KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Šarūnas Rimšelis

PROGRAMINIO VALDYMO ĮTAKOS KOMPIUTERIŲ TINKLŲ PRALAIDUMUI TYRIMAS

Baigiamasis magistro projektas

Vadovas
Doc. dr. Paulius Tervydis

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS
TELEKOMUNIKACIJŲ KATEDRA

PROGRAMINIO VALDYMO ĮTAKOS KOMPIUTERIŲ TINKLŲ PRALAIDUMUI TYRIMAS

Baigiamasis magistro projektas
Išmaniosios telekomunikacijų technologijos (621H64001)

Vadovas
(parašas) Doc. dr. Paulius Tervydis
(data)

Recenzentas
(parašas) Doc. dr. Rasa Brūzgienė
(data)

Projektą atliko
(parašas) Šarūnas Rimšelis
(data)

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Elektros ir elektronikos fakultetas

(Fakultetas)

Šarūnas Rimšelis

(Studento vardas, pavardė)

Išmaniosios telekomunikacijų technologijos (kodas 621H64001)

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Programinio valdymo įtakos kompiuterių tinklų pralaidumui tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 17 m. birželio 5 d.

Kaunas

Patvirtinu, kad mano **Šarūno Rimšelio** baigiamasis projektas tema „Programinio valdymo įtakos kompiuterių tinklų pralaidumui tyrimas“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

Rimšelis, Šarūnas. Programinio valdymo įtakos kompiuterių tinklų pralaidumui tyrimas. Telekomunikacijų inžinerijos magistro baigiamasis projektas / vadovas doc. dr. Paulius Tervydis; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas, Telekomunikacijų katedra.

Mokslo kryptis ir sritis: Elektros ir elektronikos inžinerija, Technologiniai mokslai
Reikšminiai žodžiai: Programuojami tinklai, SDN, pralaidumas, algoritmas
Kaunas, 2017. 54 p.

SANTRAUKA

Šio darbo tikslas yra sukurti programuojamų tinklų valdymo algoritmą, kuris leistų efektyviau išnaudoti tinklo pralaidumo resursus, atsižvelgiant į perduodamų duomenų prioritetus ir QoS reikalavimus. Darbe supažindinama su SDN technologijos architektūra bei nagrinėjami atskirų jos elementų tyrimai, kuriuose programuojamų tinklų sprendimai gali daryti įtaką duomenų srautų perdavimui ir tinklų pralaidumui. Taip pat, apžvelgiamos aplikacijos, kurios gali būti pritaikytos duomenų perdavimo kokybės gerinimui.

Darbe atliekami eksperimentai, kuriuose naudojama reali techninė įranga. Šių eksperimentų tikslas yra nustatyti tinklo pralaidumo padalinimo skirtingiems srautams tendencijas. Remiantis eksperimentų rezultatais, sudaromas imitacinis komutatoriaus modelis, kuris vėliau naudojamas srautų scenarijaus modeliavimui. Modeliavimas atliekamas taikant skirtingus valdymo variantus, o kiekvieno modeliavimo metu skaičiuojama aptarnauta srautų dalis bei kanalo išnaudojimas. Remiantis skaičiavimų rezultatais, sudaromas programinio srautų valdymo algoritmas.

Rimšelis, Šarūnas. Investigation of Impact of Software Defined Networking on Computer Networks Bandwidth: Master's thesis in Telecommunications engineering master degree / supervisor assoc. prof. Paulius Tervydis. Kaunas University of Technology, Faculty of Electrical and Electronics Engineering, department of Telecommunications

Research area and field: Electrical and Electronics Engineering, Technological Sciences

Key words: Software Defined Networks, SDN, throughput, algorithm

Kaunas, 2017. 54 p.

SUMMARY

The goal of this work is to create an algorithm for Software Defined Network control, which would help better utilize network throughput considering data flow priorities and demand for QoS. This paper introduces to the SDN architecture. Other SDN researches that may influence flow forwarding or network throughput are analyzed. Research of applications that may improve networking performance are also included.

My conducted experiments on real Ethernet switches are presented in this work. The goal of these experiments is to discover the tendencies that apply when throughput sharing in switch ports occurs. These tendencies are then considered in the creation of an imitational model of a network switch, which is later used for modelling network control scenarios. Different switch control scenarios are simulated while using the same input flows, while calculating flow service and channel utilization. Finally, an algorithm for SDN flow control is created based on the result of the modelling calculation.

TURINYS

SUTRUMPINIMŲ SĄRAŠAS

API	aplikacijų programavimo sąsaja (angl. *Application programming interface*)
BGP	kraštinių tarptinklinių sąsajų protokolas (angl. *Border Gateway Protocol*)
CLI	komandinės eilutės sąsaja (angl. *Command Line Interface*)
IP	interneto protokolas (angl. *Internet Protocol*)
LLDP	sujungimų lygmens aptikimo protokolas (angl. *Link Layer Discovery Protocol*)
MAC	medijos prieigos valdymas (angl. *Media Access Control*)
ML	mašininis mokymasis (angl. *Machine Learning*)
MPLS-TP	daugelio protokolų žymių komutavimo transporto profilis (angl. *Multi-Protocol Label Switching Transport Profile*)
QoS	paslaugų kokybė (angl. *Quality of Service*)
SDN	programuojami tinklai (angl. *Software Defined Networks*)
SNMP	paprastasis tinklo valdymo protokolas (angl. *Simple Network Management Protocol*)
SSH	saugaus sujungimo protokolas (angl. *Secure Socket Shell*)
TCP	perdavimo valdymo protokolas (angl. *Transmission Control Protocol*)
TLS	transportinio lygmens apsauga (angl. *Transport Layer Security*)
TTL	likęs šuolių skaičius (angl. *Time to Live*)
UDP	vartotojo datagramų protokolas (angl. *User Datagram Protocol*)
UTP	neekranuotas komutacinis kabelis (angl. *Unshielded Twisted Pair*)
VLAN	virtualus vietinis tinklas (angl. *Virtual Local Area Network*)
XML	plečiamoji aprašomoji kalba (angl. *Extensible Markup Language*)
XMPP	plečiamas signalizacijos ir būsenos protokolas (angl. *Extensible Messaging and Presence Protocol*)

ĮVADAS

Šiais laikais žmonių darbas, mokslas ir laisvalaikis tapo neatsiejamas nuo kompiuterių, telefonų ir kitų skaitmeninę informaciją apdorojančių ir perduodančių įrenginių. Vartojamų paslaugų kiekis, sudėtingumas ir reikalaujamas tinklų pralaidumas taip pat nenuvaldomai didėja. Atsiranda milžiniški interneto spartos, patikimumo ir nenutrūkstamos prieigos reikalavimai. Todėl skaitmeninės telekomunikacijų technologijos įgauna vis svarbesnį vaidmenį daugelyje šiuolaikinio gyvenimo sričių. Tai gerai iliustruoja interneto vartotojų skaičiaus didėjimo statistika. 2015 interneto vartotojų skaičius sudarė daugiau nei 40% visos žmonijos populiacijos, o 2000-aisiais metais, tai sudarė tik 7% [1]. Prognozuojama, kad 2020-aisiais metais ryšį su internetu turės net 30 milijardų įrenginių [2] . Vartotojų ir įrenginių skaičiaus augimas, be abejo, daro tiesioginę įtaką ir perduodamų duomenų kiekių augimui, o tai tampa tikru iššūkiu telekomunikacijų tinklų administratoriams, eksploatuotojams, skaitmeninių duomenų perdavimo paslaugų tiekėjams. Tradiciniai duomenų perdavimo tinklų sprendimai tampa nepajėgūs aptarnauti susidarančius masyvius srautus, kad išlaikytų vartotojus tenkinančius paslaugų kokybės reikalavimus.

Tradiciniais tapę ir dabar dažniausiai naudojami tinklų sprendimai yra sudėtingi, statiški bei dažnai nesuderinami tarpusavyje, jeigu yra naudojama skirtingų gamintojų įranga. Tai yra apribojimai, kurie trukdo efektyviai, patogiai ir lanksčiai išnaudoti tinklo resursus suteikiant vartotojams geriausią paslaugų kokybę. Dabartinių tinklų kompleksiškumas atsiranda dėl skirtingų tinklų protokolų įvairovės. Skirtingų tinklo funkcijų, kaip balso, vaizdo ar duomenų perdavimui, saugumo užtikrinimui, interneto puslapių pateikimui yra naudojami atskiri protokolai, kurie dažnai veikia tinkle vienu metu. Tinklo kompleksiškumas tampa akivaizdus, kai reikia konfigūruoti naują tinklą, įdiegti naujas paslaugas ar pakeisti tinklo topologiją. Tokių darbų atlikimui reikalingas daugelio tinklo įrenginių nustatymų konfigūravimas. Tai gali būti komutatoriai, maršrutizatoriai, ugniasienės, tinklo prieigos sistemos ir kita.

Šiais laikais naudojami tinklai yra sudėtingi. Tai sąlygoja jų statiškumą. Prijungiant naujus vartotojus ar įdiegiant naujas paslaugas į tinklą, dažnai vengiama tinklo perkonfigūravimo, dėl to, kad nebūtų sutrikdomas jau tinkle egzistuojančių paslaugų tiekimas. Vyraujant tokioms tendencijoms, tinklai tampa dar sudėtingesni. Iš dabartinių tinklų reikalaujamas daug didesnis lankstumas, ypač kuomet vis dažniau naudojami virtualūs įrenginiai. Taikant sparčiai populiarėjančius debesų kompiuterijos sprendimus, virtualūs įrenginiai gali migruoti iš vieno fizinio serverio į kitą, sukeldami papildomų problemų tradicinių tinklų operatoriams. Tai reiškia, kad tinklas turi būti iš naujo konfigūruojamas, reikalaujant administruojančio personalo įsikišimo. Tai stabdo tinklų kūrimą, plėtimąsi ir kitimą. Tinklo funkcijų įvairovės didėjimas ir kokybės gerėjimas yra taip pat ribojamas tradicinių tinklų sprendimų, nes skirtingų gamintojų įranga gali

būti nesuderinama. Tai verčia informacijos perdavimo paslaugų tiekėjus ir kompiuterių tinklus turinčias įmones rinktis vieno ar keleto gamintojų techninę įrangą ir esant poreikiui laiku negauti tinkamo funkcionalumo. Tradicinių tinklų veikimo apribojimai iškelia reikalavimą keisti tinklų veikimo principą iš esmės.

Bendrų standartų, atvirų sąsajų, centralizuoto tinklo būsenos stebėjimo ir valdymo sistemos poreikis sąlygojo programuojamų tinklų arba SDN (angl. *Software Defined Networks*) atsiradimą. Programinio tinklo įrenginių valdymo įdiegimas leistų iš vieno valdymui dedikuoto įrenginio stebėti ir matyti viso tinklo būseną, atlikti duomenų srautų valdymo operacijas, konfigūruoti esamas ir diegti naujas paslaugas. Kadangi valdymas yra laisvai programuojamas, daugelį tinklo operacijų būtų galima automatizuoti. Programuoto tinklo valdymo įvedimas leistų efektyviai išnaudoti tinklo resursus, o tai dažnai kintančiuose tinkluose gali būti sudėtinga ir reikalauti daug laiko. SDN technologijos yra naujos ir nuolat tobulėjančios. 2015 m. Google atvirai prabilo apie savo SDN tinklus, kuriuos naudoja savo duomenų centrams valdyti [3]. Todėl programinis tinklo resursų valdymas pasirinktas šio darbo tema.

Darbo tikslas – sukurti programuojamų tinklų valdymo algoritmą, kuris leistų efektyviau išnaudoti tinklo pralaidumo resursus, atsižvelgiant į perduodamų duomenų prioritetus ir QoS reikalavimus.

Darbo uždaviniai:

1) atlikti literatūros, kurioje nagrinėjami SDN valdymo klausimai analizę;
2) atlikti matavimus, kurie leistų įvertinti, kokią įtaką duomenų srautų parametrams turi skirtingi tinklo komutatorių nustatymai;
3) remiantis gautais matavimų rezultatais sudaryti imitacinį modelį, kuris leistų modeliuoti skirtingo programinio komutatorių valdymo variantus;
4) atlikti duomenų srautų scenarijaus modeliavimą, pritaikant skirtingus programinio valdymo variantus komutatoriuose;
5) remiantis gautais modeliavimo rezultatais, sudaryti SDN srautų valdymo algoritmą.

Matavimams tyrimo metu atlikti naudojami realūs Ethernet tinklo komutatoriai, bei programinė įranga, kuri skirta duomenų srautų kokybiniams parametrams nustatyti. Gauti matavimų rezultatai naudojami imitacinio modelio sukūrimui. Imitacinio modelio sudarymas leidžia nustatyti, kokią įtaką programinis valdymas turėtų duomenų srautų kokybiniams parametrams tam tikro scenarijaus atveju. Tai leidžia greitai įvertinti ir pritaikyti tinkamą SDN modelį naujai projektuojamuose tinkluose, išvengiant brangios techninės ir programinės įrangos prikimo bei gausių kompiuterinių skaičiavimo išteklių panaudojimo.

1 SDN TECHNOLIGIJOS APŽVALGA

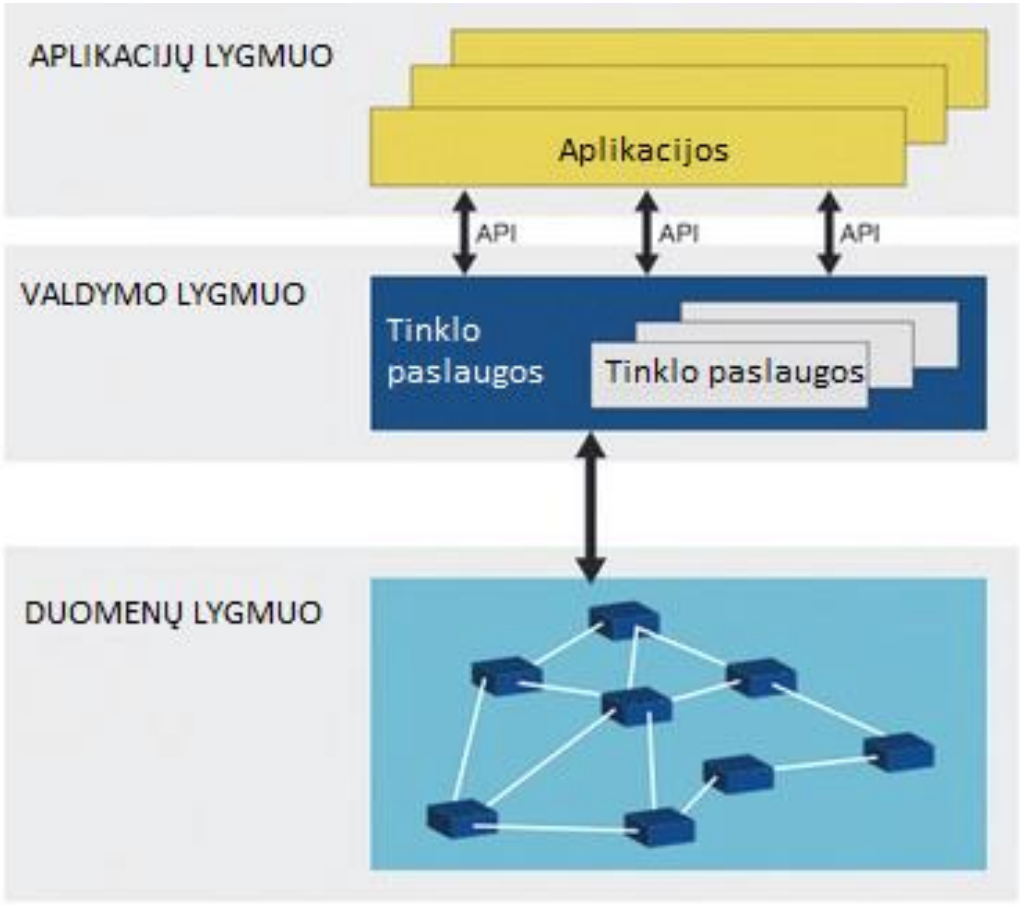
Programuojami tinklai – tai nauja duomenų perdavimo tinklų sąvoka ir principas, kuris turi padėti tinklų ir debesų kompiuterijos kūrėjams ir administratoriams nedelsiant reaguoti į dažnai kintančius tinklo reikalavimus. Tai centralizuota tinklo valdymo sistema, kuri gali apjungti skirtingas tinklo technologijas, norint sudaryti lankstesnį ir sparčiau aptarnaujamą tinklą. Tokių tinklų poreikis išaugo atsiradus moderniems duomenų centrams su virtualiais serveriais [4].

1.1 Programuojamų tinklų architektūra

Programuojami tinklai gali būti įgyvendinti pritaikant daug skirtingų architektūrų, tačiau pagrindinė SND apibūdinanti savybė yra tinklo valdymo ir duomenų perdavimo lygmenų atskyrimas. Visas SDN realizacijas sudaro tam tikras valdiklis, duomenis perduodantis įrenginys bei pietine ir šiaurine vadinamos aplikacijų programavimo sąsajos – API (angl. *Application programming interface*) [4]:

- Valdiklis – tai tarsi tinklo smegenys. Jo funkcija yra sudaryti bendrą centralizuotą tinklo vaizdą ir suteikti galimybę tinklo administratoriams nustatyti, kaip turėtų keliauti duomenų srautai, esantys apatiniame SDN architektūros lygmenyje .
- Pietinė API – tai sąsaja arba standartas, kuriuo perduodama valdymo informacija iš valdiklio į duomenų lygmenyje esančius komutatorius. Vienas iš tokių standartų yra OpenFlow. Jis laikomas pirmuoju SDN protokolu ir pirmąja pietine API bei išlieka vienu populiariausių ir plačiausiai naudojamų SDN protokolų. Kai kurie žmonės gali palaikyti SDN ir OpenFlow sinonimais, tačiau OpenFlow yra tik viena dalis, kuri gali padėti sukurti programuojamą tinklą.
- Šiaurinė API – tai sąsaja arba standartas, kuriuo komunikuoja valdiklis su išorinėmis programomis. Šių išorinių programų pagalba administratoriai kuria tinklo valdymo logiką, programiškai valdo teikiamas paslaugas ir formuoja tinklo srautus. Skirtingi valdikliai dažnai naudoja skirtingas šiaurines API. Tai dažniausiai valdiklio gamintojų sukurtos komandos, kurias galima perduoti iš skirtingų programų.
- SDN duomenų lygmuo – SDN tinklo dalis, kurioje atliekamas duomenų srautų pernešimas.

Apibendrinta SDN tinklo valdymo architektūra pavaizduota 1.1 pav.



1.1 pav. SDN architektūra [4]

1.2 SDN protokolai

Prieš programuojamų tinklų tyrimų nagrinėjimą, svarbu aptarti SDN naudojamų protokolų ir pietinių aplikacijų programavimo sąsajų savybes. Taip susidarysime bendrą suvokimą apie programuojamų tinklų veikimo principus ir iš arčiau susipažinsime su SDN architektūra.

OpenFlow

Kalbant apie SDN, dažnai išgirsime OpenFlow pavadinimą. Tai yra vienas iš pirmųjų standartų, kuris jungia SDN architektūros valdymo ir duomenų lygmenis. Jis leidžia SDN valdikliui tiesiogiai bendrauti su tokiais tinklo įrenginiais, kaip tinklo komutatoriai ar maršrutizatoriai. Tinklo įrenginiai gali būti realūs arba virtualūs, todėl OpenFlow gali būti labai lankstus ir pritapti esant kintantiems klientų reikalavimams [5].

Viena iš pagrindinių OpenFlow idėjų yra sukurti atvirą valdymo sąsają, kuri galėtų veikti skirtingų gamintojų įrangoje. Dėl šios priežasties tinko įranga privalo palaikyti OpenFlow protokolo veikimo funkciją. Dabar nėra labai daug įrenginių, palaikančių šį standartą, ir šie įrenginiai yra gana brangūs.

OpenFlow veikimo principas paremtas tam tikrų pranešimų siuntimu tarp valdiklio ir duomenis nukreipiančios įrangos. Duomenų lygmenyje esantis OpenFlow palaikantis įrenginys turi tam tikras srautų lenteles, kuriose saugomi srautų apibūdinimai, statistikos bei veiksmai, kuriuos su srautu atlieka komutatorius. OpenFlow valdiklyje yra programiškai įgyvendinti algoritmai, kurių pagalba nustatomi su srautu atliekami veiksmai. OpenFlow protokolo pagalba per tinklą siunčiamos užklausos iš komutatoriaus į valdiklį ir nurodymai iš valdiklio į komutatorių. Naujos užklausos iš duomenų lygmens siunčiamos atsiradus naujam srautui komutatoriuje [6]. Tokiu būdu vyksta nuolatinis pokalbis tarp valdiklio ir duomenis nukreipiančios įrangos, įgyvendinant reaktyvų SDN modelį. Tačiau OpenFlow susiduria su saugumo, plečiamumo problemomis ir reikalauja funkcionalumo iš techninės įrangos [7], todėl paminėsiu ir keletą kitų protokolų, kurie galėtų tarnauti SDN tinkle.

BGP

Kraštinių tarptinklinių sąsajų protokolas BGP (angl. *Border Gateway Protocol*) – tai protokolas, kurio pagalba tarptinklinių sąsajų įrenginiai apsikeičia maršrutizavimo informacija autonominėse sistemose. Šis protokolas dažnai naudojamas interneto maršrutizatoriuose ir laikomas standartizuotu išorinių tarptinklinių sąsajų protokolu. BGP dažnai klasifikuojamas kaip kelio ar atstumo vektorių protokolas. Jis naudoja maršrutizavimo lenteles, kuriose yra žinomų maršrutizatorių, jų pasiekiamų tinklo adresų sąrašas bei maršrutų svoriniai koeficientai. Apsikeitimas tokia informacija leidžia aptikti geriausią galimą maršrutą duomenų perdavimui. Kai kurie tinklų įrangos gamintojai tikisi panaudoti BGP mišriuose SDN tinkluose ir turi nuomonę, kad pietinė SDN aplikacijų perdavimo sąsają nėra tokia svarbi, kaip SDN teikiamas lankstumas ir programavimo galimybė. Todėl gamintojai įvardina BGP kaip SDN protokolą, kuris suteikia programinio valdymo galimybę [7].

NETCONF

NETCONF – tai tinklo įrenginių konfigūravimo protokolas, kurio tikslas yra pakeisti komandinės eilutės sąsają CLI (angl. *Command Line Interface*), SNMP (angl. *Simple Network Management Protocol*) ir kitus nusistovėjusius mechanizmus. Tinklo valdymo programinė įranga naudodama šį protokolą gali siųsti konfigūracijos duomenis į tinklo įrenginius bei nuskaityti informaciją apie įrenginio būseną. Informacijos apsikeitimas vykdomas saugiu SSH (angl. *Secure Socket Shell*) ar TLS (angl. *Transport Layer Security*) kanalu. NETCONF ir OpenFlow protokolai turi nemažai skirtumų ir taikomi skirtingais atvejais. NETCONF protokolas gali būti pritaikytas įvairiose tinklo architektūrose ir turi nemažai papildomų galimybių, kurios leidžia naudoti tradiciniuose tinkluose taikomus maršrutizavimo protokolus. Priešingai nei NETCONF,

OpenFlow reikalauja konkrečios tinklo architektūros, kurioje veikia OpenFlow komutatoriai ir valdiklis, kuriame kaupiama visa tinklo informacija [8].

