УНИВЕРЗИТЕТ
У НОВОМ САДУ
**ФАКУЛТЕТ ТЕХНИЧКИХ
НАУКА У НОВОМ САДУ**

Иван Нинић

СИМУЛАЦИЈА ПРОТОКОЛА РУТИРАЊА
RIP - Routing Information Protocol

ПРОЈЕКАТ

Основне академске студије

Нови Сад, 2024

Садржај

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 основих улога управо усмеравање(енг. routing) пакетског саобраћаја кроз рачунарску мрежу . Усмеравање или рутирање 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. 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 мреже, при чему сваки скок означава пролаз 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. То нам говори да је RIP протокол рутирања базиран на векторима удаљености (енг. *Distance Vector*) и представља један од првих стандардизованих протокола рутирања [1, 2].

1.2 Опис проблема

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. 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 размацима 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 „представу" о читавој мрежи, тј. о томе где да усмеравају поруке које имају одређене одредишне IP адресе у свом заглављу. Детаљније о начину функционисања 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 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 касније може правити детаљнија анализа. Формат и садржај ових датотека, биће посебно описан у наредним поглављима.

2. Опис коришћених технологија и алата

2.1 Коришћене технологије

Сав програмски код писан је у језику C/C++ користећи се општим особинама објектно оријентисане парадигме програмирања. Разлог избора C/C++ језика био је у томе да обезбеди се прецизније конструисање потребних структура података као и потенциално 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++ био приступачнији већини студената, имајући у виду да се симулационе апликације везане за рачунарске мреже традиционално пишу у C програмском језику. Такође развојно окружење које је коришћено је Visual Studio 2019. Неке коришћене библиотеке наведене су у наставку:

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 Конфигурација мреже

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. Даље подаци за сваки интерфејс састоје се од IP адресе самог интерфејса, ID-а и IP адресе рутера са којим је дати интерфејс повезан, ID-а интерфејса и његовог типа. Док подаци о уређајима из локалне мреже садрже само IP адресе тих уређаја. Изглед дела конфигурационе датотеке, приказан је на слици 3.1.

```
router_id=105
223.1.19.1|223.1.19.2|102|1051|1
223.1.12.2|223.1.12.1|106|1052|1
223.1.11.1|223.1.11.2|104|1053|1
223.1.5.7|0.0.0.0|000|1054|0
-------------------------
223.1.5.1
223.1.5.2
#########################
```

Слика 3.1. Изглед дела конфигурационе датотеке

3.2 Учитавање података мреже из конфигурационе датотеке

У оквиру класе Network метода **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 форамту подаака о рутеру који се чува у конфигурационој датотеци, ова класа детаљније је обрађена у наредним поглављима.

3.3 Класна хијерархија и организација кода

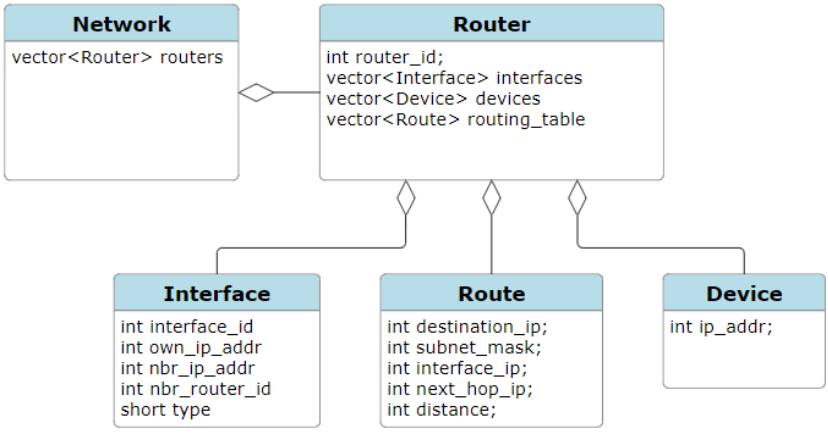
Као што је већ речено у претходним поглављима сви подаци о мрежи груписани су око класе *Router* па се тако читава мрежа може представити колекцијом објеката исте. Класа *Router* поседује следећа поља:

