



УНИВЕРЗИТЕТ
У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ
НАУКА У НОВОМ САДУ



Иван Нинић

РУТИРАЊЕ УНУТАР АУТОНОМНИХ СИСТЕМА НА ПРИМЕРУ RIP ПРОТОКОЛА

ДИПЛОМСКИ РАД

Основне академске студије

Нови Сад, 2024.



UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION


Редни број, РБР :	
Идентификациони број, ИБР :	
Тип документације, ТД :	монографска публикација
Тип записа, ТЗ :	текстуални штампани документ
Врста рада, ВР :	дипломски рад
Аутор, АУ :	Иван Нинић
Ментор, МН :	доц. др Милана Бојанић
Наслов рада, НР :	Рутирање унутар аутономних система на примеру RIP протокола
Језик публикације, ЈП :	српски
Језик извода, ЈИ :	српски
Земља публикавања, ЗП :	Србија
Уже географско подручје, УГП :	Војводина
Година, ГО :	2024.
Издавач, ИЗ :	ауторски репринт
Место и адреса, МА :	Нови Сад, Факултет техничких наука, Трг Доситеја Обрадовића 6
Физички опис рада, ФО : (поглавља/страна/ цитата/табела/слика/графика/прилога)	5/22/11/0/16/0/0
Научна област, НО :	Електротехничко и рачунарско инжењерство
Научна дисциплина, НД :	Примењене рачунарске науке и информатика
Предметна одредница/Кључне речи, ПО :	протокол рутирања, симулација, RIP, рутер, рачунарска мрежа, табела рутирања, IP адреса
УДК	
Чува се, ЧУ :	Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад
Важна напомена, ВН :	
Извод, ИЗ :	У оквиру овог рада анализирано је понашање протокола рутирања под називом RIP (енг: Routing Information Protocol). За потребе анализе имплементирана је апликација за симулацију поједностављене верзије RIP протокола, чије функционисање је детаљно објашњено. Сprovedена су тестирања за типичне случајеве и добијени резултати су упоређени са очекивањем. На основу понашања симулације и њених излаза изведени су одређени закључци.
Датум прихватања теме, ДП :	
Датум одбране, ДО :	
Чланови комисије, КО :	Председник: ванр. проф. др Александар Селаков
	Члан: доц. др Себастијан Стоја
	Члан:
	Члан, ментор: доц. др Милана Бојанић
	Потпис ментора



UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	monographic publication
Type of record, TR :	textual printed document
Contents code, CC :	Bachelor thesis
Author, AU :	Ivan Ninić
Mentor, MN :	Milana Bojanić, Ph.D.
Title, TI :	Routing within autonomus systems on the example of the RIP protocol
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian
Country of publication, CP :	Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2024
Publisher, PB :	author's reprint
Publication place, PP :	Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	5/22/11/0/16/0/0
Scientific field, SF :	Electrical and Computer Engineering
Scientific discipline, SD :	Applied Computer Science and Informatics
Subject/Key words, S/KW :	routing protocols, simulation, RIP, router, cmputer network, routing table, IP address
UC	
Holding data, HD :	Library of Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Нови Сад
Note, N :	
Abstract, AB :	This paper analyzes the behavior of the routing protocol known as RIP (Routing Information Protocol). For the purpose of the analysis, an application was implemented to simulate a simplified version of the RIP protocol, and its functionality is explained in detail. Testing was conducted for typical scenarios, and the obtained results were compared with the expected outcomes. Based on the behavior of the simulation and its outputs, certain conclusions were drawn.
Accepted by the Scientific Board on, ASB :	
Defended on, DE :	
Defended Board, DB :	President: Aleksandar Selakov, PhD, assoc. prof
	Member: Sebastijan Stoja, PhD, assist. prof
	Member:
Member, Mentor:	Milana Bojanić, PhD, assist. prof
	Menthor's sign

	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА ЗАВРШНИ РАД	Датум:

(Податке уноси предметни наставник - ментор)

Врста студија:	<input type="checkbox"/> основне академске студије
Студијски програм:	Електроенергетски софтверски инжењеринг
Руководилац студијског програма:	Ванр. проф. др Александар Селаков

Студент:	Иван Нинић	Број индекса:	Е3 19/2014
Област:	Електротехничко и рачунарско инжењерство		
Ментор:	др Милана Бојанић, доцент		
НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА: <ul style="list-style-type: none"> - проблем – тема рада; - начин решавања проблема и начин практичне провере резултата рада, ако је таква проверанеопходна; 			

НАСЛОВ ЗАВРШНОГ РАДА:

Рутирање унутар аутономних система на примеру RIP протокола

ТЕКСТ ЗАДАТКА:

Задатак дипломског рада је проучавање принципа протокола за рутирање унутар аутономних система на Интернету, конкретно на примеру RIP протокола (*Routing Information Protocol*). Симулацијом повезаних рутера унутар једног аутономног система, приказати процес размене информација о могућим рутама до одређених мрежа. Приказати формирање табеле рутирања унутар рутера и њено ажурирање током процеса размене података са суседним рутерима. На примеру различитих пакета тестирати ваљаност пронађене путање до одредишта у задатој мрежи.

Руководилац студијског програма:	Ментор рада:

Примерак за: ☐ - Студента; ☐ - Ментора

Садржај

1. Увод.....	1
1.1 Контекст проблема	1
1.2 Опис проблема.....	2
1.3 Приступ	3
2. Опис коришћених технологија и алата	4
3. Опис решења проблема	5
3.1 Структура решења	5
3.2 Конфигурисање рачунарске мреже употребом графичког интерфејса.....	5
3.3 Учитавање података мреже из конфигурационе датотеке	10
3.4 Класна хијерархија и организација кода симулационог дела апликације.....	10
3.4 Формирање и освежавање табеле рутирања.....	11
3.5 Слање IP пакета кроз мрежу.....	13
3.6 Чување добијених симулационих података.....	14
3.7 Графички приказ резултата	14
4. Опис експерименталног тока.....	16
4.1 Формирање табеле рутирања на конкретном примеру мреже	16
4.2 Праћење рутирања конкретног IP пакета на примеру дате мреже	20
5. Закључак	22
Литература	23
Списак коришћених слика и табела	24

1. Увод

1.1 Контекст проблема

У оквиру рачунарских мрежа, један од основних елемената јесте рутер. Ако рачунарску мрежу представимо OSI (енг: *Open Systems Interconnection*) моделом, онда је рутер уређај који функционише на мрежном нивоу (енг: *network layer*) и чија је једна од основних улога управо усмеравање или рутирање (енг. *routing*) пакетског саобраћаја кроз рачунарску мрежу. Усмеравање или рутирање је поступак који има за циљ да обезбеди да све послате поруке стигну на своје одредиште што краћим путем, при томе водећи рачуна да се сама мрежа што мање оптерети. Начин на који ће рутер спроводити усмеравање порука дефинисан је управо протоколом рутирања. Протокол рутирања дефинише критеријум, стратегију и процедуре помоћу којих ће се вршити избор најбоље путање до неког одредишта у мрежи. Постоји већи број оваквих протокола, а у овом раду биће представљена симулација протокола под називом RIP (енг: *Routing Information Protocol*), који спада у групу протокола за рутирање унутар аутономних система на Интернету. Као метрику при рутирању пакета, овај протокол користи број скокова потребних да би пакет стигао до неке удаљене адресе, при чему сваки скок означава пролазак кроз један рутер. Дакле, RIP протокол тражи најбољу путању за неки пакет тако што минимизује број скокова потребних да би тај пакет стигао на одредиште. RIP представља класни протокол рутирања базиран на векторима удаљености (енг. *Distance Vector*) и један је од најстаријих стандардизованих протокола рутирања [1, 2].

Да би се јасније схватила улога и суштина RIP протокола потребно је исти ставити у контекст осталих протокола рутирања и разјаснити значење претходно наведених подела. Као што је прво поменуто RIP протокол спада у групу протокола за рутирање унутар аутономних система, други назив за ову врсту протокола је IGP (енг: *Interior Gateway Protocols*). Оно што карактерише протоколе из ове групе је то да функционишу унутар мреже под контролом једне организације у којој су јасно и једнозначно дефинисана правила и политике рутирања. Конкретно RIP протокол се може користити у оквиру мањих система због својих бројних ограничења у погледу скалабилности као и споре конвергенције. Комуникација између уређаја у оквиру различитих аутономних система дефинисана је неким од протокола из групе EGP (енг: *Exterior Gateway Protocols*) [3, 4].

Даље, речено је да је RIP класни протокол, што значи да унапред је познато који део IP адресе идентификује мрежу, а који крајњи уређај. Подмрежна маска (енг: *Subnet Mask*) која се користи је фиксна и одређена класом IP адресе. Другим речима, у оквиру RIP протокола приликом слања порука о ажурирању табела рутирања, рутери не прослеђују подмрежну маску јер је она већ дефинисана и позната. У другој својој верзији, RIP протокол (RIPv2), подржао је бескласно адресирање, као и коришћење променљивих подмрежних маски што је случај и код велике већине савремених протокола рутирања [5].

