VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
KOMPIUTERIJOS IR RYŠIŲ TECHNOLOGIJŲ KATEDRA

Narūnas Kapočius

ATVIROJO KODO VIRTUALIŲJŲ TINKLO SPRENDIMŲ TYRIMAS
CASE STUDIES OF OPEN SOURCE SOFTWARE DEFINED NETWORKING SOLUTIONS

Baigiamasis magistro darbas

Telekomunikacijų inžinerijos studijų programa, valstybinis kodas 621H64003

Telekomunikacijų technologijų specializacija

Elektronikos ir elektros inžinerijos studijų kryptis

Vilnius, 2020

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
KOMPIUTERIJOS IR RYŠIŲ TECHNOLOGIJŲ KATEDRA

SUDERINTA
Katedros vedėjas Algirdas Baškys

Narūnas Kapočius

ATVIROJO KODO VIRTUALIŲJŲ TINKLO SPRENDIMŲ TYRIMAS
CASE STUDIES OF OPEN SOURCE SOFTWARE DEFINED NETWORKING SOLUTIONS

Baigiamasis magistro darbas

Telekomunikacijų inžinerijos studijų programa, valstybinis kodas 621H64003

Telekomunikacijų technologijų specializacija

Elektronikos ir elektros inžinerijos studijų kryptis

Vadovas	Doc. Dr. Arūnas Šaltis		2020-05-28
	(Moksl. laipsnis/pedag. vardas, vardas, pavardė)	(Parašas)	(Data)
Lietuvių kalbos konsultantas	Dr. Anželika Gaidienė		
	(Moksl. laipsnis/pedag. vardas, vardas, pavardė)	(Parašas)	(Data)

MB darbas peržiūrėtas lietuvių kalbos konsultantės dr. Andželikos Gaidienės:
Kompiuterijos ir ryšių technologijų katedros administratorė Jolanta Saudargaitė

Vilnius, 2020

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
KOMPIUTERIJOS IR RYŠIŲ TECHNOLOGIJŲ KATEDRA

Elektronikos inžinerijos studijų kryptis
Telekomunikacijų inžinerijos studijų programa
Telekomunikacijų technologijos specializacija

TVIRTINU
Katedros vedėjas
_____
(*parašas*)
Prof. dr. Algirdas Baškys

BAIGIAMOJO MAGISTRO DARBO UŽDUOTIS

..... Nr. .....
Vilnius

Studentui Narūnas Kapočius
(*vardas, pavardė*)

Baigiamojo darbo tema:

Atvirojo kodo virtualizuotų tinklo sprendimų tyrimas
Case studies of open source software defined networking solutions
(*lietuvių ir anglų kalbomis*)

Baigiamojo darbo užbaigimo terminas 2020 m. birželio mėn. 5 d.

BAIGIAMOJO DARBO UŽDUOTIES PRADINIAI DUOMENYS IR REIKALAVIMAI

1. Darbo tikslas

Ištirti populiariausių atvirojo kodo konteinerių tinklo sąsajos įskiepių našumą fiziniame duomenų centro tinkle.

2. Duomenys ir reikalavimai

Baigiamajame darbe turi būti atlikta tinklo virtualizacijos technologijų ir populiariausių konteinerių tinklo sąsajos įskiepių apžvalga. Palyginti įskiepių ir tinklo virtualizacijos technologijų privalumui, trūkumai bei veikimo principai. Atlikti įskiepių našumo tyrimus fiziniame duomenų centro tinkle, pateikti rezultatus bei rezultatus atitinkančias išvadas.

3. Aprašo turinys

1. Įvadas. Užduoties analizė
2. Analogiškų tyrimų apžvalga
3. Tyrimų metodikos sudarymas
4. Tinklo įskiepių tyrimas
5. Išvados

Literatūros ir informacinių šaltinių sąrašas
Priedai (CD, DVD, USB, kita)
Tyrimų metu naudotų scenarijų išeities kodas

Darbo vadovas: _____ doc. dr. Arūnas Šaltis
(*parašas*) (*mokslinis vardas ir laipsnis, vardas, pavardė*)

Užduotį gavau: 2020-05-29 _____ Narūnas Kapočius
(data) (*parašas*) (*vardas, pavardė*)

Vilniaus Gedimino technikos universitetas
Elektronikos fakultetas
Kompiuterijos ir ryšių technologijų katedra

ISBN ISSN
Egz. sk. .....
Data .....-.....-.....