Ateities sprendimai

MPLS-TP (angl. *Multi-Protocol Label Switching Transport Profile*) – tai standartas, kuris sukurtas duomenų srautų problemos spręsti. Tai MPLS protokolo patobulinimas. MPLS naudoja komutavimą pritaikant žymes, tai padeda išvengti IP tinklų tolimesnio šuolio paieškos ir palaiko interaktyvaus balso bei vaizdo taikymą, norint užtikrinti gerus paslaugų kokybės parametrus. MPLS-TS turi papildomų funkcijų, kaip tinklo palaikymas bei galimybė perduoti bet kokio pobūdžio tradicinių tinklų duomenis. Šis protokolas palaiko statiškus bei dinamiškus valdymo lygmenis, kas leidžia naujų maršrutų sukūrimą be administratoriaus įsikišimo. Šis protokolas dar tik siejamas su diegimu SDN tinkluose, tačiau galėtų sumažinti SDN tinklų su dinamiškais valdymo lygmenimis kompleksiškumą [9].

XMPP (angl. *Extensible Messaging and Presence Protocol*) – protokolas, kurio veikimas pagrįstas XML kalba. Pagrindinis šio protokolo tikslas buvo momentinių žinučių siuntimas, bei prisijungimo būsenos aptikimas. Šis protokolas funkcionuoja tarp serverių ir veikia artimu realiam laikui greičiu. XMPP neseniai iškilo kaip OpenFlow alternatyva mišriuose programuojamuose tinkluose. Jis gali pernešti valdymo ir priežiūros informaciją tarp serverių visais abstrakcijos lygiais [7].

Programuojami tinklai gausūs savo realizacijų ir architektūrų įvairove, tačiau pagrindiniai SDN bruožai yra galimybė suteikti tinklui lankstumo, našumo ir prisitaikyti esant skirtingiems tinklų reikalavimams pritaikant išorinius programuojamo valdymo sprendimus. Todėl galima teigti, kad OpenFlow nėra tas pats, kas SDN ir tai nėra vienintelė pietinė aplikacijų programavimo sąsaja.

1.3 Duomenų lygmens tyrimai

Programuojamuose tinkluose atskiriant duomenų persiuntimo ir valdymo lygmenis, atsiranda daug tyrimo objektų, kurie iššaukia SDN tinklų realizavimo galimybių gausą. Tai tokie objektai, kaip komutatoriai duomenų lygmenyje, skirtingi valdikliai, tinklo topologijos, valdiklių aplikacijos, skirtingi veikimo režimai, tinklo reikalavimai. Dėl to ypatingą svarbą įgauna kitų autorių atliktų tyrimų nagrinėjimas. Šioje dalyje nagrinėjami tyrimai, kuriuose aptariama SDN darbingumas, taikant skirtingus programuojamų tinklų veikimo režimus.

Reaktyvaus modelio nagrinėjimas

SDN tinkluose, kitaip nei tradiciniuose tinkluose, gali atsirasti dideli kontrolinės ir valdymo informacijos srautai. Dėl šios priežasties svarbu žinoti, kokie yra šie susidarantys srautai, kaip juos generuoja ir aptarnauja programuojamų tinklų duomenų lygmuo.

A. R. Curtis atliktame tyrime [10] nagrinėjami OpenFlow sistemos trūkumai, kuomet pritaikomas reaktyvus OpenFlow modelis. Reaktyvaus modelio veikimo principas paremtas naujų srautų aptikimu ir valdiklio klausinėjimu. Taikant tokį modelį, kiekvieną kartą, kuomet komutatoriuje aptinkamas naujas srautas, iš komutatoriaus siunčiamas užklausos pranešimas į valdiklį. Valdiklis tuomet nusprendžia, kaip srautas turi būti aptarnautas, bei siunčia atsakymą į komutatorių. Taip susidaro papildomi kontrolinės ir valdymo informacijos srautai tinkle. Šių srautų siuntimas ir aptarnavimas labai apkrauna tinklo įrenginius. Ši tendencija yra ypatingai ryški, kuomet SDN tinklas yra didelis, bei jame aptarnaujama daug skirtingų srautų, pavyzdžiui internete. Dėl valdymo informacijos atsirandantys duomenų lygmens pralaidumą trikdantys veiksniai [10]:

- Papildomas tinklo apkrovimas duomenimis. Kuomet tinkle nuosekliai sujungta N komutatorių, vienos krypties srauto įrašo sudarymas sukuria $94+144N$ baitų papildomų duomenų.
- Tarp komutatoriaus ir valdiklio yra riboto pralaidumo kanalas. Tai gali trukdyti naujų įrašų siuntimui ir priėmimui. Geriausi žinomi OpenFlow komutatorių sprendimai gali atlikti tik keletą šimtų naujų srautų įrašų įvedimų per sekundę.
- Statistinės informacijos rinkimas tampa dar vienu duomenų generavimo šaltiniu. Kuo dažniau renkama statistinė informacija iš šaltinio, tuo mažiau ateinančių srautų įrašų gali aptarnauti komutatorius.
- OpenFlow taisyklės užima daug vietos komutatoriaus atmintyje. OpenFlow komutatorius savo atmintyje gali laikyti apie 1 500 taisyklių, kuomet įprastas Ethernet komutatorius gali palaikyti iki 64 000 duomenų nukreipimo įrašų.
- Ribotas valdiklio sugebėjimas aptarnauti valdymo informacijos srautus.

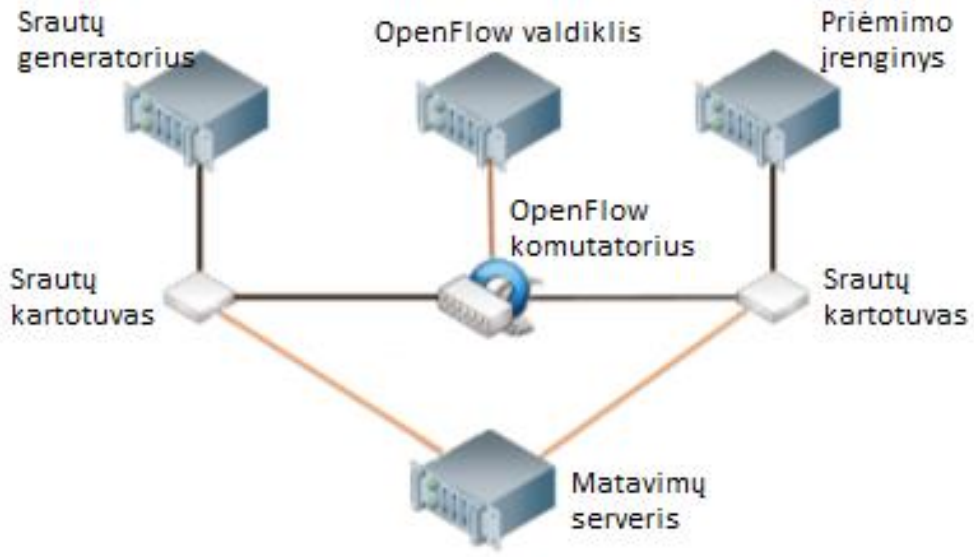
Dėl reaktyvaus OpenFlow modelio apribojimų, galima teigti, kad visiškai reaktyvus srautų valdymo modelis yra netinkamas nuolat didėjančių tinklo informacijos kiekių aptarnavimui. Kyla reikalavimas rasti sprendimą, kurio metu stebėjimo ir valdymo informacijos kiekiai būtų sumažinami, bei aptarnaujami efektyviau.

Proaktyvaus modelio nagrinėjimas [11]

Jarschel „SDN aplikacijų ir darbingumo įvertinimas“ [11] yra vienas iš tyrimų, kuriame nagrinėjamos skirtingos programuojamų tinklų duomenų lygmens realizacijos. Darbe testuojami trys skirtingi komutatoriai:

1) Open vSwitch – tai atviro kodo, programinis daugialygio komutatoriaus įgyvendinimas. Šis komutatorius neturi jam sukurto specialaus fizinio įrenginio, jis yra virtualus. OpenFlow palaikymas yra viena iš šio komutatoriaus galimybių. Šis programinis sprendimas turi galimybę veikti keliuose įrenginiuose, naudojant įrenginių susiejimą hipervizorių pagalba;
2) NetFPGA – sąlyginai pigus eksperimentinis programuojamas fizinio komutatoriaus sprendimas tyrėjams, turintis 1 Gbps pralaidumo sąsajas. Šis įrenginys palaiko nemažai eksperimentinių projektų, tarp kurių yra ir OpenFlow komutatoriaus realizacija;
3) Pronto 3290 – eksperimentinis komutatorius dedikuotas tik OpenFlow veikimui su 1Gbps spartos sąsajomis.

Komutatorių darbingumo nustatymo eksperimentai buvo atliekami pritaikant keletą scenarijų, kuriuose taikomi skirtingi taisyklių rinkiniai ir matuojamas komutatorių sukuriamas vėlinimas perduodant duomenų paketus. Duomenų lygmens tyrimuose srautų įrašai buvo įvedami komutatoriuose taikant proaktyvų SDN modelį. Tai reiškia, kad taisyklės buvo aprašomos ir įvedamos prieš duomenų srauto generavimą, todėl valdiklis daug įtakos komutatorių darbui neturi. Šio tyrimo eksperimentinė tinklo struktūra pateikta 1.2 paveiksle.



1.2 pav. Jarschel tyrimo eksperimentuose naudota tinklo struktūra [11]

Tarp srautą generuojančio ir priimančio įrenginio bei tarp komutatoriaus ir valdiklio sudaromas 1 Gbps pralaidumo sujungimas. Komutatorių apdorojamas srautas buvo generuojamas

Pktgen programos. Šis srautas išnaudojo turimą tinklo pralaidumą. Testai buvo atliekami siunčiant 2 mln. 18, 60, 200, 600, 1000, 1400, 1472 baitų dydžio paketų. Naudojamas NOX valdiklis ir nekintama topologija su vienu komutatoriumi, paketų generatoriumi, valdikliu ir paketus priimančiu kompiuteriu.

Visuose bandymuose programinis komutatoriaus įgyvendinimo variantas rodė prastesnius rezultatus nei techninės įrangos sprendimai. Open vSwitch bandymuose, kuriuose valdiklis neturėjo įtakos, turėjo 0.2-1 ms vėlinimą, Pronto – 0.004-0.01 ms, NetFPGA – 0.002-0.01 ms. Esant mažiems paketams virtualus komutatorius veikė 100 kartų lėčiau, tačiau esant dideliems paketams techninės įrangos sprendimų vėlinimas padidėjo, o Open vSwitch išliko panašus.

Eksperimentų metu nustatyta, kad Open vSwitch ir Pronto komutatoriaus darbingumas nepriklauso nuo srautų įrašų skaičiaus komutatoriuje. Tačiau NetFPGA komutatorius vėlinimas suprastėjo, kuomet srautų įrašų buvo daug. Be to, Open vSwitch bei Pronto komutatorius vienodai sparčiai aptarnavo srautus, kuomet srautų įrašai tiksliai apibūdina srautą ir kuomet srauto įrašas atitinkantis srautą yra aprašomas naudojant mažesnį skaičių kriterijų. NetFPGA komutatoriuje daugelio kriterijų naudojimas aprašant srautus sumažino komutatoriaus darbingumą.

Analizuodamas tyrimą [10] pastebėjau, kad verta atsižvelgti į komutatoriaus palaikomų funkcijų skaičių. OpenFlow SDN dedikuotas komutatorius rodė geriausius darbingumo rezultatus. Kitas fizinis įrenginys bei programinis komutatoriaus sprendimas rodė prastesnius rezultatus, nors šie komutatoriai palaiko daugybę kitų paketinio perdavimo protokolų. Taip pat, kuriant SDN tinklą reikia atsižvelgti į naudojamų taisyklių skaičių, bei jų aprašymo detalumą. Daug ir detalių srautų įrašų gali padidinti vėlinimą tinkle.

1.4 Valdymo lygmens tyrimai

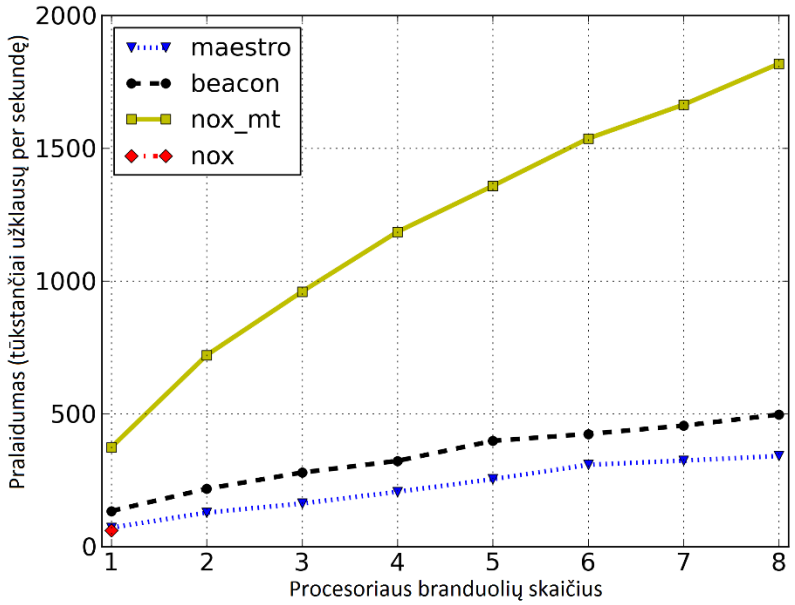
Programuojamų tinklų tyrimuose dažniausiai pagrindinis tyrimų objektas yra valdymo lygmuo. Šiuose tyrimuose dažnai bandoma įvertinti skirtingų valdiklių galimybė susidoroti su valdymo informacija iš daugelio komutatorių.

Vienas iš SDN valdiklius testuojančių tyrimų „SDN aplikacijų ir darbingumo įvertinimas“ [11]. Jame buvo lyginami tokie skirtingų OpenFlow valdiklių darbingumo parametrai, kaip vidutinis paketų keliavimo pirmyn ir atgal laikas bei variacijos koeficientas, paketų siuntimo ir priėmimo sparta, neapdorotų paketų skaičius tam tikrame laiko intervale. Valdiklių parametrų matavimui buvo naudojama straipsnio autorių sukurta testavimo priemonė – OFCBenchmark, o tiriami valdikliai buvo NOX, Floodlight ir Maestro. Kadangi NOX valdiklis nesugeba naudoti daugiau nei vieną kompiuterio procesoriaus branduolį vienu metu, visi valdikliai

buvo apriboti vienu procesoriaus branduoliu. Srautų nukreipimo valdymui valdikliai naudojo savo „besimokančio komutatoriaus“ valdymo aplikacijas.

Tyrimo rezultatuose nurodyta, kad Maestro valdiklis turėjo geriausius parametrų, kurie apibūdina valdymo lygmens darbingumą, rezultatus, nors paketo kelio pirmyn atgal variacijos koeficientas buvo labai nestabilus ir kintantis. Lyginant su kitų valdiklių, šis koeficientas labai svyruoja kol prie valdiklio prijungiama apie 50 virtualių komutatorių.

Tootoonchian atliktame valdiklių tyrime [12], valdikliams buvo suteikta galimybė naudoti daugiau nei vieną procesoriaus branduolį. Šiame tyrime buvo analizuojami NOX, Beacon, ankstesniame tyrime nagrinėtas, Maestro bei patobulintas NOX valdiklis – NOX-MT. Valdiklių testavimui buvo naudojama Cbench programinė įranga. Šiame tyrime Beacon valdiklis pasirodė geriau, nei anksčiau nagrinėtame tyrime [11] geriausiai pasirodęs – Maestro, o geriausiai pasirodė NOX-MT. Tootoonchian tyrime [12] gautos valdiklių užklausų aptarnavimo spartos, pateiktos 1.3 paveiksle.



1.3 pav. Valdiklių darbingumas [12]

Analizuojant skirtingus šaltinius, galime pastebėti, jog juose yra lyginami skirtingi valdikliai, naudojami skirtingi srautų generavimo įrankiai, todėl tyrimų rezultatus vertinti vienareikšmiškai yra sunku.

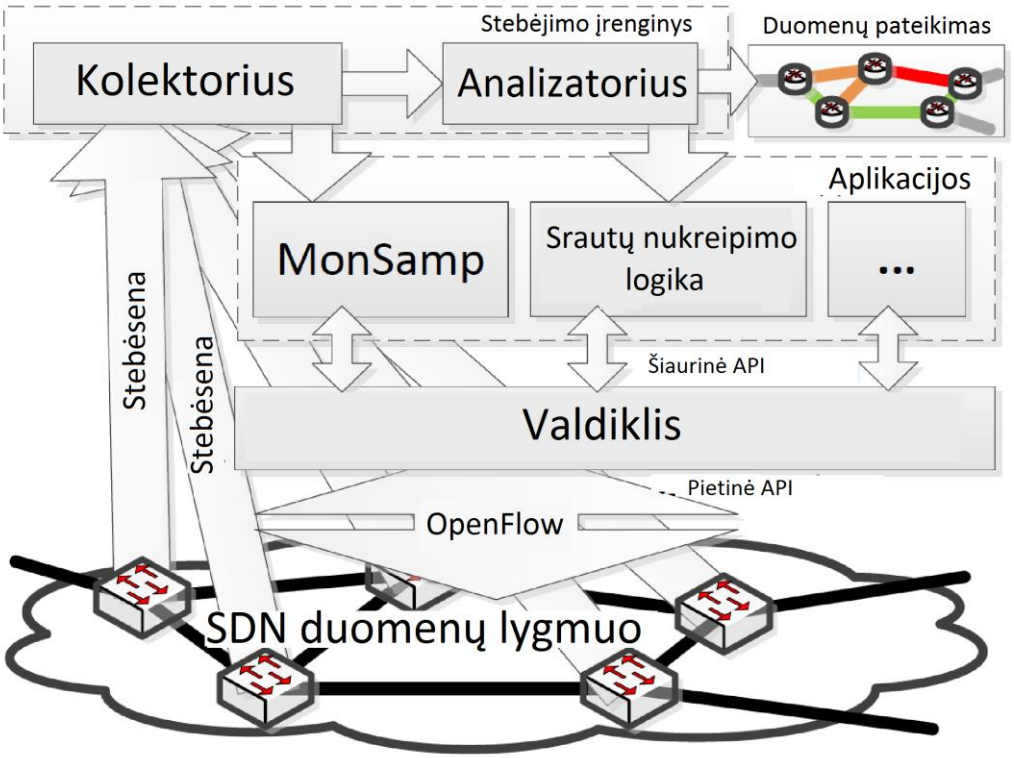
1.5 SDN aplikacijos

Kompiuterių tinkluose įdiegus programinį valdymą, tinklo srautų nukreipimui galima naudoti skirtingas aplikacijas bei algoritmus, kurie galėtų padėti efektyviai išnaudoti tinklo

resursus bei užtikrinti kokybinius srautų parametrus. Kitų autorių nagrinėjamų aplikacijų analizė tampa svarbi, norint sukurti SDN srautų valdymo algoritmą.

QoS stebėjimo sistema

MonSamp [13] – tai tinklo srautų spartos stebėjimo sistema, kuri pritaikoma OpenFlow programuojamuose tinkluose. SDN tinklo, kuriame įdiegta ši kontrolės sistema loginė struktūra pateikta 1.4 paveiksle.



1.4 pav. SDN tinklas su apkrovimo stebėjimo sistema [13]

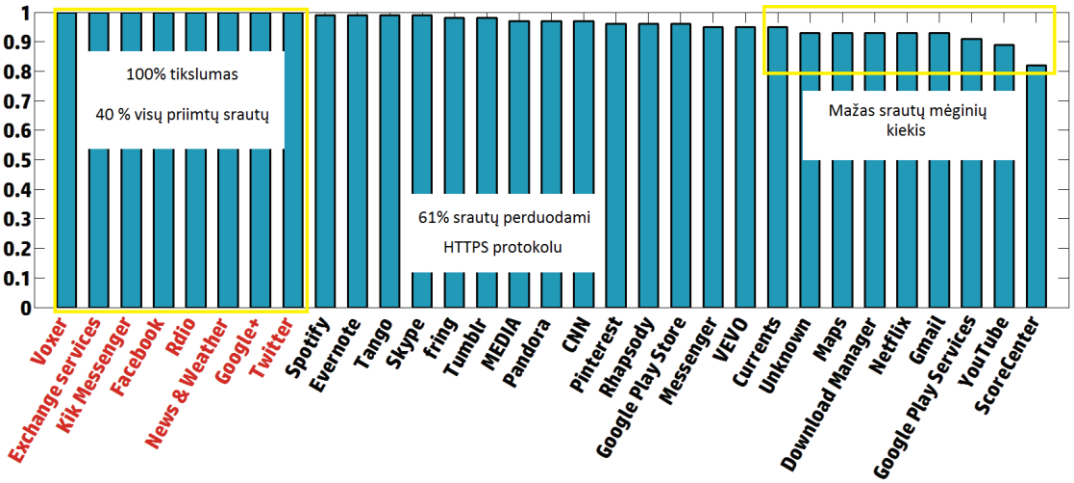
Šioje stebėjimo sistemoje, naudojamas atskiras srautų parametrų stebėjimo įrenginys, į kurį nukreipiami srautai tiesiai iš duomenų lygmens. MonSamp aplikacija gali nukreipti duomenų srautus į atskirus tinklo monitoringo įrenginius, pateikdama valdymo komandas į SDN valdiklį per šiaurinę API. Valdiklis tuomet siunčia srautų nukreipimo taisykles į duomenų lygmenį. MonSamp aplikacija iš stebėjimo įrenginio kolektoriaus gauna informaciją apie stebėjimo įrenginio apkrovimą. Tuomet aplikacija gali nuspręsti, kokią dalį srautų siųsti į kolektorių. Analizatorius atlieka kokybinių parametrų skaičiavimą, bei perduoda ją atvaizdavimui, bei nuorodas į srautų nukreipimo logikos algoritmą. Tokia tinklo stebėsenos sistema išnaudoja jau esantį OpenFlow funkcionalumą, įvertinant stebėjimo įrenginių apkrovimą, egzistuojančias taisykles duomenų lygmenyje [13].

Aplikacijų atpažinimas

ATLAS [14] – tai srautų identifikavimo sistema, kuri atpažįsta, kokia aplikacija generuoja tinkle perduodamą srautą. Šios sistemos veikimo principas paremtas mašininio mokymosi arba ML (angl. *Machine Learning*) mechanizmu, bei OpenFlow protokolu. Sistema sukurta ir išbandyta bevielės prieigos tinkle, kuriame naudojamas OpenFlow protokolą palaikantis prieigos taškas.

ATLAS naudoja tam tikrus agentus, kurie yra įdiegti į vietiniame tinkle esančius klientų įrenginius. Šie agentai kaupia „netstat“ žurnalus, kuriuose nurodyti įrenginiuose vykstančių tinklo sesijų IP adresai ir naudojami TCP/UDP prievadai. Žurnalai yra siunčiami į ML apmokymo modulį, kuris veikia programuojamo tinklo valdiklyje. Kartu su žurnalais OpenFlow protokolu siunčiami ir pirmųjų srauto paketų duomenys (pavyzdžiui pirmųjų N paketų dydžiai). Šie duomenys apmokymo modelyje yra kombinuojami ir sudaromas tam tikrą aplikaciją apibūdinantis duomenų rinkinys. Šių rinkinių komplektas sudaromas rankiniu būdu pateikiant ML mokymosi moduliui didelį kiekį (bent 200) kiekvienos aplikacijos sukuriamų srautų. Sukurtas aplikacijas apibūdinantis masyvas vėliau naudojamas, ML srautų identifikavimo modulyje, kuris veikia bevielės prieigos taške.