RIP протокол је означен да припада групи протокола заснованих на векторима удаљености. Протоколи из ове групе функционишу тако што рутери размењују векторе удаљености са својим непосредним суседима, на основу којих ажурирају своје табеле рутирања и тај поступак се периодично понавља. Овде треба приметити да рутери немају увид о потпуној топологији мреже, једини податак који добијају од суседних рутера јесу руте из њихових табела рутирања и удаљености тих рута. На основу тога, коришћењем Белман-Фордовог алгоритма спроводи се ажурирање табела, детаљније о овом процесу биће речи у наредним поглављима. Са друге стране, група протокола, која је заснована на стањима везе (енг: *Link State*) и користи се Дијкстриним алгоритмом за проналажење најкраћег пута у графу, ради тако што рутери међусобно размењују податке о стањима веза са својим суседима (енг: *LSAs - Link State Advertisements*) и на тај начин стичу увид о читавој топологији мреже и на основу тога доносе одлуке о избору најбољих путања до свих чворова исте. И код једне и код друге група протокола, критеријум за избор путање најчешће није само број скокова, односно број рутера у путањи, као што је то случај код RIP протокола. Код новијих протокола тај критеријум је обично сложенији и може да укључује више различитих параметара, као што су пропусни опсег везе, кашњење, проток и максимална дужина пакета. Генерално гледано протоколи засновани на векторима удаљености су једноставнији и захтевају мање рачунарских ресурса и мање оптерећују мрежу, док су протоколи везани за стања везе комплекснији, брже конвергирају и имају мање ограничења у смислу практичних ситуација које ови први нису у стању адекватно да разреше [6, 7].

На крају треба нагласити да се RIP протокол, у великој мери може сматрати превазиђеним и застарелим и његово коришћење у пракси представља реткост и углавном је замењено употребом напреднијих протокола попут OSPF (енг: *Open Shortest Path First*) и iBGP(енг: *Internal Border Gateway Protocol*) који нуде боље перформансе и флексибилност. Међутим овај протокол и даље има своје место у контексту разумевања рутирања и због своје релативне једноставности представља одличну почетну тачку за изучавање протокола рутирања генерално.

1.2 Опис проблема

Зарад израде симулације било потребно је прво дефинисати шта ће тачно од стандардне верзије RIP протокола бити имплементирано, јер је због комплексности теме било неопходно увести бројна ограничења, која би омогућила да се процес имплементације заврши у разумном временском року, али не на штету сагледавања и разумевања суштине самог протокола. Дакле у оквиру симулације биће имплементирана мрежа у којој сваки рутер поседује табелу рутирања која кратко речено представља списак IP адреса дестинација које дати рутер познаје уз додатне податке који говоре о томе куда и како проследити поруку која је приспела и која у свом заглављу има неку одредишну адресу. Имплементирани RIP протокол пре свега дефинише начин на који се табеле рутирања формирају и њихово ажурирање кроз време. Функционише користећи се Белман-Фордовим алгоритмом, дакле сваки рутер у одређеним временским размацама обавештава своје директне суседе о променама у својој табели рутирања и при томе сам ажурира своју табелу

рутирања на основу података које њему шаљу суседни рутери. На овај начин комуницирајући само са својим суседима рутери постепено добијају податке о чворовима из читаве мреже, тј. о томе где да усмеравају поруке које имају одређене одредишне IP адресе у свом заглављу без да умају увид у то каква је топологија мреже. Детаљније о начину функционисања имплементираног протокола у наредним поглављима.

1.3 Приступ

Основна идеја била је да се развије апликација која ће симулирати понашање рутера у мрежи који за генерисање и ажурирање својих табела рутирања користи RIP протокол. У симулацији сваки рутер је представљен посебном програмском нити и посебним UDP (енг: *User Datagram Protocol*) сервером. Тиме је постигнута аутономност функционисања рутера која се дешава и у реалном систему где сваки уређај функционише као посебна физичка, али пре свега као посебна логичка целина. На основу конфигурационог фајла, за сваки рутер, учитава се листа његових директних суседа са којима он даље комуницира искључиво слањем UDP порука, на тај начин рутер постепено добија податке о остатку мреже, проширује своју табелу рутирања новим уносима и евентуално прави измене у постојећим. Поред симулације самог RIP протокола, одрађена је и симулација слања једног IP пакета кроз мрежу у којој су рутери већ формирали своје табеле, тиме је тестирана исправност односно „квалитет” табела која су настале у претходно описаном процесу. Сви добијени симулациони подаци чувају се у текстуалним датотекама на основу којих се касније може правити детаљнија анализа. Формат и садржај ових датотека, биће посебно описан у наредним поглављима. Конфигурациони фајл из ког симулациони програм учитава топологију мреже и све податке о њој, добија се коришћењем посебне апликације у оквиру које преко интерактивног графичког интерфејса корисник креира мрежу и уноси податке о њој користећи понуђене опције којима визуелно и логички повезује елементе мреже, а из поменуте апликације, након што се креира конфигурациони фајл, покреће се и сам симулациони програм. Такође у овој апликацији се након завршене симулације, а на основу генерисаних излазних датотека, даје приказ добијених табела рутирања за сваки рутер, као и графички приказ путање пакета кроз мрежу за изабрану одредишну и изворишну адресу.

У наставку, у оквиру поглавља 2, наведене су кључне технологије и алати који су употребљени приликом израде решења. Поглавље 3 се односи на само решење, односно детаљно описује начин на који су израђени симулациони програм и корисничка апликација, пратећи ток њиховог коришћења, односно ток кретања података у систему, почевши од уноса конфигурације мреже од стране корисника, преко извршавања симулације, односно ажурирања табела рутирања и кретања пакета кроз мрежу, па све до приказа добијених резултата. Поглавље 4 се бави анализом симулационог дела апликације на конкретном примеру мреже, где се прати и објашњава како рутери у датом примеру формирају табеле рутирања и прослеђују пакете. У поглављу 5, читав проблем поново је сагледан у целини и изражени су коначни закључци, као и идеје за потенцијална проширења и наставак истраживања.

2. Опис коришћених технологија и алата

Сав програмски код симулационог програма писан је у језику C/C++ користећи се општим особинама објектно оријентисане парадигме програмирања. Разлог избора C/C++ језика био је у томе да се обезбеди прецизније конструисање потребних структура података као и потенцијално боље перформансе извршавања симулације уколико би била симулирана мрежа са великим бројем рутера. Такође сам пројекат је замишљен да буде коришћен и у едукативне сврхе, па би тако код написан у C/C++ језику био приступачнији већини студената, имајући у виду да се симулационе апликације везане за рачунарске мреже традиционално пишу у C програмском језику. Са друге стране, корисничка апликација преко које се задају параметри и управља симулацијом, писана је у језику C# уз коришћење WPF (енг: *Windows Presentation Foundation*) фрејмворка за креирање *desktop* апликација са графичким интерфејсом. Такође развојно окружење које је коришћено је *Visual Studio 2019*. У наставку су наведене неке од библиотека коришћених током израде симулационог програма.

Standard Template Library (STL) - Омогућава стандардизовано коришћење структура података у овом раду коришћене су STL библиотеке **<vector>** и **<string>**.

<thread> - Библиотека за рад са програмским нитима.

<filesystem> - Библиотека која омогућава манипулацију системима фајлова.

<winsock> - Библиотека која пружа могућност креирања мрежних апликација.

<regex> - Омогућава једноставно парсирање текстуалног садржаја

3. Опис решења проблема

3.1 Структура решења

Главни задатак био је омогућити кориснику да конфигурише рачунарску мрежу сачињену од рутера и крајњих уређаја, на једноставан и лак начин, затим да покрене симулацију која ће одредити табеле рутирања за све рутере у мрежи и симулирати кретање једног пакета кроз такву мрежу. Затим је било потребно приказати добијене резултате у форми која би била једноставна и читљива, а да се притом не изгуби ништа од детаља који су од значаја. Имајући све ово у виду, као што је већ речено, решење је подељено у две одвојене апликације, једну која врши саму симулацију и која ће у наставку бити означавана као симулациони програм или симулациони део апликације и другу која омогућава интеракцију са корисником и графички приказ резултата на коју ће се даље у тексту реферисати као на кориснички програм или кориснички део апликације. У поглављима која следе ове две апликације биће анализирани упоредно, пратећи ток извршавања програма, почевши од тога да је на почетку потребно генерисати конфигурациони фајл што представља задатак корисничке апликације, преко учитавања конфигурационог фајла и саме симулације што је задатак симулационог програма, па до тога да је на крају неопходно добијене резултате приказати кориснику, што опет представља одговорност корисничке апликације.