Antrosios pakopos studijų **Telekomunikacijų inžinerijos** programos magistro baigiamasis darbas

Pavadinimas	**Atvirojo kodo virtualiųjų tinklo sprendimų tyrimas**
Autorius	**Narūnas Kapočius**
Vadovas	**Arūnas Šaltis**

Kalba: lietuvių

Anotacija

Magistro darbo tikslas – ištirti keturių CNCF rekomenduojamų „Kubernetes“ CNI įskiepių našumą ir jo priklausomybę nuo įvairių paketų dydžio ir „Linux“ „teamd“ tvarkyklėmis jungiamų sąsajų skaičiaus fiziniame duomenų centro tinkle. Darbe nagrinėjamas CNI įskiepių TCP protokolo vidutinis pralaidumas, vidutinis ICMP paketų vėlinimas, CPU apkrova ir OS kontekstų keitimas. Tyrimo rezultatai yra lyginami su fizinės terpės rezultatais.

Darbo apimtis – 71 p. teksto be priedų, 47 iliustracijos, 6 lentelės, 51 bibliografinis šaltinis.

Atskirai pridedami darbo priedai.

Prasminiai žodžiai: „Kubernetes“, CNI, „Linux“, TCP, pralaidumas, vėlinimas, našumas.

Vilnius Gediminas Technical University
Faculty of Electronics
Department of Computer Science and Communications Technologies

ISBN ISSN
Copies No. .....
Date .....-.....-.....

Master Degree Studies **Telecommunications Engineering** study programme Master Graduation Thesis

Title	**Case Studies of Open Source Software Defined Networking Solutions**
Author	**Narūnas Kapočius**
Academic supervisor	**Arūnas Šaltis**

Thesis language: Lithuanian

Annotation

In this Master thesis 4 CNCF (Cloud Native Computing Foundation) recommended CNI plugins are benchmarked in a physical datacentre environment. CNI plugins performance is evaluated in terms of latency, average TCP throughput and OS context switching for various Maximum Transmission Unit (MTU) sizes, the different number of aggregated network interfaces, and different interfaces segmentation offloading conditions. Plugins' performance is compared to baremetal baseline results.

Thesis consist of: 71 p. text without extras, 47 pictures, 6 tables, 51 bibliographical entries.

Appendixes included.

Keywords: Kubernetes, CNI, Linux, TCP, throughput, latency, performance.

Vilniaus Gedimino technikos universiteto egzaminų sesijų ir baigiamųjų darbų rengimo bei gynimo organizavimo tvarkos aprašo
2 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Narūnas Kapočius, 20146343
(Studento vardas ir pavardė, studento pažymėjimo Nr.)
Elektronikos fakultetas
(Fakultetas)
Telekomunikacijų inžinerija, TETfm-18
(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO) SĄŽININGUMO DEKLARACIJA

2020 m. gegužės 30 d.

Patvirtinu, kad mano baigiamasis darbas tema „Atvirojo kodo virtualiųjų tinklo sprendimų tyrimas“ patvirtintas 2018 m. lapkričio 07 d. dekano potvarkiu Nr. 250el, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Mano darbo vadovas docentas daktaras Arūnas Šaltis.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

(Parašas)

Narūnas Kapočius
(Vardas ir pavardė)

TURINYS

SANTRUMPOS

ACI – „Cisco“ virtualiųjų tinklų sprendimas (angl. *Application Centric Infrastructure*)

AKS – „Azure“ „Kubernetes“ paslauga (angl. *Azure Kubernetes Service*)

API – programų sąsaja (angl. *Application Programming Interface*)

ARM – itin efektyvios kompiuterių architektūros tipas (angl. *Advandced RISC Machine*)

AWS – „Amazon“ saityno paslaugos (angl. *Amazon Web Services*)

BGP – išorinių vartų protokolas (angl. *Border Gateway Protocol*)

BPF – „Berkley“ paketų filtras (angl. *Berkely Packet Filter*)

CDN – turinio paskirstymo tinklas (angl. *Content Delivery Network*)

CNI – konteinerių tinklo sąsaja (angl. *Container Network Interface*)

CPU – centrinis procesorius (angl. *Central Proccesing Unit*)

CSV – failų formatas, dažniausiai naudojamas išsaugoti skaičiavimų duomenis (angl. *Comma Separated Values*)

DAC – tiesioginio prijungimo sąsaja (angl. *Directly Attached Copper*)

DPDK – „Intel“ įmonės greito paketų apdorojimo biblioteka (angl. *Data Plane Development Kit*)

eBPF – „Berkely“ įmonės platinta programinė įranga (angl. *Extended Berkley Packet Filter*)