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

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

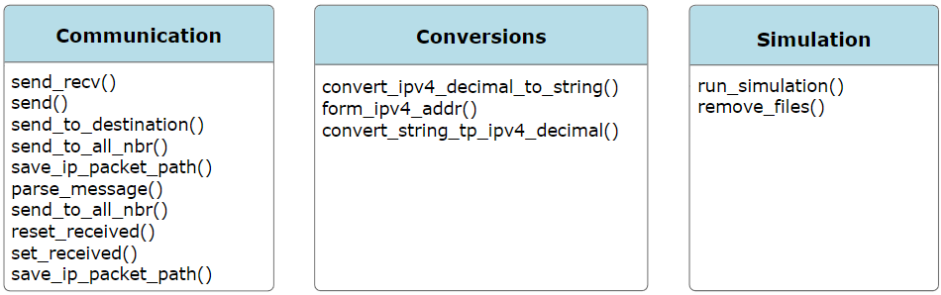
vector<Route> routing_table - Вектор путања које рутер познаје

Организација читавог кода везаног за представу мреже приказана је на слици 3.2.



Слика 3.2. 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 3.3.



Слика 3.3. Класе задижене 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 3.4.

```
Destination: 0.0.0.0        Subnet mask: 0.0.0.0        Next hop: 223.1.15.1     Interface: 223.1.15.2     Distance: 0
Destination: 223.1.2.0      Subnet mask: 255.255.255.0  Next hop: On-link        Interface: 223.1.2.2      Distance: 0
Destination: 223.1.3.0      Subnet mask: 255.255.255.0  Next hop: 223.1.20.2     Interface: 223.1.20.1     Distance: 1
Destination: 223.1.1.0      Subnet mask: 255.255.255.0  Next hop: 223.1.15.1     Interface: 223.1.15.2     Distance: 1
Destination: 223.1.5.0      Subnet mask: 255.255.255.0  Next hop: 223.1.19.1     Interface: 223.1.19.2     Distance: 1
Destination: 223.1.8.0      Subnet mask: 255.255.255.0  Next hop: 223.1.16.2     Interface: 223.1.16.1     Distance: 1
Destination: 223.1.6.0      Subnet mask: 255.255.255.0  Next hop: 223.1.19.1     Interface: 223.1.19.2     Distance: 2
Destination: 223.1.4.0      Subnet mask: 255.255.255.0  Next hop: 223.1.19.1     Interface: 223.1.19.2     Distance: 2
Destination: 223.1.9.0      Subnet mask: 255.255.255.0  Next hop: 223.1.16.2     Interface: 223.1.16.1     Distance: 2
Destination: 223.1.7.0      Subnet mask: 255.255.255.0  Next hop: 223.1.15.1     Interface: 223.1.15.2     Distance: 2
```

Слика 3.4. 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 иницијализује 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 тебеле 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

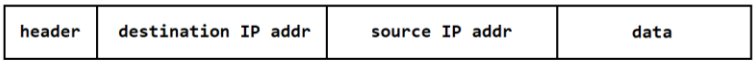
методи **update_table** ова порука разбија на појединачне руте из табеле. Затим се за сваку путању проверава да ли таква већ постоји у табели рутирања текућег рутера, ако не постоји одмах се додаје, а уколико постоји проверава се да ли предложена рута има мању удаљеност до дестинације, од руте која се већ налази у табели, ако је то тачно, онда се у уместо постојеће у табелу додаје нова путања, а ако није, задржава се стара.

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-Фордовог 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 [3, 4].

3.5 Слање IP пакета кроз мрежу

Поред претходно описаног процеса добијања табела рутирања, у оквиру симулације 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 рутирање стигне у стање конвергенције, 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 приспела 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 приспели 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 3.5.



Слика 3.5. Формат IP пакета коришћен у симулацији

Са слике 3.5 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.