3.2 Конфигурисање рачунарске мреже употребом графичког интерфејса

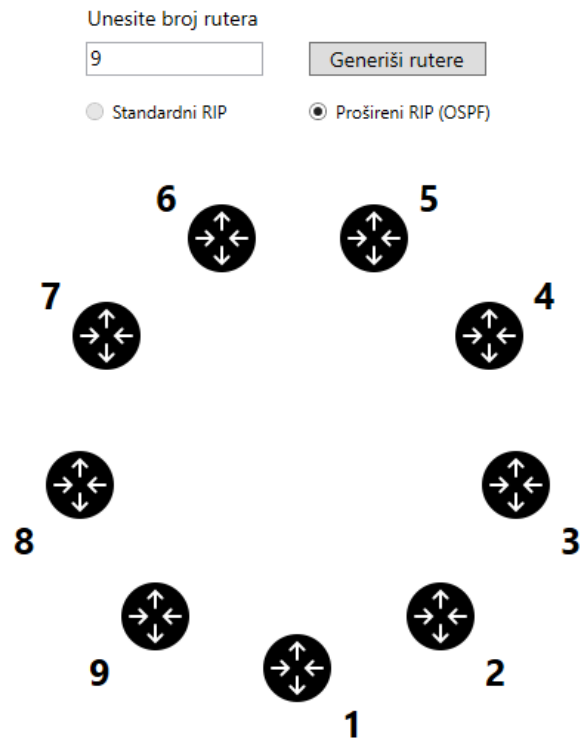
На самом почетку било је потребно конфигурисати рачунарску мрежу над којом ће се спроводити симулација. Мрежа је представљена скупом рутера који су међусобно повезани. Сваки рутер има своју локалну мрежу сачињену од рачунара који представљају крајње кориснике, рутер повезује локалну мрежу са другим рутерима, а преко њих и са свим осталим локалним мрежама. Потпун опис читаве мреже дат је у виду конфигурационе текстуалне датотеке **config.txt**. Подаци у датотеци **config.txt** груписани су око сваког појединачног рутера. За сваки рутер наводи се његов идентификациони број, листа интерфејса које рутер поседује као и листа уређаја који се налазе у локалној мрежи која је повезана на дати рутер. Изглед дела конфигурационе датотеке који се односи на један рутер, приказан је на слици 3.1.

```
router_id=105
105.1.4.1|104.1.2.1|104|1054|1|1
105.1.3.1|106.1.2.1|106|1053|1|1
105.1.2.1|102.1.2.1|102|1052|1|1
105.1.1.1|0.0.0.0|000|1051|0|0
-----
105.1.1.2
105.1.1.3
#####
```

Слика 3.1. Изглед дела конфигурационе датотеке

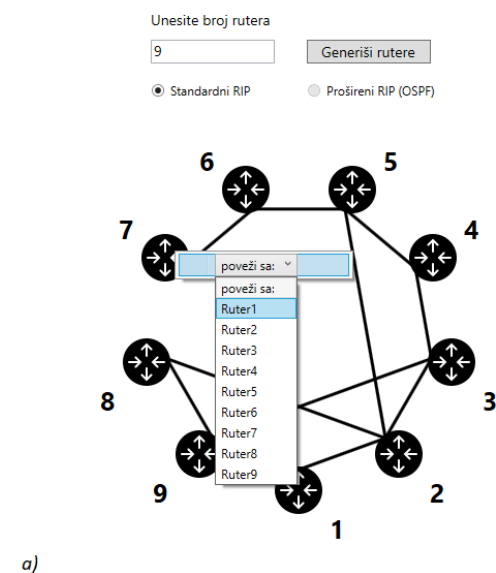
Као што се може уочити на слици 3.1, подаци за сваки интерфејс дати су у једној врсти раздвојени знаковима усправне црте „|” и састоје се редом од: IP адресе самог интерфејса, IP адресе интерфејса на другој страни везе, ID-а рутера са којим дати интерфејс омогућава везу, ID-а интерфејса, типа интерфејса и вредности која представља тежину везе. Тип интерфејса може имати вредност 0 или 1, што означава да се ради о интерфејсу ка локалној, односно глобалној мрежи. Уколико је интерфејс типа 1, тј. ради се о интерфејсу који повезује рутер са другим мрежама, додаје му се и вредност „тежине везе” која у случају стандардног RIP протокола увек има вредност 1, док у случају да се ради о проширеној верзији протокола ова вредност је неки позитиван цео број. Уколико је у питању интерфејс типа 0, односно интерфејс који рутер спаја са локалном мрежом тежина везе се увек поставља на вредност 0. Такође треба напоменути и да је ID интерфејса податак искоришћен искључиво за интерну употребу у симулацији и није од значаја за опис конфигурације мреже. Након листе интерфејса, следи Листа уређаја из локалне мреже, подаци о овим уређајима састоје се само од њихових IP адреса.

Конфигурациони фајл **config.txt** настаје на основу поставки које се бирају у оквиру корисничке апликације, употребом графичког интерфејса. Корисник пре свега бира број рутера у мрежи и верзију протокола рутирања која ће бити коришћена приликом симулације, а затим преко интерактивног приказа међусобно повезује рутере, такође омогућено му је и додавање уређаја који ће бити прикључени на рутере. Када корисник изабере број рутера у мрежи и верзију протокола рутирања и кликне на дугме „Generiši rutere”, позива се метода **private void GenerateRouters(object sender, RoutedEventArgs e)** у оквиру које се праве објекти класе *Router* који поред података о рутеру потребних за израду конфигурационог фајла садрже и објекте за визуелно представљање истих у оквиру графичког интерфејса. Дакле након избора жељеног броја рутера, на екрану се појављују иконице рутера распоређене у кружном поретку и нумерисане бројевима, почевши од броја 1, а уколико је раније већ биран број рутера и прављена нека конфигурација мреже, њен приказ се брише и поставља се приказ нове. Овако распоређене рутере сада је могуће међусобно повезивати и додавати им уређаје, такође у сваком тренутку могуће је видети и све податке о сваком од њих. Изглед дела корисничког интерфејса након што се одабере жељени број рутера и верзија протокола рутирања приказан је на слици 3.2.

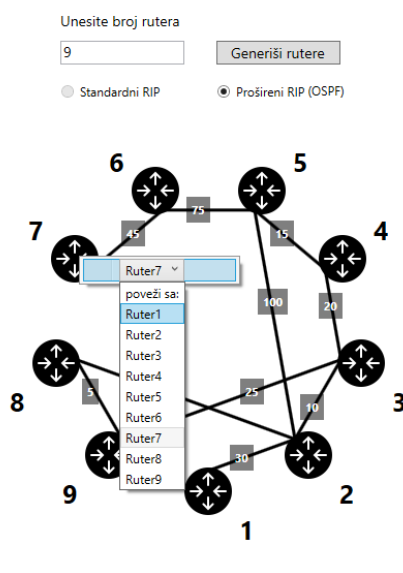


Слика 3.2. Изглед дела корисничког интерфејса, након избора броја рутера

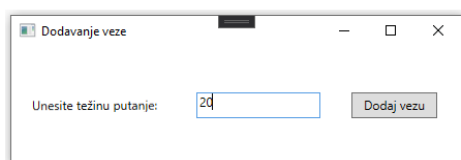
Повезивање рутера, спроводи се на следећи начин: десним кликом на иконицу неког од рутера, добија се падајући мени у оквиру кога се појављују „имена“ свих осталих рутера, затим се избором једног од понуђених остварује њихово повезивање, а уколико је верзија протокола рутирања „проширени“ RIP након избора рутера за повезивање потребно је још задати и „трошкове линка“ између њих уношењем целоброје вредности у поље које се појављује у новоотвореном прозору. Унета тежина биће придружена интерфејсима рутера који се повезују и наћи ће се у конфигурационом фајлу, а у случају стандардног RIP протокола, ова вредност ће се увек постављати на 1, као што је већ раније речено. Веза односно линк се графички представља појављивањем линије између иконица та два рутера, на којој у случају да је биран проширени RIP протокол, стоји и број који представља нумеричку вредност „трошка“ дате везе. Логичко повезивање рутера спроводи се у методи `private void Connect_routers_logically(Router r1, Router r2, Line myLine)` и своди се на додавање интерфејса у оквиру рутера `r1` и `r2` којима се дефинише да је рутер `r1` повезан са `r2` и обрнуто. Док се графичко повезивање обавља у методи `private void ConnectRouters(double x1, double y1, double x2, double y2, Line myLine)` којом се исцртава линија која представља физичку везу између два рутера. Прикази начина на који корисник повезује рутере у мрежи, дати су на слици 3.3.



a)



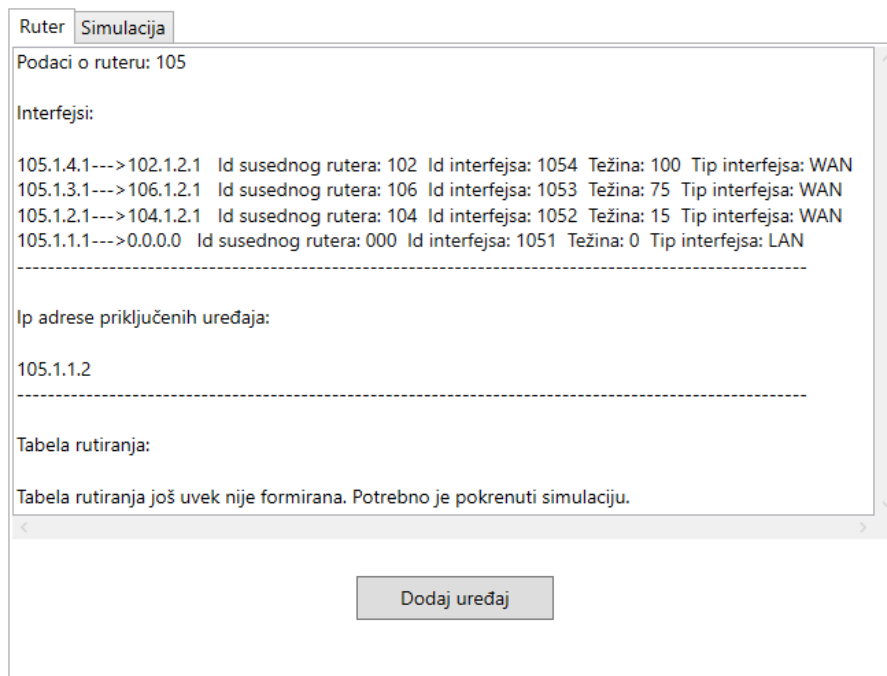
b)



e)

Слика 3.3. приказ а) Изглед интерфејса за повезивање рутера ако је изабран протокол стандардни RIP, б) изглед интерфејса за повезивање рутера ако је изабран протокол проширени RIP в) прозор који се отвара након избора рутера за повезивање у случају да је изабран протокол проширени RIP

Када су рутери додати у мрежу, податке о сваком од њих могуће је добити једноставним кликом на иконицу рутера, након чега се у десној половини прозора апликације, у оквиру картице „Ruter“ појављују сви подаци о изабраном рутеру, који укључују идентификациони број рутера, списак свих интерфејса, листу крајњих уређаја прикључених на локалну мрежу рутера, као и табелу рутирања, коју је могуће видети тек након што се симулација формирања табела заврши. Генерисање приказа података о рутеру извршава се у оквиру методе **private void ViewRouterInfo(Router r)**. Изглед картице „Ruter“ у оквиру које су приказани сви подаци о изабраном рутеру, пре пократања симулације дат је на слици 3.4.