EKS – „Amazon“ valdoma „Kubernetes“ paslauga (angl. *Elastic Kubernetes Service*)

FTP – bylų perdavimo protokolas (angl. *Filer Transfer Protocol*)

GCP – „Google“ viešojo duomenų centro platforma (angl. *Google Cloud Platform*)

GENEVE – vienas iš užklotojo tinklo sprendimų (angl. *Generic Network Virtualziation Encapsualtion*)

GKE – „Google“ valdoma „Kubernetes“ paslauga (angl. *Google Kubernetes Engine*)

GRE – viena iš paketų inkapsuliacijos technologijų (angl. *Generic Routing Encapsulation*)

GSO – TCP segmentų našumo optimizavimo mechanizmas (angl. *Generic Segmentation Offloading*)

HTTP – saityno protokolas (angl. *Hyper Text Transfer Protocol*)

IaaC – infrastruktūros aprašymas / kūrimas kodu (angl. *Infrastructure as a Code*)

ICT – informacinės ir ryšių technologijos (angl. *Information and Communication Technologies*)

IP – interneto protokolas (angl. *Internet Protocol*)

IPAM – IP adresų valdymas (angl. *IP Address Management*)

IPIP – viena iš paketų inkapsuliavimo technologijų (angl. *IP-in-IP*)

IPv4 – IP protokolo 4 versija

IPv6 – IP protokolo 6 versija

IT – informacinės technologijos (angl. *Information Technologies*)

KVM – „Linux“ virtualizacijos technologija (angl. *Kernel-based Virtual Machine*)

L{1..7} – OSI lygmenys

MTU – maksimalus perdavimo vienetas (angl. *Maximum Transmission Unit*)

NAT – tinklo adresų transliavimas (angl. *Network Address Translation*)

NFV – tinklo funkcijų virtualizavimas (angl. *Network Function Virtualisation*)

NVGRE – viena iš tinklo virtualizacijos technologijų (angl. *Network Virtualisation using GRE*)

OS – operacinė sistema (angl. *Operating System*)

OSI – atvirųjų sistemų sujungimo modelis (angl. *Open System Interconnection*)

OVS – atvirasis virtualusis perjungiklis (angl. *Open Virtual Switch*)

PCIe – patobulinta periferinė komponentų sąsaja (angl. *Peripheral Component Interconnection express*)

PF – fizinė funkcija (angl. *Physical function*)

QoS – paslaugos kokybė (angl. *Quality of Service*)

RAM – operatyvioji atmintis (angl. *Random Access Memory*)

RAN – radijo prieigos tinklas (angl. *Radio Access Network*)

RDMA – tiesioginės atminties prieiga (angl. *Remote Direct Memory Access*)

REST – saityno programų architektūros tipas (angl. *Representational State Transfer*)

RTT – paketo išsiuntimo ir grįžimo trukmė (angl. *Round Trip Time*)

SFP – tinklo sąsajos tipas (angl. *Small form-factor Pluggable*)

SR-IOV – PCIe resursų izoliavimo technologija (angl. *Single Root Input/Output Virtualization*)

SSH – nuotolinio prisijungimo tinklo protokolas (angl. *Secure Shell*)

TCP – transporto valdymo protokolas (angl. *Transmission Control Protocol*)

UDP – vartotojo duomenų perdavimo protokolas (angl. *User Datagaram Protocol*)

VF – virtualioji funkcija (angl. *Virtual Function*)

VLAN – virtualusis vietinis tinklas (angl. *Virtual Local Area Network*)

VNI – VXLAN tinklo identifikatorius (angl. *VLAN Network Identifier*)

VPC – virtualusis privatusis tinklas (angl. *Virtual Private Cloud*)

VTEP – VXLAN tunelio prieigos taškas (angl. *VXLAN Tunnel Enpoints*)

VXLAN – virtualusis išplėstasis vietinis tinklas (angl. *Virtual Extensible Local Area Network*)

XML – duomenų struktūrų aprašymo kalba (angl. *Extensible Markup Language*)

