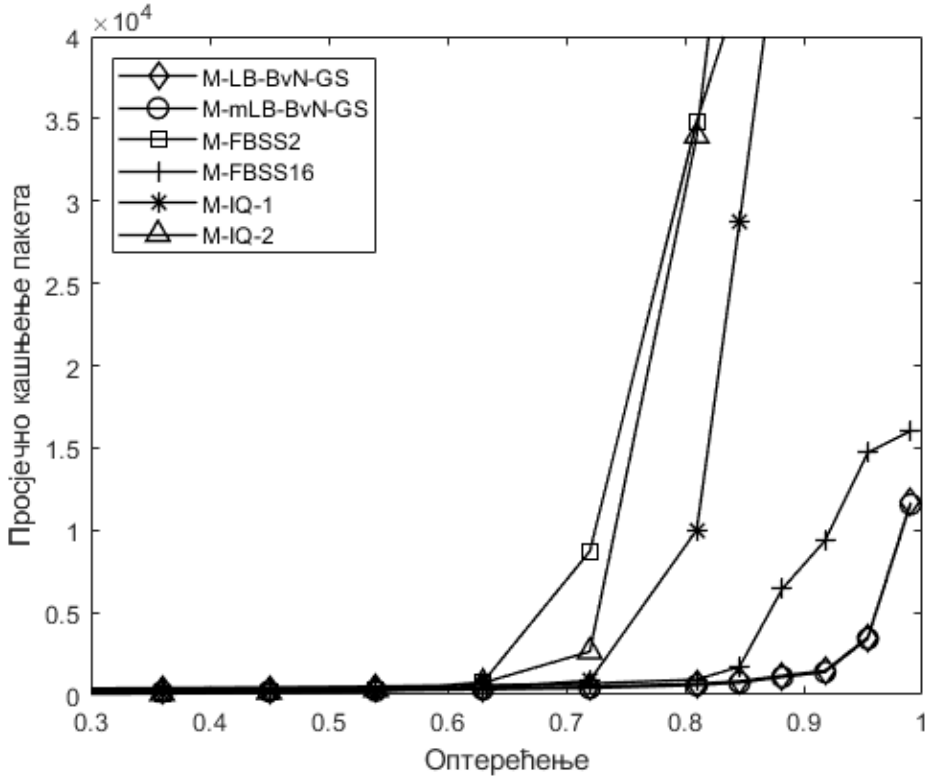
На слици 10.3 су представљени резултати за Бернулијев униформни саобраћај при чему је сав саобраћај мултикаст. На слици су дати резултати за велика оптерећења јер се разлике између комутатора примјећују тек при оптерећењима већим од 90%. Закључци су слични као у сценарију са Бернулијевим унфиформним моделом саобраћаја гдје је 50% саобраћаја уникаст, а 50% мултикаст. Перформансе *M-FBSS* комутатора зависе од броја мултикаст *VOQ* редова, док комутатори са баферима на улазним портовима постижу добре перформансе осим при највећем оптерећењу. И у овом сценарију *M-LB-BvN-GS* и *M-mLB-BvN-GS* постижу најбоље перформансе под великим оптерећењем.



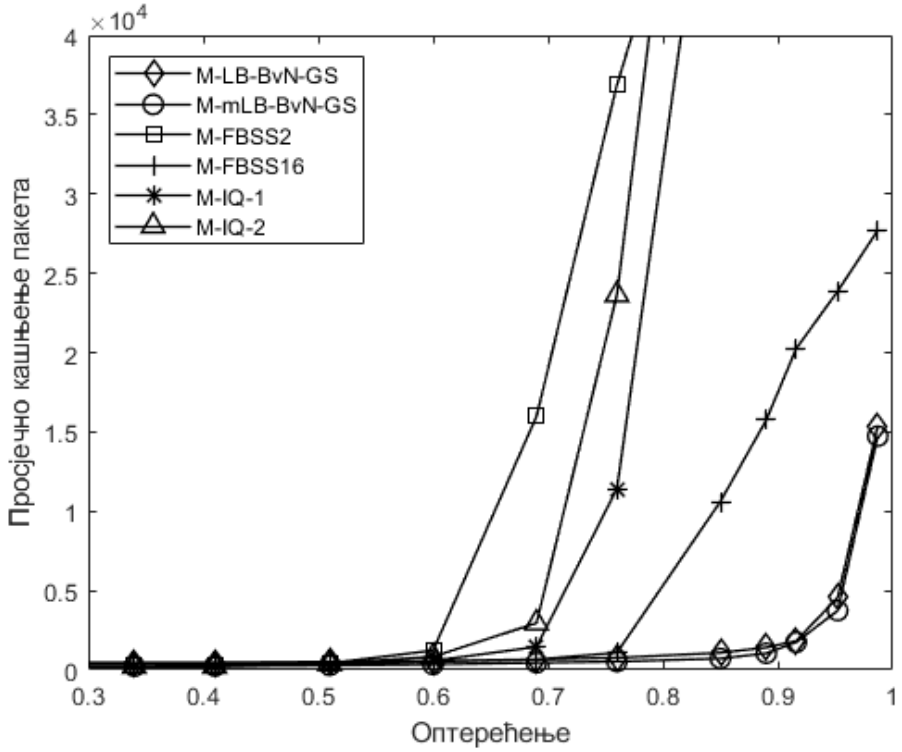
Слика 10.4. Зависност средњег кашњења пакета од оптерећења за Бернулијев униформни саобраћај са 100% мултикаст саобраћаја за 64x64 комутаторе

На слици 10.4 су дати резултати за исти саобраћајни сценарио за 64x64 комутаторе. Као што се види са слике, односи између различитих рјешења су слична као и у случају 32x32 комутатора. И у овом сценарију *M-LB-BvN-GS* и *M-mLB-BvN-GS* постижу најбоље перформансе у односу на друга рјешења.

На слици 10.5 су представљени резултати за униформни *bursty* саобраћај при чему је 50% саобраћаја уникаст, а 50% мултикаст. Овај сценарио представља веће оптерећење за комутатор због спорадичне природе саобраћаја. Зато и сва рјешења имају веће средње кашњење него у сценарију са Бернулијевим униформним саобраћајем. *M-IQ-2* и *M-FBSS2* први постају нестабални, док је *M-IQ-1* комутатор нешто бољи у односу на њих. *M-LB-BvN-GS*, *M-mLB-BvN-GS* и *M-FBSS16* су стабилни чак и при највећим оптерећењима при чему опет *M-LB-BvN-GS* и *M-mLB-BvN-GS* постижу најмање средње кашњење при највећем оптерећењу.



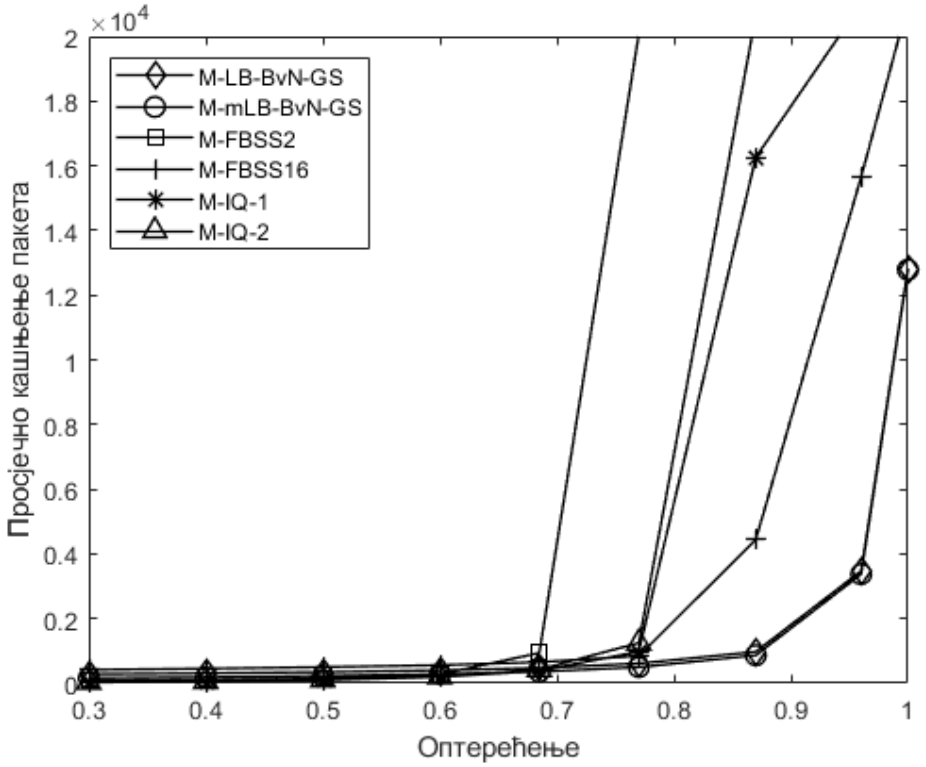
Слика 10.5. Зависност средњег кашњења пакета од оптерећења за униформни *bursty* саобраћај са 50% уникаст и 50% мултикаст саобраћаја за 32x32 комутаторе