Atliktame ATLAS sistemos tyrime [14] buvo bandoma atpažinti 40 populiariausių Android operacinės sistemos aplikacijų. Tyrime, ML apmokymo laikotarpiu, 200 arba daugiau srautų mėginių pavyko surinkti tik trisdešimčiai minėtų aplikacijų. Per 3 savaičių testavimo laikotarpį, buvo surinkta virš 100 tūkst. srautų mėginių, kurių aplikacijas buvo bandoma atpažinti. Bandyme buvo pasiektas 94% vidutinis aplikacijų aptikimo tikslumas, skaičiuojant ir tas aplikacijas, kurių mėginių kiekis apmokymo laikotarpiu nesiekė 200. 100% aptikimo tikslumas buvo pasiektas aštuonioms aplikacijoms, kurios kartu sudarė 40% visų priimtų srautų [14].



1.5 pav. Aplikacijų atpažinimo tikslumas [14]

Pažiūrėję į 1.5 paveiksle pateiktus duomenis, galime pastebėti, kad 61% visų srautų yra perduodami HTTPS protokolu. Tai reiškia, kad sistema gali atpažinti aplikacijas, kurių duomenys

tinkle yra šifruojami. Tokia ar panaši srautų identifikavimo sistema galėtų būti labai naudinga, kuomet tam tikro srauto perdavimui reikalingas kokybės parametrų užtikrinimas ir kokybės užtikrinimo funkcija turi būti atliekama automatiškai.

Tinklo topologijos aptikimas

Modernūs SDN tinklai žada centralizuotą tinklo valdymą, kuriame sudaromas bendras viso tinklo vaizdas ir skaidrumas. Tokiuose tinkluose didelę svarbą sudaro tinklo topologijos, tai yra ryšių ir sujungimų, aptikimas. Tinklo struktūros nustatymas, taip pat, turi būti greitas bei mažai apkrauti tinklo valdiklį.

Ethernet technologijoje naudojamas LLDP (angl. *Link Layer Discovery Protocol*). Šis protokolas remiasi Ethernet kadrų, kurių duomenų lange nurodyta komutatoriaus, bei prievado, iš kurių siunčiamas LLDP kadras, identifikacijos numeriai bei likęs šuolių skaičius – TTL (angl. *Time to live*). Komutatoriai nuolat generuoja LLDP paketus kiekvienai komutatoriaus sąsajai [15].

LLDP protokolu remiasi daugelyje modernių OpenFlow valdiklių naudojamos topologijos aptikimo sistemos, tačiau veikimo principas yra šiek tiek modifikuotas, kadangi OpenFlow komutatoriai neturi teisės keisti kadro duomenų lango informacijos. OpenFlow SDN atveju valdiklis generuoja LLDP paketus kiekvienai komutatoriaus sąsajai ir siunčia juos į komutatorius, su taisykle, kad kiekvienas paketas turi būti nukreiptas iš komutatoriaus per atitinkamą sąsają, kuri nurodyta LLDP pakete. Komutatoriuose yra nurodyta iš anksto aprašyta taisyklė, kurioje nurodyta, kad gavus LLDP paketą reikia, išsiųsti jį į valdiklį (išskyrus tuos atvejus, kai paketas ateina iš valdiklio). Tokiu metodu valdiklis susidaro bendrą tinklo vaizdą. Esant dideliam tinklui valdiklis turi generuoti paketus į visas kiekvieno komutatoriaus sąsajas. Šis procesas atliekamas periodiškai, dažniausiai kas 5 sekundes. Tai sudaro nemažą duomenų srautą ir eikvoja turimus valdymo resursus. Šis mechanizmas naudojamas tokiuose valdikliuose kaip NOX, POX, Floodlight, Ryu, Beacon ir kituose. Skirtingose valdiklių realizacijose skiriasi tik LLDP paketų siuntimo pobūdis laike [15].

Straipsnyje „Efektyvus topologijos aptikimas SDN tinkluose“ [15], siūlomas ryšių sudarymo algoritmas OFPDv2 (angl. *OpenFlow Discovery Protocol*), kuriame valdiklio apkrovimas yra sumažinamas iki 45%. Šis metodas siūlo keletą pakeitimų aptartoje SDN topologijos aptikimo sistemoje [15]:

1) LDDP paketo kadre, prievado identifikacijos numerio lango reikšmė prilyginama nuliui ir į kiekvieną komutatorių siunčiama po vieną LLDP paketą;
2) komutatoriuose įrašoma nauja LLDP paketų priėmimo taisyklė, kurioje nurodoma, kad kiekvienas iš valdiklio priimtas paketas turi būti persiunčiamas į visus komutatoriaus

prievadus, pakeičiant šaltinio MAC adreso lauką antraštėje atitinkamo išsiuntimo prievado MAC adresu;

3) valdiklio LLDP apdorojimo modulyje, vietoj paketų duomenų lauko analizės atliekama antraštės analizė, kurioje prievado identifikacijos numerį atitinka prievado MAC adresas.

Pakeitimų įvedimas leidžia sumažinti valdiklio procesoriaus apkrovimą bei topologijų aptikimo laiką. Efektyvus topologijos aptikimo mechanizmas gali būti taikomas programinio tinklo valdymo sistemose, kuomet ieškoma alternatyvių maršrutų tinklo srautų perdavimui.

1.6 Apibendrinimas

Atlikę SDN duomenų lygmens tyrimų nagrinėjimą, pastebėjome, kad egzistuoja daug skirtingų duomenų lygmens realizacijų, o pagrindiniai duomenų lygmens tyrimų objektai yra – komutatoriuose sukuriamas vėlinimas nukreipiant srautus, sugebėjimas generuoti ir siųsti užklausas apie srautus į valdiklį bei valdiklio komandų priėmimo sparta. Pastebėjime, jog norint užtikrinti spartų skirtingų srautų perdavimą, kyla reikalavimas mažinti tinklo valdymo srautus. Nagrinėjant proaktyvaus modelio SDN tinklus, pamatėme, jog egzistuojantys duomenų lygmens sprendimai yra gana efektyvūs, kuomet naudojamas tikslingas valdymo informacijos įvedimo metodas.

Analizuojant programuojamų tinklų valdiklius, pastebėjome, jog pagrindinis skirtingus valdiklius apibūdinantis parametras yra jo sugebėjimas aptarnauti didelį kiekį užklausų iš komutatorių. Tačiau šis parametras daugiau apibūdina, kokio mąsto tinklas gali būti aptarnaujamas vieno valdiklio, bet nenusako valdiklio srautų nukreipimo logikos efektyvumo. Tai kelia reikalavimą valdiklyje realizuoti aplikacijas, kurios efektyviai išnaudotų turimą duomenų lygmens pralaidumą.

Valdiklio aplikacijų nagrinėjimas atskleidė, kad egzistuoja skirtingų SDN aplikacijų, kurios gali būti panaudotos efektyviam duomenų lygmens pralaidumo bei valdymo lygmens skaičiavimo resursų išnaudojimui. Galime pastebėti, jog norint sukurti pažangų SDN tinklą, jame reikia kombinuoti skirtingus aplikacijų sprendimus, kurie harmoningai veiktų techninėje įrangoje ir tarpusavyje, parenkant bendras programavimo kalbas bei aplikacijų programavimo sąsajas.

SDN tyrimuose dažniausiai analizuojami atsirandantys valdymo informacijos srautai, SDN įrenginių sugebėjimas aptarnauti šiuos srautus bei efektyvus skaičiavimo resursų išnaudojimas, dažnai akcentuojamas SDN tinklų valdymo patogumas, automatizavimo galimybė, valdymo centralizavimo įvedimas, tačiau nėra kalbama, kokius sprendimus reikia priimti, kad duomenų lygmenyje perduodami duomenų srautai būtų aptarnaujami atsižvelgiant į kokybės reikalavimus

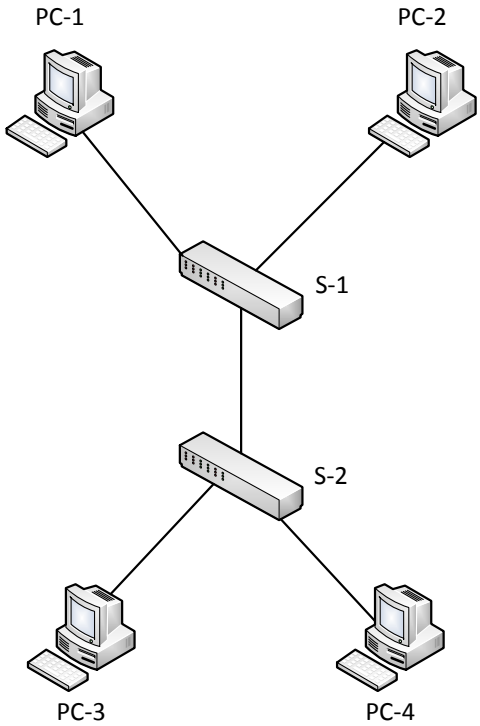
ir turimą duomenų lygmens pralaidumą. Dėl šios priežasties, darbe tiriama, kokią įtaką duomenų srautai daro vienas kitam, ir kaip tinklo valdymas gali šią įtaką sumažinti.

2 TINKLŲ PRALAIDUMO ANALIZĖ

Šiame skyriuje aprašomi eksperimentai, kurie atlikti naudojant realią techninę įrangą, aptariamos gautos priklausomybės, kurios naudojamos imitacinio modelio kūrime. Taip pat, pateikiamas imitacinio modelio veikimo principas, modeliavimo scenarijai ir rezultatai.

2.1 Eksperimentai su technine įranga

Perduodant duomenis komutaciniuose tinkluose, komutatoriams tenka aptarnauti daugelį duomenų srautų vienu metu, dažnais juos nukreipiant į tuos pačius komutatoriaus prievadus. Norėdamas sudaryti duomenų srautų valdymo algoritmą, nagrinėjau, kokią įtaką skirtingi duomenų srautai daro vienas kitam, kuomet taikomi skirtingi komutatoriaus nustatymai, todėl sudariau kompiuterių tinklą, kuriame atlikau eksperimentus. Loginė tinklo struktūra yra pateikta 2.1 pav.



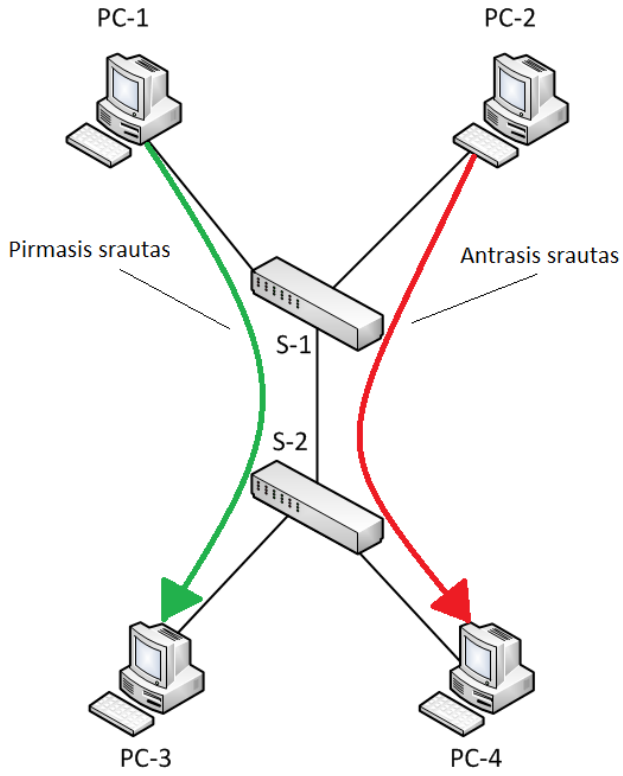
2.1 pav. Eksperimentinis kompiuterių tinklas

Šių eksperimentų rezultatai, darbe naudojami imitacinio modelio kūrimui. Eksperimento tinklą sudaro keturi kompiuteriai su Gigabit Ethernet tinklo sąsajomis, du „Extreme Networks Summit X440-8t“ Gigabit Ethernet komutatoriai ir 6-osios kategorijos UTP kabeliai, kuriais sujungiami įrenginiai. Kompiuteriai žymimi taip: PC-1 – pirmasis kompiuteris, PC-2 antrasis

kompiuteris ir t.t. Atitinkamai pirmasis ir antrasis komutatoriai – S-1 ir S-2. Pirmasis ir antrasis kompiuteriai yra prijungti prie pirmojo komutatoriaus, o antrasis ir ketvirtasis kompiuteriai – prie antrojo komutatoriaus. Komutatoriai tarpusavyje sujungti naudojant vieną perdavimo liniją.

Duomenų perdavimo spartos matavimas, kuomet naudojami standartiniai komutatorių nustatymai

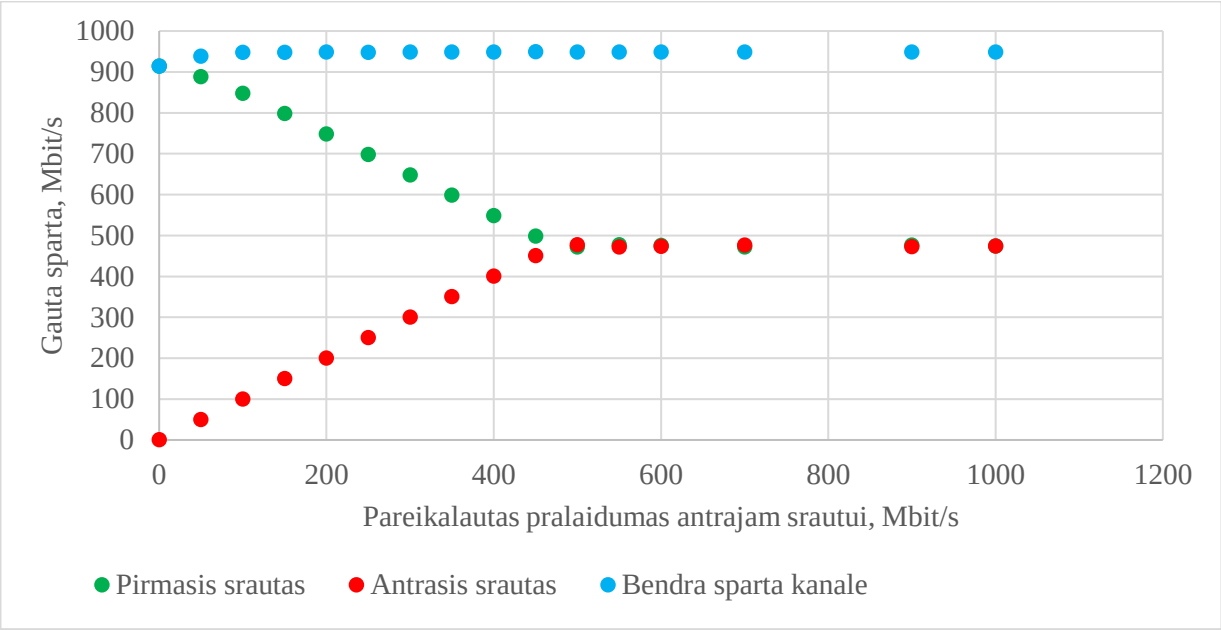
Pirmojo eksperimento metu buvo naudojama standartinė komutatorių konfigūracija bei analizuojama, kaip skirtingi srautai pasidalina suteikiamu komutatoriaus sąsajos pralaidumu. Šiame eksperimente visų komutatoriaus sąsajos yra 1 Gbit/s pralaidumo. Bandymo metu tarp pirmojo ir trečiojo kompiuterių, Iperf programinės įrangos pagalba, generuojamas didžiausios galimos spartos duomenų srautas. Tokiu būdu sukuriamas TCP sujungimas, kuris bando maksimaliai išnaudoti jam suteikiamą kanalo pralaidumą. Šis duomenų srautas yra perduodamas viso eksperimento metu. Tolesniame eksperimento aprašyme jį vadinsime pirmuoju srautu.



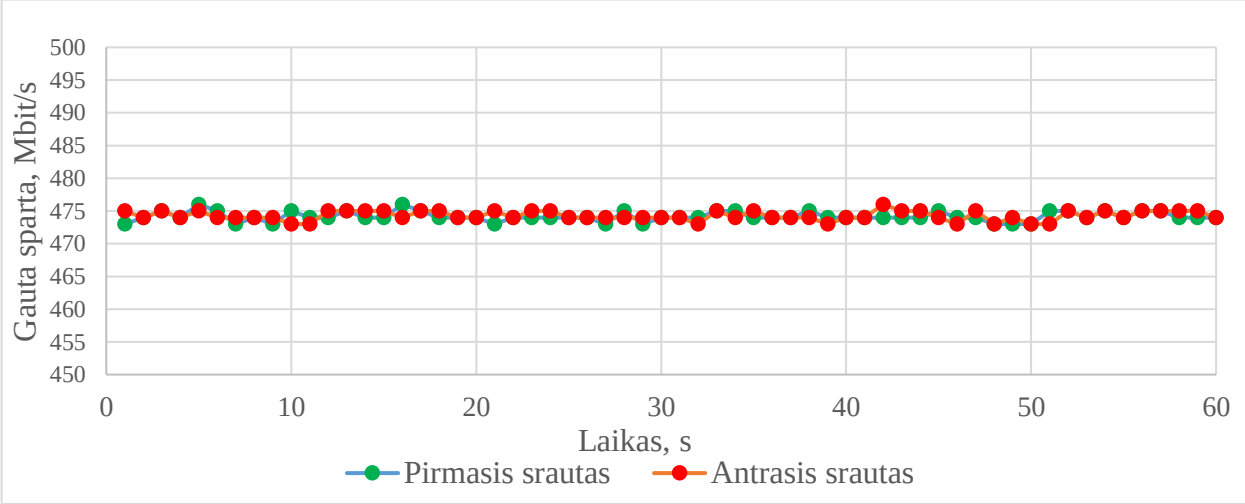
2.2 pav. Loginė eksperimento schema

Nenutraukiant pirmojo srauto perdavimo, tarp antrojo ir ketvirtojo kompiuterio sukuriamas papildomas duomenų srautas, kurio pareikalaujama sparta didinama nuo 50 Mbit/s iki 1000 Mbit/s (pareikalaujama sparta nustatoma Iperf programoje). Šį srautą toliau vadinsime antruoju arba papildomu srautu. Kiekvieną kartą padidinus duomenų perdavimo greitį, atliekame abiejų srautų pasiekiamos duomenų perdavimo spartos matavimą. Duomenų srautai ir jų perdavimo kryptys pateikti 2.2 pav.

Eksperimento, kurio metu naudojami standartiniai komutatoriaus, matavimų rezultatai pateikti 2.3 ir 2.4 paveiksluose. 2.3 pav. pateiktoje diagramoje, kiekvienas taškas atitinka vienos minutės trukmės matavimo rezultatų vidurkį.



2.3 pav. Perdavimo kanalo pralaidumo pasidalinimas, esant standartiniams nustatymams ir dviem duomenų srautams



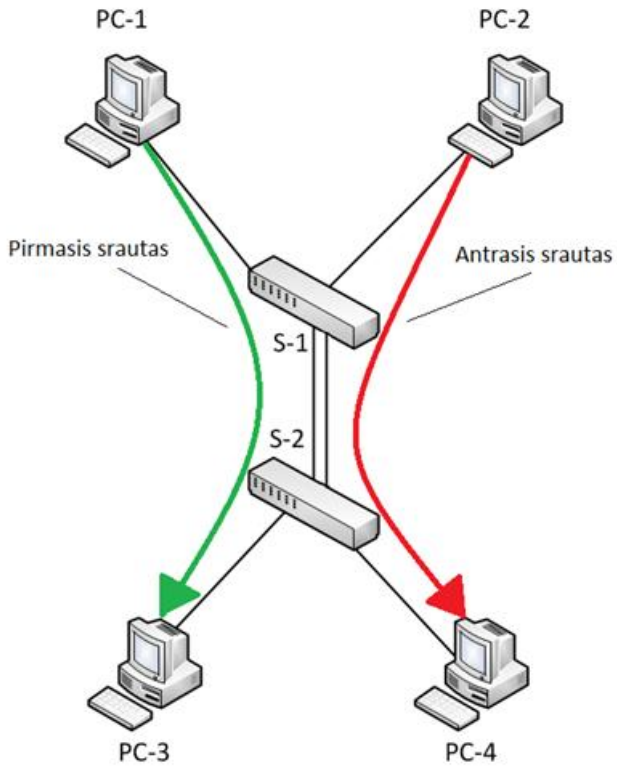
2.4 pav. Duomenų spartos kitimas laike, kuomet perdavimo kanalas maksimaliai išnaudojamas

Pažvelgę į 2.3 pav. pateiktus matavimo rezultatus, galime matyti, jog pirmasis srautas galėjo išnaudoti visą kanalo suteikiamą pralaidumą, kol nepradėjome perduoti duomenis antruoju srautu. Inicijavus antrąjį srautą, jam buvo suteikiamas reikalaujamas pralaidumas, o pirmasis srautas buvo atitinkamai sumažinamas. Šis procesas vyko gana tiesiškai, kol srautai susilygino savo sparta. Todėl galime sakyti, kad, esant standartiniams nustatymams, kanalo pralaidumas komutatoriuje dalinamas taikant sąlyginį prioritetą mažesnės spartos srautui. Tai reikštų, kad, perduodant du atskirus duomenų srautus per vieną komutatoriaus sąsają ir esant visiškam kanalo

išnaudojimui, mažesnės spartos srautui bus suteikiamas reikalaujamas pralaidumas, kol minėtieji srautai netampa vienodo didumo. 2.4 pav. galime pastebėti, kad duomenų perdavimo spartos fliuktuacijos laike yra labai nežymios.

Duomenų perdavimo spartos matavimas, kuomet sudaromos dvi perdavimo linijos tarp komutatorių

Sekančiame eksperimente sudarome papildomą perdavimo liniją tarp komutatorių. Tokiu būdu perduodami srautai yra izoliuojami vienas nuo kito, tai yra perduodami skirtingomis perdavimo linijomis (skirtingais maršrutais). Eksperimento schema pateikta 2.5 pav.