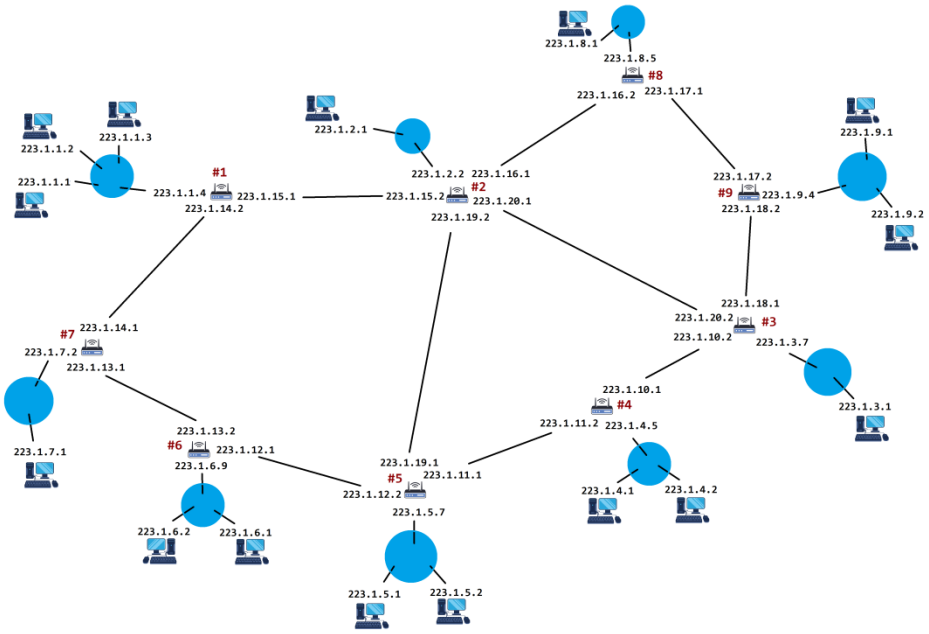
3.6 Чување добијених симулационих података

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. И овај фајл детаљније се анализира у поглављу 4.

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. Покретањем симулације за овакву мрежу добијени су одговарајући симулациони подаци који ће бити детаљно анализирани на примеру рутера бр. 6. Потребно је још нагласити и да IP адресе уређаја у мрежи са слике 4.2 почињу увек са два иста октета (223.1.), разлог за то је искључиво лакше праћење добијених резултата симулације. У реалним условима мреже могуће су наравно и адресе које у прва два октета имају неке друге вредности.



Слика 4.1. Конфигурација мреже коришћене при симулацији

Дакле посматрамо садржај излазног текстуалног фајла који се односи на рутер 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, RIP протокол рутирања улази у

стабилно стање већ након 4 итерације, па је у остатку садржаја датотеке само понављана табела рутирања из 4. итерације.

```
373
Destination: 0.0.0.0        Subnet mask: 0.0.0.0        Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 0
Destination: 223.1.6.0      Subnet mask: 255.255.255.0  Next hop: On-link       Interface: 223.1.6.9      Distance: 0
Destination: 223.1.7.0      Subnet mask: 255.255.255.0  Next hop: 223.1.13.1    Interface: 223.1.13.2     Distance: 1
Destination: 223.1.5.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 1
696
Destination: 0.0.0.0        Subnet mask: 0.0.0.0        Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 0
Destination: 223.1.6.0      Subnet mask: 255.255.255.0  Next hop: On-link       Interface: 223.1.6.9      Distance: 0
Destination: 223.1.7.0      Subnet mask: 255.255.255.0  Next hop: 223.1.13.1    Interface: 223.1.13.2     Distance: 1
Destination: 223.1.5.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 1
Destination: 223.1.4.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 2
Destination: 223.1.2.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 2
Destination: 223.1.1.0      Subnet mask: 255.255.255.0  Next hop: 223.1.13.1    Interface: 223.1.13.2     Distance: 2
964
Destination: 0.0.0.0        Subnet mask: 0.0.0.0        Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 0
Destination: 223.1.6.0      Subnet mask: 255.255.255.0  Next hop: On-link       Interface: 223.1.6.9      Distance: 0
Destination: 223.1.7.0      Subnet mask: 255.255.255.0  Next hop: 223.1.13.1    Interface: 223.1.13.2     Distance: 1
Destination: 223.1.5.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 1
Destination: 223.1.4.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 2
Destination: 223.1.2.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 2
Destination: 223.1.1.0      Subnet mask: 255.255.255.0  Next hop: 223.1.13.1    Interface: 223.1.13.2     Distance: 2
Destination: 223.1.3.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 3
Destination: 223.1.8.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 3
1333
Destination: 0.0.0.0        Subnet mask: 0.0.0.0        Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 0
Destination: 223.1.6.0      Subnet mask: 255.255.255.0  Next hop: On-link       Interface: 223.1.6.9      Distance: 0
Destination: 223.1.7.0      Subnet mask: 255.255.255.0  Next hop: 223.1.13.1    Interface: 223.1.13.2     Distance: 1
Destination: 223.1.5.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 1
Destination: 223.1.4.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 2
Destination: 223.1.2.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 2
Destination: 223.1.1.0      Subnet mask: 255.255.255.0  Next hop: 223.1.13.1    Interface: 223.1.13.2     Distance: 2
Destination: 223.1.3.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 3
Destination: 223.1.8.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 3
Destination: 223.1.9.0      Subnet mask: 255.255.255.0  Next hop: 223.1.12.2    Interface: 223.1.12.1     Distance: 4
```