Слика 10.6. Зависност средњег кашњења пакета од оптерећења за униформни *bursty* саобраћај са 50% уникаст и 50% мултикаст саобраћаја за 64x64 комутаторе

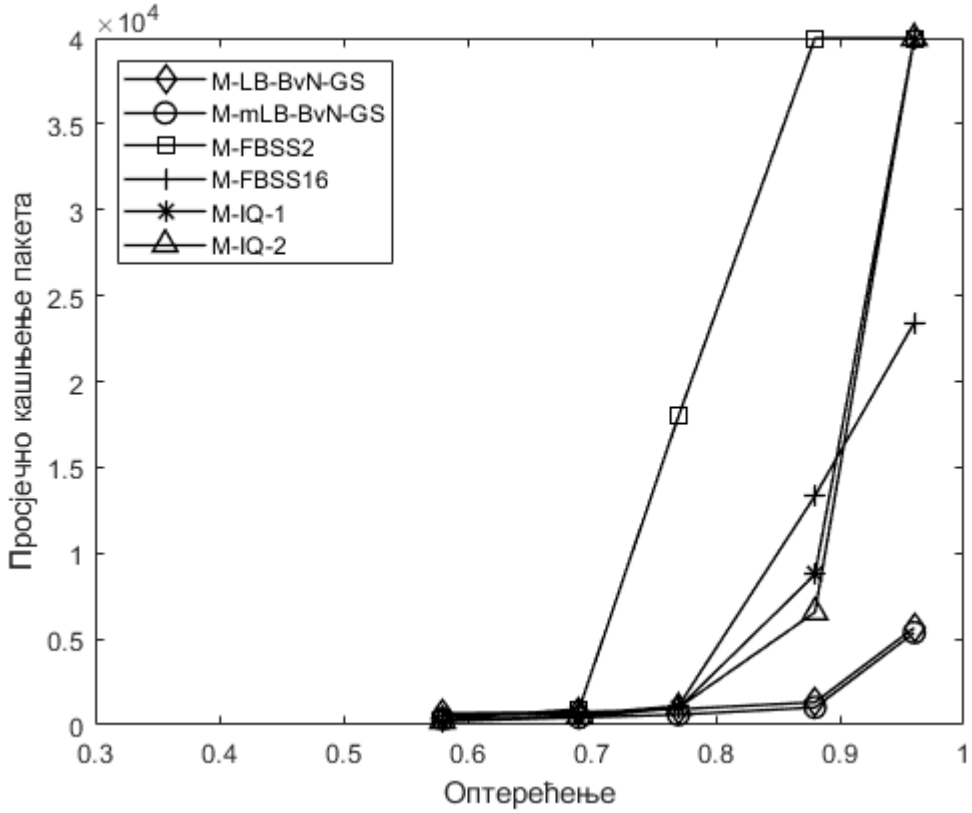
На слици 10.6 су дати резултати за исти саобраћајни сценарио за 64x64 комутаторе. Као и у претходним случајевима, закључци остају исти као и за исти сценарио за 32x32 комутаторе.

На слици 10.7 су представљени резултати за униформни *bursty* саобраћај при чему је сав саобраћај мултикаст. И овдје закључци остају слични као и у претходном сценарију. *M-IQ-2* и *M-FBSS2* први постају нестаблни, док је *M-IQ-1* комутатор тек нешто бољи. *M-FBSS16* је врло стабилан осим за највећа оптерћења. Коначно, *M-LB-BvN-GS* и *M-mLB-BvN-GS* опет постижу најмање средње кашњење при највећем оптерећењу.



Слика 10.7. Зависност средњег кашњења пакета од оптерећења за униформни *bursty* саобраћај са 100% мултикаст саобраћаја за 32x32 комутаторе

На слици 10.8 су дати резултати за исти саобраћајни сценарио за 64x64 комутаторе. Као и у претходним случајевима, закључци остају исти као и за исти сценарио за 32x32 комутаторе. Резултати свих сценарија за двије различите димензије комутатора показују да је мултикаст рјешење предложено у овом раду веома скалабилно. Имајући у виду да је као основа искоришћено предложено рјешење за уникаст саобраћај, а мултикаст допуна тог рјешења није комплексна што је показано у претходном поглављу, може се закључити да су предложени *M-LB-BvN-GS* и *M-mLB-BvN-GS* комутатори добри кандидати за решавање проблема комутације мултикаст саобраћаја.



Слика 10.8. Зависност средњег кашњења пакета од оптерећења за униформни *bursty* саобраћај са 100% мултикаст саобраћаја за 64x64 комутаторе

11. ЗАКЉУЧАК

Захваљујући развоју различитих апликација и сервиса, Интернет је постао неизоставни дио наше свакодневнице. Постављање каблова са оптичким влакнима је обезбиједило огроман капацитет за пренос података, па су мрежни уређаји, тј. рутери и свичеви постали ограничавајући фактор кад су у питању перформансе Интернет мрежа. Зато је развој и унапређење перформанси ових уређаја од кључног значаја за квалитет рада мрежа. У посљедњих неколико деценија је предложено више различитих рјешења која су користила различите приступе рјешавању проблема комутације пакета. Рјешења која су користила бафере на излазним портовима нису погодна, јер се од ових бафера захтијева да раде вишеструко брже од линкова, што је технолошки немогуће имплементирати за велике брзине које нуде оптичка влакна. Зато је предност дата рјешењима која користе бафере на улазним портовима, јер се код њих захтијева рад на нивоу брзине линкова, што је технолошки остварив. Међутим, овај тип комутатора захтијева да се у сваком временском слоту врши прорачун конфигурације комутатора, што је врло комплексан задатак који треба урадити у врло кратком временском року. Рјешење које не захтијева прорачун конфигурације је *LB-BvN* архитектура, која се састоји од два комутациона модула који имају детерминистичке конфигурације. Међутим, недостатак овог типа комутатора је што у њему долази до поремећаја редослиједа пакета, што је у *IP* мрежама недопустиво јер изазива велики број ретрансмисија пакета у случају *TCP* саобраћаја што значајно обара перформансе мрежа.

Циљ рада на овој дисератцији је био да се предложи архитектура пакетског свича који ће да омогући ефикасно прослиједивање уз прихватљив ниво хардверске комплексности. Основна идеја је да се пронађе рјешење које би користило најбоље особине постојећих рјешења, а да се при том избјегну њихови недостаци. Да би се ријешио проблем повременог загушења излазних портова до којих у *IP* мрежама може доћи због спорадичног карактера *IP* саобраћаја, идеја је да се користе бафери на улазним портовима који раде на ниоу брзине линкова. С друге стране, да би се избјегла потреба за прорачуном конфигурација комутатора, одлучено је да се користи *LB-BvN* архитектура са два комутациона модула који имају детерминистичке конфигурације. Кључни изазови које је још требало ријешити јесте избјегавање или рјешавање проблема поремећаја редослиједа пакета, што је карактеристично за *LB-BvN* комутаторе. У току рада на дисертацији предложена су и истражена два рјешења.