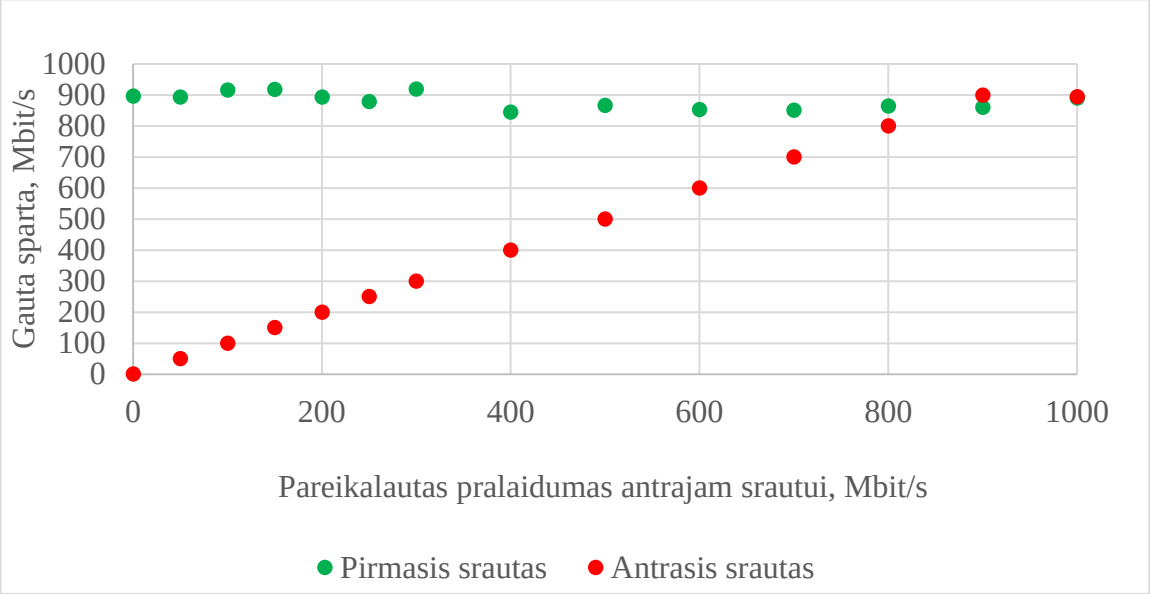


2.5 pav. Eksperimento, kuriame naudojamos dvi perdavimo linijos, loginė tinklo struktūra

Naudojant standartinius nustatymus, sudarius dvi perdavimo linijas tarp komutatorių, būtų inicijuojamas STP (angl. Spanning Tree Protocol) protokolas ir viena iš linijų būtų deaktyvuojama. Todėl virtualiai izoliuojame prievadus, į kuriuos patenka perduodami srautai, juos priskiriant atskiriems virtualiems tinklams VLAN (angl. Virtual Local Area Network). Pirmasis ir trečiasis kompiuteriai bei pirmoji perdavimo linija yra priskiriama pirmajam virtualiam tinklui – VLAN1, o antrasis ir ketvirtasis kompiuteriai bei antroji perdavimo linija yra priskiriama antrajam virtualiam tinklui – VLAN2. Eksperimento metu siekiama ištirti ar tame pačiame komutatoriuje, vienas nuo kito izoliuoti srautai daro įtaką vienas kitam.

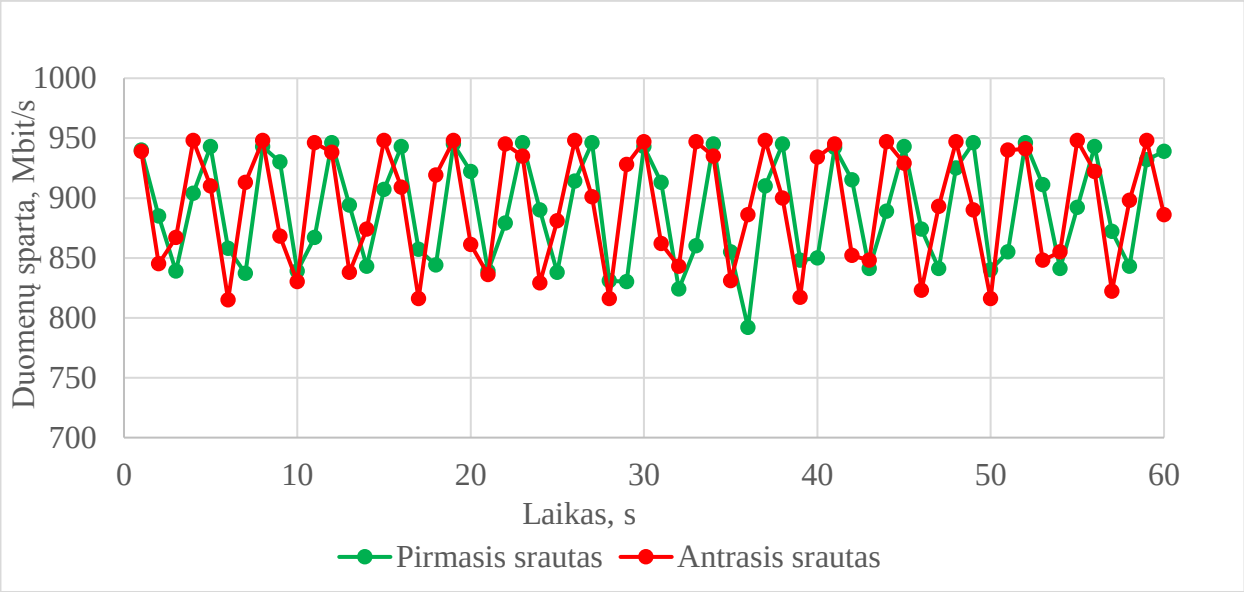
Srautų generavimas atliekamas tokiu pačiu principu, kaip ir ankstesniame eksperimente. Pradžioje sukuriamas pastovus srautas maksimaliai išnaudojantis kanalo pralaidumą (pirmasis

srautas), o vėliau pridedamas antrasis srautas, kurio pareikalaujamas pralaidumas didinamas nuo 50 Mbit/s iki 1000 Mbit/s. Eksperimento metu gauti duomenys yra atvaizduoti 2.6 ir 2.7 paveiksluose.



2.6 pav. Perdavimo kanalo pralaidumo pasidalinimas, kuomet srautai perduodami skirtingomis perdavimo linijomis

2.7 pav. pateiktoje diagramoje matome, kad duomenų spartos vertės abiejų srautų atveju kinta maždaug 800 – 950 Mbit/s intervale. Srautų perdavimo greičio vidurkiai, prie maksimalios pareikalautos spartos minutės intervale, yra atitinkamai 890 ir 894 Mbit/s pirmajam ir antrajam srautui. Be to, didėjant antrojo srauto spartai, pirmojo srauto sparta neturi aiškios kitimo tendencijos, todėl galime teigti, kad atskiromis linijomis perduodami duomenų srautai, vienas kitam įtakos neturi.

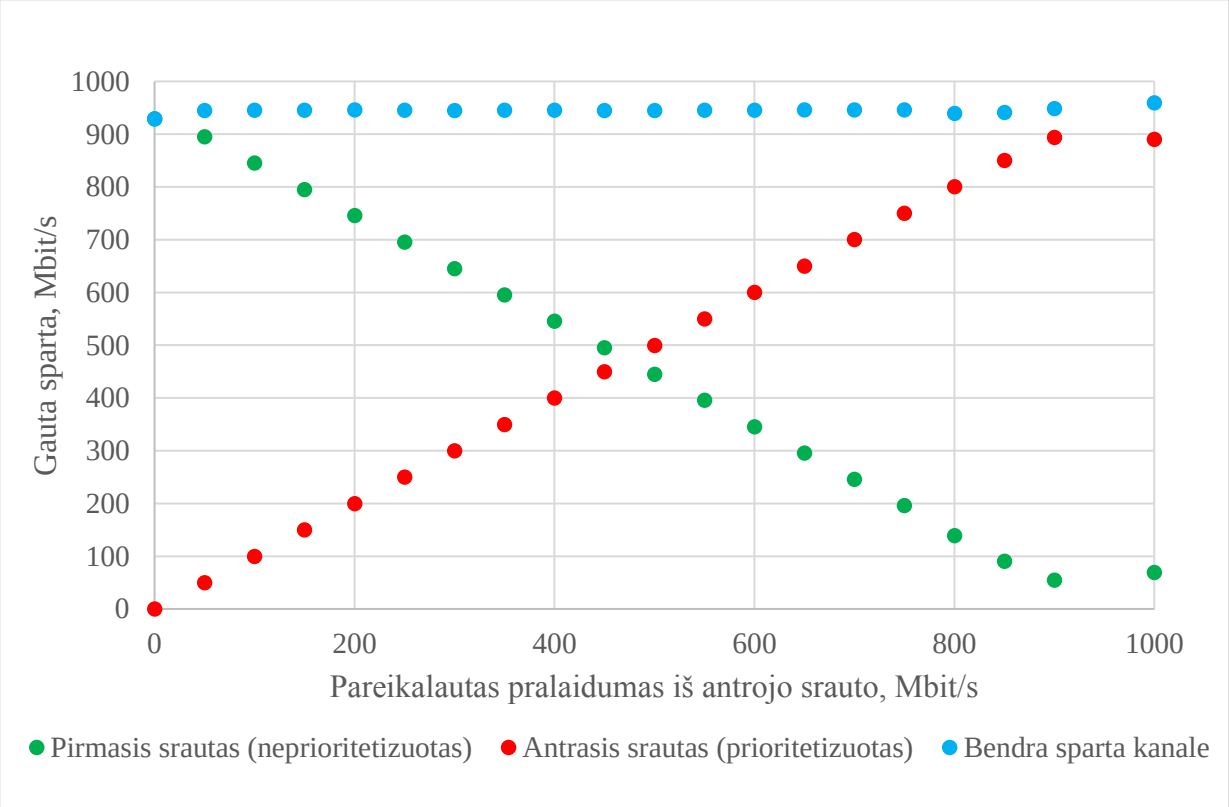


2.7 pav. Perdavimo spartos kitimas laike, kuomet duomenys perduodami didele sparta

Duomenų perdavimo spartos matavimas, kuomet srautams taikomi skirtingi prioritetai

Dar vienas srautų valdymo variantas yra suteikti skirtingus prioritetus srautams, kurie perduodami ta pačia komutatoriaus sąsają. Eksperimento schema atitinka pirmajame eksperimente naudotą sujungimą (2.1 pav.). Norėdami atlikti eksperimentą, pirmąjį ir trečiąjį kompiuterius priskiriame pirmajam virtualiam tinklui – VLAN1, o antrąjį ir ketvirtąjį – VLAN2. Perdavimo linija priskiriama abiem virtualiems tinklams. Ji elgsis kaip kamieninė (angl. *trunk*) linija ir atliks tik VLAN1 ir VLAN2 esančių duomenų perdavimą. Prioritetų nustatymui naudojame 802.1p prioriteto antraštę, kuri yra pridedama į antrojo virtualaus tinklo prievadus patenkantiems paketams. Pirmasis virtualus tinklas šios antraštės eksperimente nenaudoja. VLAN2 paketai turėtų būti aptarnaujami pirmiau, nes yra statomi į aukštesnio prioriteto eilę.

Duomenų perdavimas atliekamas, kaip ir pirmajame eksperimente (2.2 pav.), o srautai generuojami sukuriant pastovų maksimalų srautą tarp pirmojo ir trečiojo kompiuterio (šis srautas prioriteto antraščių nenaudoja). Nepertraukiant pirmojo srauto, generuojamas antrasis srautas, kurio pareikalaujamas pralaidumas didinamas nuo 50 Mbit/s iki 1000 Mbit/s. Šio srauto paketai žymimi 802.1p prioriteto antrašte. Šio eksperimento metu gauti, duomenys yra grafiškai pateikti 2.8 pav. esančioje diagramoje.

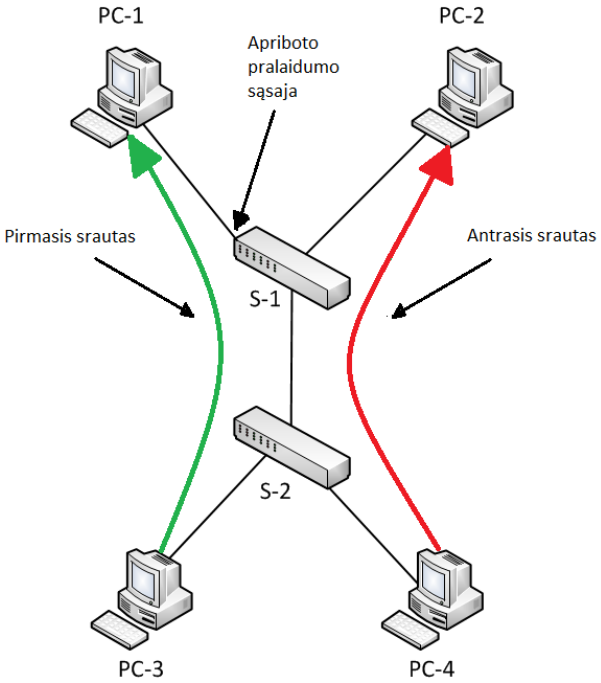


2.8 pav. Perdavimo kanalo pralaidumo pasidalinimas, kuomet perduodami srautai yra skirtingo prioriteto lygmens

Šiame eksperimente gautus rezultatus palyginsime su eksperimento, kuriame naudojami standartiniai komutatoriaus nustatymai, rezultatais (2.3 pav.). Taikant standartinius komutatoriaus nustatymus, mažesnės spartos srautas gauną visą reikalingą pralaidumą, kol srautai išsilygina. Tą patį pastebime tuomet, kai antrajam srautui suteikiame prioritetinę antraštę. Pagrindinis skirtumas atsiranda, kuomet srauto spartą didiname virš susilyginimo ribos. Kaip ir tikėtasi, srautas turintis aukštesnį prioritetą, išnaudoja didesnę kanalo suteikiamo pralaidumo dalį. Tokiu būdu srauto, kuris neturi prioritetinės antraštės, sparta yra mažinama tiek, kiek to reikalauja aukštesnio prioriteto srautas. Pavyzdžiui, esant standartiniams nustatymams, reikalaujant iš pirmojo ir antrojo srauto 700 Mbit/s spartos, abu srautai pasidalintų kanalo pralaidumą po lygiai, t.y. vidutiniškai po maždaug 474 Mbit/s (vidutinis maksimalus kanalo pralaidumas pirmojo eksperimento metu buvo 948 Mbit/s). Perduodant duomenis skirtingo prioriteto srautais 700 Mbit/s perdavimo greičiu, prioritetinis srautas gaus visus 700 Mbit/s, o mažesnio prioriteto, arba neprioritetinis, srautas gaus likusią kanalo pralaidumo dalį, tai yra apie 248 Mbit/s.

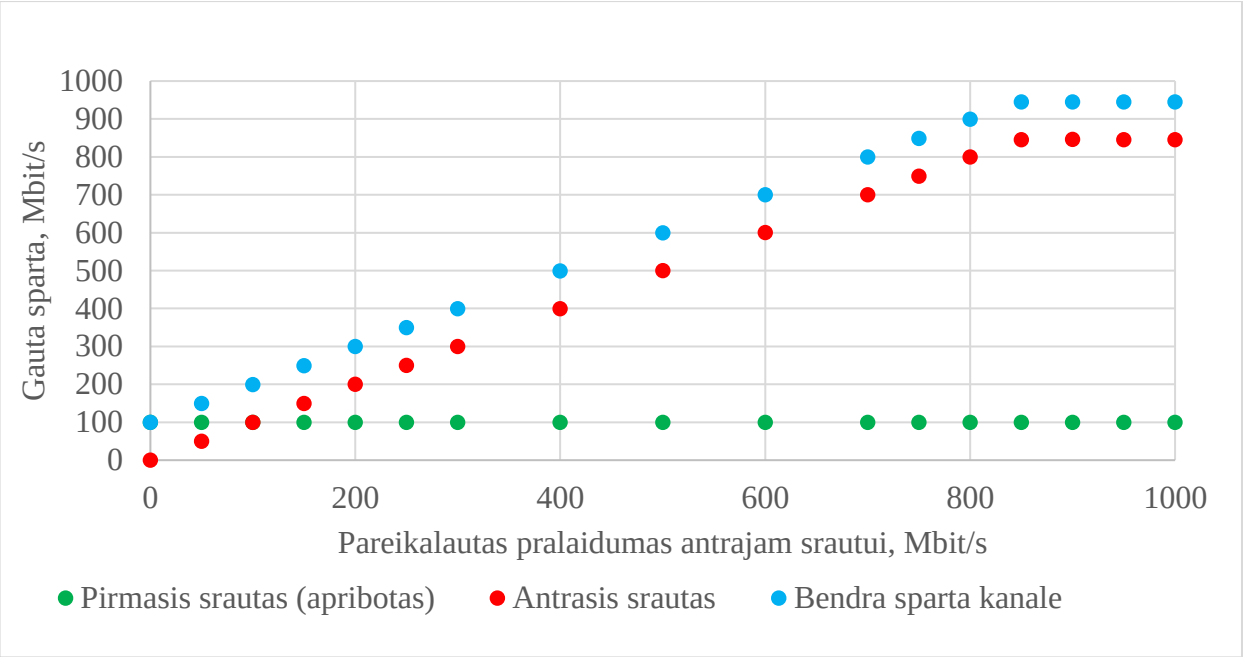
Duomenų perdavimo spartos matavimas, kuomet srautų spartos apribojamos įėjime

Komutatoriaus sąsajos suteikiamo pralaidumo apribojimas, taip pat yra duomenų srautų valdymo variantas. Norėdamas nustatyti, kaip srautai dalinasi kanalo pralaidumu, kuomet vienas iš srautų yra apribojimas, atlikau eksperimentus. Bandymų metu buvo ribojamas S-1 komutatoriaus PC-1 prievado pralaidumas ir naudojama viena perdavimo linija tarp komutatorių (2.9 pav.).

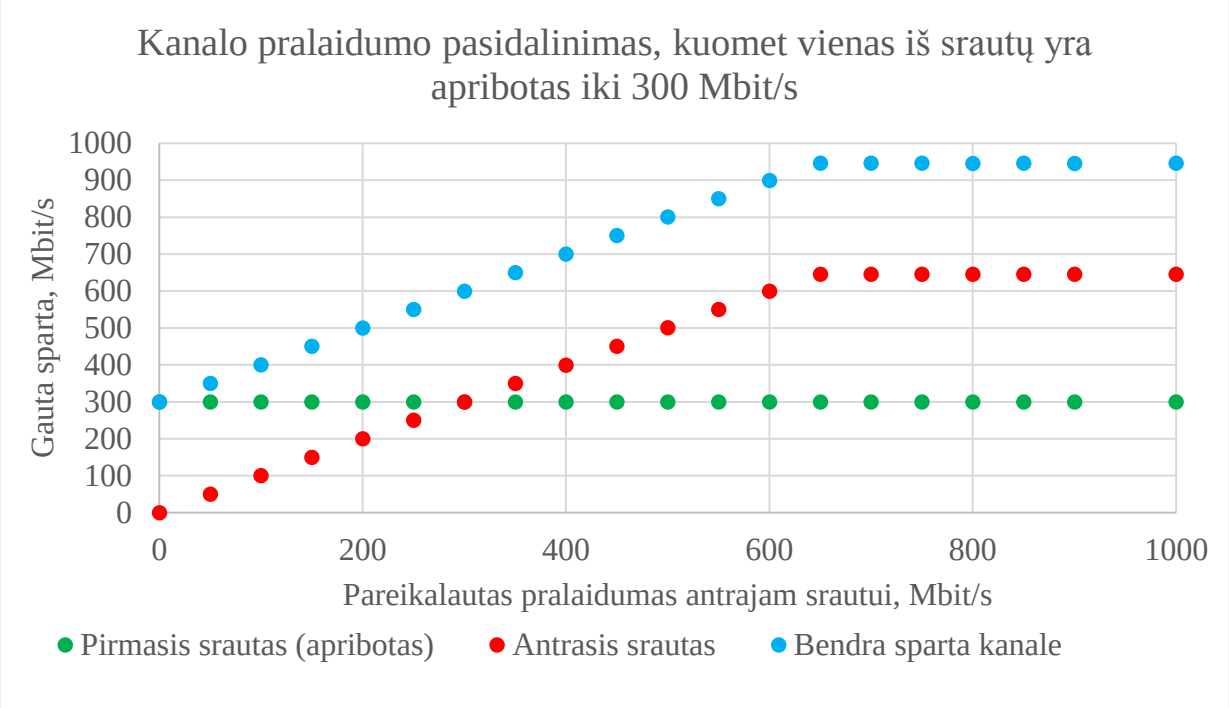


2.9 pav. Eksperimento, kuriame apribojamas vieno iš komutatoriaus prievadų pralaidumas, loginė schema

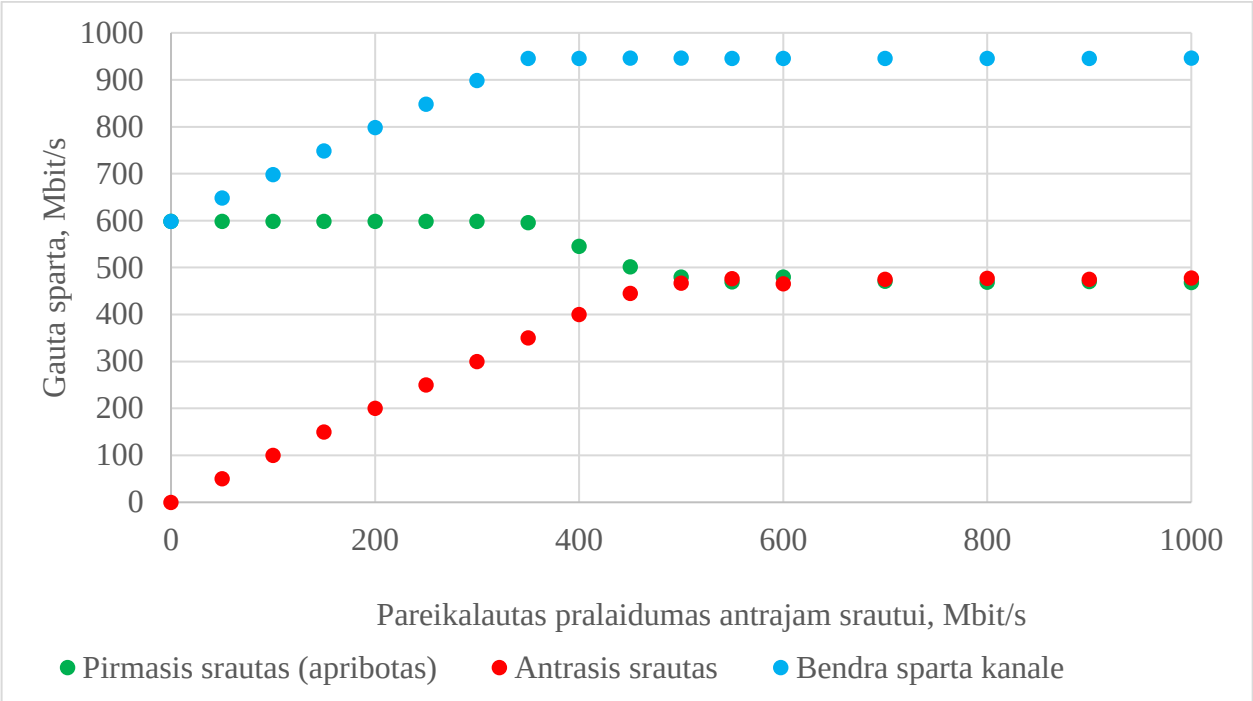
Bandymu metu pirmasis srautas (tarp PC-1 ir PC-3) buvo apriboto pralaidumo ir nuolat mėginantis visiškai išnaudoti suteikiamą kanalo pralaidumą. Antrojo srauto (tarp PC-2 ir PC-4) pareikalaujama sparta buvo didinama nuo 50 iki 1000 Mbit/s. Sąsajos leidžiama perdavimo sparta buvo ribojama prie 100, 300, 600 ir 700 Mbit/s reikšmių. Reikšmės pasirinktos dėl srautų spartų išsilyginimo prie maždaug 470 Mbit/s, esant standartiniams nustatymams. 2.10 – 2.13 paveiksluose atvaizduoti prievado spartos ribojimo eksperimento duomenys.