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

ĮVADAS. UŽDUOTIES ANALIZĖ

Tyrimo objektas

Informacinių ir komunikacijų technologijų sritis ICT sparčiai tobulėja kiekvienais metais. Ne taip seniai vartotojams siūlomus sprendimus sudarė didelės, monolitinės programos. Tačiau monolitinė programų architektūra nėra lanksti ir neturi galimybės lengvai keisti programų mastelį. Kadangi lankstumas ir lengvas programų mastelio keitimas yra pagrindinės šiuolaikinių IT sprendimų, paremtų HTTP ir REST API technologijomis, savybės, vis dažniau IT sprendimuose yra naudojama mikropaslaugų architektūra. Mikropaslaugų architektūros nauda itin pastebima taikant programų konteinerizaciją, kuri yra viena iš virtualizacijos formų, suteikianti lankstumo ir patogumo programuotojams. Taip pat tokios sistemos yra lengviau diegiamos ir turi galimybę prisitaikyti prie kintančio vartotojų ar duomenų srauto. Be to, konteinerizacija pasižymi mažesniu skaičiavimo perviršiu lyginant su tradicine virtualiųjų mašinų monitoriaus virtualizacija. Kita vertus, mikropaslaugų architektūra paremti IT sprendimai yra kompleksiški ir pasižymi dideliu tarpusavyje komunikuojančių mikropaslaugų, o kartu ir konteinerių skaičiumi. Todėl atsirado natūralus poreikis automatizuotai valdyti ir prižiūrėti mikropaslaugų sprendimus naudojant konteinerių orkestratorius, tokius kaip ,,Kubernetes“, ,,Docker Swarm“, ,,Apache Mesos“ ir kt. „Kubernetes“ šiuo metu yra labiausiai rinkoje naudojamas atvirojo kodo konteinerių orkestratorius, siūlantis geriausiomis praktikomis paremtą konteinerizuotų programų gyvavimo valdymo ciklą. Nors itin svarbi mikropaslaugų architektūros dalis yra mikropaslaugų tarpusavio komunikacija, ,,Kubernetes“ neturi vieno numatytojo tinklo įgyvendinimo sprendimo, o tik nusako bendras taisykles, kurių būtina laikytis įgyvendinant ,,Kubernetes“ tinklo modelį, kuris yra paremtas CNI standartu. Dėl to yra svarbu ištirti galimų ,,Kubernetes“ tinklo sprendimų savybes ir šiuos virtualiojo tinklo sprendimus palyginti tarpusavyje, todėl šio darbo tyrimo objektas yra CNI įskiepių našumo tyrimas fiziniame duomenų centro tinkle.

Temos naujumas ir aktualumas

Konteinerizacija ir konteinerių orkestravimas tampa vis populiaresnis, kadangi mikropaslaugomis paremti sprendimai yra lengvai įdiegiami, galima lengvai keisti jų mastelį, sprendimai nėra priklausomi nuo vienos platformos ir užima nedaug vietos. Tačiau didelis nepriklausomų mikropaslaugų kiekis sukuria iššūkį, kaip užtikrinti ir valdyti ryšį tarp didelio skaičiaus konteinerių ir mikropaslaugų. Todėl viena svarbiausių mikropaslaugų sričių yra jų tarpusavio ryšio įgyvendinimas. ,,Kubernetes“ tinklo modelis yra paremtas keliomis aiškiomis taisyklėmis, tačiau nėra pateikiamas vienas galimas sprendimo variantas. Labiausiai prigijęs ,,Kubernetes“ tinklo sprendimas yra CNI, kuris yra paremtas iš karto į sprendimą įtrauktais įskiepiais (angl. *built-in plugins*) ir trečiųjų šalių sukurtais įskiepiais. CNI yra atsakingas už konteinerių

pridėjimą prie tinklo, reikiamų maršrutų sukūrimą ir jų pasidalijimą. Kiekvienas įskiepis šias užduotis gali spręsti skirtingai, todėl CNI trečiųjų šalių įskiepiai ir jų našumas yra svarbi „Kubernetes“ tinklo modelio tyrimo dalis. Pastarųjų metų publikacijose „Kubernetes“ CNI įskiepių našumas buvo lyginimas tiek virtualizuotoje, tiek fizinėje duomenų centro aplinkoje įvairių scenarijų atveju ir yra pakankamai gerai išnagrinėtas. Tačiau iki šiol atlikti CNI įskiepių tyrimai nevertino įvairių paketų MTU dydžio įtakos įskiepių našumui. Todėl šiame darbe yra nagrinėjamas CNI įskiepių našumas fiziniame 1 Gbit/s spartos duomenų centro tinkle esant įvairiems paketų dydžiams.

Tyrimo tikslas ir uždaviniai

Šio tyrimo tikslas yra ištirti keturių CNCF rekomenduojamų „Kubernetes“ CNI įskiepių našumą ir jo priklausomybę nuo įvairių paketų dydžio ir „Linux“ „teamd“ tvarkyklėmis jungiamų sąsajų skaičiaus. Tyrimo rezultatai suteikia galimybę įvertinti CNI įskiepių skirtumus ir suteikia informacijos, kokiu atveju geriausia yra naudoti vieną ar kitą CNI įskiepį.