У потпоглављу 4.1 је представљен *BvN-DLB* комутатор, који пакете са улазних портова дефлектује на све портове чиме се врши равномјерна дистрибуција улазног оптерећења по свим портовима. Проблем поремећаја редослиједа пакета рјешава коришћењем ресеквенционих бафера на излазним портовима. Ово рјешење има предност у односу на већину осталих јер омогућава да се коришћењем *network calculus* теорије математички прорачуна граница кашњења пакета и величина бафера које никад неће бити прекршене тј. могу да се гарантују. Како би се ови резултати провјерили, урађена је софтверска симулација предложеног рјешења за различите моделе саобраћаја. Симулирани су Бернулијев униформни (*u*), Бернулијев *hot-spot* (*h*) и дијагонални (*d*) саобраћајни сценарији. Бернулијев униформни модел саобраћаја одговара изразито симетричном саобраћају, док с друге стране, дијагонални модел саобраћаја представља изразито

асиметричан модел саобраћаја. Бернулијев *hot-spot* модел саобраћаја представља својеврсну средину два претходна модела саобраћаја у смислу симетричности. Таквим избором сценарија се покривају све крајности са становишта симетричности саобраћаја. Резултати симулације су показали да теоријске границе, које су добијене уз помоћ *network calculus* математичког апарата, никад нису прекршене чак ни у најгорем сценарију. Недостатак овог рјешења је величина ресеквенционих бафера од N^2 пакета, што може бити имплементационо проблематично за велико N. Предложени *BvN-DLB* комутатор заједно са математичким моделом овог рјешења представља први допринос ове дисертације.

У току даљег рада на дисертацији предложен је *LB-BvN-GS* комутатор који је за циљ имао избегавање поремећаја редослиједа пакета. И ово рјешење користи два комутациона модула са детерминистичким конфигурацијама, као и бафере на улазним портовима. Овдје је проблем поремећаја редослиједа пакета ријешен развојем логике за одабир пакета који гарантује да ће пакети на излазне портове стићи у правилном редослиједу. Алтернативно рјешење, *mLB-BVN-GS*, умјесто ове логике за одабир пакета користи ресеквенционе бафере на излазним портовима, али је њихова величина ограничена на N пакета, што је знатно мање него код *BvN-DLB* комутатора, али и других *LB-BvN* рјешења која користе ресеквенционе бафере. Такође, битно је напоменути да сва три предложена рјешења у овој дисертацији могу користити тзв. *folded* архитектуру, гдје се умјесто два комутациона модула може користити само један физички уређај тј. комутациони модул. Урађене су софтверске симулације за различите саобраћајне моделе које су показале да *LB-BvN-GS* и *mLB-BvN-GS* комутатори у већини сценарија постижу боље перформансе у односу на *BvN-DLB* комутатор. Предложени *LB-BvN-GS* и *mLB-BvN-GS* за уникаст пакетски саобраћај представљају други допринос ове дисертације

У петом поглављу, *LB-BvN-GS* и *mLB-BvN-GS* комутатори су упоређени са најбољим постојећим уникаст рјешењима која користе различите приступе комутацији пакета. Показано је да у свим саобраћајним сценаријима ови комутатори постижу врло добре перформансе, нарочито при највећим оптерећењима. Такође, урађена је и хардверска имплементација предложених *LB-BvN-GS* и *mLB-BvN-GS* ради процјене њихове хардверске комплексности, и која је показала да су њихове потребе за хардверским ресурсима прихватљиве. Компарација предложених рјешења са постојећим уникаст рјешењима, као и процјена њихове хардверске комплексности представљају трећи допринос ове дисертације.

Као што је већ речено, симулације су показале да *LB-BvN-GS* и *mLB-BvN-GS* комутатори постижу веома добре перформансе у различитим регуларним саобраћајним сценаријима који подразумевају да ниједан излазни порт није преоптерећен. С друге стране, за Интернет саобраћај је типична спорадичност, као и различите ситуације које могу довести довести до тога да су поједини излазни портови рутера преоптерећени. Зато је веома битно и понашање комутатора под нерегуларним условима, односно кад су неки од излазних портова преоптерећени. Наиме, веома често се дешава да у случају преоптерећења излазног порта неки од токова добију знатно већи удио у капацитету него што је фер према другим токовима. Урађене су анализе које су показале да постојећа рјешења заснована на *LB-BvN* архитектури имају лоше перформансе са становишта фер опслуживања. Приликом почетних анализа *LB-BvN-GS* рјешења се показало да он има углавном добре перформансе у погледу фер опслуживања, али не и идеалне, нарочито у одређеним испитиваним сценаријима. Стога је предложено унапријеђење *LB-BvN-GS* које омогућава бољу подршку за фер сервис. Ово рјешење је тестирано за различите сценарије у којима долази до преоптерећења неких излазних портова и на основу добијених резултата закључено је да предложено унапријеђење (*fLB-BvN-GS*) остварује боље перформансе од других алгоритама, а такође постиже резултате

који одговарају идеалном фер сервису. Предложено унапријеђење *fLB-BvN-GS*, као и фер сервис анализа предложеног рјешења, као и постојећих, представљају четврти допринос ове дисертације.

Развој нових сервиса и апликација на Интернету је у последњих пар деценија утицао да се повећа количина мултикаст саобраћаја у мрежи. С обзиром на то да је већина свичева или рутера била развијена за управљање уникаст саобраћајем који је до тада био доминантан, њихове перформансе у присуству мултикаст саобраћаја су слабије. Зато је пажња истраживача посвећена проналажењу рјешења која би на ефикасан начин управљали и уникаст и мултикаст саобраћајем. Већина рјешења су заправо унапријеђења постојећих комутатора, којима су додати посебни бафери за чување мултикаст пакета. Такође, развијени су и нови алгоритми за одабир пакета за слање који су прилагођени и за мултикаст саобраћај. Један од доприноса ове дисератције јесте предлог комутатора који је прилагођен за ефикасно управљање уникаст и мултикаст саобраћајем. Као основа су искоришћени *LB-BvN-GS* и *mLB-BvN-GS* комутатори који су првобитно развијени само за уникаст саобраћај. Додатно је развијен модул за управаљње мултикаст пакетима. Принцип рада овог модула је да се за сваки мултикаст ток формира одговарајуће мултикаст стабло, које служи да се оптерећење са неког улазног порта који прима мултикаст пакет распореди по другим портовима који су дио тог мултикаст стабла. Овако је на једноставан начин ријешен проблем управљања мултикаст саобараћајем. Софтверске симулације које су урађене су показале да ово рјешење постиже врло добре перформансе за различите саобраћајне сценарије, нарочито при великим оптерећењима. Предложено проширење за мултикаст подршку, као и извршене анализе представљају пети допринос ове дисертације.

Развој 5G мрежа ће омогућити повезивање готово свих уређаја на Интернет, што је познато као концепт Интернет ствари (*Internet of things*). Такође, развој вјештачке интелигенције утиче на појаву нових сервиса као што су аутономна божња, телемедицина, итд. Све ово ће утицати да количина саобраћаја у Интернет мрежама настави експоненцијални раст, па је пред провајдерима изазов да осигурају квалитетан рад својих мрежа. Како су каблови са оптичким влакнима ријешили проблем капацитета, перформансе мрежа ће у највећој мјери зависити од перформанси мрежних уређаја. Доприноси ове дисертације су резултирали архитектуром пакетског свича која покрива све битне аспекте (уникаст, мултикаст, фер опслуживање) једног свича, и која у поређењу са другим рјешењима постиже врло добре перформансе, при чему је ниво хардверских захтијева на прихватљивом нивоу што ово рјешење чини атрактивним.

ЛИТЕРАТУРА

[1] J. Markoff, „An Internet Pioneer Ponders the Next Revolution,“ Dec. 1999. Available: https://archive.nytimes.com/www.nytimes.com/library/tech/99/12/biztech/articles/122099outlook-bobb.html [Accessed 21.04.2022.]

[2] L. Kleinrock, „Information flow in large communication nets,“ *RLE Quarterly Progress Report* 1, Massachusetts Institute of Technology, Apr. 1962.

[3] Cisco, „Cisco Nexus 9300-GX2 Series Fixed Switches Data Sheet,“ Jan. 2022. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/datasheet-c78-743854.html [Accessed 21.04.2022.]

[4] Juniper Networks, „400G“. Available: https://www.juniper.net/us/en/solutions/400g.html [Accessed 21.04.2022.]