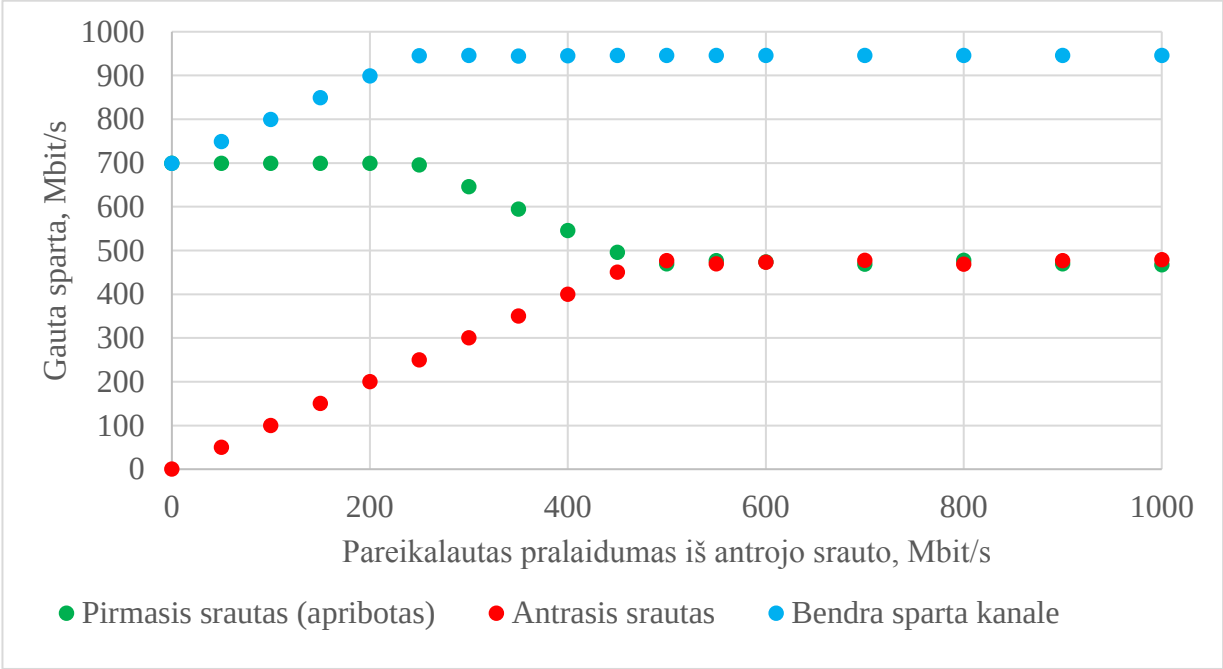
2.10 pav. Kanalo pralaidumo pasidalinimas, kai vienas iš srautų apribotas iki 100 Mbit/s



2.11 pav. Kanalo pralaidumo pasidalinimas, kai vienas iš srautų apribotas iki 300 Mbit/s



2.12 pav. Kanalo pralaidumo pasidalinimas, kai vienas iš srautų apribotas iki 600 Mbit/s



2.13 pav. Kanalo pralaidumo pasidalinimas, kai vienas iš srautų apribotas iki 600 Mbit/s

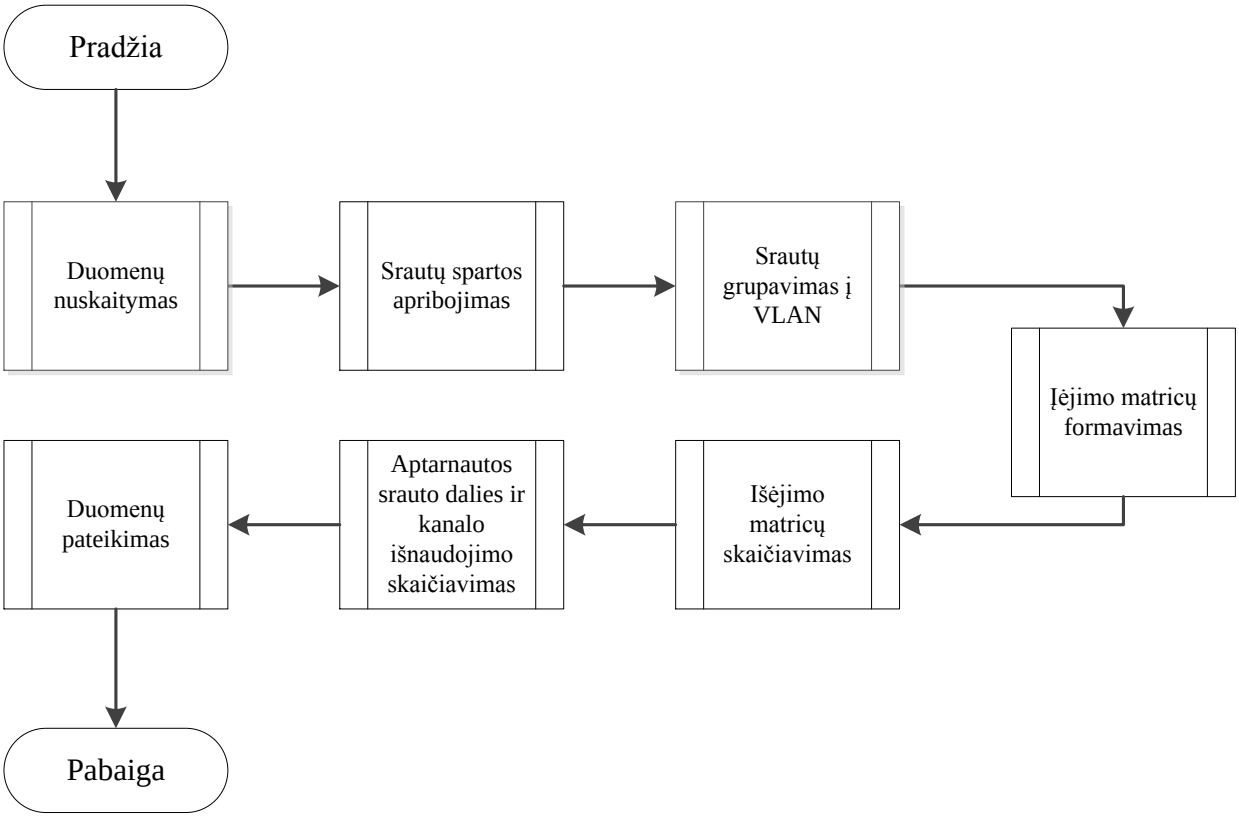
Analizuodami gautas diagramas, galime pastebėti, kad esant mažai apriboto srauto spartai (sparta mažesnė nei pusė maksimalaus kanalo pralaidumo), tuomet apribotas srautas išlaiko savo duomenų perdavimo greitį, kad ir kiek didintume antrojo srauto spartą (2.10 ir 2.11 pav.). Kuomet apribotas srautas yra didelis (didesnis, nei pusė maksimalaus pralaidumo), išnaudojus turimą kanalo pralaidumą, primojo srauto sparta mažėja, kol abiejų srautų spartos tampa vienodo didumo (2.12 ir 2.13 pav.). Šio eksperimento metu, pastebime, kad apriboto ir neribojamo srautų kanalo pasidalinimo tendencija išlieka tokia pati, kaip ir esant standartiniams nustatymams – mažesnio

perdavimo greičio srautas turi sąlyginį prioritetą didesniojo atžvilgiu, kol srautai išsilygina, dėl maksimalaus kanalo išnaudojimo.

Šiame skyriuje atliktų eksperimentų rezultatai naudojami imitacinio modelio kūrimui.

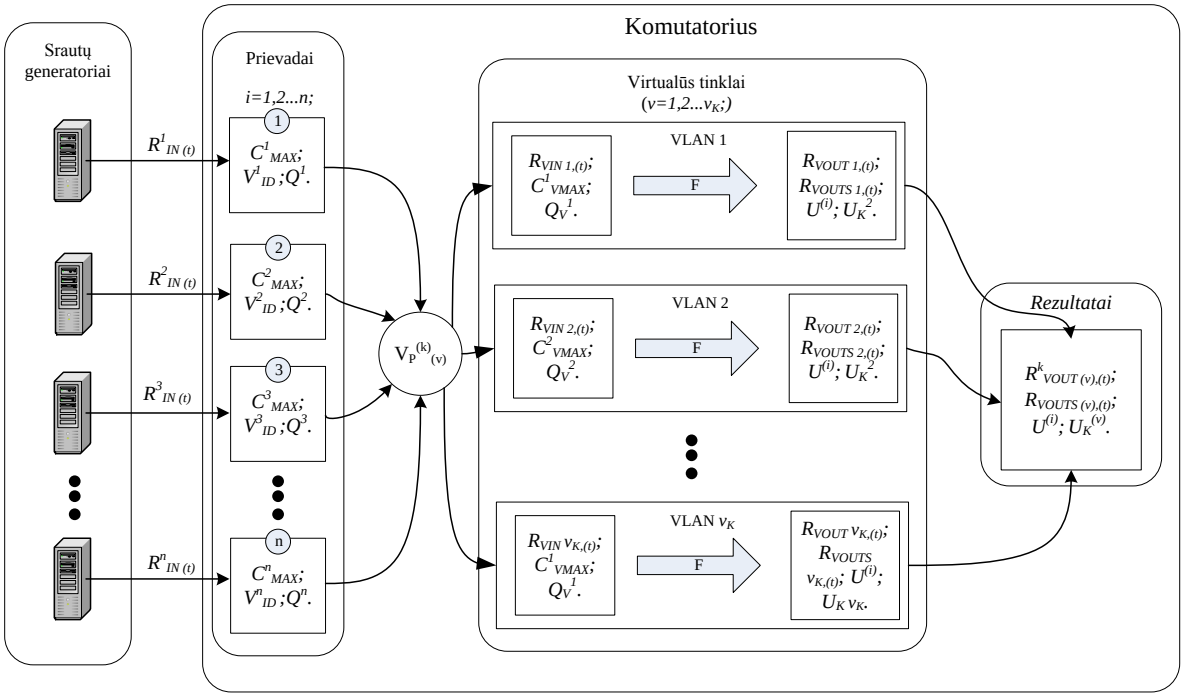
2.2 Imitacinio modelio kūrimas

Norėdamas geriau iliustruoti komutatorių valdymo sprendimus, MATLAB aplinkoje buvo sukurtas imitacinis komutatoriaus modelis, kurį naudojant galima apskaičiuoti srautų spartas komutatoriaus išėjimuose, įvertinti aptarnautą srauto dalį bei perdavimo kanalo išnaudojimą. Modelyje atsižvelgiama į komutatoriaus sąsajų prioritetus, išėjimo kanalų skaičių, įėjimo sąsajų pralaidumą. Šis modelis sukurtas remiantis anksčiau atliktuose eksperimentuose, kuriuose naudojama techninė įranga, gautomis priklausomybėmis. Imitacinis modelis suteikia galimybę stebėti, kaip kinta kanale perduodamų srautų spartos, kuomet perduodama daugiau srautų, be to leidžia apibūdinti srautų kitimo pobūdį laike, kas suteikia papildomą naudą atvaizduojant srautų kitimo priklausomybes. Imitacinio modelio veikimo algoritmas pateiktas 2.14 pav.



2.14 pav. Imitacinio modelio veikimo algoritmas

Šį modelį, tai pat atvaizdavau schema, kurioje atsispindi modeliuojamas komutatorius, modelio įgyvendinimui naudojami įvesties duomenys, skaičiavimo kintamieji ir matricos, išvesties duomenys (2.15 pav.).



2.15 pav. Imitacinio modelio elementų schema

Modelyje naudojami įvesties kintamieji:

- $R^{(i)}_{IN\,(t)}$ – srautų įėjimo spartų matrica. Šioje matricoje pateikta *i* srauto sparta (Mbit/s), *t* laiko momentu (čia $i \in 1,2,..\,n;\ t \in 1,2,..\,l$), kai *n* – prievadų skaičius, *l* – laiko atskaitų skaičius:

$$R^{i}_{IN\,t} = \begin{vmatrix} R^{i=1}_{IN\ t=1} & R^{i=1}_{IN\ t=2} & \cdots & R^{i=1}_{IN\ t=l} \\ R^{i=2}_{IN\ t=1} & R^{i=2}_{IN\ t=2} & \cdots & R^{i=2}_{IN\ t=l} \\ \vdots & \vdots & \vdots & \vdots \\ R^{i=n}_{IN\ t=1} & R^{i=n}_{IN\ t=2} & \cdots & R^{i=n}_{IN\ t=l} \end{vmatrix}, \tag{2.1}$$

pavyzdžiui

$$R^{i=1,2,3}_{IN\ \ t=1,2..5} = \begin{vmatrix} 500 & 1000 & 1000 & 1000 & 1000 \\ 0 & 100 & 200 & 600 & 1000 \\ 0 & 200 & 400 & 600 & 800 \end{vmatrix}; \tag{2.2}$$

- $C^{(i)}_{max}$ – maksimalus įvesties prievado pralaidumas (Mbit/s):

$$C^{i}_{\max} = \left| C^{i=1}_{\max} \quad C^{i=2}_{\max} \quad \cdots \quad C^{i=n}_{\max} \right|, \tag{2.3}$$

pavyzdžiui

$$C^{i=1,2,3}_{\max} = \left| 1000 \quad 1000 \quad 1000 \right|; \tag{2.4}$$

- $V^{(i)}_{ID}$ – virtualus vietinis tinklas, kuriam priklauso prievadas:

$$V^{i}_{ID} = \left| V^{i=1}_{ID} \quad V^{i=2}_{ID} \quad \cdots \quad V^{i=n}_{ID} \right|, \tag{2.5}$$

pavyzdžiui

$$V_{ID}^{i=1,2,3} = \left|1 \quad 1 \quad 2\right|; \tag{2.6}$$

- Q^i – prievado prioriteto lygis:

$$Q^i = \left|Q^{i=1} \quad Q^{i=2} \quad \cdots \quad Q^{i=n}\right|, \tag{2.7}$$

pavyzdžiui

$$Q^i = \left|2 \quad 1 \quad 1\right|. \tag{2.8}$$

Kiti modelyje naudojami žymėjimai:

- V_P – VLAN priklausančių prievadų indeksų masyvas;
- R_{VIN} – VLAN priklausančių prievadų įėjimo spartų matrica;
- C_{VMAX} – išėjimo kanalo pralaidumas;
- Q_V – VLAN tinklo prioritetų matrica;
- F – pralaidumo padalinimo funkcija;
- R_{VOUT} – išėjimo spartų matrica;
- R_{VOUTS} – bendra sparta perdavimo kanale;
- U – aptarnauta srauto dalis procentais;
- U_K - kanalo išnaudojimas procentais.

Pagal 2.14 pav. pateiktą imitacinio modelio algoritmą, po įvesties duomenų nuskaitymo vykdomas srautų spartų apribojimas, t.y. jeigu srauto sparta viršija maksimalų prievado pralaidumą, ji yra sumažinama iki maksimalios leistinos perdavimo spartos. Modelyje atliekamas prievadų spartų stebėjimas ir valdymas iš anksto žinomam signalui įėjime:

$$R_{IN\,t}^i = \left|\begin{array}{l} i \leftarrow 1; \\ kol \quad i \le n \\ \quad \left|\begin{array}{l} t \leftarrow 0; \\ kol \quad t \le l \\ \quad \left|\begin{array}{l} R_{IN\,t}^i \leftarrow C_{\max}^i, \quad jei \quad R_{IN\,t}^i > C_{\max}^i; \\ t \leftarrow t+1; \end{array}\right. \\ i \leftarrow i+1; \end{array}\right. \\ rez.: \quad R_{IN\,t}^i; \end{array}\right. \tag{2.9}$$

čia i – prievado numeris, t – laiko momentas ($i \in 1,2,..\,n;\ t \in 1,2,..\,l$), o „←“ – priskyrimo ženklas, kuris parodo, kad kairėje ženklo pusėje esančiam kintamajam yra priskiriama dešinėje jo pusėje esanti vertė arba veiksmų rezultatas.

Į spartų apribojimo funkciją pateikus 2.2 ir 2.3 formulėse esančius pavyzdinius duomenis, gauname naują įėjimo spartų matricą:

$$R_{IN\ t=1,2..5}^{i=1,2,3} = \begin{vmatrix} 500 & 1000 & 1000 & 1000 & 1000 \\ 0 & 100 & 200 & 600 & 1000 \\ 0 & 200 & 400 & 600 & 800 \end{vmatrix}. \quad (2.10)$$

Norint suskirstyti prievadus ir į juos patenkančius duomenų srautus į atskirus VLAN tinklus, imitaciniame modelyje yra sukuriami indeksų masyvai V_P, kuriuose yra nuorodos į virtualiems tinklams priklausančius prievadus:

$$V_{P(v)}^{k} = \left| V_{P(v)}^{k=1} \quad V_{P(v)}^{k=2} \quad \cdots \quad V_{P(v)}^{k=h_{(v)}} \right|, \quad (2.11)$$

čia k – prievado numeris VLAN tinkle, h – VLAN tinkle esančių prievadų kiekis, v – VLAN numeris, v_K – VLAN tinklų skaičius ($k \in 1,2, .. h;\ v \in 1,2, .. v_K$).

Indeksų matricos sukūrimo funkcija:

$$[V_{P(v)}^{(k)}, h_{(v)}, v_K] = \begin{vmatrix} h \leftarrow 0; \quad i \leftarrow 1; \\ kol \quad i \le n \\ \quad \begin{vmatrix} h_{V_{ID}^i} \leftarrow h+1; \\ V_{PV_{ID}^i} \leftarrow i; \\ i \leftarrow i+1; \end{vmatrix} \\ v_K \leftarrow \max(V_{id}); \\ rez.: \quad V_{P\ (v)}^{(k)}; h_{(v)}; \quad v_K. \end{vmatrix}. \quad (2.12)$$

Į indeksų matricos funkciją pateikus 2.5 formulėje pateiktus pavyzdinius duomenis, gauname skirtingo ilgio masyvus, jų ilgį, bei VLAN tinklų skaičių:

$$V_{P\,v=1}^{k} = [1 \quad 2]; \quad (2.13)$$

$$h_{v=1} = 2; \quad (2.14)$$

$$V_{P\,v=2}^{k} = [3]; \quad (2.15)$$

$$h_{v=2} = 2; \quad (2.16)$$

$$v_K = 2. \quad (2.17)$$

Naudojant gautus indeksų masyvus, pradinės prievadų prioritetų (2.7 formulė) ir įėjimo spartų matrica (2.1 formulė) sugrupuojamos į atskiras VLAN prioritetų ir įėjimo matricas, pagal VLAN prievadų priklausomybes. Prioritetų matrica $Q_{V\,(v)}$ sudaroma įrašant „1“ į tas matricos pozicijas, kur stulpelis atitinka v VLAN tinkle esančio prievado numerį k, o eilutė atitinka prievado prioriteto lygį $Q^{V_{P(k)}}$. VLAN įėjimo srautų matrica $R_{VIN\,(v)}$ sudaroma iš R_{IN} matricos surašant tik tas eilutes, kurios nurodo prievadus priklausančius virtualiam tinklui v:

$$\left[Q_V^{k,V_P^k}{}_v, R^k_{VIN\,v,t}\right]= \left| \begin{array}{l} v \leftarrow 1; \\ kol \quad v \le v_K \\ \quad \left| \begin{array}{l} Q_V \leftarrow zeros(z, h_v); \\ R_{VIN} \leftarrow zeros(z, h_v); \\ k \leftarrow 1; \\ kol \quad k \le h_v \\ \quad \left| \begin{array}{l} Q_V^{k,Q_P^{V_k}}{}_v \leftarrow 1; \\ t \leftarrow 0; \\ kol \quad t \le l \\ \quad \left| \begin{array}{l} R^k_{VIN\,v,t} \leftarrow R^{V_P^k}_{IN\,t}; \\ t \leftarrow t+1; \end{array} \right. \\ k \leftarrow k+1; \end{array} \right. \\ v \leftarrow v+1; \end{array} \right. \\ rez.: Q^k_{V\,v}; \quad R^k_{VIN\,v,t}. \end{array} \right. , \qquad (2.18)$$

čia *z* – skirtingų prioritetų skaičius, funkcija *zeros(x,y)* - atlieka matricos užpildymą nuliais, kai *x* – matricos eilučių skaičius, *y* – stulpelių skaičius.

2.12 formulėje gautus skaičiavimo rezultatus (2.13 – 2.17 formulės) bei pavyzdinius duomenis iš pradinių *Q* ir R_{IN} matricų (2.2 ir 2.8), pateikus į 2.18 formulėje esančią funkciją, gauname naujus VLAN prioritetų ir įėjimo masyvus:

$$R_{VIN\,v=1,t=1,2..5} = \begin{vmatrix} 500 & 1000 & 1000 & 1000 & 1000 \\ 0 & 100 & 200 & 600 & 1000 \end{vmatrix}; \qquad (2.19)$$

$$Q_{V\,v=1} = \begin{vmatrix} 1 & 1 \\ 0 & 0 \end{vmatrix}; \qquad (2.20)$$

$$R_{VIN\,v=2,t=1,2..5} = \begin{vmatrix} 0 & 200 & 400 & 600 & 800 \end{vmatrix}; \qquad (2.21)$$

$$Q_{V\,v=2} = \begin{vmatrix} 0 \\ 1 \end{vmatrix}. \qquad (2.22)$$

Sudarytose Q_V matricose eilutės numeris nurodo prioriteto lygį *q* ($q \in 1,2, .. z$), o stulpelis – srauto numerį VLAN tinkle *k* ($k \in 1,2, .. h$).

Virtualių tinklų įėjimo spartų ir prioritetų matricos, naudojamos išėjimo matricų skaičiavimui. Skaičiavimas atliekamas pateikiant duomenis į pralaidumo padalinimo funkciją. Į šią funkciją pateikiame srautus kiekviename VLAN ir kiekvienu laiko momentu atskirai. Pradžioje pateikiami aukščiausio prioriteto srautai, vėliau žemesnio ir t.t. Funkcijos išvestyje, gauname

srautų, kurie patenka į išėjimo kanalą, spartos reikšmes bei likusį perdavimo kanalo pralaidumą. Pralaidumo padalinimo funkcija:

$$
F(R_{FIN\,(v)}, C^{(t)}_{VL\,(v)}) = \left|\begin{array}{l}
N_{nz} \leftarrow nnz(R_{FIN\,(v)});\\
R_{CS} \leftarrow 1;\\
R_L \leftarrow C^{(k)}_{L\,(t)};\\
kol \quad R_{CS} \neq 0\\
\quad \left|\begin{array}{l}
R_{CS} \leftarrow 0;\\
N_C \leftarrow 0;\\
k \leftarrow 1;\\
kol \quad k \le h_v\\
\quad \left|\begin{array}{l}
jei \quad R^k_{FIN\,(v)} \neq 0, \quad R_L \neq 0, \quad R^k_{FIN\,(v)}\\
\quad \left|\begin{array}{l}
R^k_{FOUT} \leftarrow R^k_{FIN\,(v)}\\
R^k_{FIN\,(v)} \leftarrow 0;\\
R_{CS} \leftarrow R_{CS} + R^k_{FOUT}\\
N_C \leftarrow N_C + 1;
\end{array}\right.\\
k \leftarrow k + 1;
\end{array}\right.\\
R_L \leftarrow R_L - R_{CS}; \quad N_{nz} \leftarrow N_{nz} - N_C;
\end{array}\right.\\
k \leftarrow 1;\\
jei \quad N_{nz} \neq 0\\
\quad \left|\begin{array}{l}
kol \quad k \le h_v\\
\quad \left|R^k_{FOUT} \leftarrow R_L / N_{nz}; \quad jei \quad R^k_{FIN\,(v)} \neq 0;\right.\\
R_L \leftarrow 0;
\end{array}\right.\\
\\
rez: R^{(k)}_{FOUT}; \quad R_L.
\end{array}\right. , \qquad (2.23)
$$

čia funkcija *nnz(A)* – pateikia masyve esančių narių, kurie nėra lygūs nuliui, skaičių.