Слика 4.2. Приказ табеле рутирања рутера бр. 6 (након четири итерације)

Видимо да је у првој итерацији рутер бр. 6 сазнао за мреже **223.1.7.0** и **223.1.5.0** 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 непосердних суседа, односно мрежаама **223.1.2.0** и **223.1.4.0** преко рутера бр. 5, а **223.1.1.0**. 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 5 сазнао за мрежу **223.1.3.0** преко рутера 4, да би је затим у трећој итерацији проследио посматраном рутеру 6, што управо и видимо са слике 4.2, дакле мрежа **223.1.3.0** се у табели рутирања рутера 6, појављује тек након треће итерације. У четвртој итерацији процес размене података између рутера је конвергирао до стабилног стања, 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.

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 **223.1.6.1** док је за одредиште изабран уређај са IP адресом **223.1.8.1**. Овај део симулације започет је након 5. итерације симулационог процеса анализираног 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 4.3**.**

```
Router: 106
Source: 223.1.6.1
Destination: 223.1.8.1
Next hop : 223.1.12.2
Interface: 223.1.12.1
----------------------
Router: 105
Lan: 223.1.5.7
Destination: 223.1.8.1
Next hop : 223.1.19.2
Interface: 223.1.19.1
----------------------
Router: 102
Lan: 223.1.2.2
Destination: 223.1.8.1
Next hop : 223.1.16.2
Interface: 223.1.16.1
----------------------
Router: 108
Lan: 223.1.8.5
Destination: 223.1.8.1
Next hop : On-Link
Message received!
```

Слика 4.3. Приказ путање коју је имао изабрани пакет записан 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. Овакво понашање не представља ограничење ове симулације јер оваква стохастичност постоји и у реалном систему на основу ког је модел и конструисан.

5. Закључак

5.1 Општи закључци

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. 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.

5.2 Предлози за даља проширења

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 програма, који тренутно функционише као конзолна апликација чији су сви улази и излази текстуалне датотеке, које корисник ручно попуњава и чији садржај тумачи. Дакле коришћење саме апликације доста би се олакшало уз пригодан графички интерфејс који би пре свега олакшао унос података, а онда омогућио и неки смисленији приказ резултата симулације од сувопарних података у текстуалним датотекама. Други правац односи се на сам протокол који јесте заснован на оригиналном RIP протоколу, али ипак уноси и нека ограничења и не покрива све могуће случајеве. Тако на пример, ситуација у којој долази до отказа неког рутера овде није разматрана иако је то ситуација која је реална и од интереса је видети како се посматрани протокол понаша у том случају.

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] K. Brush. „Routing Information Protocol (RIP)" www.techtarget.com. [Online]. Dostupno na: https://www.techtarget.com/searchnetworking/definition/Routing-Information-Protocol (pristupljeno: maj 28, 2024).

[4] Wikipedia, the free encyclopedia, „Bellman–Ford algorithm" www.wikipedia.org. [Online]. Dostupno na: https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm (pristupljeno: maj 28, 2024).

Списак коришћених слика и табела