[5] T. Neal, „Nokia’s 400G Everywhere: Optimizing IP/Optical Networks with Next Generation Coherent Optics,“ Jul. 2021. Available: https://www.thefastmode.com/technology-solutions/20214-nokia-s-400g-everywhere-optimizing-ip-optical-networks-with-next-generation-coherent-optics [Accessed 21.04.2022.]

[6] З.Чича, Материјали за предмет Комутациони системи. Доступно: http://telit.etf.rs/predmeti-elektronski-dokumenti/ [Accessed 21.04.2022.]

[7] N. Mckeown, „A fast switched backplane for a gigabit switched router,“ *Business Communications Review*, vol. 27, no. 12, Dec. 1997.

[8] R.T. El-Maghraby, N.M. Abd Elazim, A.M. Bahaa-Eldin, „A survey on deep packet inspection,“ in proc. of *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, Cairo, Egypt, Dec. 2017, pp. 188-197, doi: 10.1109/ICCES.2017.8275301.

[9] P. Jiang, S. Zhang, Q. Liu, C. Zheng, „A P4-Based Packet Scheduling Approach for Clustered Deep Packet Inspection Appliances,“ in proc. of *2021 International Conference on Computer Communications and Networks (ICCCN)*, Athens, Greece, Jul. 2021, pp. 1-9, doi: 10.1109/ICCCN52240.2021.9522193.

[10] W. Song, M. Beshley, K. Przystupa, H. Beshley, O. Kochan, A. Pryslupskyi, D. Pieniak, J. Su, „A Software Deep Packet Inspection System for Network Traffic Analysis and Anomaly Detection,“ *Sensors*, vol. 20(6):1637, Mar. 2020, doi: 10.3390/s20061637.

[11] Z. Cheng, M. Beshley, H. Beshley, O. Kochan, O. Urikova, „Development of Deep Packet Inspection System for Network Traffic Analysis and Intrusion Detection,“ in proc. of *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, Feb. 2020, pp. 877-881, doi: 10.1109/TCSET49122.2020.235562.

[12] B. Yang, D. Liu, „Research on Network Traffic Identification based on Machine Learning and Deep Packet Inspection,“ in proc. of *2019 IEEE 3rd Information Technology,*

Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, Mar. 2019, pp. 1887-1891, doi: 10.1109/ITNEC.2019.8729153.

[13] M. Alsaeedi, M.M. Mohamad, A.A. Al-Roubaiey, „Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey," *IEEE Access*, vol. 7, pp. 107346-107379, Aug. 2019, doi: 10.1109/ACCESS.2019.2932422.

[14] R. Amin, M. Reisslein, N. Shah, „Hybrid SDN Networks: A Survey of Existing Approaches," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3259-3306, May 2018, doi: 10.1109/COMST.2018.2837161.

[15] E. Haleplidis, K. Pentikousis, S. Denazis, J. Hadi Salim, D. Meyer, O. Koufopavlou, „Software-Defined Networking (SDN): Layers and Architecture Terminology," RFC 7426, Jan. 2015, doi: 10.17487/RFC7426.

[16] K. Bakshi, „Considerations for Software Defined Networking (SDN): Approaches and use cases," in proc. of *2013 IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2013, pp. 1-9, doi: 10.1109/AERO.2013.6496914.

[17] З.Чича, „Имплементација функција пакетског процесирања у интернет рутерима великог капацитета," докторска дисертација, Универзитет у Београду - Електротехнички факултет, 2012, doi: 10.2298/bg20120712cica.

[18] L. Ferdouse, A. Anpalagan, S. Erkucuk, „Joint Communication and Computing Resource Allocation in 5G Cloud Radio Access Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 9122-9135, Sept. 2019, doi: 10.1109/TVT.2019.2927904.

[19] S.T. Arzo, R. Bassoli, F. Granelli, F.H.P. Fitzek, „Study of Virtual Network Function Placement in 5G Cloud Radio Access Network," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2242-2259, Dec. 2020, doi: 10.1109/TNSM.2020.3020390.

[20] S. Zhou, X. Liu, D. Fu, X. Cheng, B. Fu and Z. Zhao, „5G Core Network Framework Based on Artificial Intelligence," in proc. of *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, Exeter, United Kingdom, Dec. 2020, pp. 1460-1464, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00219.

[21] D. Lake et al., „Virtualising and orchestrating a 5G evolved packet core network," in proc. of *2017 IEEE Conference on Network Softwarization (NetSoft)*, Bologna, Italy, pp. 1-5, July 2017, doi: 10.1109/NETSOFT.2017.8004215.

[22] ETSI, „System architecture for the 5G System (5GS) (3GPP TS 23.501 version 16.6.0 Release 16)," Oct. 2020. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf [Accessed 21.04.2022.]

[23] Z. Čiča, „Analysis and implementation of packet processing functions in Internet routers," in proc. of *2012 20th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, pp. 218-225, Nov. 2012, doi: 10.1109/TELFOR.2012.6419186.

[24] T. Yang et al., „Constant IP Lookup With FIB Explosion," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1821-1836, Aug. 2018, doi: 10.1109/TNET.2018.2853575.

[25] M.I. Islam, J.I. Khan, „C2RTL: A High-level Synthesis System for IP Lookup and Packet Classification," in proc. of *22nd International Conference on High Performance Switching and Routing (HPSR 2021)*, Paris, France, pp. 1-8, June 2021, doi: 10.1109/HPSR52026.2021.9481810.

[26] B. Indira, K. Valarmathi, D. Devaraj, „A Trie based IP Lookup Approach for High Performance Router/Switch," in proc. of *2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, Tamilnadu, India, pp. 1-6, Apr. 2019, doi: 10.1109/INCOS45849.2019.8951425.

[27] W. Li, D. Li, X. Liu, T. Huang, X. Li, W. Le, Wenxia, H. Li, „A power-saving pre-classifier for TCAM-based IP lookup," *Computer Networks*, vol. 164, 106898, Dec. 2019, doi: 10.1016/j.comnet.2019.106898.

[28] A. Smiljanić, Z. Čiča, „A comparative review of scalable lookup algorithms for IPv6," *Computer Networks*, vol. 56, no. 13, pp. 3040-3054, Sept. 2012, doi: 10.1016/j.comnet.2012.04.027

[29] C. Toal, D. Burns, K. McLaughlin, S. Sezer, S. O'Kane, „An RLDRAM II Implementation of a 10Gbps Shared Packet Buffer for Network Processing," in proc. of *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*, Edinburgh, UK, pp. 613-618, Aug. 2007, doi: 10.1109/AHS.2007.30.

[30] J. Garcia-Vidal, M. March, L. Cerda, J. Corbal, M. Valero, „A DRAM/SRAM memory scheme for fast packet buffers," *IEEE Transactions on Computers*, vol. 55, no. 5, pp. 588-602, May 2006, doi: 10.1109/TC.2006.63.

[31] Y. Song, J. Hwang, I. Jo, H. Lee, „Highly Available Packet Buffer Design With Hybrid Nonvolatile Memory," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 11, pp. 2008-2012, Nov. 2021, doi: 10.1109/TVLSI.2021.3116272.

[32] K. Kobayashi, „A DRAM-friendly priority queue Internet packet scheduler implementation and its effects on TCP," in proc. of *2020 IFIP Networking Conference (Networking)*, Paris, France, pp. 713-718, June 2020.

[33] C. Hu, X. Chen, W. Li, B. Liu, „Fixed-length switching vs. variable-length switching in input-queued IP switches," in proc. of *2004 IEEE International Workshop on IP Operations and Management*, Beijing, China, pp. 117-122, Oct. 2004, doi: 10.1109/IPOM.2004.1547602.

[34] J. Duato, J. Flich, T. Nachiondo, „A cost-effective technique to reduce HOL blocking in single-stage and multistage switch fabrics," in proc. of *12th Euromicro Conference on Parallel, Distributed and Network-Based Processing 2004*, Coruna, Spain, pp. 48-53, Feb, 2004, doi: 10.1109/EMPDP.2004.1271426.

[35] T. Nachiondo, J. Flich, J. Duato, „Buffer Management Strategies to Reduce HoL Blocking," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 6, pp. 739-753, June 2010, doi: 10.1109/TPDS.2009.63.