Pralaidumo padalinimo funkcija veikia suteikiant aptarnavimo pirmenybę mažiausiems srautams. Funkcijoje tikrinama, ar srauto spartos reikšmė yra mažesnė, nei likęs kanalo pralaidumas padalintas iš aktyvių srautų skaičiaus. Jei reikšmė tenkina šią sąlygą, srautas priskiriamas išėjimo masyvui R_{FOUT}, prilyginamas nuliui ir sumažinamas likusių aktyvių srautų skaičius. Ciklas kartojamas, kol nėra srautų, kurie tenkina minėtą sąlygą. Tuomet tikrinama, ar yra likusių aktyvių srautų. Jeigu yra, tuomet likęs kanalo pralaidumas padalinamas likusiems srautams po lygiai.

2.23 formulėje matome, jog pralaidumo padalinimo funkcija reikalauja tam tikro masyvo R_{FIN}. Šis masyvas gaunamas sudauginant *v* VLAN prioritetų matricos stulpelį $Q_V{}^v{}_q$ su įėjimo matricos eilute $R_{VIN\ v,\ t}$. Masyvo R_{FIN} reikšmės atitinka *v* virtualaus tinklo, *t* laiko momentu, *q*

prioriteto lygmens srautų spartos reikšmes, o srautai, kurie nėra *q* prioriteto lygmens, šiame masyve pateikiami kaip neaktyvūs (0 Mbit/s spartos). Pralaidumo padalinimo funkcija įvestyje, taip pat, reikalauja likusio kanalo pralaidumo C_{VL}. Pradžioje, šis pralaidumas atitinka pradinį kanalo pralaidumą, kuris yra perskaičiuojamas po kiekvieno aptarnauto srauto. Srautų patekimo į pralaidumo funkciją eiliškumo nustatymui (R_{FIN} masyvo formavimui), bei virtualių tinklų išėjimo matricų (R_{VOUT} ir R_{VOUTS}) sudarymui naudojama funkcija, kuri pateikta 2.24 formulėje:

$$\left[R^{v}_{VOUT\,t}, R^{v}_{VOUTS\,t}\right] = \left|\begin{array}{l} v \leftarrow 1; \\ kol \quad v \le v_K \\ \quad \left|\begin{array}{l} C^{v}_{VL} \leftarrow C^{v}_{VMAX} \cdot zeros(1,l); \\ R^{v}_{VOUT} \leftarrow zeros(l, h_v); \\ q \leftarrow 1; \\ kol \quad q \le z \\ \quad \left|\begin{array}{l} kol \quad t \le l \\ \quad \left|\begin{array}{l} R_{FIN} \leftarrow Q^{q}_{V\,v} \circ R_{VIN\,v,t}{}^{T}; \\ R_{FOUT}, C^{v}_{VL\,t} \leftarrow F[R_{FIN}, C^{v}_{VL\,t}]; \\ R^{v}_{VOUT\,t} \leftarrow R^{v}_{VOUT\,t} + R_{FOUT}{}^{T}; \\ R^{v}_{VOUTS\,t} \leftarrow sum(R^{v}_{VOUT\,t}); \\ t \leftarrow t+1; \end{array}\right. \\ q \leftarrow q+1; \end{array}\right. \\ v \leftarrow v+1; \\ rez.: \quad R^{v}_{VOUT\,t}, R^{v}_{VOUTS\,t}. \end{array}\right. , \qquad (2.24)$$

čia R^{v}_{VOUT} – virtualaus tinklo išėjimo matrica, kurioje pateiktos VLAN srautų spartos reikšmės, R^{v}_{VOUTS} – bendra sparta kanale, C_{VL} – likusio kanalo pralaidumo masyvas, funkcija *sum(A)* – pateikia masyvo narių sumą. Mano imitaciniame modelyje, kiekvienam naujam VLAN priskiriamas naujas 1 Gbit/s pralaidumo perdavimo kanalas ($C_{VMAX\,(v)}$=1 000Mbit/s).

Gautas pavyzdines $Q_{V\,(v)}$ ir $R_{VIN\,(v)}$ reikšmes (2.19 – 2.22 formulės), pateikiame į išėjimo matricų skaičiavimo bloką (2.24 formulė). Gautos skaičiavimo reikšmės:

$$R_{VOUT\,v=1,t=1,2..5} = \begin{vmatrix} 500 & 900 & 800 & 400 & 0 \\ 0 & 100 & 200 & 600 & 1000 \end{vmatrix}; \qquad (2.25)$$

$$R_{VOUTS\,v=1,t=1,2..5} = \begin{vmatrix} 500 & 1000 & 1000 & 1000 & 1000 \end{vmatrix}; \qquad (2.26)$$

$$R_{VOUT\,v=2,t=1,2..5} = \begin{vmatrix} 0 & 200 & 400 & 600 & 800 \end{vmatrix}; \qquad (2.27)$$

$$R_{VOUTS\,v=1,t=1,2..5} = \begin{vmatrix} 0 & 200 & 400 & 600 & 800 \end{vmatrix}. \qquad (2.28)$$

Turint išėjimo bei įėjimo matricas galima apskaičiuoti, kokia srautų dalis buvo aptarnauta U^k_v, bei bendrą perdavimo kanalo išnaudojimą $U_{K\,v}$:

$$\left[U^i, U^v_K\right] = \begin{array}{l} v \leftarrow 1; \\ kol \quad v \le v_K \\ \quad \begin{array}{l} kol \quad k \le h_v \\ \quad \begin{array}{l} U_v^{V_P^k} \leftarrow sum(R^v_{VOUT\,k}) / sum(R^v_{VIN\,k}); \\ k \leftarrow k+1; \end{array} \\ U_{K\,v} \leftarrow sum(R^v_{VOUTS}) / (C^v_{VMAX} \cdot l); \\ v \leftarrow v+1; \end{array} \\ rez.: \quad U^i, U^v_K; \end{array} . \tag{2.29}$$

Aptarnauta srauto dalis ir kanalo išnaudojimas, kuomet į imitacinį modelį pateikiame apskaičiuotus išėjimo matricų rezultatus, kurie nurodyti 2.25 – 2.28 formulėse:

$$U^{i=1,2,3} = \left[0{,}55 \quad 1 \quad 1\right]; \tag{2.30}$$

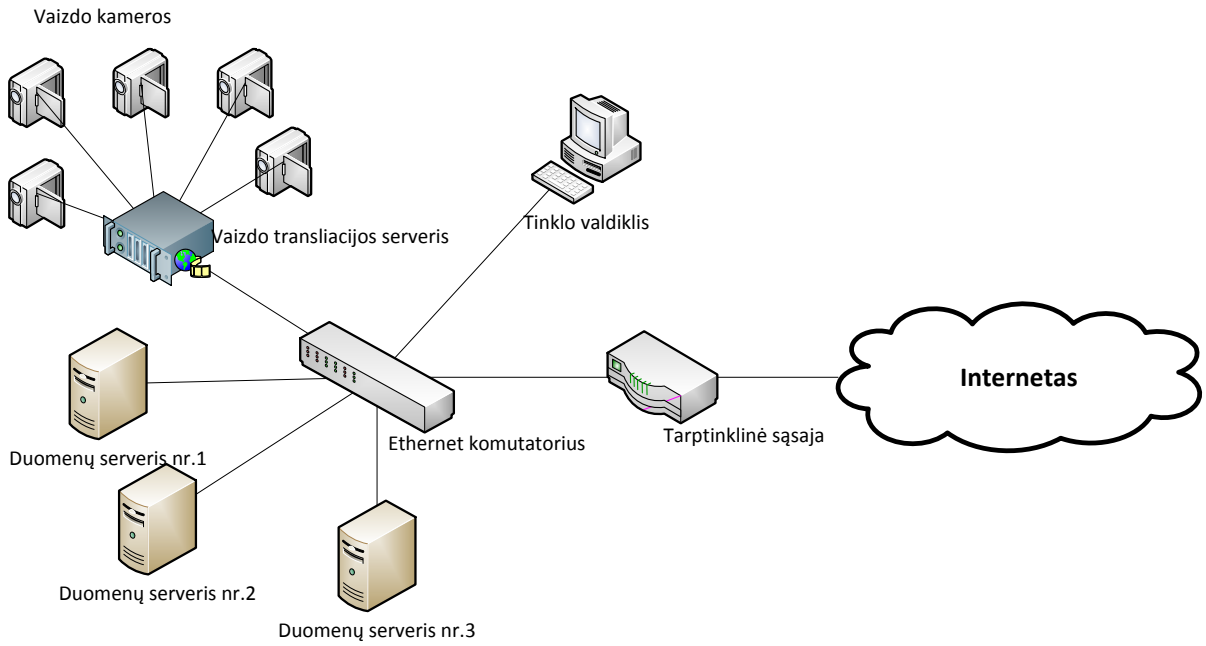
$$U^v_K = \left[0{,}9 \quad 0{,}4\right]. \tag{2.31}$$

Modelio kodas MATLAB aplinkoje pateiktas 1 priede. Sekančiame skyriuje atliksime srautų modeliavimą, taikydami aptartą imitacinį modelį.

2.3 Srautų modeliavimas

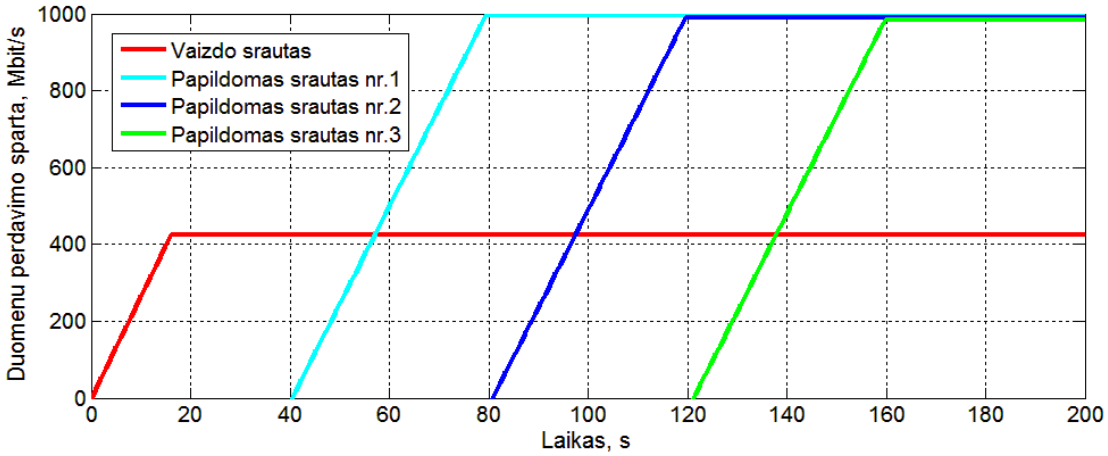
Sukurtas imitacinis modelis leidžia įvertinti, kokią įtaką duomenų srautams daro skirtingi komutatorių nustatymai, esant keletui duomenų srautų. Skirtingas komutatoriaus nustatymų konfigūracijas galima modeliuoti nenaudojant daugelio kompiuterių srautų generavimui. Valdymo įtakos nustatymui buvo sugalvotas scenarijus, kuriame reikalaujama užtikrinti reikiamą pralaidumą labai aukštos kokybės vaizdo duomenų srautui, kuomet kanale egzistuoja pašaliniai duomenų srautai. Modeliuojama tinklo topologija pateikta 2.16 pav. Tinklo struktūrą sudaro:

- Vaizdo kameros – aukštos kokybės skaitmeninio vaizdo signalus generuojantys įrenginiai.
- Medijos paslaugų serveris – įrenginys, apjungiantis vaizdo informaciją į vieną duomenų srautą.
- Duomenų serveriai – pašalinį signalą generuojantys įrenginiai.
- Ethernet komutatorius – mano nagrinėjamas, imitacinis modelis.
- Valdiklis – kompiuteris, kurio pagalba siunčiami komutatoriaus konfigūracijos nustatymai.
- Tarptinklinė sąsaja – įrenginys, jungiantis vietinį tinklą su internetu.



2.16 pav. Modeliuojamo tinklo topologija

Transliuojant 4K (8,3 milijono taškų viename kadre) raiškos, 60 kadrų per sekundę skaitmeninį vaizdą į Youtube portalą, Google teigimu, vaizdo transliacijos sparta gali pasiekti iki 85 Mbit/s [16]. Sakykime, kad į komutatorių per vieną sąsają atvyksta penki maksimalios spartos srautai. Šie srautai mūsų vaizduojamame scenarijuje bus apibūdinami, kaip vienas 425 Mbit/s spartos srautas. Be minėto vaizdo transliacijos srauto, į komutatorių taip pat patenka trys papildomi duomenų srautai, kurių sparta auga ir bando maksimaliai išnaudoti 1 Gbit/s perdavimo kanalą. Norint užtikrinti vaizdo duomenų kokybę, vaizdo srautams turime užtikrinti 425 Mbit/s pralaidumą komutatoriaus išėjime. Duomenų srautų, kurie papuola į skirtingus komutatoriaus prievadus įėjime, spartos kitimas laike yra pateiktas 2.17 paveiksle.



2.17 pav. Duomenų srautų perdavimo sparta komutatoriaus įėjimo prievaduose

Pirmasis srautas, kuris atkeliauja į komutatorių aptarnavimui yra vaizdo duomenų srautas. Jo sparta auga nuo pačios pirmosios sekundės iki 170 sekundės, kuomet lieka pastovios 425 Mbit/s

spartos. Nuo 40-osios sekundės pradedamas generuoti pirmasis pašalinis srautas, kuris iki 60-osios sekundės auga iki maksimalios galimos, tai yra 1 Gbit/s spartos. Atitinkamai antrasis ir trečiasis pašaliniai srautai pradedami generuoti 80-ąją ir 120-ąją sekundę ir, taip pat, pasiekia 1 Gbit/s spartą. Visus šiuos srautus turime perduoti į sekantį tinklo mazgą, efektyviai išnaudodami turimus tinklo resursus ir užtikrindami pralaidumą vaizdo duomenų srautui. Panagrinėsime atskirus komutatoriaus konfigūracijos variantus.

Modeliavimas, kuomet naudojami standartiniai komutatoriaus nustatymai

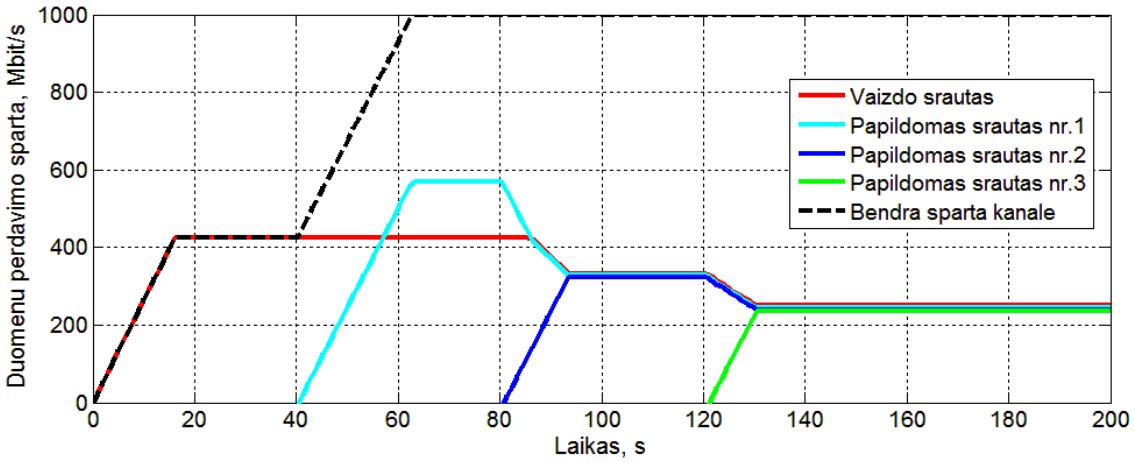
Esant standartiniams komutatoriaus nustatymams, visi komutatoriaus prievadai priklauso vienam standartiniam virtualiam vietiniam tinklui, kurio identifikacijos numeris yra „1“. Taip pat, visi srautai yra vienodo prioriteto lygmens bei komutatoriaus prievadai veikia 1 Gbit/s sparta. Duomenys iš komutatoriaus siunčiami vienu perdavimo kanalu. Komutatoriaus prievadų nustatymai pateikti 2.1 lentelėje .

2.1 lentelė. Komutatoriaus nustatymai

i	**Srauto pobūdis**	C^i_{max}	**Tipas**	V^i_{ID}	Q_i
1	Vaizdo srautas	1000	Full duplex	1	1
2	Papildomas duomenų srautas nr.1	1000	Full duplex	1	1
3	Papildomas duomenų srautas nr.2	1000	Full duplex	1	1
4	Papildomas duomenų srautas nr.3	1000	Full duplex	1	1
5	Išėjimo kanalas	1000	Full duplex	1	-

2.1 lentelėje: i – prievado numeris, C^i_{max} – kanalo pralaidumas, V^i_{ID} – VLAN, kuriam priklauso prievadas, Q_i – prievadui suteiktas prioriteto lygmuo.

Aukščiau aprašytus įeinančius duomenų srautus apdorojame imitaciniame modelyje. 2.18 paveiksle pavaizduotos srautų spartos, kurios gaunamos komutatoriaus išėjime.



2.18 pav. Išeinančių duomenų srautų spartų pasiskirstymas kanale, kuomet naudojami standartiniai komutatoriaus nustatymai

2.2 lentelėje pateiktos modeliuojamų srautų spartos įėjime ir išėjime, bei bendras kanalo išnaudojimas esminiais laiko momentais, kuomet naudojami standartiniai komutatoriaus nustatymai.

2.2 lentelė. Srautų perdavimo sparta bei aptarnauta srauto dalis

Laiko momentas, s	Vaizdo srauto sparta, Mbit/s		Papildomo srauto nr.1 sparta, Mbit/s		Papildomo srauto nr.2 sparta, Mbit/s		Papildomo srauto nr.3 sparta, Mbit/s		Perdavimo kanalo išnaudojimas, %
	Įėjime	Išėjime	Įėjime	Išėjime	Įėjime	Išėjime	Įėjime	Išėjime	
40	425	425	0	0	0	0	0	0	42,5
80	425	425	1000	575	0	0	0	0	100
120	425	333	1000	333	1000	333	0	0	100
200	425	250	1000	250	1000	250	1000	250	100
Bendra aptarnauta srauto dalis, %	79,96		36,76		31,45		31,09		-
							Bendras kanalo išnaudojimas, %		83,32

Norėdamas išanalizuoti eksperimento rezultatus, diagramą suskirstysiu į keletą dalių:

- Iki 40-osios sekundės matome, kad kol nėra papildomų duomenų srautų, vaizdo duomenis perduodantis srautas komutatoriaus išėjime yra toks pat, kaip ir buvo komutatoriaus įėjime, o bendra sparta perdavimo kanale lygi šio srauto spartai.
- Nuo 40-osios iki 63-osios sekundės vaizdą perduodantis srautas, taipogi nekinta, netgi atsiradus pirmajam pašaliniam srautui. Taip yra dėl to, kad kanalas nėra visiškai išnaudojamas.
- Pasiekus kanalo įsisotinimą (nuo 63-tosios sekundės), papildomo duomenų srauto sparta tampa lygi 575 Mbit/s ir nekinta, kol nėra generuojamas sekantis srautas. Vaizdo srautas nekinta ir kanalui įsisotinus, nes yra mažesnis nei pirmasis papildomas srautas, o kanalo pralaidumas dalinamas pagal tendencijas, kurios gautos realios įrangos tyrime, 2.3 paveiksle (mažesnis srautas aptarnaujamas pirmiau, kol srautai neišsilygina).
- Nuo 80-osios sekundės pradedamas generuoti antrasis papildomas srautas. Kadangi kanalo pralaidumas yra visiškai išnaudotas, antrojo papildomo srauto sparta auga, nes pradžioje jis yra mažesnis už kitus srautus. Atitinkamai pradžioje mažėja pirmojo papildomo srauto sparta, nes jis yra didžiausias. Tai vyksta kol ji išsilygina su vaizdo

srauto sparta. Tuomet mažėja vaizdo srauto ir pirmojo papildomo srauto sparta, kol visi srautai išsilygina ir vienodai pasidalina kanalo pralaidumu, tai yra po ~333,33 Mbit/s.

- 120-ąją sekundę pradedamas generuoti trečiasis papildomas srautas. Šio srauto sparta auga, o kitų srautų mažėja, kol visi srautai tampa lygūs, tai yra po 250 Mbit/s.

Matome, jog tokiame scenarijuje vaizdo srautui tenkantis pralaidumas, tampa nepakankamas kuomet kanale atsiranda bent du dideli srautai. Didėjant didelių srautų skaičiui kanale vaizdo srautui tenkantis pralaidumas mažėja.

Papildomų srautų apribojimas

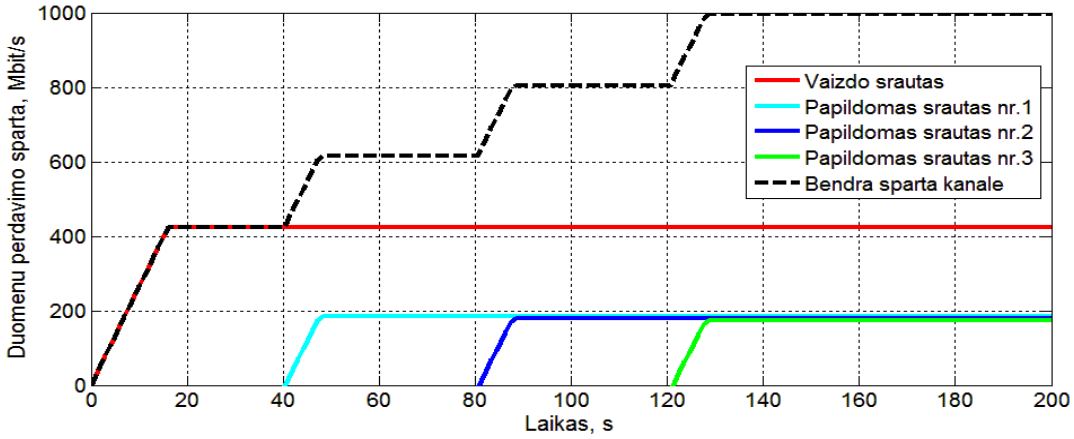
Norint suteikti reikalingą pralaidumą vaizdo duomenų srautui, galime apriboti pašaliniams srautams suteikiamą pralaidumą. Turimame 1 Gbit/s pralaidumo kanale, perduodant tą patį, 425 Mbit/s spartos vaizdo duomenų srautą, dar lieka 575 Mbit/s neišnaudoto pralaidumo. Šią reikšmę padaliname papildomiems srautams po 191,66 Mbit/s. Vaizdo srautui suteikiama sparta neribojama. 2.3 lentelėje pateikti komutatoriaus prievadų nustatymai, kurie naudojami sekančiame modeliavime.