Esminiai šio darbo uždaviniai, būtini darbo tikslui įgyvendinti, yra šie:

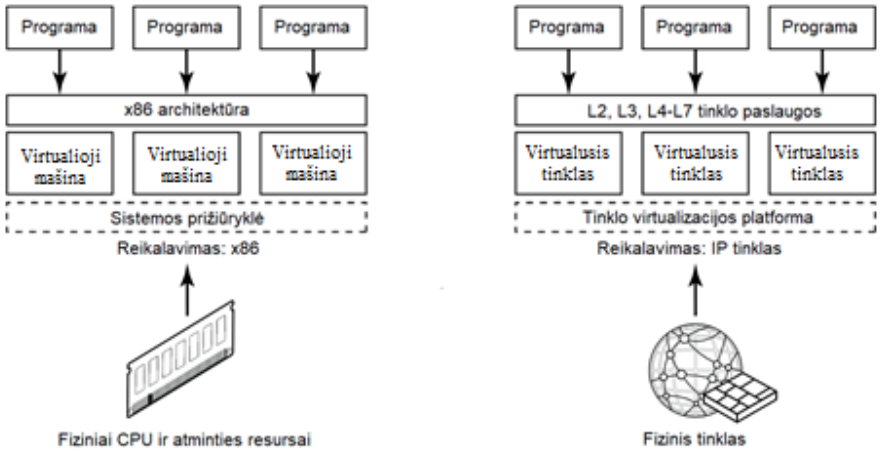
1. Ištirti CNI įskiepių vidutinį ICMP paketų vėlinimą fiziniame duomenų centro tinkle.
2. Palyginti CNI įskiepių vidutinį TCP protokolo pralaidumą, kai yra naudojami 76–8960 baitų dydžio MSS paketai.
3. Ištirti CNI įskiepių našumą (vidutinę CPU apkrovą, TCP pralaidumą ir vidutinį OS kontekstų keitimą), kai yra naudojamos 2–5 „Linux“ „teamd“ tvarkykle jungtosios tinklo sąsajos.
4. Išsiaiškinti fizinių tinklo sąsajų TCP GSO mechanizmo įtaką CNI įskiepių našumui abiem atvejais, kai nėra taikomas sąsajų jungimas ir kai sąsajų jungimas yra taikomas.

Tyrimo metodai

Anksčiau aptartiesiems uždaviniams ir darbo tikslui pasiekti buvo naudojamas trijų fizinių serverių „Kubernetes“ telkinys. Visi fiziniai serveriai buvo sujungti į atskirą izoliuotą 1 Gbit/s teorinės spartos fizinį tinklą. Našumo tyrimams pasirinktas TCP protokolas, kadangi didžioji dalis Internete veikiančių programų yra paremtos šiuo protokolu. TCP srautas buvo generuojamas „iperf3“ sintetinio srauto generatoriumi, o CPU apkrovos ir OS kontekstų keitimo rezultatai buvo gauti naudojant „sysstat“ programinio paketo „sar“ įrankį. Tyrimams automatizuoti buvo sukurti „Bash“ scenarijai, kurie buvo paleidžiami pagal „Chef“ scenarijų sukurtus tvarkaraščius.

1. TINKLO VIRTUALIZACIJOS TECHNOLOGIJŲ APŽVALGA

Kompiuterinių resursų virtualizacijos koncepcija yra pakankamai sena ir siekia pirmųjų centrinių kompiuterių (angl. *mainframe*) laikotarpį. Nors dabartinių technologijų ir virtualizacijos sprendimų negalima lyginti su kompiuterių eros pradžia, virtualizacijos koncepcija ir jos siūlomi privalumai nepasikeitė. Kompiuterinių resursų virtualizacija suteikia galimybę vykdyti keletą programų vienoje fizinėje terpėje, paskirstyti fizinius resursus ir atlikti programų izoliaciją (VMWare, 2020). Ši koncepcija suteikia itin daug lankstumo ir galimybių efektyviau panaudoti fizinius kompiuterių resursus IT sprendimų kūrėjams. Dėl šių privalumų virtualizacijos technologijų taikymas duomenų centruose tampa neišvengiamas. Kita vertus, dažniausiai nagrinėjant virtualizaciją yra kalbama apie skaičiavimo resursų virtualizaciją ir virtualiųjų mašinų naudojimą, tačiau taip pat itin svarbi tema yra kompiuterių tinklo virtualizacija. Tinklo virtualizacija turi labai daug panašumų į skaičiavimo resursų virtualizaciją (žr. 1.1 pav.). Virtualusis tinklas kaip ir virtualioji mašina yra įgyvendinama programine įranga ir teikia loginio virtualiojo tinklo paslaugas – virtualųjį maršruto parinkimą, paketų perjungimą, tinklo perskirstymą, užkardas ir pan. (Gozani, 2016).