[36] H.J Chao, B. Liu, *High Performance Switches and Routers*, John Wiley & Son, 2006.

[37] R. Smith, „SK Hynix Announces Its First HBM3 Memory: 24GB Stacks, Clocked at up to 6.4Gbps," Oct. 2021. Available: https://www.anandtech.com/show/17022/sk-hynix-announces-its-first-hbm3-memory-24gb-stacks-at-up-to-64gbps [Accessed 21.04.2022.]

[38] J.R. Wilson, „SK Hynix HBM3 Memory Module Revealed During OCP Summit 2021 - 12-Hi Stack, 24GB Module With 6400 Mbps Transfer Speeds," Nov. 2021. Available: https://wccftech.com/sk-hynix-hbm3-memory-module-revealed-during-ocp-summit-2021-12-hi-stack-24gb-module-with-6400-mbps-transfer-speeds/ [Accessed 21.04.2022.]

[39] Sourcengine Team, „Micron Technology RLDRAM 3," June 2020. Available: https://www.sourcengine.com/blog/rldram-3-micron-technology [Accessed 21.04.2022.]

[40] Micron Technology, Inc., „576Mb: x18, x36 RLDRAM 3 Features," 2016. Available: https://media-www.micron.com/-/media/client/global/documents/products/data-sheet/dram/576mb_rldram3.pdf?rev=c2b8d61687174305af387cfb1e5aafee [Accessed 21.04.2022.]

[41] Infineon, „Infineon launches industry's highest density QML-V-certified QDR-II+ SRAM to simplify on-system satellite image processing," May 2021. Available: https://www.infineon.com/cms/en/about-infineon/press/market-news/2021/INFATV202105-069.html [Accessed 21.04.2022.]

[42] D. Bertsekas, R. Gallager, *Data Networks*, Prentice-Hall, 1992.

[43] I. Radusinovic, M. Radonjic, M. P. Djurisic, „Analysis of WRR scheduling algorithm frame size impact on CQ switch performance," in proc. of *2013 Africon*, Pointe aux Piments, Mauritius, pp. 1-5, Sept. 2013, doi: 10.1109/AFRCON.2013.6757797.

[44] S.J. Golestani, „A framing strategy for congestion management," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp. 1064-1077, Sep. 1991, doi: 10.1109/49.103553.

[45] C.R. Kalmanek, H. Kanakia, S. Keshav, „Rate controlled servers for very high-speed networks," in proc. of *IEEE GLOBECOM '90*, San Diego, CA, USA, pp. 12-20, Dec. 1990, doi: 10.1109/GLOCOM.1990.116472.

[46] A. Baranowska, W. Kabacinski, „Hierarchical round-robin matching for virtual output queuing switches," in proc. of *Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05)*, Lisbon, Portugal, pp. 196-201, July 2005, doi: 10.1109/AICT.2005.46.

[47] C.L. Liu, J.W. Wayland, „Scheduling algorithms for multi-programming in a hard real-time environment," *Journal of ACM*, vol. 20, no. 1, pp. 46-61, Jan. 1973, doi: 10.1145/321738.321743.

[48] D. Ferrari, D. Verma, „A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, pp. 368-379, Apr. 1990, doi: 10.1109/49.53013.

[49] D.C. Verma, H. Zhang, D. Ferrari, „Guaranteeing delay jitter bounds in packet switching networks," in proc. of *TRICOMM `91: IEEE Conference on Communications Software: Communications for Distributed Applications and Systems*, Chapel Hill, NC, USA, pp. 35-43, Apr. 1991, doi: 10.1109/TRICOM.1991.152873.

[50] M. Shreedhar, G. Varghese, „Efficient fair queueing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, issue 3, pp. 375-385, June 1996, doi: 10.1109/90.502236.

[51] K. Nichols, V. Jacobson, „Controlling queue delay," *Communications of the ACM*, vol. 5, no. 7, pp. 42-50, July 2012, doi: 10.1145/2209249.2209264

[52] A.K. Parekh, R.G. Gallager, „A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 2, pp. 344-357, June 1993, doi: 10.1109/90.234856.

[53] A.K. Parekh, R.G. Gallager, „A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 137-150, Apr. 1994, doi: 10.1109/90.298432.

[54] X. Yu, L.J. Thng, Y. Jiang, „Generalized processor sharing with long-range dependent traffic input," in proc. of *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Cincinnati, OH, USA, pp. 224-231, Aug. 2001, doi: 10.1109/MASCOT.2001.948872.

[55] A.J. Khan, A. Sahoo, D. Manjunath, „Implementation of WFQ in a distributed open software router," in proc. of *2011 IEEE 36th Conference on Local Computer Networks*, Bonn, Germany, pp. 519-527, Oct. 2011, doi: 10.1109/LCN.2011.6115515.

[56] S.J. Golestani, „A self-clocked fair queueing scheme for broadband applications," in proc. of *IEEE INFOCOM '94*, Toronto, ON, Canada, pp. 636-646, June 1994, doi: 10.1109/INFCOM.1994.337677.

[57] H. Halabian, H. Saidi, R. Changiz, „LVT-SCFQ: A Modified Self Clocked Fair Queueing Algorithm for Broadband Networks," in proc. of *2008 Third International Conference on Broadband Communications, Information Technology & Biomedical Applications*, Pretoria, South Africa, pp. 175-180, Nov. 2008, doi: 10.1109/BROADCOM.2008.77.

[58] J.C.R. Bennett, H. Zhang, „WF/sup 2/Q: worst-case fair weighted fair queueing," in proc. of *IEEE INFOCOM '96*, San Francisco, CA, USA, pp. 120-128, Mar. 1996, doi: 10.1109/INFCOM.1996.497885.

[59] J.C.R. Bennett, H. Zhang, „Hierarchical packet fair queuing algorithms," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 675-689, Oct. 1997, doi: 10.1109/90.649568.

[60] T. Balogh, M. Medvecký, „Performance evaluation of WFQ, WF2Q+ and WRR queue scheduling algorithms," in proc. of *2011 34th International Conference on Telecommunications and Signal Processing (TSP)*, Budapest, Hungary, pp. 136-140, Aug. 2011, doi: 10.1109/TSP.2011.6043756.

[61] K. Shin, S. Choi, H. Kim, „Flit Scheduling for Cut-Through Switching: Towards Near-Zero End-to-End Latency," *IEEE Access*, vol. 7, pp. 66369-66383, May 2019, doi: 10.1109/ACCESS.2019.2916651.

[62] S. Patel, P. Gupta, G. Singh, „Performance measure of Drop tail and RED algorithm," in proc. of *2010 2nd International Conference on Electronic Computer Technology*, Kuala Lumpur, Malaysia, pp. 35-38, May 2010 doi: 10.1109/ICECTECH.2010.5479996.

[63] G. Raina, S. Manjunath, S. Prasad, K. Giridhar, „Stability and Performance Analysis of Compound TCP With REM and Drop-Tail Queue Management," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 1961-1974, Aug. 2016, doi: 10.1109/TNET.2015.2448591.

[64] T.V. Lakshman, A. Neidhardt, T.J. Ott, „The drop from front strategy in TCP and in TCP over ATM," in proc. of *IEEE INFOCOM '96*, San Francisco, CA, USA, pp. 1242–1250, Mar. 1996, doi: 10.1109/INFCOM.1996.493070.

[65] B. Briscoe et al., „Reducing Internet Latency: A Survey of Techniques and Their Merits," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2149-2196, thirdquarter 2016, doi: 10.1109/COMST.2014.2375213.

[66] S. Floyd, V. Jacobson, „Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993, doi: 10.1109/90.251892.

[67] F. Baker, G. Fairhurst, „IETF Recommendations Regarding Active Queue Management," RFC 7567, July 2015, doi: 10.17487/RFC7567.

[68] S.B. Danladi, F.U. Ambursa, „DyRED: An Enhanced Random Early Detection Based on a new Adaptive Congestion Control," in proc. of *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)*, Abuja, Nigeria, pp. 1-5, Dec. 2019, doi: 10.1109/ICECCO48375.2019.9043276.

[69] D. Lin, R. Morris, „Dynamics of random early detection," in proc. of *SIGCOMM*, Cannes, France, pp. 127–137 (Sept. 1997).