2.3 lentelė. Komutatoriaus nustatymai

i	**Srauto pobūdis**	**$C^{(i)}_{max}$**	**Tipas**	**$V_{ID}^{(i)}$**	**$Q_{(i)}$**
1	Vaizdo srautas	1000	Full duplex	1	1
2	Papildomas duomenų srautas nr.1	191,66	Full duplex	1	1
3	Papildomas duomenų srautas nr.2	191,66	Full duplex	1	1
4	Papildomas duomenų srautas nr.3	191,66	Full duplex	1	1
5	Išėjimo kanalas	1000	Full duplex	1	-

Apdorojimui pateikiame tokius pačius įėjimo srautus, kaip ir esant standartiniams nustatymams (2.17 pav.) . Gauti modeliavimo rezultatai pateikti 2.19 pav, o 2.4 lentelėje pateiktos modeliuojamų srautų spartos įėjime ir išėjime bei bendras kanalo išnaudojimas esminiais laiko momentais, kuomet taikomas papildomų srautų spartų apribojimas.

Taikant šį valdymo metodą, visą modeliavimo laiką, vaizdą perduodantis srautas nėra veikiamas papildomų srautų, tačiau išėjimo kanalas yra visiškai išnaudojamas tik tuomet, kai visi srautai yra perduodami vienu metu. Perdavimo linija didžiąją dalį laiko turi atliekamo pralaidumo, kuris galėtų būti išnaudojamas perduodant informaciją iš duomenų serverių.



2.19 pav. Išeinančių duomenų srautų spartų pasiskirstymas kanale, kuomet apribojamas papildomiems srautams suteikiamas pralaidumas

2.4 lentelė. Srautų perdavimo sparta bei aptarnauta srauto dalis

Laiko momentas, s	Vaizdo srauto sparta, Mbit/s		Papildomo srauto nr.1 sparta, Mbit/s		Papildomo srauto nr.2 sparta, Mbit/s		Papildomo srauto nr.3 sparta, Mbit/s		Perdavimo kanalo išnaudojimas, %
	Įėjime	Išėjime	Įėjime	Išėjime	Įėjime	Išėjime	Įėjime	Išėjime	
40	425	425	0	0	0	0	0	0	42,5
80	425	425	1000	191,66	0	0	0	0	61,67
120	425	425	1000	191,66	1000	191,66	0	0	80,83
200	425	425	1000	191,66	1000	191,66	1000	191,66	100
Bendra aptarnauta srauto dalis, %	100		21,25		22		24		-
							Bendras kanalo išnaudojimas, %		73,77

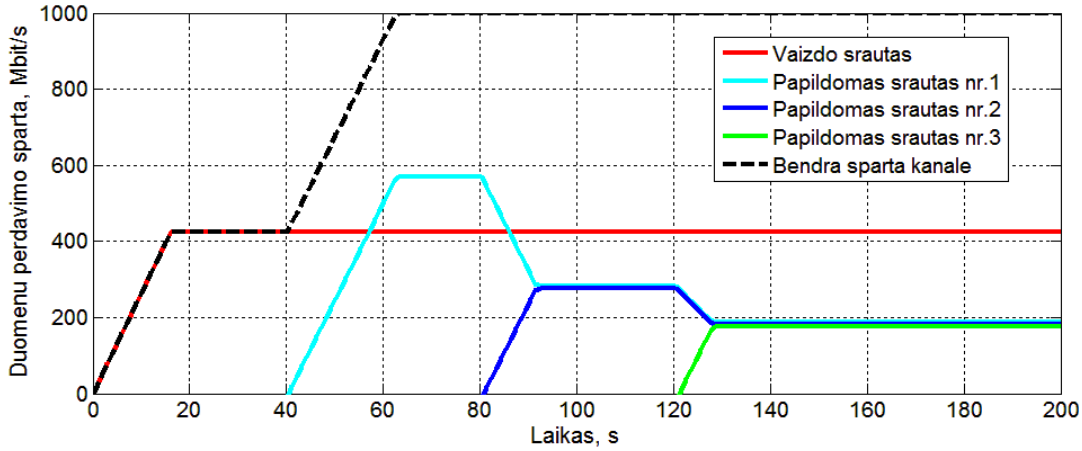
Prioritetų taikymas srautams

Kitas duomenų srautų valdymo variantas yra suteikti srautams skirtingus prioritetus. Tai atlikti komutatoriuose leidžia 802.1p signalizacijos standartas. Šiame eksperimente atliktas srautų modeliavimas, kuomet į komutatoriaus prievadus patenkantys paketai yra žymimi prioriteto antrašte. Vaizdo duomenis perduodantis srautas žymimas aukštesnio prioriteto antrašte nei papildomi duomenų srautai. Papildomų srautų paketams suteikiama vienodo lygmens prioriteto žymė. Komutatoriaus prievadų nustatymai, kurie naudojami eksperimente pateikti 2.5 lentelėje.

Modeliavimo rezultatai pateikti 2.20 pav, o 2.6 lentelėje pateiktos modeliuojamų srautų spartos įėjime ir išėjime, bei kanalo išnaudojimas esminiais laiko momentais.

2.5 lentelė. Komutatoriaus nustatymai

i	Srauto pobūdis	$C^{(i)}_{max}$	Tipas	$V_{ID}^{(i)}$	$Q_{(i)}$
1	Vaizdo srautas	1000	Full duplex	1	5
2	Papildomas duomenų srautas nr.1	1000	Full duplex	1	1
3	Papildomas duomenų srautas nr.2	1000	Full duplex	1	1
4	Papildomas duomenų srautas nr.3	1000	Full duplex	1	1
5	Išėjimo kanalas	1000	Full duplex	1	-



2.20 pav. Išeinančių duomenų srautų spartų pasiskirstymas kanale, kuomet srautams taikomi skirtingi prioritetai

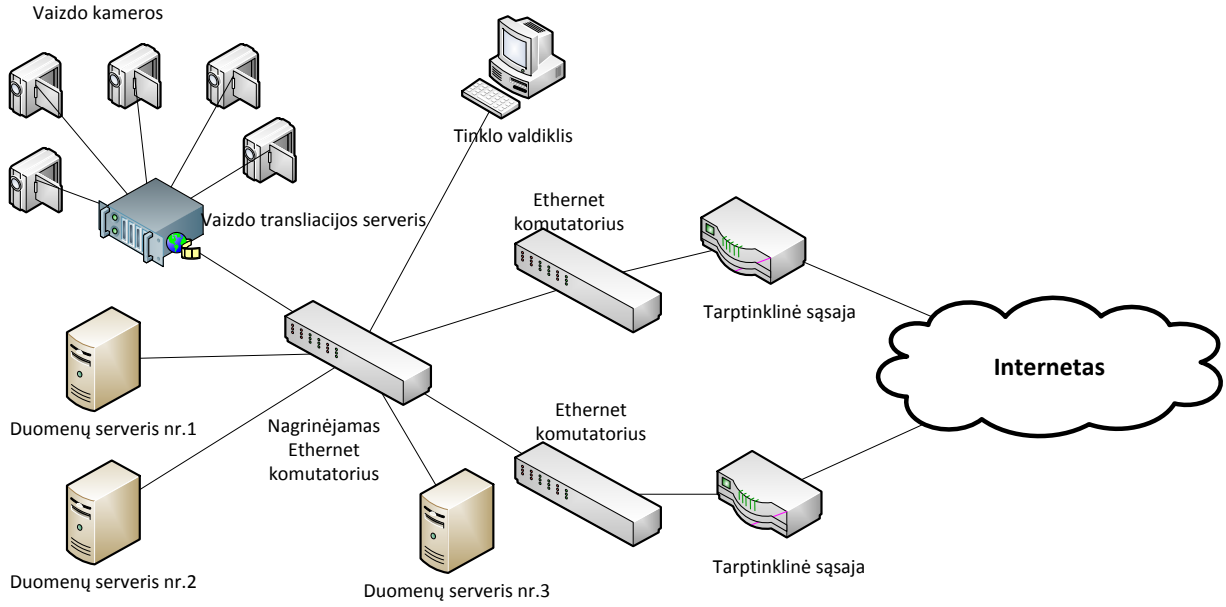
2.6 lentelė. Srautų perdavimo sparta bei aptarnauta srauto dalis

Laiko momentas, s	Vaizdo srauto sparta, Mbit/s		Papildomo srauto nr.1 sparta, Mbit/s		Papildomo srauto nr.2 sparta, Mbit/s		Papildomo srauto nr.3 sparta, Mbit/s		Perdavimo kanalo išnaudojimas, %
	Įėjime	Išėjime	Įėjime	Išėjime	Įėjime	Išėjime	Įėjime	Išėjime	
40	425	425	0	0	0	0	0	0	42,5
80	425	425	1000	575	0	0	0	0	100
120	425	425	1000	287,5	1000	287,5	0	0	100
200	425	425	1000	191,66	1000	191,66	1000	191,66	100
Bendra aptarnauta srauto dalis, %	100		32,33		25,48		24,2		-
							Bendras kanalo išnaudojimas, %		83,32

Pažvelgę į modeliavimo rezultatus galime pastebėti, jog viso modeliavimo metu vaizdo duomenų srautas nėra veikiamas pašalinių srautų. Jis yra aptarnaujamas pirmiausiai, o papildomi duomenų srautai pasidalina liekančia kanalo pralaidumo dalimi. Kaip ir papildomų srautų apribojimo atveju, papildomieji srautai modeliavimo pabaigoje aptarnaujami 191,66 Mbit/s sparta. Pagrindinis skirtumas yra tai, kad srautai gauna didesnę pralaidumo dalį tuo metu, kai tik dalis srautų yra generuojami, pavyzdžiui 80-ąją ir 120-ąją sekundę.

Fizinis ir loginis atskyrimas

Perduodant duomenų srautus, svarbu įvertinti ar yra galimybė paketus perduoti keletu perdavimo kanalų. Tinklo topologija, kurioje galimi du perdavimo maršrutai, pateikta 2.21 pav.

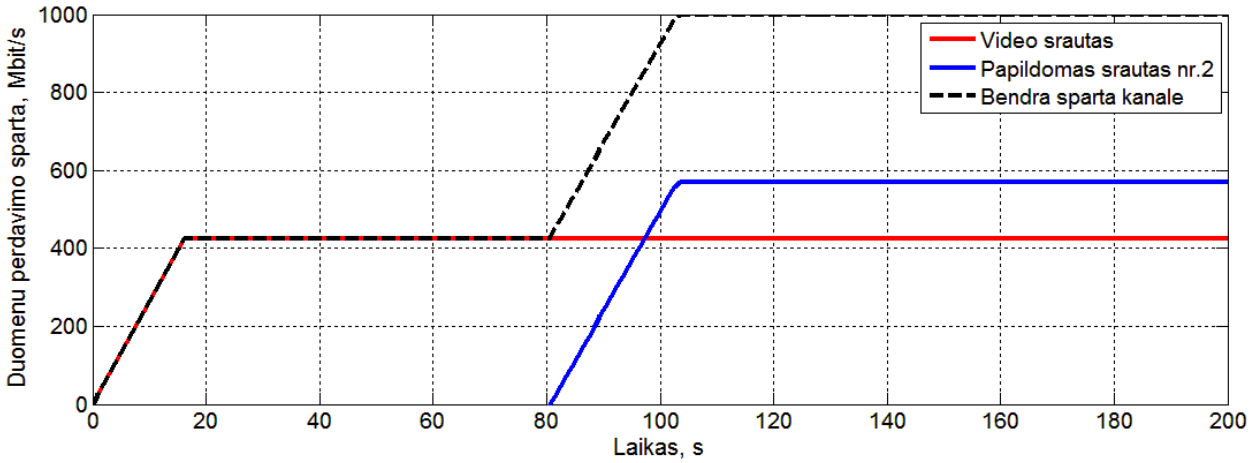


2.21 pav. Modeliuojamo tinklo topologija taikant duomenų srautų atskyrimą

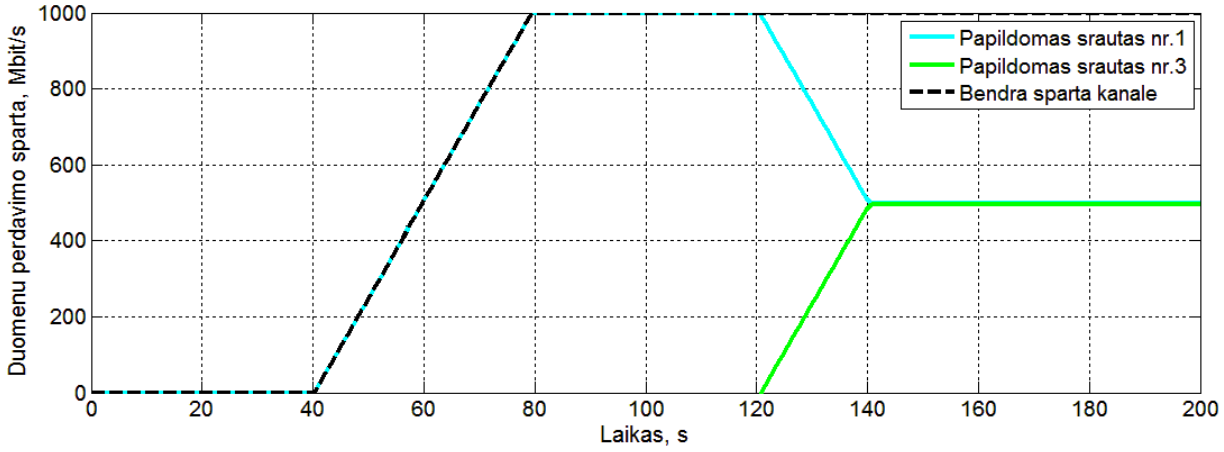
Tokioje tinklo struktūroje, esant standartiniams komutatoriaus nustatymams, būtų išnaudojama tik viena perdavimo linija, nes komutatoriuje pradėtų veikti STP protokolas. Programinio valdymo įvedimas leidžia aptikti keletą perdavimo linijų ir srautus logiškai izoliuoti bei perduoti skirtingomis perdavimo linijomis. Šiame modeliavimo eksperimente srautai sugrupuoti į atskirus virtualius tinklus. Vaizdo duomenis perduodantis srautas ir antrasis papildomas srautas bus perduodami per pirmąjį perdavimo kanalą, o pirmasis ir trečiasis papildomi srautai – per antrąjį kanalą. Prioritetinių antraščių ir prievado spartos ribojimas yra nenaudojamas. Komutatoriaus sąsajų nustatymai pateikti 2.7 lentelėje. Modeliavimo rezultatai pateikti 2.22 ir 2.23 paveiksluose.

2.7 lentelė. Komutatoriaus nustatymai

i	Srauto pobūdis	$C^{(i)}_{max}$	Tipas	$V_{ID}^{(i)}$	$Q_{(i)}$
1	Vaizdo srautas	1000	Full duplex	1	1
2	Papildomas duomenų srautas nr.1	1000	Full duplex	2	1
3	Papildomas duomenų srautas nr.2	1000	Full duplex	1	1
4	Papildomas duomenų srautas nr.3	1000	Full duplex	2	1
5	Išėjimo kanalas nr.1	1000	Full duplex	1	-
6	Išėjimo kanalas nr.2	1000	Full duplex	2	-



2.22 pav. Išeinančių duomenų srautų spartų pasiskirstymas pirmajame perdavimo kanale, kuomet taikomas srautų atskyrimas



2.32 pav. Išeinančių duomenų srautų spartų pasiskirstymas antrajame perdavimo kanale, kuomet taikomas srautų atskyrimas

Ateinančius duomenų srautus padalinus į atskirus perdavimo kanalus, srautams tenka daug didesnis pralaidumas. Kadangi vaizdo srautas dalinasi kanalo pralaidumu tik su vienu papildomu srautu, jis gauna visą reikalingą pralaidumą. Kitų srautų aptarnavimas, taip pat, yra pagerinamas. Srautų aptarnavimo bei kanalo išnaudojimo duomenys pateikti 2.8 lentelėje.

2.8 lentelė. Srautų perdavimo sparta bei aptarnauta srauto dalis

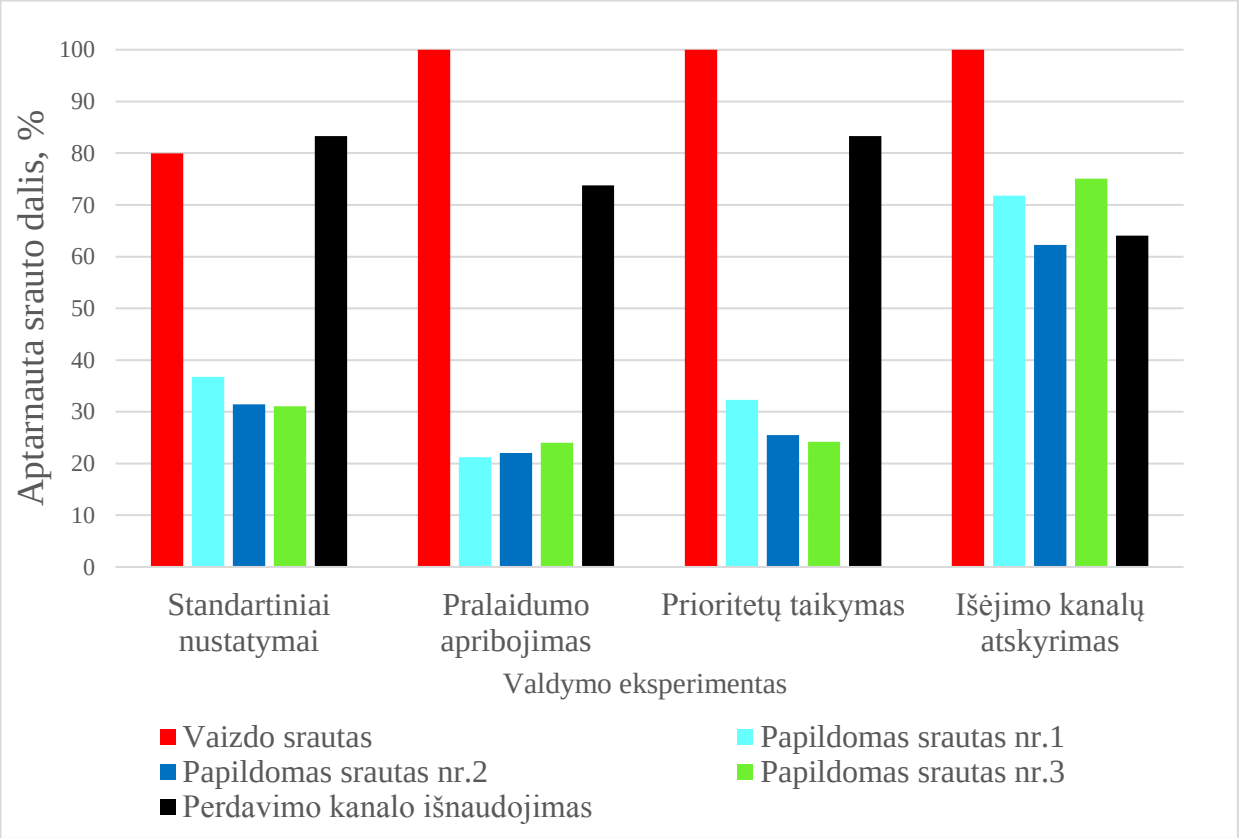
Laiko momentas, s	Vaizdo srauto sparta, Mbit/s		Papildomo srauto nr.1 sparta, Mbit/s		Papildomo srauto nr.2 sparta, Mbit/s		Papildomo srauto nr.3 sparta, Mbit/s		Perdavimo kanalo išnaudojimas, %
	Įėjime	Išėjime	Įėjime	Išėjime	Įėjime	Išėjime	Įėjime	Išėjime	
40	425	425	0	0	0	0	0	0	21,25
80	425	425	1000	1000	0	0	0	0	71,25
120	425	425	1000	1000	1000	575	0	0	100
200	425	425	1000	500	1000	575	1000	500	100
Bendra aptarnauta srauto dalis, %	100		71,82		62,26		75,09		-
							Bendras kanalo išnaudojimas, %		64,06

3 PROGRAMINIO VALDYMO ALGORITMO SUDARYMAS

Šiame skyriuje aptariami gauti eksperimentų rezultatai, pateikiamas siūlomas srautų valdymo algoritmas, bei galimas valdymo rezultatas, kuomet taikomas sukurtas algoritmas.

3.1 Eksperimentų rezultatų apibendrinimas

Norėdamas apibendrinti gautus modeliavimo rezultatus, sudariau diagramą, kurioje atsispindi skirtingų valdymo scenarijų rezultatai (3.1 pav.).

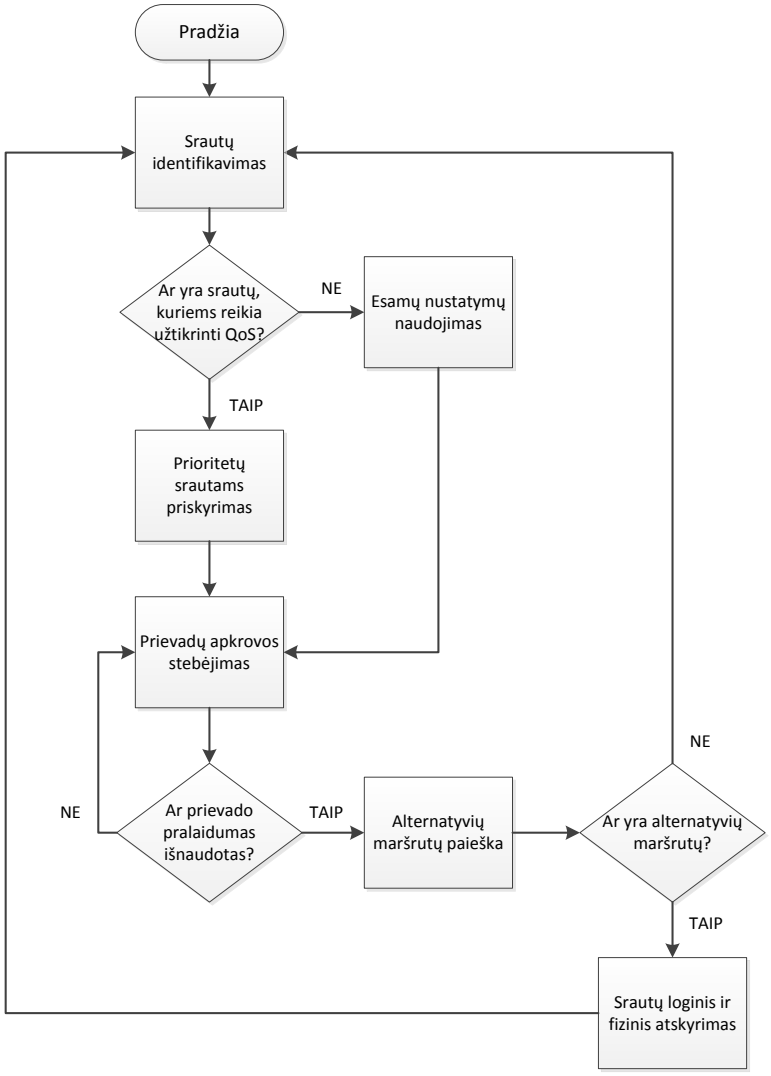


3.1 pav. Aptarnauta srautų dalis ir perdavimo kanalo išnaudojimas skirtinguose srautų valdymo eksperimentuose

Pažiūrėję į aptarnautos srautų dalies diagramą (3.1 pav.) galime matyti, kad standartinių nustatymų atveju, aptarnaujama tik apie 80% vaizdo duomenų srauto, o tai yra netenkinantis rezultatas. Atskiriant srautus į skirtingas perdavimo linijas, gauname geriausią srautų įėjimo ir išėjimo spartų santykį, tačiau dažnai alternatyvių maršrutų tinkle gali nebūti, todėl turėtų būti taikomas tikslingas prioritetų taikymas, kuris rodė geresnius rezultatus nei sąsajų pralaidumo apribojimas.