Слика 3.4. Изглед картице “Ruter” пре пократања симулације

Сваком додатом рутеру аутоматски се придружује по један кориснички уређај прикључен на његову локалну мрежу, а по потреби је могуће додати и произвољан број додатних уређаја. Додавање нових уређаја спроводи се кликом на дугме „Dodaj uređaj” које се налази у делу екрана непосредно испод приказа информација о рутеру. IP адресе уређаја који се придружују неком рутеру конструишу се у форми: **id_rutera + “.1.1.” + redni_br_uređaja + 1**, па би тако адреса првог уређаја додатог рутера са идентификационим бројем 101 била 101.1.1.2, другог 101.1.1.3, итд. Овакав начин одабира IP адреса превасходно служи томе да кориснику олакша даље спровођење симулације и тумачење добијених резултата. Претходно описано додавање уређаја неком рутеру одвија се у методи **private void AddDevice(object sender, RoutedEventArgs e)**.

Симулација формирања табела рутирања и слања једног пакета кроз мрежу су у оквиру симулационог дела апликације обједињене и дешавају се паралелно једна са другом, с тим што слање пакета започиње након предефинисаног броја већ извршених итерација симулације која се односи на освежавање табела рутирања. Стога је пре покретања симулације неопходно изабрати и параметре који се односе на део симулације везан за слање једног пакета кроз мрежу. Под овим се подразумева избор почетне и крајње адресе путање којом ће пакет проћи након што буду формиране табеле рутирања. Адресе је потребно унети у текст поља која се налазе у картици “Simulacija” и неопходно је изабрати адресе неких од рутерима већ додатих крајњих уређаја. Тек након тога могуће је покренути симулациони процес кликом на дугме „Pokreni simlaciju”. Симулациони програм покреће се методом **private void StartSimulation(object sender, RoutedEventArgs e)** која на основу изабраних поставки прво генерише улазни симулациони фајл **config.txt**, а онда покреће апликацију написану у програмском језику C++ чији рад ће бити детаљно описан у наредним поглављима. Такође, поред датотеке **config.txt**, генерише се још и датотека **routers_info.txt** у којој се чува број рутера над којима се врши симулација као и датотека **src_dest.txt** у којој се

чувају претходно поменуте одредишна и крајња адреса путање које је корисник унео. Ове три датотеке заједно представљају улазне податке симулације.

Након покретања симулационе апликације, чека се њен завршетак да би се потом на основу њених излазних фајлова генерисали и графички и текстуално приказали добијени резултати. Ови резултати представљају пре свега израчунате табеле рутирања за сваки од рутера, као и на основу њих изабрану најкраћу путању којом се пакет кретао да би прешеао пут од изворишне до одредишне адресе. Детаљнији опис приказа добијених резултата и метода које извршавају овај задатак, дат је у поглављу 3.6.

3.3 Учитавање података мреже из конфигурационе датотеке

Симулациони програм у оквиру класе *Network* методом **`void load_routers(const char* filename)`** учитава податке из текстуалне датотеке **`config.txt`**, парсира их и учитава у радну меморију у виду вектора рутера, који је поље саме класе *Network*. На основу садржаја конфигурационе датотеке праве се објекти класе *Router* који се затим додају у већ поменути вектор рутера у оквиру класе *Network*. Композиција класе *Router* осмишљена је тако да одговара формату података о рутеру који се чува у конфигурационој датотеци, ова класа детаљније је обрађена у наредним поглављима.

3.4 Класна хијерархија и организација кода симулационог дела апликације

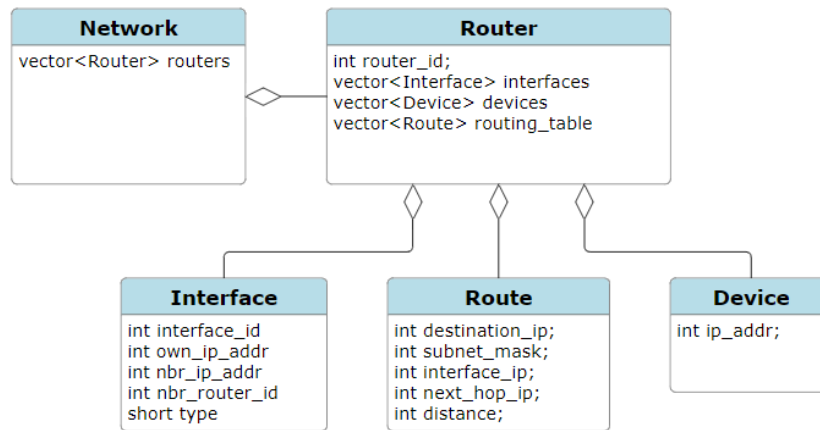
Као што је већ речено у претходним поглављима сви подаци о мрежи груписани су око класе *Router* па се тако читава мрежа може представити колекцијом објеката исте. Класа *Router* поседује следећа поља:

`vector<Interface> interfaces` - Вектор интерфејса које дати рутер поседује

`vector<Device> devices` - Вектор уређаја који се налазе у локалној мрежи рутера

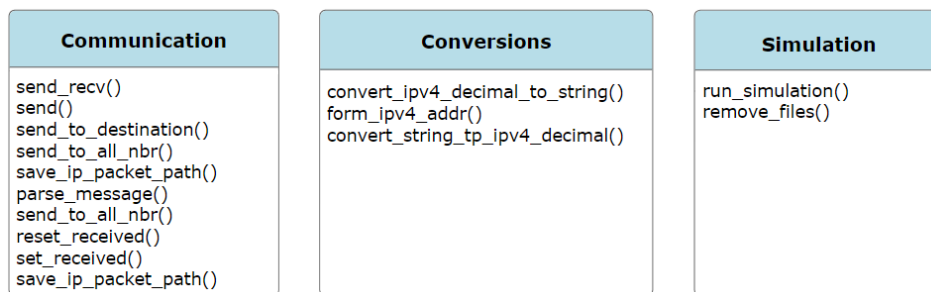
`vector<Route> routing_table` - Вектор путања које рутер познаје

Организација читавог кода везаног за представу мреже приказана је на слици 3.5.



Слика 3.5. Дијаграм кључних класа за дефинисање мрежне топологије

Поред наведених класа које су директно везане за модел мреже, коришћене су још три помоћне класе *Simulation*, *Communication* и *Conversions* у овим класама смештен је код везан за покретање саме симулације, комуникацију између рутера која се одвија по дефинисаном протоколу као и све потребне конверзије података. Садржај ових класа приказан је на слици 3.6.



Слика 3.6. Класе задужене за управљање симулацијом, размену података и конверзије бројева

3.4 Формирање и освежавање табеле рутирања

Пре детаљније анализе поступка добијања табела рутирања односно анализе симулације самог протокола који се овде разматра потребно је дефинисати форму табеле рутирања која ће у овом случају бити коришћена. Дакле основна улога табеле рутирања је у томе да чува податке о путањама до мрежа које један рутер „познаје“. Ово значи да је свакој мрежи која се налази у табели придружена IP адреса „следећег скока“ (енг. *next hop*) која представља адресу на коју рутер прослеђује пакет адресиран на дату мрежу. Поред овог податка у табели се наводи преко ког интерфејса рутера се пакет прослеђује као и удаљеност, у броју скокова, до мреже. Такође табела садржи и колону која представља коришћену маску којом се операцијом логичког „И“ (логичке конјункције) IP адреса адресираног уређаја своди на локалну мрежу у којој се уређај налази, што омогућава памћење података само о мрежама, а не о свим уређајима појединачно. Додатно, ако се адресирани уређај налази у мрежи која је директно повезана на рутер, за поље следећег скока се наводи посебна вредност (енг. "On-link"). Поред тога, у табели је потребно дефинисати и посебну руту тзв. предефинисану руту којом ће се прослеђивати пакети који су

адресирани на мрежу коју рутер у датом тренутку не познаје. Унутар предефинисане руте за адресу следећег скока у овом случају изабрана је адреса првог суседног рутера. Такође треба приметити да су поља *Destination* и *Subnet mask* за ову предефинисану руту **0.0.0.0** што значи да ће се приликом примене операције логичког „И” маском над било којом адресом добити вредност **0.0.0.0**. Пример изгледа табеле рутирања за један рутер дата је на слици 3.7.