[70] T. Ott, T. Lakshman, L. Wong, „SRED: stabilized RED," in proc. of *IEEE INFOCOM*, New York, NY, USA, pp. 1346-1355, Mar. 1999, doi: 10.1109/INFCOM.1999.752153.

[71] S. Patel, „Performance analysis of RED for stabilized queue," in proc. of *2014 Seventh International Conference on Contemporary Computing (IC3)*, Noida, India, pp. 306-311, Aug. 2014, doi: 10.1109/IC3.2014.6897191.

[72] B. Abbasov, S. Korukoglu, „Effective RED: An algorithm to improve RED's performance by reducing packet loss rate," *Journal of Network and Computer Applications*, vol. 32, no. 3, pp. 703-709, May 2009, doi: 10.1016/j.jnca.2008.07.001.

[73] R. Adams, „Active Queue Management: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1425-1476, Third Quarter 2013, doi: 10.1109/SURV.2012.082212.00018.

[74] K. Al-Bawani, M. Englert, M. Westermann, „Comparison-Based Buffer Management in QoS Switches," *Algorithmica*, vol. 80, no. 3, pp. 1073-1092, Mar. 2018, doi: 10.1007/s00453-017-0393-2.

[75] Cisco, „Cisco Catalyst 9600 Series Switches Data Sheet," Apr. 2022. Available: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-9600-series-switches/nb-06-cat9600-series-data-sheet-cte-en.html [Accessed 21.04.2022.]

[76] Juniper Networks, „PTX1000, PTX10001, PTX10002, and PTX10003 Fixed Configuration Routers". Available: https://www.juniper.net/us/en/products/routers/ptx-series/ptx1000-ptx10001-ptx10002-ptx10003-fixed-configuration-packet-transport-routers-datasheet.html [Accessed 21.04.2022.]

[77] H. Lee, K.H. Kook, C.S. Rim, K.P. Jun, S.K. Lim, „A limited shared output buffer switch for ATM," in proc. of *Fourth Int. Conf. on Data Commun. Syst. and their Performance*, Barcelona, Spain, pp. 163-179, June 1990, doi: 10.1016/B978-0-444-88756-6.50019-0.

[78] K.J. Schultz, P.G. Gulak, „CAM-based single-chip shared buffer ATM switch," in proc. of *IEEE ICC/SUPERCOMM'94*, New Orleans, LA, USA, pp. 1190-1195, May 1994, doi: 10.1109/ICC.1994.368914.

[79] K. Oshima, H. Yamanaka, H. Saito, H. Yamada, S. Kohama, H. Kondoh, and Y. Matsuda, „A new ATM switch architecture based on STS-type shared buffering and its LSI implementation," in proc. of *XIV International Switching Symposium*, Yokohama, Japan, Oct. 1992.

[80] T. Cheney, J.A. Fingerhut, M. Flucke, J.S. Turner, „Design of a Gigabit ATM switch," in proc. of *IEEE INFOCOM'97*, Kobe, Japan, pp. 2–11, Apr. 1997, doi: 10.1109/INFCOM.1997.635108.

[81] K. Eng, „State of the art in gigabit ATM switching," in proc. of *8th IEEE Workshop on Computer Communications*, Del Mar, CA, USA, pp. 0_66-0_66, Oct. 1993, doi: 10.1109/CMPCMM.1993.659075.

[82] J. Garcia-Haro, A. Jajszczyk, „ATM shared-memory switching architectures," *IEEE Network*, vol. 8, no. 4, pp. 18-26, July 1994, doi: 10.1109/65.298160.

[83] P. Eugster, K. Kogan, S. Nikolenko, A. Sirotkin, „Shared Memory Buffer Management for Heterogeneous Packet Processing," in proc. of *2014 IEEE 34th International Conference on Distributed Computing Systems*, Madrid, Spain, pp. 471-480, July 2014, doi: 10.1109/ICDCS.2014.55.

[84] C. Effiong, G. Sassatelli, A. Gamatie, „Distributed and dynamic shared-buffer router for high-performance interconnect," in proc. of *2017 Eleventh IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, Seoul, South Korea, pp. 1-8, Oct. 2017.

[85] C. Effiong, G. Sassatelli, A. Gamatie, „Scalable and Power-Efficient Implementation of an Asynchronous Router with Buffer Sharing," in proc. of *2017 Euromicro Conference on Digital System Design (DSD)*, Vienna, Austria, pp. 171-178, Sept. 2017, doi: 10.1109/DSD.2017.55.

[86] A.T. Tran, B.M. Baas, „Achieving High-Performance On-Chip Networks With Shared-Buffer Routers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 6, pp. 1391-1403, June 2014, doi: 10.1109/TVLSI.2013.2268548.

[87] M.J. Karol, M. Hluchyj, S. Morgan, „Input versus output queuing on a space-division packet switch," *IEEE Transactions on Communications*, vol. 35, no. 12, pp. 1347–1356, Dec. 1987, doi: 10.1109/TCOM.1987.1096719.

[88] A.L. Gupta, N.D. Georganas, „Analysis of a packet switch with input and output buffers and speed constraints," in proc. of *IEEE INFOCOM'91*, Bal Harbour, FL, USA, pp. 694-700, Apr. 1991, doi: 10.1109/INFCOM.1991.147573.

[89] J.S. Chen, T.E. Stern, „Throughput analysis, optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 3, pp. 439-449, Apr. 1991, doi: 10.1109/49.76643.

[90] S.T. Chuang, A. Goel, N. McKeown, B. Prabhakar, „Matching output queueing with a combined input/output-queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1030-1039, June 1999, doi: 10.1109/49.772430.

[91] B. Prabhakar, N. McKeown, „On the speedup required for combined input and output queued switching," *Automatica*, vol. 35, no. 12, pp. 1909-1920, Dec. 1999, doi: 10.1016/S0005-1098(99)00129-6.

[92] S. Iyer, N. McKeown, „Using constraint sets to achieve delay bounds in CIOQ switches," *IEEE Communications Letters*, vol. 7, no. 6, pp. 275–277, June 2003, doi: 10.1109/LCOMM.2003.812712.

[93] Y. Doi, N. Yamanaka, „A high-speed ATM switch with input and cross-point buffers," *IEICE Transactions on Communications*, vol. E76-B, no. 3, pp. 310–314, Mar. 1993.

[94] K. Yoshigoe, „The Crosspoint-Queued switches with virtual crosspoint queueing," in proc. of *2011 5th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Honolulu, HI, USA, pp. 1-5, Dec. 2011, doi: 10.1109/ICSPCS.2011.6140853.

[95] B. Wang, „A fair queueing scheduling algorithm for port trunking CICQ switch," in proc. of *2010 IEEE 12th International Conference on Communication Technology*, Nanjing, China, pp. 877-880, Nov. 2010, doi: 10.1109/ICCT.2010.5688717.

[96] H. Hu, Z. Xu, „A Load Balanced Differentiated Services Support CICQ Switch," in proc. of *2012 Third International Conference on Digital Manufacturing & Automation*, Guilin, China, pp. 66-71, Aug. 2012, doi: 10.1109/ICDMA.2012.16.

[97] M. Nabeshima, „Performance evaluation of a combined input- and crosspoint-queued switch," *IEICE Transactions on Communications*, vol. E83-B, no. 3, pp. 737–741, Mar. 2000.

[98] T. Javidi, R. Magill, T. Hrabik, „A high throughput scheduling algorithm for a buffered crossbar switch fabric," in proc. of *IEEE International Conference on Communications (ICC 2001)*, Helsinki, Finland, pp. 1586–1591, June 2001, doi: 10.1109/ICC.2001.937187.

[99] L. Mhamdi, M. Hamdi, „MCBF: a high-performance scheduling algorithm for buffered crossbar switches," *IEEE Communications Letters*, vol. 7, no. 9, pp. 451–553, Sept. 2003, doi: 10.1109/LCOMM.2003.815665.

[100] Wolfram MathWorld, „Bipartite Graph". Available: https://mathworld.wolfram.com/BipartiteGraph.html [Accessed 21.04.2022.]

[101] N. McKeown, A. Mekkittikul, V. Anantharam, J. Walrand, „Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260-1267, Aug. 1999, doi: 10.1109/26.780463.