3.2 Srautų valdymo algoritmas

Vadovaujantis eksperimentuose gautais rezultatais suformuotas programinio valdymo algoritmas (3.2 pav.).



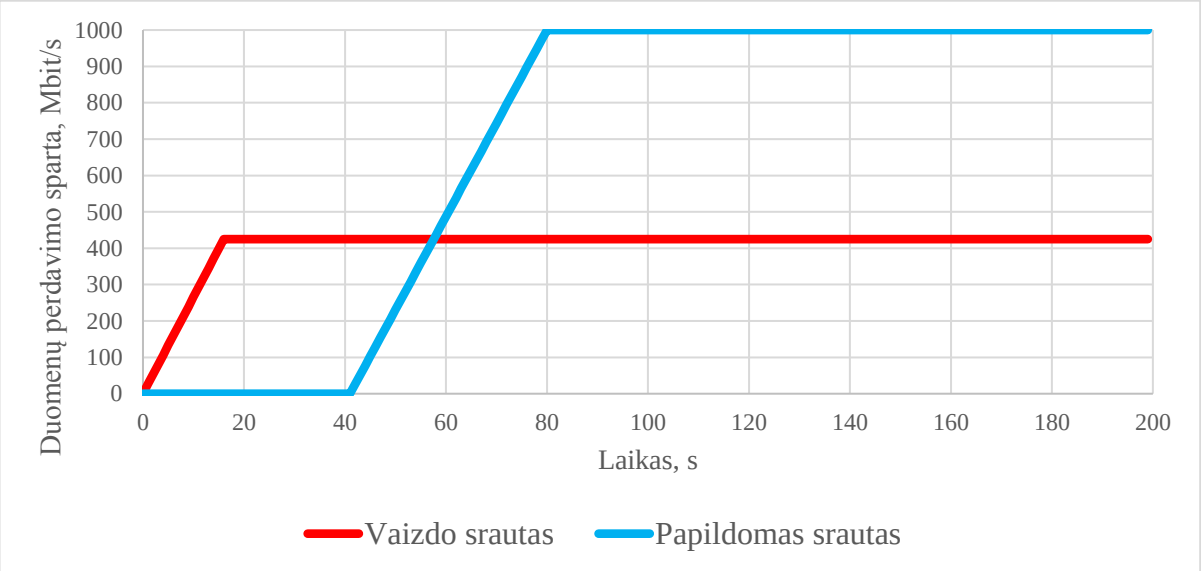
3.2 pav. Programinio tinklo valdymo algoritmas

Programinio valdymo algoritme pritaikomos prioritetų nustatymo ir srautų atskyrimo funkcijos, dėl tyrime gauto geriausio efektyvumo. Algoritmą sudaro tam tikri funkciniai blokai:

- Srautų identifikavimas – tai pradinių ir naujų srautų pobūdžio nustatymas. Už šią tinklo funkciją turėtų būti atsakingas administratorius arba valdiklyje įdiegtas aplikacijų atpažinimo mechanizmas (galimai panašus į aptartą ATLAS [14]). Srautų identifikavimas turėtų būti vykdomas tinklo pradžioje ir atsiradus naujiems sujungimams komutatoriuose.
- Prioritetų srautams priskyrimas – veiksmas, kuris vykdomas kuomet žinoma srautų prigimtis. Jis atliekamas nustatant santykinę srauto svarbą, kitų srautų atžvilgiu. Prioritetai priskiriami srautams, kurie reikalauja kokybinių parametrų užtikrinimo.

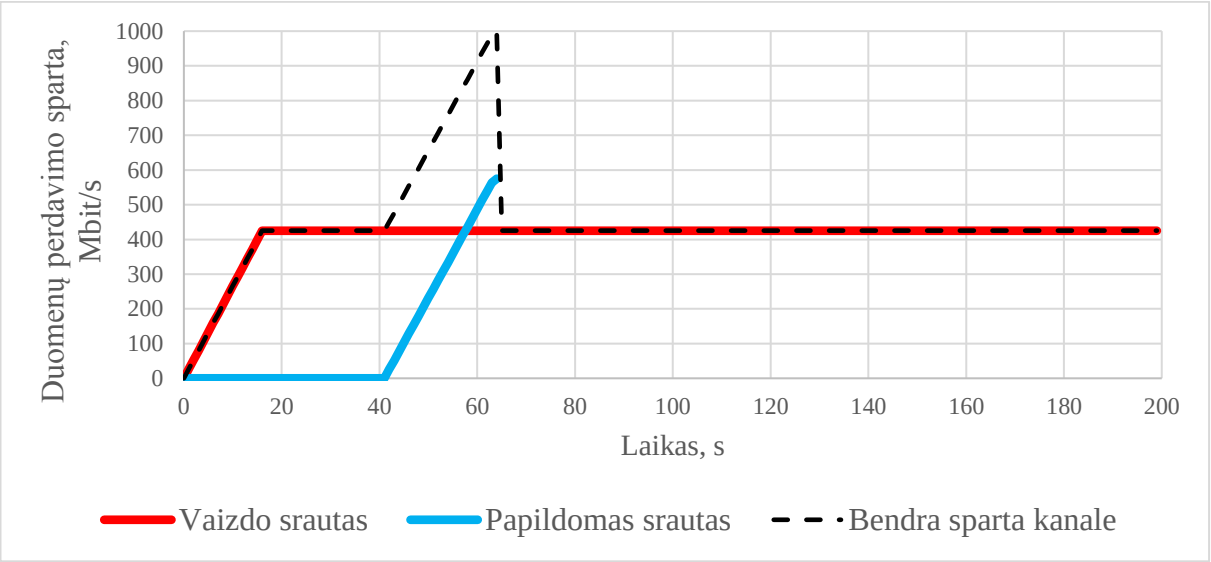
- Prievadų apkrovos stebėjimas – nuolatinis procesas, kuriame matuojama komutatoriaus prievado momentinė duomenų perdavimo spartos reikšmė. Stebėjimo tikslas yra nustatyti, kuomet prievado apkrova pasiekia jo maksimalaus pralaidumo reikšmę. Įvykdžius šią sąlygą, įvyksta pertrauktis, kurios metu ieškoma alternatyvių duomenų perdavimo maršrutų. Stebėjimo funkcija atliekama automatiškai tinklo valdiklyje. Gali būti naudojama panaši sistema į anksčiau aptartą MonSamp [13].
- Alternatyvių maršrutų paieška – tai programiškai įgyvendinta funkcija, kurios tikslas surasti kitą komutatoriaus prievadą, kuriuo galima perduoti informaciją į tą patį galutinį tinklo mazgą. Tai maršrutizavimo įgyvendinimas tinklo komutatoriuose. Topologijos nustatymui gali būti naudojama panaši sistema į aptartą OFDPv2 [15].
- Srautų loginis ir fizinis atskyrimas – tai sekantis žingsnis po alternatyvaus maršruto radimo. Jo metu komutatoriaus prievadai suskirstomi į virtualius tinklus ir nukreipiami į skirtingas sąsajas, išvengiant STP protokolo veikimo.

Norint iliustruoti programinio valdymo algoritmo veikimą nagrinėsime 2.21 pav. pateiktos tinklo topologijos atvejį, kuomet į komutatorių ateina du srautai. Pirmasis – anksčiau nagrinėtas, 425 Mbit/s vaizdo transliacijos srautas, o antrasis – papildomas duomenų srautas, kurio sparta auga iki 1000 Mbit/s. Duomenų srautų spartos komutatoriaus įėjime pavaizduotas 3.3 pav.

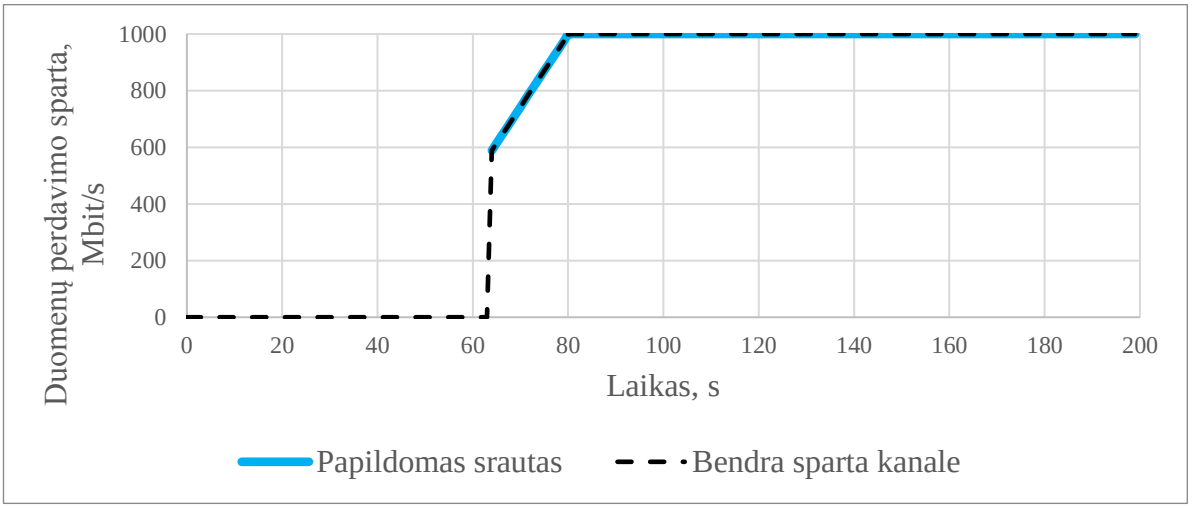


3.3 pav. Duomenų srautų perdavimo spartos komutatoriaus įėjimo prievaduose

Pagal sukurtą programinio valdymo algoritmą, kuomet komutatoriuje pasiekiama maksimali perdavimo sąsajos sparta (1 Gbit/s), valdiklio pagalba aptinkamas alternatyvus maršrutas ir papildomas srautas yra nukreipiamas į kitą perdavimo kanalą. Duomenų srautų spartos komutatoriaus išėjimuose pateiktos 3.4 ir 3.5 paveiksluose.



3.4 pav. Duomenų srautų spartos kitimas pirmajame perdavimo kanale



3.5 pav. Duomenų srautų spartos kitimas antrajame perdavimo kanale

Pirmajame perdavimo kanale (3.4 pav.) vaizdo duomenų srautas gauna visą pareikalautą pralaidumą. Atsiradus papildomajam srautui, jo sparta auga, kol pasiekiama maksimali sparta kanale ir įvykdoma maršruto paieška bei srautų atskyrimas. Papildomas srautas, 64-iąją sekundę, yra nukreipiamas į kitą perdavimo kanalą (3.5 pav.), kur jam, taip pat, suteikiamas visas pareikalautas pralaidumas, todėl abiejų srautų aptarnavimas yra lygus 100%. Valdymo modelyje alternatyvaus paieškos ir sudarymo trukmė nėra įvertinama, nes ši funkcija nebuvo išbandyta naudojant realią įrangą.

IŠVADOS

1. Analizuojant kitų autorių tyrimus pastebėta, jog nagrinėjama SDN įrenginių galimybė susidoroti su valdymo informacijos srautais, efektyvus valdiklio skaičiavimo resursų panaudojimas aplikacijose, tačiau nėra aptariama, kaip efektyviai išnaudoti duomenų lygmenyje turimus tinklo pralaidumo resursus.

2. Atlikti eksperimentai, kuriuose naudota reali tinklo įranga, leido nustatyti tendencijas, kuriomis vadovaujasi komutatorius, kuomet vykdomas prievado pralaidumo dalinimas duomenų srautams, esant skirtingiems komutatoriaus nustatymams. Pagrindinė tendencija – vykdant, pralaidumo padalinimą komutatoriuje, kuomet perduodami vienodo prioriteto lygio srautai, mažesnės spartos srautas gauna jam reikalingą pralaidumą, kol srautų spartos tampa vienodo didumo.

3. Remiantis pralaidumo padalinimo tendencijomis buvo sukurtas komutatoriaus imitacinis modelis, kuris leidžia daugelio srautų modeliavimą, bei pateikia kiekvieno srauto aptarnavimo ir perdavimo kanalų išnaudojimo statistiką.

4. Naudojant sukurtą imitacinį modelį, buvo atliktas tinklo srautų scenarijaus modeliavimas, kuomet pritaikomi skirtingi komutatoriaus valdymo variantai. Modeliavimas leido nustatyti, kad efektyviausias valdymo būdas yra srautų atskyrimas į skirtingus perdavimo kanalus, o kuomet tai yra neįmanoma, turėtų būti naudojamas prioritetų priskyrimas. Abu valdymo būdai padėjo pasiekti 100% modeliuoto vaizdo srauto aptarnavimą, taip pat, atitinkamai 71,82% ir 32,33% pirmojo papildomo srauto aptarnavimą. Naudojant standartinius nustatymus, vaizdo srautas buvo aptarnaujamas tik 80%, o pirmasis papildomas srautas 36,76%.

5. Remiantis modeliavimo rezultatais, buvo sudarytas programinio duomenų srautų valdymo algoritmas. Algoritme nurodyta, kad svarbu stebėti perduodamų srautų pobūdį, komutatorių prievaduose esančią apkrovą, tinklo topologiją ir pagal tai, atitinkamai nustatyti srautams taikomus prioritetus bei nurodyti perdavimo linijas.

INFORMACIJOS ŠALTINIŲ SĄRAŠAS

1. DAVIDSON, J. Here's How Many Internet Users There Are. Time.com, 2015 [žiūrėta 2017-05-25]. Prieiga per internetą: <http://time.com/money/3896219/internet-users-worldwide/>
2. NORDRUM, A. Popular Internet Of Things Forecast Of 50 Billion Devices By 2020 Is Outdated. IEEE Spectrum: Technology, Engineering, and Science News, 2016. [žiūrėta 2017-05-25]. Prieiga per internetą: <http://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated>
3. DUFFY, J. Google Opens Up On Its SDN. Network World, 2017 [žiūrėta 2017-05-25]. Prieiga per internetą: <http://www.networkworld.com/article/2937656/cisco-subnet/google-opens-up-on-its-sdn.html>
4. What’s Software-Defined Networking (SDN)?, SDxCentral [žiūrėta 2017-05-25]. Prieiga per internetą: <https://www.sdxcentral.com/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/>
5. What Is Openflow? Definition And How It Relates To SDN [žiūrėta 2017-05-25]. Priega per internetą: <https://www.sdxcentral.com/sdn/definitions/what-is-openflow/>
6. FERRO, G. Openflow And Software Defined Networking: Is It Routing Or Switching? [žiūrėta 2017-05-25]. Priega per internetą: <http://etherealmind.com/openflow-software-defined-networking-routing-or-switching/ >
7. MCNICKLE, M. Five SDN Protocols Other Than Openflow, SearchSDN, [žiūrėta 2017-05-25]. Priega per internetą: <http://searchsdn.techtarget.com/news/2240227714/Five-SDN-protocols-other-than-OpenFlow>
8. JACOBS, D. Can The NETCONF Protocol Give Openflow A Run For Its Money?, SearchSDN, 2015 [žiūrėta 2017-05-25]. Priega per internetą: <http://searchsdn.techtarget.com/tip/Can-the-NETCONF-protocol-give-OpenFlow-a-run-for-its-money>
9. JACOBS, D. With MPLS-TP And SDN, A Simpler Dynamic Control Plane, SearchSDN, 2014. [žiūrėta 2017-05-25]. Priega per internetą:

<http://searchsdn.techtarget.com/tip/With-MPLS-TP-and-SDN-a-simpler-dynamic-control-plane>

10. CURTIS, A.R. MOGUL, J.C. TOURRILHES J. DevoFlow: Scaling Flow Management for High-Performance Networks. SIGCOMM'11, 2011, ACM 978-1-4503-0797-0/11/08
11. JARSCHEL, M. An Assessment of Applications and Performance Analysis of Software Defined Networking. Julius-Maximilian University of Würzburg, 2014, p.19-43
12. TOOTOONCHIAN A, GORBUNOV S, GANJALI Y, CASADO M, SHERWOOD R. On Controller Performance in Software Defined Networks, 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, San Jose, USENIX, 2012
13. RAUMER, D. SCHWAIGHOFER, L. CARLE, G. MonSamp: A Distributed SDN Application for QoS Monitoring. Conference on Computer Science and Information Systems. Varšuva: IEEE, 2014, vol. 2, pp. 961-968. DOI: 10.15439/2014F175
14. QAZI, Z. LEE, J. JIN, T. Application-Awareness in SDN. Hong Kong: SIGCOMM'13, 2013. ACM 978-1-4503-2056-6/13/08.
15. PAKZAD, F. PORTMANN, M. TAN, W. INDULSKA, J. Efficient Topology Discovery in Software Defined Networks. Conference on Signal Processing and Communication Systems (ICSPCS). Gold Coast: IEEE, 2014, DOI: 10.1109/ICSPCS.2014.7021050
16. Recomended Upload Encoding Settings – Youtube Help, Google Support [žiūrėta 2017-05-25]. Priega per internetą: <https://support.google.com/youtube/answer/1722171?hl=en>

PRIEDAI

1 priedas. Imitacinio modelio MATLAB kodas

main.m

```
clear all; clc; close all;
srautas = port;
kanalas(10) = port;
VLAN(10) = vlanas;
PR_sk=8;
[N, srautas]=video_default();

colors=['r'; 'c'; 'b'; 'g'];
markers=['.','s','o','x'];
figure(50);

for i=1:length(srautas)
   plot(linspace(0,N,N),srautas(i).Rin-(i-1)*5,'LineWidth',3,'Color',colors(i));
   hold on;
   grid on;
   axis([0 N 0 1000]);
   xlabel('Laikas, s', 'FontSize', 15);
   ylabel('Duomenu perdavimo sparta, Mbit/s', 'FontSize', 15);
   set(gca,'fontsize',14)
end
hold off
legend('Vaizdo srautas','Papildomas srautas nr.1','Papildomas srautas nr.2','Papildomas srautas nr.3')

for i=1:length(srautas)
   srautas(i).Util=sum(srautas(i).Rin);
   VLAN(srautas(i).VID).ports(end+1)=i;
   for j=1:N
   if srautas(i).Rin(j) > srautas(i).Cmax
     srautas(i).Rin(j)=srautas(i).Cmax;
   end
   end
end

for V = 1:length(VLAN)
   if length(VLAN(V).ports) ~= 0
      VLAN(V).PrMat=zeros(PR_sk,length(VLAN(V).ports));
      VLAN(V).RinMat=zeros(N,length(VLAN(V).ports))';
      VLAN(V).RoutMat=zeros(N,length(VLAN(V).ports))';
      kanalas(V).Cmax=1000;
      kanalas(V).Rl=kanalas(V).Cmax*ones(1,N);
      for i = 1:length(VLAN(V).ports)
         VLAN(V).PrMat(srautas(VLAN(V).ports(i)).Prio,i)=1;
         for j=1:N
            VLAN(V).RinMat(i,j)=srautas(VLAN(V).ports(i)).Rin(j);
         end
      end
   end
end

for V = 1:length(VLAN)
```

```
   if length(VLAN(V).ports) ~= 0
      for i = 1:PR_sk
         for j =1:N
         kanalas(V).Rin=VLAN(V).PrMat(i,:).*VLAN(V).RinMat(:,j)';
         [kanalas(V).Rout,kanalas(V).Rl(j)]=padalinimas(kanalas(V).Rin,kanalas(V).Rl(j));
         VLAN(V).RoutMat(:,j)=VLAN(V).RoutMat(:,j)+kanalas(V).Rout';
         VLAN(V).RoutSum(j)=sum(VLAN(V).RoutMat(:,j));
         VLAN(V).RoutSum(j)=sum(VLAN(V).RoutMat(:,j));
         end
      end
      for i = 1:length(VLAN(V).ports)
         for j=1:N
            srautas(VLAN(V).ports(i)).Rout(j)=VLAN(V).RoutMat(i,j);
         end

srautas(VLAN(V).ports(i)).Util=sum(srautas(VLAN(V).ports(i)).Rout)/srautas(VLAN(V).ports(i)).Util;
         srautas(VLAN(V).ports(i)).Util
      end
      VLAN(V).Util=sum(VLAN(V).RoutSum)/(kanalas(V).Cmax*N);
      VLAN(V).Util
   end
end

for V = 1:length(VLAN)
   if length(VLAN(V).ports) ~= 0
      figure(V);
      for i = 1:length(VLAN(V).ports)
          plot(linspace(0,N,N), srautas(VLAN(V).ports(i)).Rout-(i-
1)*5,'LineWidth',3,'Color',colors(VLAN(V).ports(i)));
          hold on;
      end
      plot(linspace(0,N,N),VLAN(V).RoutSum,'LineWidth',3,'Color','k', 'LineStyle','--');
      axis([0 N 0 1000]);
      xlabel('Laikas, s', 'FontSize', 15);
      ylabel('Duomenu perdavimo sparta, Mbit/s', 'FontSize', 15);
      set(gca,'fontsize',14)
      grid on;
      hold off;
   end
end
figure(1)
legend('Video srautas','Papildomas srautas nr.2','Bendra sparta kanale');
figure(2)
legend('Papildomas srautas nr.1','Papildomas srautas nr.3','Bendra sparta kanale');
```

vlanas.m

```
classdef vlanas
   properties
      PrMat
      RinMat
      RoutMat
      RoutSum
      Util
      ports
   end
```

```
end
```

port.m

```
classdef port
   properties
      Rin
      Rout
      Prio
      Cmax
      VID
      Rl
      Util
   end
end
```

padalinimas.m

```
function [Rout,Rl] = padalinimas(Rin, Rk)
N=nnz(Rin);
RcSum=1;
Rout=0*ones(1,length(Rin));
Rl=Rk;
   while RcSum~=0
      RcSum=0;
      Nc=0;
      for i=1:length(Rin)
         if Rin(i)~=0
            if Rl~=0
               if Rin(i)<=Rl/N
                  Rout(i)=Rin(i);
                  Rin(i)=0;
                  RcSum=RcSum+Rout(i);
                  Nc=Nc+1;
               end
            end
         end
      end
   Rl=Rl-RcSum;
   N=N-Nc;
   end
   if N~=0
         for i=1:length(Rin)
            if Rin(i)~=0
              Rout(i)=Rl/N;
            end
         end
         Rl=0;
   end
end
```