1.1 pav. Skaičiavimo ir tinklo resursų virtualizacijų palyginimas (Gozani, 2016)

Literatūroje galima rasti įvairių tinklo virtualizacijos apibrėžimų ir technologijų skirstymų, tačiau bendrai galima išskirti šias technologijas naudojamas kurti virtualizuotus tinklus (Gozani, 2016):

- Užklotojo tinklo sprendimai (angl. *overlay networks*):
  - VXLAN
  - NVGRE
  - GENEVE
  - IPIP
- Kiti tinklo sprendimai

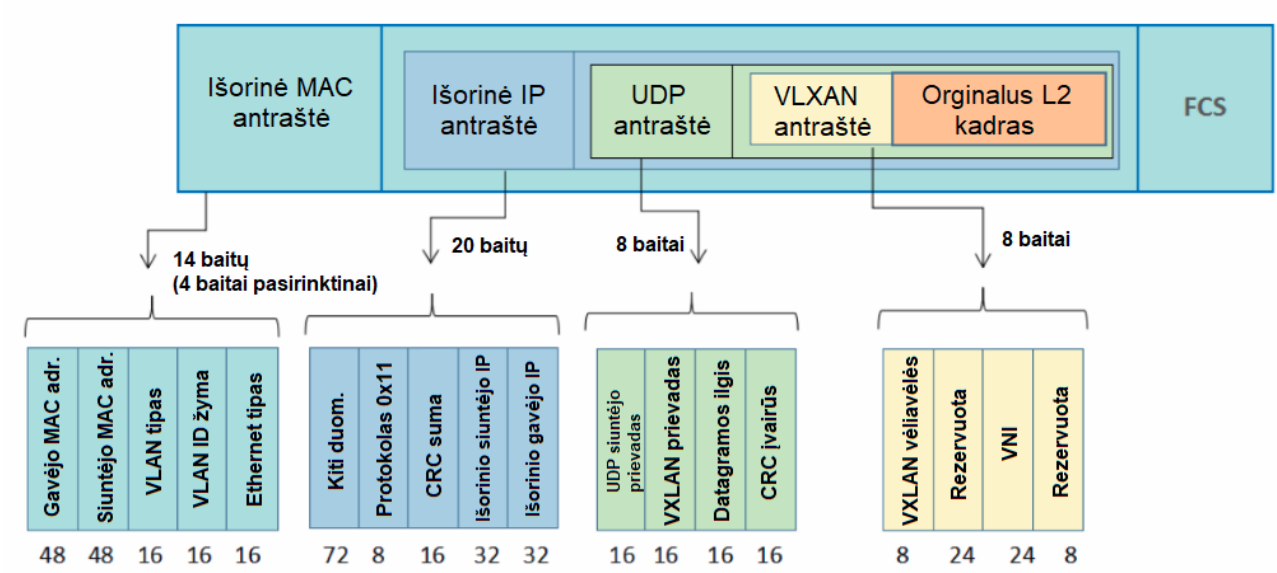
Toliau šiame skyriuje yra apžvelgiamos kompiuterių tinklo virtualizacijos ir jungimo technologijos nagrinėjant jų privalumus, trūkumus ir veikimo principus.

1.1. Užklotojo tinklo sprendimai

Užklotojo tinklo sprendimai suteikia galimybę visą tinklą įgyvendinti naudojant tik programinę įrangą ir nesudėtingos konfigūracijos IP fizinį tinklą (Gozani, 2016). Taip pat taikant užklotojo tinklo technologijas virtualųjį tinklą galima įgyvendinti jį parėmus kitu virtualiuoju tinklu, todėl bendruoju atveju užklotojo tinklo apibrėžimas apima tinklo įgyvendinimą pasinaudojus kitu tinklu (Huang ir Wu, 2017). Įdomu tai, kad užklotojo tinklo technologijas galima išskirti į L2 ir L3 OSI lygmens sprendimus. Tai reiškia, kad į fizinio tinklo L3 paketus yra inkapsuliuojami L2 arba L3 virtualiojo tinklo paketai taip sukuriant L2 arba L3 virtualiuosius tunelius tarp tinklo mazgų. L2 OSI lygmens užklotojo tinklo technologijoms galima priskirti VXLAN ir NVGRE ir GENEVE, o L3 OSI lygmens technologijoms IPIP ir GRE technologijas.