[102] L. Tassiulas, A. Ephremides, „Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992, doi: 10.1109/9.182479.

[103] R.E. Tarjan, *Data structures and network algorithms*, Society for Industrial and Applied Mathematics, 1983.

[104] A. Mekkittikul, N. Mckeown, „A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in proc. of *INFOCOM'98*, San Francisco, CA, USA, pp. 792–799, Mar. 1998, doi: 10.1109/INFCOM.1998.665102.

[105] J.E. Hopcroft, R.M. Karp, „An n5/2 algorithm for maximum matchings in bipartite graphs," *SIAM J. Comput*., vol. 2, no. 4, pp. 225–231, Dec. 1973, doi: 10.1137/0202019.

[106] N. McKeown, „Scheduling algorithms for input-queued cell switches," *Ph.D. thesis*, UC Berkeley, 1995.

[107] J. G. Dai, B. Prabhakar, „The throughput of data switches with and without speedup," in proc. of *INFOCOM'00*, Tel Aviv, Israel, pp. 556–564, Mar. 2000, doi: 10.1109/INFCOM.2000.832229.

[108] E. Leonardi, M. Mellia, F. Neri, M.A. Marsan, „On the stability of input-queued switches with speed-up," *IEEE/ACMTransactions on Networking*, vol. 9, no. 1, pp. 104-118, Feb. 2001, doi: 10.1109/90.909028.

[109] T.E. Anderson, S.S. Owicki, J.B. Saxe, C.P. Thacker, „High speed switch scheduling for local area networks," *ACM Transactions on Computer Systems*, vol. 11, no. 4, pp. 319-352, Nov. 1993, doi: 10.1145/161541.161736.

[110] N. McKeown, „The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, Apr. 1999, doi: 10.1109/90.769767.

[111] Y. Li, S. Panwar, H.J. Chao, „On the performance of a dual round-robin switch," in proc. of *IEEE INFOCOM 2001*, Anchorage, AK, USA, pp. 1688–1697, Apr. 2001, doi: 10.1109/INFCOM.2001.916666.

[112] Y. Li, „Design and analysis of schedulers for high speed input queued switches," *Ph.D. Dissertation*, Polytechnic University, 2004.

[113] H.J. Chao, J.S. Park, „Centralized contention resolution schemes for a large-capacity optical ATM switch," in proc. of *1998 IEEE ATM Workshop 'Meeting the Challenges of Deploying the Global Broadband Network Infrastructure*, Fairfax, VA, USA, pp. 11-16 May 1998, doi: 10.1109/ATM.1998.675103.

[114] H.J. Chao, „Saturn: A terabit packet switch using dual round-robin," *IEEE Communications Magazine*, vol. 38, no. 12, pp. 78–84, Dec. 2000, doi: 10.1109/35.888261.

[115] B. Hu, F. Fan, K.L. Yeung, S. Jamin, „Highest Rank First: A New Class of Single-Iteration Scheduling Algorithms for Input-Queued Switches," *IEEE Access*, vol. 6, pp. 11046-11062, Feb. 2018, doi: 10.1109/ACCESS.2018.2800686.

[116] L. Wang, T. Ye, T.T. Lee, W. Hu, „A Parallel Complex Coloring Algorithm for Scheduling of Input-Queued Switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1456-1468, July 2018, doi: 10.1109/TPDS.2018.2802523.

[117] D.B. West, *Introduction to graph theory*, Prentice hall, 2000.

[118] M. Petrovic, A. Smiljanic, M. Blagojevic, „Design of the Switching Controller for the High-capacity Non-blocking Internet Router," *IEEE Transactions on VLSI Systems*, vol. 17, no. 8, pp. 1157-1161, Aug. 2009, doi: 10.1109/TVLSI.2009.2019817.

[119] M. Petrović, A. Smiljanić, „Optimization of the scheduler for the non-blocking high-capacity router," *IEEE Communications Letters*, vol. 11, no. 6, pp. 534–536, June 2007, doi: 10.1109/LCOMM.2007.061653.

[120] M. Petrović, A. Smiljanić, „Design of the scheduler for the high-capacity nonblocking packet switch," in proc. of *IEEE Workshop on High Performance Switching and Routing* 2006, Poznan, Poland, June 2006, doi: 10.1109/HPSR.2006.1709742.

[121] G. Birkhoff, „Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucum Rev. Ser. A*, vol. 5, pp. 147–150, 1946.

[122] J. von Neumann, „A certain zero-sum two-person game equivalent to the optimal assignment problem," from the book *Contributions to the Theory of Games* (pp. 5–12), Princeton University Press, 1953.

[123] C.S. Chang, W.J. Chen, H.Y. Huang, „On service guarantees for input buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann," in proc. of *IEEE IWQOS'99*, London, UK, June 1999, pp. 79–86, doi: 10.1109/IWQOS.1999.766481.

[124] C.S. Chang, W.J. Chen, H.Y. Huang, „Birkhoff-von Neumann input buffered crossbar switches," in proc. of *IEEE INFOCOM'00*, Tel Aviv, Israel, Mar. 2000, pp. 1614–1623, doi: 10.1109/INFCOM.2000.832560.

[125] C.S. Chang, D.S. Lee, Y.S. Jou, „Load balanced Birkhoff-von Neumann switches. Part I: One-stage buffering," *Computer Communications*, vol. 25, no. 6, pp. 611–622, Apr. 2002, doi: 10.1016/S0140-3664(01)00427-3.

[126] C.S. Chang, D.S. Lee, Y.S. Jou, „Load balanced Birkhoff–von Neumann switche. Part II: Multistage buffering," *Computer Communications*, vol. 25, no. 6, pp. 623–634, Apr. 2002, doi: 10.1016/S0140-3664(01)00428-5.

[127] Y. Shen, S. Jiang, S.S. Panwar, H.J. Chao, „Byte-focal: a practical load balanced switch," in proc. of *2005 Workshop on High Performance Switching and Routing (HPSR 2005)*, Hong Kong, China, May 2005, pp. 6-12, doi: 10.1109/HPSR.2005.1503184.

[128] I. Keslassy, N. McKeown, „Maintaining packet order in two-stage switches," in proc. of *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'02)*, New York, NY, USA, June 2002, pp. 1032–1041, doi: 10.1109/INFCOM.2002.1019351.

[129] I. Keslassy, S. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, N. McKeown, „Scaling Internet routers using optics," in proc. of *2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM 2003)*, Karlsruhe, Germany, Aug. 2003, pp. 189–200, doi: 10.1145/863955.863978.

[130] I. Keslassy, „The load-balanced router," *Ph.D. Dissertation*, Stanford University, 2004.

[131] J.J. Jaramillo, F. Milan, R. Srikant, „Padded Frames: A Novel Algorithm for Stable Scheduling in Load-Balanced Switches," *IEEE/ACM Transactions on Networking*, vol. 16, no. 5, pp. 1212-1225, Oct. 2008, doi: 10.1109/TNET.2007.906654.

[132] C. Yu, C. Chang, D. Lee, „CR Switch: A Load-Balanced Switch With Contention and Reservation," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1659-1671, Oct. 2009, doi: 10.1109/TNET.2008.2010624.

[133] B. Hu, K.L. Yeung, „Feedback-Based Scheduling for Load-Balanced Two-Stage Switches," *IEEE/ACM Transactions on Networking*, vol. 18, no. 4, pp. 1077-1090, Aug. 2010, doi: 10.1109/TNET.2009.2037318.

[134] A. Huang, B. Hu, „The optimal joint sequence design in the feedback-based two-stage switch," in proc. of *2014 IEEE International Conference on Communications (ICC)*, Sydney, NSW, Australia, June 2014, pp. 3031-3036, doi: 10.1109/ICC.2014.6883786.

[135] S. Durković, Z. Čiča, „Birkhoff-von Neumann switch with deflection based load balancing," in proc. of *2016 24th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, Nov. 2016, pp. 1-4, doi: 10.1109/TELFOR.2016.7818731.

[136] J.Y. Le Boudec, P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, Springer, 2001, doi: 10.1007/3-540-45318-0.