Destination: 0.0.0.0	Subnet mask: 0.0.0.0	Next hop: 101.1.2.1	Interface: 102.1.5.1	Metric: 0
Destination: 102.1.1.0	Subnet mask: 255.255.255.0	Next hop: On-link	Interface: 102.1.1.1	Metric: 0
Destination: 101.1.1.0	Subnet mask: 255.255.255.0	Next hop: 101.1.2.1	Interface: 102.1.5.1	Metric: 1
Destination: 105.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.2.1	Interface: 102.1.4.1	Metric: 1
Destination: 108.1.1.0	Subnet mask: 255.255.255.0	Next hop: 108.1.2.1	Interface: 102.1.2.1	Metric: 1
Destination: 103.1.1.0	Subnet mask: 255.255.255.0	Next hop: 103.1.2.1	Interface: 102.1.3.1	Metric: 1
Destination: 109.1.1.0	Subnet mask: 255.255.255.0	Next hop: 103.1.2.1	Interface: 102.1.3.1	Metric: 2
Destination: 104.1.1.0	Subnet mask: 255.255.255.0	Next hop: 103.1.2.1	Interface: 102.1.3.1	Metric: 2
Destination: 107.1.1.0	Subnet mask: 255.255.255.0	Next hop: 101.1.2.1	Interface: 102.1.5.1	Metric: 2
Destination: 106.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.2.1	Interface: 102.1.4.1	Metric: 2

Слика 3.7. Изглед табеле рутирања једног рутера

Након учитавања података из конфигурационе датотеке, над сваким објектом класе *Router* позива се метода **void init_routes()** која има за циљ да табелу рутирања постави у иницијално стање, односно да у табелу рутера дода локалну мрежу која је директно прикључена на тај рутер и да постави одговарајуће податке за њу, као и предефинисану руту за коју се бира први суседни рутер. Затим се покреће симулација током које рутери међусобно комуницирају, размењују податке о својим суседима и на тај начин освежавају своје табеле рутирања.

Симулација се покреће методом **static void run_simulation(Network& network)** класе *Simulation*. У овој методи се за сваки рутер из мреже прави и покреће посебна програмска нит. Програмска нит као параметар прима методу **static void send_recv(Router& router, int source_ip, int destination_ip)** која је задужена слање и пријем података. У методи **send_recv** дефинисан је UDP сервер који све време рада прихвата и акумулира поруке добијене од суседних рутера, а такође је дефинисан и UDP клијент који у одређеним тренуцима шаље податке свим суседним рутерима. Када се симулација покрене њен ток изгледа овако: Сваки рутер одмах након што иницијализише свој UDP сервер, шаље свим суседима поруке које садрже све податке о његовој табели рутирања, након тога прелази у стање чекања у којем прикупља поруке које му шаљу суседни рутери, када је рутер примио поруке од свих суседа, започиње ажурирање своје табеле рутирања након чега, опет шаље, сада ажуриране податке из табеле рутирања свим својим суседима и опет прелази у стање чекања. Овај процес се понавља током читавог трајања симулације. Само ажурирање табеле рутирања спроводи се методом **void update(vector<string>& messages)** класе *Router* која као параметар прима вектор стрингова који представљају добијене поруке од суседних рутера. Ова метода за сваку поруку из вектора позива посебну методу из класе *Router*, **void update_table(string message)** у којој се порука парсира и на основу добијених података се освежава табела рутирања. Порука коју рутер добија од свог суседа садржи увек читаву табелу рутирања тог суседа, док се у методи **update_table** ова порука разбија на појединачне руте из табеле. Затим се за сваку путању проверава да ли таква већ постоји у табели рутирања текућег рутера, ако не постоји одмах се додаје, а уколико постоји проверава се да ли предложена рута има мању удаљеност до дестинације, од руте која се већ налази у табели, ако је то

тачно, онда се у уместо постојеће у табелу додаје нова путања, а ако није, задржава се стара. Овде треба нагласити да уколико су тежине свих веза постављене на вредност 1 што је случај код стандардног RIP протокола, алгоритам ће за најбољу руту увек бирати ону са најмање скокова, док ће у случају да тежине неких веза имају вредности веће од 1, што ће се догађати код проширеног RIP протокола, увек бити бирана она путања која има најмању суму тежина.

На овај начин из итерације у итерацију рутер добија нове податке о мрежи, док алгоритам рутирања не конвергира у стабилно стање, тј. до тренутка када сваки рутер у својој табели има адекватне информације о читавој мрежи. Овакав приступ формирању табела и размени информација између рутера суштински представља имплементацију Белман-Фордовога алгоритма, што нам даје и теоријску основу да претпостављамо да се до стабилног стања заиста стиже у коначно малом броју итерација. У наставку, у оквиру приказа експерименталног тока, биће детаљније анализиран конкретан пример табеле рутирања и њеног ажурирања кроз време и за случај да су вредности свих веза истих тежина (стандардни RIP) и за случај у коме се те тежине разликују (проширени RIP) [8, 9].

3.5 Слање IP пакета кроз мрежу

Поред претходно описаног процеса добијања табела рутирања, у оквиру симулациониг програма постоји и део који се односи на пролазак једног пакета кроз мрежу између два претходно одабрана уређаја у различитим локалним мрежама. Овај део симулације спроводи се паралелно са претходним с тим што слање пакета има више смисла започети тек кад формирање табела рутирања конвергира у стабилно стање, односно када су табеле у потпуности формиране. На овај начин, пратећи кретање пакета који се шаље проверава се да ли су табеле рутирања формиране на задовољавајући начин. Део кода који је одговоран за симулацију прослеђивања IP пакета такође се налази у методи **send_recv** али његово извршавање започиње тек када се испуни довољан, унапред дефинисани број итерација алгоритма описаног у претходном поглављу. Када симулација прослеђивања IP пакета започне за сваку поруку која пристигне на UDP сервер рутера, проверава се њено заглавље, чиме се одређује да ли је у питању порука суседног рутера везана за ажурирање табеле рутирања или је у питању IP пакет који треба проследити. Уколико се испостави да је прispела порука IP пакет, исти се пре прослеђивања парсира методом **static void parse_message(string message, Router& router)** и на тај начин се сазнаје одредишна адреса у чијем правцу ће затим пакет бити прослеђен. Метода задужена за само слање IP пакета назива се **static void send_to_destination(int destination_ip, string message, Router& router, string log_message)** и њено извршавање директно се ослања на претходно формиране табеле рутирања, тако да сваки рутер након што прими IP пакет на основу своје табеле сам одреди адресу на коју ће проследити прispели пакет. Када се одредишна адреса пакета поклопи са адресом локалне мреже прикључене на текући рутер, прослеђивање се прекида и исписује се садржај из дела пакета у коме се налазе подаци, што је знак да је пакет стигао на одредиште. Формат IP пакета коришћеног у сврхе овог пројекта приказан је на слици 3.8.

header	destination IP addr	source IP addr	data
--------	---------------------	----------------	------

Слика 3.8. Формат IP пакета коришћен у симулацији

Са слике 3.8 видимо да пакет има форму која је суштински слична форми IP пакета какав он заиста и јесте у реалном систему, односно поседује IP адресу одредишта (*destination*), IP адресу извора (*source*), као и део који се односи на саме податке који се преносе (*data*), уз изостављање неких других техничких детаља, који нису битни за саму суштину симулације рутирања. Такође додатно, пакет има и заглавље (*header*) које је додато да би олакшало разликовање порука које представљају IP пакет од порука које преносе информацију о ажурирању табеле рутирања, о којима је било речи у претходном поглављу.

3.6 Чување добијених симулационих података

Сви резултати добијени симулацијом чувају се у одговарајућим текстуалним датотекама. За сваки рутер прави се по једна датотека у којој се чувају све промене табеле рутирања тог рутера. Након сваког ажурирања табеле позива се метода **void export_routing_table (chrono:: high_resolution_clock:: duration d)** која снима тренутни садржај читаве табеле у текстуални фајл, такође поред садржаја табеле, чува се и симулационо време, па се на тај начин може пратити и сам темпо промена које су се догађале. Детаљнији опис и анализа добијених фајлова, следе у поглављу 4 у коме ће бити обрађен један конкретан симулациони случај. Поред чувања података о табелама, такође се снимају и резултати симулације који се односе на кретање IP пакета кроз мрежу. Ови подаци чувају се у фајлу под називом **packet_path.txt**. У овој датотеци редом су излистане информације о рутерима кроз које је IP пакет пролазио. Овај фајл детаљније се анализира у поглављу 4.2.

3.7 Графички приказ резултата

Као што је већ речено подаци добијени након симулације, чувају се у фајловима чији називи одговарају идентификационим бројевима рутера, у овим датотекама налазе се све верзије табела рутирања сваког појединачног рутера, тј. може се водити како се табела мењала кроз време. У оквиру корисничког дела апликације и приказа података о рутеру као што је наглашено у поглављу 3.1 након што се симулација заврши, приказује се и табела рутирања изабраног рутера, у овом случају узима се последња верзија табеле из излазног фајла, остале верзије се не приказују, али корисник им зарад неке дубље анализе може приступити директно, а у наредном поглављу биће управо анализирано, како се табела мења у времену и како се стиже до коначног изгледа исте. Приказ табеле у корисничкој апликацији дат је у истој форми у којој је и излазни фајл јер је такав приказ био сасвим погодан за ту сврху, пример приказане табеле рутирања за изабрани рутер, дат је на слици 3.9.