1.1.1. VXLAN technologija

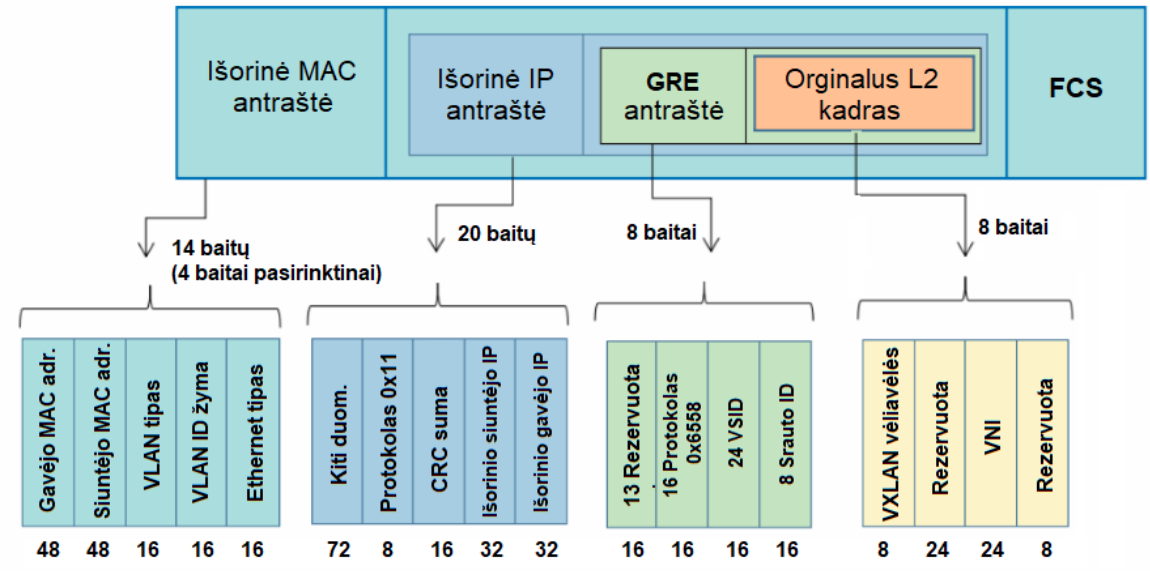
VXLAN technologija, laikoma *de facto* užklotojo tinklo numatytuoju standartu (VM Ware, 2020), buvo sukurta įmonės ,,Cisco“ (Fiber Cabling Solution, 2018) išspręsti VLAN technologijos apribojimus, tokius kaip STP protokolo konfigūravimo iššūkiai (Eclavea ir Allen, 2020) ar tinklo dydžio ribojimas dėl VLAN žymos ilgio iki 4094 tinklų (Huang ir Wu, 2017; DCLessons, 2019). VXLAN atveju yra naudojama ,,Ethernet“ kadro struktūra, panaši į VLAN kadro struktūrą (žr. 1.2 pav.), tačiau virtualiojo tinklo „Ethernet“ kadras yra inkapsuliuojamas į fizinio tinklo UDP datagramą. UDP atveju yra naudojamas 4789-asis gavėjo numatytasis prievadas ir tarp VXLAN galinių mazgų, kuriais gali būti fiziniai serveriai arba sistemos prižiūryklės (angl. *hypervisors*), yra sudaromas L2 tunelis (Eclavea ir Allen, 2020). VXLAN tinklo išplečiamumą užtikrina iki 16 milijonų tinklų taikant 24 bitų ilgio VXLAN tinklo identifikatorių VNI. Šis tinklo identifikatorius yra susiejamas su virtualiuoju IP potinkliu panašiai kaip ir VLAN atveju (VM Ware, 2020) ir yra perduodamas su kiekvienu paketu VXLAN antraštėje. Verta pastebėti, kad papildomos L2–L4 antraštės, būtinos įgyvendinti VXLAN technologiją, prideda pastebimą kiekį tarnybinės informacijos ir todėl VXLAN technologijos atveju tinklo sparta gali pastebimai sumažėti lyginant ją su fizinio tinklo pralaidumu. Jeigu fizinis tinklas turi galimybę, yra rekomenduojama padidinti MTU iki 1600 B, kad būtų kompensuojamas tinklo našumo sumažėjimas dėl papildomos VXLAN tarnybinės informacijos (Gozani, 2016).