[137] S. Durkovic, Z. Cica, „Birkhoff-Von Neumann Switch Based on Greedy Scheduling," *IEEE Computer Architecture Letters*, vol. 17, no. 1, pp. 13-16, Jan.-Jun. 2018, doi: 10.1109/LCA.2017.2707082.

[138] Xilinx Inc., „Virtex UltraScale+ HBM FPGA: A Revolutionary Increase in Memory Performance," July 2019. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/white_papers/wp485-hbm.pdf [Accessed 21.04.2022.]

[139] S. Durkovic, Z. Cica, „Load balanced Birkhoff-von Neumann switch with output congestion detection," in proc. of *2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*, Nis, Serbia, Oct. 2017, pp. 283-286, doi: 10.1109/TELSKS.2017.8246281.

[140] S. Durkovic, Z. Cica, „Fair Service Analysis of Load Balanced Birkhoff-von Neumann Switches," in proc. of *icETRAN 2017*, Kladovo, Serbia, June 2017

[141] J. Xiao, K.L. Yeung, „Iterative multicast scheduling algorithm for input-queued switch with variable packet size," in proc. of *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, Windsor, ON, Canada, pp. 1-4, May 2017, doi: 10.1109/CCECE.2017.7946605.

[142] J. Xiao, K.L. Yeung, S. Jamin, „Pipelined Scheduler for Unicast and Multicast Traffic in Input-Queued Switches," in proc. of *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, USA, pp. 1-6. Dec. 2016, doi: 10.1109/GLOCOM.2016.7842148.

[143] B. Hu, K.L. Yeung, „Multicast scheduling in feedback-based two-stage switch," in proc. of *2009 International Conference on High Performance Switching and Routing*, Paris, France, pp. 1-6, June 2009, doi: 10.1109/HPSR.2009.5307438.

[144] S. Durkovic, Z. Cica, „Multicast Load-Balanced Birkhoff-Von Neumann Switch With Greedy Scheduling," *IEEE Access*, vol 8, pp. 120654-120667, July 2020, doi: 10.1109/ACCESS.2020.3006370.

СПИСАК СКРАЋЕНИЦА

ATM	Asynchronous Transfer Mode
BF	Byte-Focal
BvN	Birkhoff-von Neumann
BvN-DLB	Birkhoff-von Neumann with Deflection-Based Load Balancing
CHRF	Coded Highest Rank First
CR	Contention and Resolution
CRC	Cyclic Redundancy Check
DQ	Department Queue
DRAM	Dynamic Random-Access Memory
DRR	Deficit Round-Robin
DRRM	Dual Round-Robin Matching
DTS	Dynamic Threshold Scheme
EDD	Earliest Due Date
EDF	Earliest Deadline First
FBSS	Feedback Staggered Symmetry
FCFS	First-Come-First-Serve
fEDF	flit-based Earliest Deadline First
FFF	Full Frame First
FIFO	First-In-First-Out
fLB-BvN-GS	fair LB-BvN-GS
FOFF	Full Ordered Frame First
FTP	File Transfer Protocol
FTS	Fixed Threshold Scheme
GPS	Generalized Processor Sharing
GS	Greedy Scheduling
HBM	High Bandwidth Memory
HoF	Head of Flow
HoL	Head-of-Line
HRF	Highest Rank First
HRR	Hierarchical Round-Robin
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ISO	International Organization for Standardization
LB-BvN	Load-Balanced Birkhoff-von Neumann
LB-BvN-GS	Load-Balanced Birkhoff-von Neumann with Greedy Scheduling
LPF	Longest Port First
LQF	Longest Queue First

LUT	LookUp Table
M-LB-BvN-GS	Multicast LB-BvN-GS
MIT	Massachusetts Institute of Technology
mLB-BvN-GS	modified LB-BvN-GS
MPLS	Multi Protocol Label Switching
MSM	Maximum Size Matching
MWM	Maximum Weight Matching
NoC	Network on Chip
OCF	Oldest Cell First
OSI	Open System Interconnection
PCC	Parallel Complex Coloring
PF	Padded Frames
PIM	Parallel Iterative Matching
QDR SRAM	Quad Data Rate Static Random-Access Memory
QoS	Quality of Service
RAM	Random Access Memory
RLDRAM	Reduced Latency Dynamic Random-Access Memory
RR	Round-Robin
SAFA	Success: Advance / Failure: Advance
SAFP	Success: Advance / Failure: Persist
SCFQ	Self-Clocked Fair Queuing
SDN	Software Defined Networking
SERDES	Serializer/Deserializer
SGS	Sequential Greedy Scheduling
SMTP	Simple Mail Transfer Protocol
SPFA	Success: Persist / Failure: Advance
SPFA-LMQ	Success: Persist / Failure: Advance - Longer than or equal to Medium Queue
SPFP	Success: Persist / Failure: Persist
SRAM	Static Random-Access Memory
TCP	Transmission Control Protocol
UFS	Uniform Frame Spreading
VHDL	VHSIC Hardware Description Language
VIQ	Virtual Input Queue
VOQ	Virtual Output Queue
WF2Q	Worst-Case Fair Weighted Fair Queuing
WFQ	Weigthed Fair Queuing
WRR	Weighted Round-Robin

СПИСАК СЛИКА

СПИСАК ТАБЕЛА

Биографија аутора

Срђан Дурковић је рођен 29.07.1991. у Подгорици. Основну школу „Максим Горки“ и Гимназију „Слободан Шкеровић“ је завршио у Подгорици с одличним успјехом. Електротехнички факултет у Подгорици је уписао 2009. године, на којем је и дипломирао 2013. године, на смјеру за телекомуникације. Мастер академске студије је уписао 2013. године на Електротехничком факултету у Београду, модул системско инжењерство и радио комуникације. Диплому мастер инжењера електротехнике и рачунарства је стекао 2014. године. Докторске академске студије је уписао на Електротехничком факултету у Београду, 2015. године - модул Телекомуникације. Област истраживања током докторских студија је првенствено била архитектура рутера и свичева. Као плод рада на истраживањима везаним за тему докторске дисертације објавио је више радова међу којима и два рада у међународним часописима са SCI листе. Срђан Дурковић је тренутно запослен у компанији „Ериксон“, на позицији инжењера за процјену перформанси мобилних мрежа.

образац изјаве о ауторству

Изјава о ауторству

Име и презиме аутора Срђан Дурковић

Број индекса 2015/5041

Изјављујем

да је докторска дисертација под насловом

Архитектура пакетског свича за ефикасно комутирање уникаст и мултикаст саобраћаја

- резултат сопственог истраживачког рада;
- да дисертација у целини ни у деловима није била предложена за стицање друге дипломе према студијским програмима других високошколских установа;
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио/ла интелектуалну својину других лица.

Потпис аутора

У Београду, 26.04.2022.

Дурковић Срђан

образац изјаве о истоветности штампане и електронске верзије докторског рада

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Срђан Дурковић

Број индекса 2015/5041

Студијски програм Електротехника и рачунарство

Наслов рада Архитектура пакетског свича за ефикасно комутирање уникаст и мултикаст саобраћаја

Ментор проф. др Зоран Чича

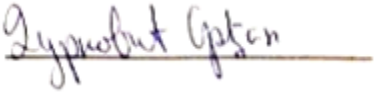
Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањивања у **Дигиталном репозиторијуму Универзитета у Београду**.

Дозвољавам да се објаве моји лични подаци везани за добијање академског назива доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис аутора

У Београду, 26.04.2022.



образац изјаве о коришћењу

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Архитектура пакетског свича за ефикасно комутирање

уникаст и мултикаст саобраћаја

која је моје ауторско дело

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигиталном репозиторијуму Универзитета у Београду и доступну у отвореном приступу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство (CC BY)
2. Ауторство – некомерцијално (CC BY-NC)
3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)
4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)
5. Ауторство – без прерада (CC BY-ND)
6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци.
Кратак опис лиценци је саставни део ове изјаве).

Потпис аутора

У Београду, 26.04.2022.

Зурковић Срђан

1. **Ауторство.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. **Ауторство – некомерцијално.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. **Ауторство – некомерцијално – без прерада.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. **Ауторство – некомерцијално – делити под истим условима.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. **Ауторство – без прерада.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. **Ауторство – делити под истим условима.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.