Ruter

Simulacija

Tabela rutiranja:

Destination: 0.0.0.0	Subnet mask: 0.0.0.0	Next hop: 106.1.2.1	Interface: 105.1.4.1	Metric: 0
Destination: 105.1.1.0	Subnet mask: 255.255.255.0	Next hop: On-link	Interface: 105.1.1.1	Metric: 0
Destination: 102.1.1.0	Subnet mask: 255.255.255.0	Next hop: 102.1.2.1	Interface: 105.1.2.1	Metric: 1
Destination: 104.1.1.0	Subnet mask: 255.255.255.0	Next hop: 104.1.3.1	Interface: 105.1.3.1	Metric: 1
Destination: 106.1.1.0	Subnet mask: 255.255.255.0	Next hop: 106.1.2.1	Interface: 105.1.4.1	Metric: 1
Destination: 107.1.1.0	Subnet mask: 255.255.255.0	Next hop: 106.1.2.1	Interface: 105.1.4.1	Metric: 2
Destination: 101.1.1.0	Subnet mask: 255.255.255.0	Next hop: 102.1.2.1	Interface: 105.1.2.1	Metric: 2
Destination: 108.1.1.0	Subnet mask: 255.255.255.0	Next hop: 102.1.2.1	Interface: 105.1.2.1	Metric: 2
Destination: 103.1.1.0	Subnet mask: 255.255.255.0	Next hop: 102.1.2.1	Interface: 105.1.2.1	Metric: 2
Destination: 109.1.1.0	Subnet mask: 255.255.255.0	Next hop: 104.1.3.1	Interface: 105.1.3.1	Metric: 3

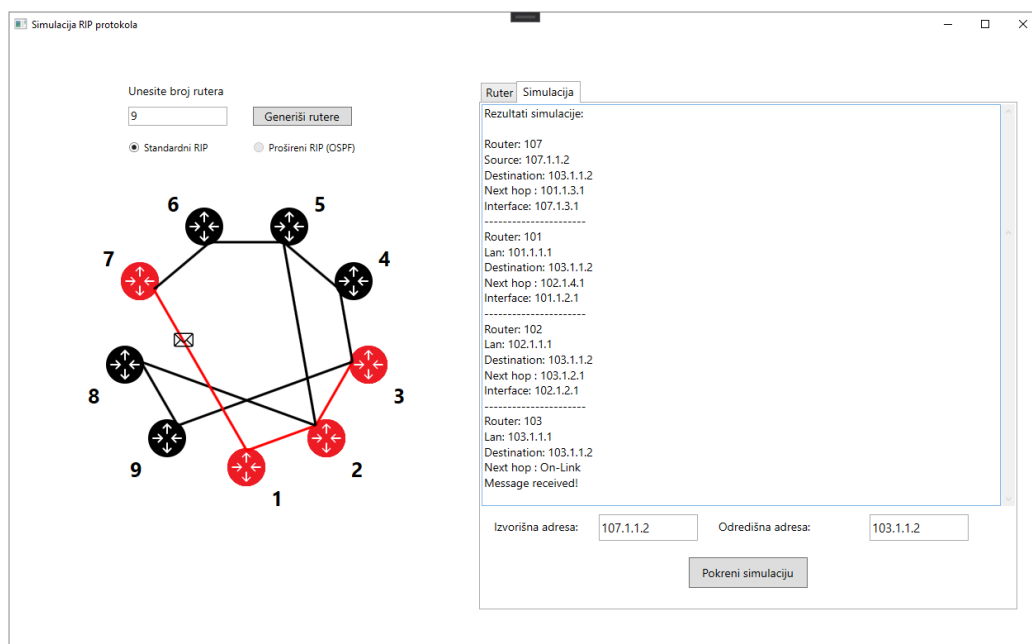
<

>

Dodaj uredaj

Слика 3.9. Пример приказане табеле рутирања за избрани рутер

Када је реч о добијеним резултатима везаним за кретање пакета кроз мрежу они се такође директно приказују у форми у којој су сачуване у фајлу **packet_path.txt**. Ови подаци приказују се у оквиру картице „Симулација“ након што се симулациони програм заврши. Поред самог текстуалног приказа који редом наводи све рутере и интерфејсе преко којих је пакет усмераваан до свог одредишта, обезбеђен је и графички приказ саме путање на дијаграму мреже, тако што је путања којом је пакет прошао обојена црвеном бојом, као и анимација кретања пакета дуж путање која је израчуната. Исцртавање путање на основу добијених података из датотеке **packet_path.txt** дешава се у методи **private void DrawPath(string path_str)** која се позива одмах након што је завршен симулациони програм, док је анимација кретања пакета реализована у оквиру методе **public async void MoveImage(Image message_img)**. Пример графичког и текстуалног приказа резултата везаних за део симулације који се односи на кретање пакета кроз мрежу дат је на слици 3.10.

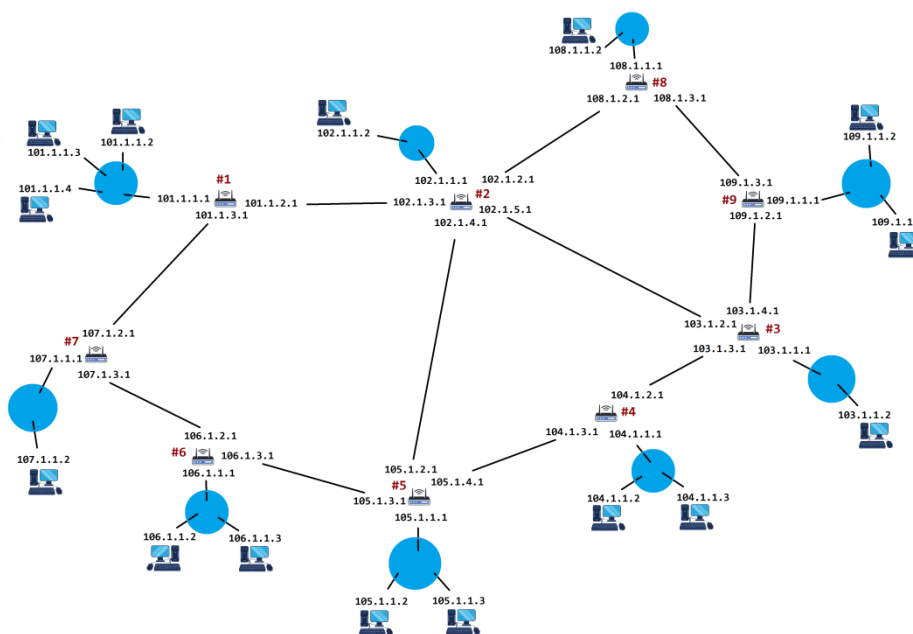


Слика 3.10. Пример графичког и текстуалног приказа резултата везаних за кретање пакета

4. Опис експерименталног тока

4.1 Формирање табеле рутирања на конкретном примеру мреже

Посматраће се конкретна мрежа чија је конфигурација дата на слици 4.1. Уношењем података и покретањем симулације за овакву мрежу и за стандардни RIP протокол добијени су одговарајући симулациони подаци који ће бити детаљно анализирани на примеру рутера бр. 6.



Слика 4.1. Конфигурација мреже коришћене при симулацији

Дакле посматрамо садржај излазног текстуалног фајла који се односи на рутер бр. 6 (на слици означен са #6, поред иконице рутера), након завршене симулације која је трајала 10 итерација. Приказ садржаја датотеке за прве 4 итерације дат је на слици 4.2. За посматрани рутер у датој мрежи, стандардни RIP протокол рутирања улази у стабилно стање већ након 4 итерације, па је у остатку садржаја датотеке само понављана табела рутирања из 4. итерације.

```

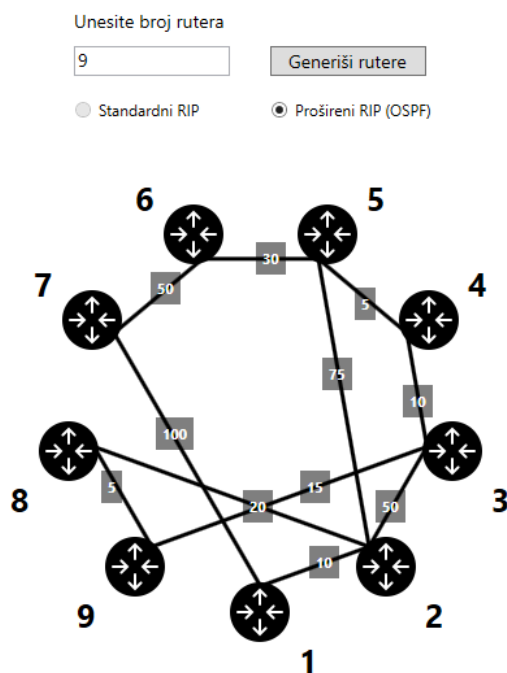
183
Destination: 0.0.0.0      Subnet mask: 0.0.0.0      Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 0
Destination: 106.1.1.0    Subnet mask: 255.255.255.0  Next hop: On-link        Interface: 106.1.1.1      Metric: 0
Destination: 107.1.1.0    Subnet mask: 255.255.255.0  Next hop: 107.1.3.1      Interface: 106.1.2.1      Metric: 1
Destination: 105.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 1
308
Destination: 0.0.0.0      Subnet mask: 0.0.0.0      Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 0
Destination: 106.1.1.0    Subnet mask: 255.255.255.0  Next hop: On-link        Interface: 106.1.1.1      Metric: 0
Destination: 107.1.1.0    Subnet mask: 255.255.255.0  Next hop: 107.1.3.1      Interface: 106.1.2.1      Metric: 1
Destination: 105.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 1
Destination: 104.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 2
Destination: 102.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 2
Destination: 101.1.1.0    Subnet mask: 255.255.255.0  Next hop: 107.1.3.1      Interface: 106.1.2.1      Metric: 2
439
Destination: 0.0.0.0      Subnet mask: 0.0.0.0      Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 0
Destination: 106.1.1.0    Subnet mask: 255.255.255.0  Next hop: On-link        Interface: 106.1.1.1      Metric: 0
Destination: 107.1.1.0    Subnet mask: 255.255.255.0  Next hop: 107.1.3.1      Interface: 106.1.2.1      Metric: 1
Destination: 105.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 1
Destination: 104.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 2
Destination: 102.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 2
Destination: 101.1.1.0    Subnet mask: 255.255.255.0  Next hop: 107.1.3.1      Interface: 106.1.2.1      Metric: 2
Destination: 103.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 3
Destination: 108.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 3
566
Destination: 0.0.0.0      Subnet mask: 0.0.0.0      Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 0
Destination: 106.1.1.0    Subnet mask: 255.255.255.0  Next hop: On-link        Interface: 106.1.1.1      Metric: 0
Destination: 107.1.1.0    Subnet mask: 255.255.255.0  Next hop: 107.1.3.1      Interface: 106.1.2.1      Metric: 1
Destination: 105.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 1
Destination: 104.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 2
Destination: 102.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 2
Destination: 101.1.1.0    Subnet mask: 255.255.255.0  Next hop: 107.1.3.1      Interface: 106.1.2.1      Metric: 2
Destination: 103.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 3
Destination: 108.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 3
Destination: 109.1.1.0    Subnet mask: 255.255.255.0  Next hop: 105.1.3.1      Interface: 106.1.3.1      Metric: 4

```

Слика 4.2. Промене табеле рутирања рутера бр. 6 током прве четири итерације за стандардни RIP протокол

Видимо да је у првој итерацији рутер бр. 6 сазнао за мреже **107.1.1.0** и **105.1.1.1** које су директно повезане на његове непосредне суседе, рутер бр. 7 и рутер бр. 5. У следећој итерацији добио је податке и о мрежама доступним преко суседних рутера његових непосредних суседа, односно мрежама **102.1.1.0** и **104.1.1.0** преко рутера бр. 5, а **101.1.1.0** преко рутера бр. 7. Даље, у трећој и четвртој итерацији настављен је исти процес, рутер је постепено добијао нове податке од својих суседа, оним темпом којим су и ти рутери проширивали своје табеле рутирања. Па је тако на пример, у другој итерацији рутер бр. 5 сазнао за мрежу **103.1.1.0** преко рутера бр. 4, да би је затим у трећој итерацији проследио посматраном рутеру бр. 6, што управо и видимо са слике 4.2, дакле мрежа **103.1.3.0** се у табели рутирања рутера бр. 6, појављује тек након треће итерације. У четвртој итерацији процес размене података између рутера је конвергирао до стабилног стања, односно рутер бр. 6 је сазнао за све локалне мреже које постоје и уједно и најближе путеве до њих, па је стога као што је већ речено, остатак излазне датотеке изостављен због понављања записа. Након сваке итерације, пре исписа података из табеле рутирања, додатно је исписиван број који представља време протекло од почетка симулације изражено у милисекундама. Овај податак овде нема посебну сврху осим да пружи информацију о протоку времена, али би могао бити употребљен у неком даљем проширењу апликације, где би послужио за израду анимације која би приказивала промене табеле рутирања у времену.

Сада ћемо посматрати мрежу идентично повезану као мрежа са слике 4.1, само са подешеним тежинама грана односно трошковима веза. Њен приказ, овај пут у самој корисничкој апликацији, дат је на слици 4.3.



Слика 4.3. Конфигурација мреже са подешеним тежинама грана

Поново ћемо у фокус ставити рутер бр. 6 и посматраћемо како се његова табела рутирања мењала кроз време. У овом случају до коначне верзије табеле рутирања стиже се у 7 итерација, на слици 4.4 приказане су промене табеле рутирања током прве 4 итерације.

146				
Destination: 0.0.0.0	Subnet mask: 0.0.0.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 0
Destination: 106.1.1.0	Subnet mask: 255.255.255.0	Next hop: On-link	Interface: 106.1.1.1	Metric: 0
Destination: 107.1.1.0	Subnet mask: 255.255.255.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 50
Destination: 105.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 30
352				
Destination: 0.0.0.0	Subnet mask: 0.0.0.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 0
Destination: 106.1.1.0	Subnet mask: 255.255.255.0	Next hop: On-link	Interface: 106.1.1.1	Metric: 0
Destination: 107.1.1.0	Subnet mask: 255.255.255.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 50
Destination: 105.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 30
Destination: 104.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 35
Destination: 102.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 105
Destination: 101.1.1.0	Subnet mask: 255.255.255.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 150
416				
Destination: 0.0.0.0	Subnet mask: 0.0.0.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 0
Destination: 106.1.1.0	Subnet mask: 255.255.255.0	Next hop: On-link	Interface: 106.1.1.1	Metric: 0
Destination: 107.1.1.0	Subnet mask: 255.255.255.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 50
Destination: 105.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 30
Destination: 104.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 35
Destination: 102.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 105
Destination: 101.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 115
Destination: 108.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 125
Destination: 103.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 45
703				
Destination: 0.0.0.0	Subnet mask: 0.0.0.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 0
Destination: 106.1.1.0	Subnet mask: 255.255.255.0	Next hop: On-link	Interface: 106.1.1.1	Metric: 0
Destination: 107.1.1.0	Subnet mask: 255.255.255.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 50
Destination: 105.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 30
Destination: 104.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 35
Destination: 102.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 95
Destination: 101.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 115
Destination: 108.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 125
Destination: 103.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 45
Destination: 109.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 60

Слика 4.4. Промене табеле рутирања за рутер бр. 6 током прве 4 итерације за проширени RIP протокол

Као и у претходном примеру видимо да у првој итерацији рутер сазнаје за мреже својих директних суседа односно мреже **107.1.1.0** и **105.1.1.0**. У другој итерацији преко рутера бр. 5 добија информације о мрежама **104.1.1.0** и **102.1.1.0**, а преко рутера бр. 7 о мрежи **101.1.1.0**. Такође видимо и да се за мреже појављују одговарајуће метрике, па тако

вредност метрике рецимо за мрежу **101.1.1.0** износи 150 јер се до те мреже за сада стиже преко рутера бр. 7 и бр. 1 а вредности веза на тој путањи у збиру заиста дају 150. Након треће итерације у табели се појављује и путања до мреже **108.1.1.0** до које се стиже преко рутера бр. 6 као и путања до мреже **103.1.1.0**, до које се стиже преко рутера бр. 5. Такође треба приметити да је током треће итерације измењена већ постојећа путања ка мрежи **101.1.1.0** и до ње се више не стиже преко рутера бр. 7 већ преко рутера бр. 5 чиме је обезбеђена „јефтинија“ рута чија вредност метрике сада износи 115 што одговара ситуацији да се до мреже долази редом, преко рутера бр. 5, бр. 2 и бр. 1. У четвртој итерацији у табелу је додата још и путања до мреже **109.1.1.0**, а поред тога измењена је рута ка мрежи **102.1.1.0**, пакети ка овој мрежи се и даље прослеђују на рутер бр. 5 међутим рутер бр. 5 је пронашао „бољи“ пут до те мреже и сада пакете ка **102.1.1.0** прослеђује преко рутера бр. 4, па је у табели рутирања рутера бр. 6 за посматрану руту ажурирана само вредност метрике. На слици 4.5 која следи дате су измене у табели током наредне 3 итерације.

1000					
Destination: 0.0.0.0	Subnet mask: 0.0.0.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 0	
Destination: 106.1.1.0	Subnet mask: 255.255.255.0	Next hop: On-link	Interface: 106.1.1.1	Metric: 0	
Destination: 107.1.1.0	Subnet mask: 255.255.255.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 50	
Destination: 105.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 30	
Destination: 104.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 35	
Destination: 102.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 95	
Destination: 101.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 105	
Destination: 108.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 65	
Destination: 103.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 45	
Destination: 109.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 60	
1309					
Destination: 0.0.0.0	Subnet mask: 0.0.0.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 0	
Destination: 106.1.1.0	Subnet mask: 255.255.255.0	Next hop: On-link	Interface: 106.1.1.1	Metric: 0	
Destination: 107.1.1.0	Subnet mask: 255.255.255.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 50	
Destination: 105.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 30	
Destination: 104.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 35	
Destination: 102.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 85	
Destination: 101.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 105	
Destination: 108.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 65	
Destination: 103.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 45	
Destination: 109.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 60	
1742					
Destination: 0.0.0.0	Subnet mask: 0.0.0.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 0	
Destination: 106.1.1.0	Subnet mask: 255.255.255.0	Next hop: On-link	Interface: 106.1.1.1	Metric: 0	
Destination: 107.1.1.0	Subnet mask: 255.255.255.0	Next hop: 107.1.2.1	Interface: 106.1.3.1	Metric: 50	
Destination: 105.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 30	
Destination: 104.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 35	
Destination: 102.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 85	
Destination: 101.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 95	
Destination: 108.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 65	
Destination: 103.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 45	
Destination: 109.1.1.0	Subnet mask: 255.255.255.0	Next hop: 105.1.3.1	Interface: 106.1.2.1	Metric: 60	

Слика 4.5. Промене табеле рутирања за рутер бр. 6 током пете, шесте и седме итерације за проширени RIP протокол

Током наредне 3 итерације у табели рутирања нису се више појављивале руте ка новим мрежама, али јесу ажуриране постојеће. Па су тако у петој итерацији ажуриране путање ка мрежама **108.1.1.0** и **101.1.1.0**, где је поново остао исти рутер на који се прослеђује саобраћај ка тим мрежама, али је дошло до измена у рутирању негде у остатку путање о чему је рутер бр. 6 обавештен од стране суседног рутера бр. 5 што је резултовало изменама метрике за одговарајуће путање. Током шесте итерације, рутер је обавештен да је пронађена боља путања ка мрежи **102.1.1.0** и постављена је нова метрика за исту, а у седмој итерацији још једном је ажурирана метрика за путању ка мрежи **101.1.1.0**. Током наредних итерација више није било промена у табели рутирања рутера бр. 6.

4.2 Праћење рутирања конкретног IP пакета на примеру дате мреже

У овом сегменту анализираће се садржај излазне датотеке packet_path.txt у којој су сачувани симулациони подаци везани за прослеђивање тестног пакета кроз мрежу чија конфигурација одговара примеру са слике 4.1. У овом случају за уређај који започиње слање пакета изабран је уређај са IP адресом **106.1.1.2** док је за одредиште изабран уређај са IP адресом **108.1.1.2**. Овај део симулације започет је након 5. итерације симулационог процеса анализираних у поглављу 4.1 и надаље је текао упоредо са њим. Након завршетка симулације у потпуности, у излазном фајлу добијени су резултати као на слици 4.6.

```
Router: 106
Source: 106.1.1.2
Destination: 108.1.1.2
Next hop : 105.1.4.1
Interface: 106.1.2.1
-----
Router: 105
Lan: 105.1.1.1
Destination: 108.1.1.2
Next hop : 102.1.2.1
Interface: 105.1.2.1
-----
Router: 102
Lan: 102.1.1.1
Destination: 108.1.1.2
Next hop : 108.1.2.1
Interface: 102.1.5.1
-----
Router: 108
Lan: 108.1.1.1
Destination: 108.1.1.2
Next hop : On-Link
Message received!
```

Слика 4.6. Приказ путање којом је прошао пакет, записане у излазној датотеци packet_path.txt

На основу добијених резултата видимо да је пакет до одредишта заиста стигао најкраћом могућом путањом, што нам потврђује да су табеле рутирања коректно формиране. У конкретном примеру који смо овде посматрали добијена путања је заиста најкраћа и јединствена, мада није увек случај да је најкраћа путања јединствена. Наиме, врло често је могуће да постоји више рута које имају исту најмању метрику (исту вредност растојања) и у том случају можемо добити другачију путању пакета приликом сваког наредног покретања симулације. Резултати ће увек бити исправни, али избор путање зависиће од стохастичке природе размене порука између рутера, јер постоји случајно време које сваки рутер проводи у стању чекања пре него што пошаље нову верзију табеле својим суседима. Добијање више различитих путања исте вредности до једног одредишта јесте очекивана ситуација и оригинални RIP протокол је у том случају предвидео да се саобраћај равномерно распоређује на све добијене путање (енг: *Equal cost load balancing*). Број путања на које се саобраћај распоређује може се мењати, максималан број путања је дефинисан

ограничењем у поставкама самог рутера, а минималан је 1, у том случају не постоји распоређивање и сав саобраћај се испоручује првом рутом која која за дато одредиште постоји у табели рутирања [10]. Решење које је у овом раду имплементирано, не води евиденцију о алтернативним рутама, тако да се увек узима само једна од потенцијално више еквивалентних путања.

5. Закључак

Као што је већ речено на почетку, RIP протокол је најстарији и један је основних протокола рутирања. У овом раду представљена је апликација која симулира рад овог протокола у рачунарској мрежи са променљивим бројем рутера и њиховом међусобном повезаношћу. Показано је да у општем случају алгоритам на малим мрежама функционише добро, односно рутери доносе исправне закључке о мрежи и формирају адекватне табеле рутирања. Такође примећено је да алгоритам није детерминистички и да за исте примере потенцијално добијамо различита, али увек исправна решења. Додатно, самом софтверском архитектуром којом су рутери представљени независним програмским нитима и UDP серверима наглашена је аутономност рутера као уређаја која је присутна и у реалном систему.

Поред тога тестирано је и како би се RIP протокол понашао у случају да се уместо броја скокова изабере метрика која уводи трошкове везе као критеријум по коме се врши избор путање. У том случају, протокол даје добре резултате и заиста проналази „најбоље“ руте, бирајући их по истом критеријуму по ком то ради OSPF протокол, али и даље користећи Белман-Фордов алгоритам, за разлику од OSPF-а који се ослања на Дијкстрин алгоритам. Овакав приступ иако даје добре резултате, ипак се не користи у пракси, због потенцијалних проблема који могу настати, као што је пре свега, проблем „бројања до бесконачности“ (енг: *Count to infinity problem*) који се догађа приликом испада неког од рутера или прекида везе између њих. Код стандардног RIP протокола проблем бројања до бесконачности је донекле решен ограничењем максималног броја скокова који нека путања може имати на 15 [11]. Коначно постигнута је основна идеја да се овим радом приближи и боље разуме један комплексан систем, чије функционисање није увек лако сагледиво из његовог обичног формалног описа.

Рад који је управо представљен тежио је да заокружи једну малу али компактну целину, међутим захваљујући комплексности и ширини теме ипак је отворено и неколико додатних питања као и могућности за проширења. Ове могућности иду у два првца. Први је везан за саму функционалност корисничке апликације која би могла бити проширена тако да кориснику додатно олакша конфигурисање мреже као и измене на њој, као што су уклањање и мењање вредности за појединачне рутере и везе између њих, ван и у току времена симулације. Овакве измене отвориле би простор за други правац проширења која се односе на имплементацију самог протокола, односно симулациони део апликације, где би се дефинисало како протокол реагује на отказе појединих елемената мреже или додавање нових у реалном времену. Такође симулација би се могла проширити и тако да у случају добијања више путања једнаких метрика до истог одредишта врши расподелу опетерећења (енг: *Load balancing*) што би додатно допринело разумевању самог протокола али и функциосања рачунарске мреже као система.

Литература

- [1] J. F. Kurose, K. W. Ross, *Computer Networking: A Top-Down Approach*, Seventh edition, Hoboken, New Jersey: Pearson, 2017, ISBN 0133594149.
- [2] C. Hedrick, „*Request for Comments: 1058 - Routing Information Protocol*”, Rutgers University, 1988.
- [3] T. Keary, „*Types of Routing Protocols – The Ultimate Guide*” www.comparitech.com. [Online] Доступно на: <https://www.comparitech.com/net-admin/routing-protocol-types-guide/> (Приступљено: август 10, 2024).
- [4] V. Fluber-Garcia, „*Routing: IGP and EGP Protocols*” www.baeldung.com. [Online] Доступно на: <https://www.baeldung.com/cs/routing-igp-egp-protocols> (Приступљено: септембар 2, 2024).
- [5] R. Bhardwaj, „*RIP v1 vs v2 : Detailed Comparison*” www.ipwithease.com. [Online] Доступно на: <https://ipwithease.com/rip-v1-vs-rip-v2/> (Приступљено: септембар 5, 2024).
- [6] S. Antoniou, „*Dynamic Routing Protocols: Distance Vector and Link State*” www.pluralsight.com. Доступно на: <https://www.pluralsight.com/blog/it-ops/dynamic-routing-protocol> (Приступљено: август 5, 2024).
- [7] T. Gupta, „*Distance Vector Routing v/s Link State Routing*” www.scaler.com. [Online] Доступно на: <https://www.scaler.com/topics/distance-vector-routing-vs-link-state-routing/> (Приступљено: август 17, 2024).
- [8] K. Brush, „*Routing Information Protocol (RIP)*” www.techtarget.com. [Online]. Доступно на: <https://www.techtarget.com/searchnetworking/definition/Routing-Information-Protocol> (Приступљено: мај 28, 2024).
- [9] Wikipedia, the free encyclopedia, „*Bellman–Ford algorithm*” www.wikipedia.org. [Online] Доступно на: https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm (Приступљено: мај 28, 2024).
- [10] R. Molenaar, „*RIP Maximum-Paths*” www.networklessons.com [Online] Доступно на: <https://networklessons.com/rip/rip-maximum-paths> (Приступљено: август 11, 2024).
- [11] J. Doyle, J. DeHaven Carroll, *CCIE Professional Development: Routing TCP/IP, Volume I*, Cisco Press, 2005, ISBN 9781587052026.

Списак коришћених слика и табела

Слика 3.1. Изглед дела конфигурационе датотеке - стр. 5

Слика 3.2. Изглед дела корисничког интерфејса, након избора броја рутера - стр. 6

Слика 3.3. Приказ начина на који корисник повезује рутере у мрежи - стр. 7

Слика 3.4. Изглед картице "Ruter" пре пократања симулације - стр. 8

Слика 3.5. Дијаграм кључних класа за дефинисање мрежне топологије - стр. 10

Слика 3.6. Класе задивжене за управљање симулацијом, размену података и конверзије бројева - стр. 10

Слика 3.7. Изглед табеле рутирања једног рутера - стр. 11

Слика 3.8. Формат IP пакета коришћен у симулацији - стр. 13

Слика 3.9. Пример приказане табеле рутирања за изабрани рутер - стр. 14

Слика 3.10. Пример графичког и текстуалног приказа резултата везаних за кретање пакета - стр. 14

Слика 4.1. Конфигурација мреже коришћене при симулацији - стр. 15

Слика 4.2. Промене табеле рутирања рутера бр. 6 током прве четири итерације за стандардни RIP протокол - стр. 16

Слика 4.3. Конфигурација мреже са подешеним тежинама грана - стр. 17

Слика 4.4. Промене табеле рутирања за рутер бр. 6 током прве четири итерације за проширени RIP протокол - стр. 17

Слика 4.5. Промене табеле рутирања за рутер бр. 6 током пете, шесте и седме итерације за проширени RIP протокол - стр. 18

Слика 4.6. Приказ путање којом је прошао пакет, записане у излазној датотеци packet_path.txt . стр. - 19