1.2 pav. Fizinio kadro struktūra naudojant VXLAN technologiją (DC Lessons, 2019)

1.1.2. NVGRE technologija

NVGRE technologija yra alternatyvi technologija VXLAN. Ši technologija buvo sukurta ir yra palaikoma įmonės ,,Microsoft“ (Fiber Cabling Solution, 2018) siekiant išspręsti VLAN išplečiamumo problemą. Kaip ir VXLAN technologijos atveju NVGRE virtualiojo tinklo L2 kadrai yra inkapsuliuojami į fizinio tinklo L3 paketus (Garg ir Wang, 2015). Pagrindinis skirtumas tarp VXLAN ir NVGRE technologijų yra tas, kad VXLAN atveju prie virtualiojo tinklo L2 kadrų yra pridedama VXLAN antraštė ir vėliau visas kadras yra inkapsuliuojamas į UDP datagramą. NVGRE atveju prie virtualiojo tinklo L2 kadro yra pridedama GRE antraštė ir vėliau visas modifikuotas virtualiojo tinklo L2 kadras yra inkapsuliuojamas į fizinio tinklo L3 paketą (žr. 1.3 pav.).

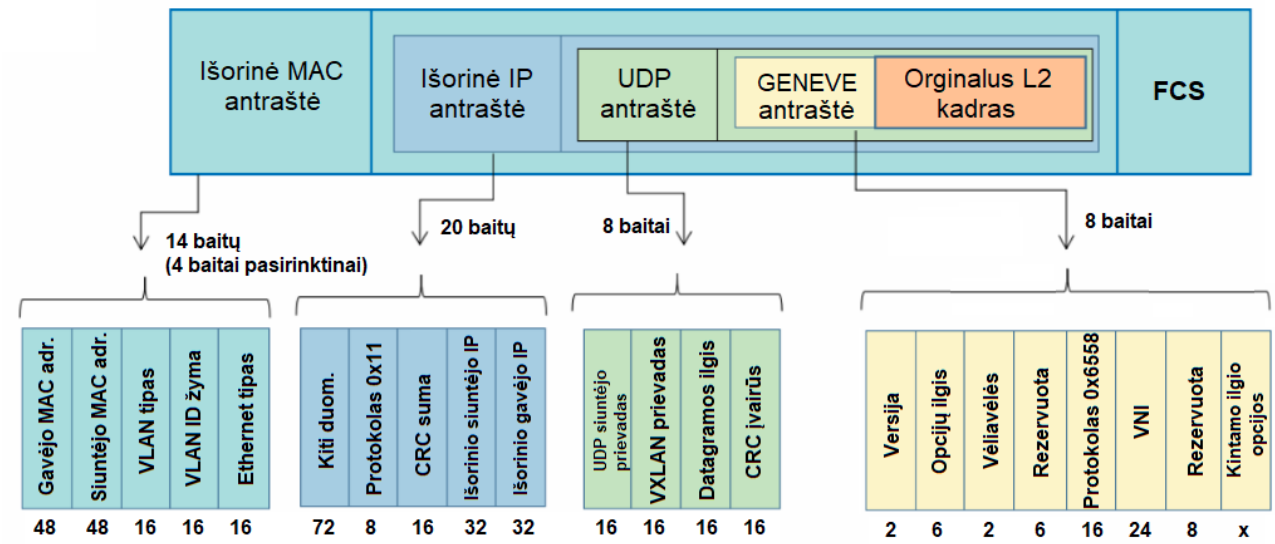


1.3 pav. Fizinio kadro struktūra naudojant NVGRE technologiją (DC Lessons, 2019)

NVGRE taip pat kaip ir VXLAN palaiko iki 16 milijonų virtualiųjų potinklių, kur potinklio numeris yra nurodomas GRE antraštės 24 bitų ilgio VSID lauke. Įdomu tai, kad NVGRE atveju bendras papildomos tarnybinės informacijos kiekis yra 8 baitais mažesnis nei VXLAN atveju ir siekia 42 baitus, o tai turėtų lemti efektyvesnį tinklo resursų išnaudojimą. Kita vertus, yra pabrėžiama, kad VXLAN technologija, be papildomų priemonių, gali garantuoti srauto paskirstymą ir garantuoti nustatytą paketų pristatymo tvarką tarp virtualiųjų mazgų, o NVGRE atveju norint užtikrinti šias savybes yra būtina nurodyti srauto numerį GRE antraštėje ir NVGRE mazgas turi turėti kelis IP adresus (Fiber Cabling Solution, 2018).

1.1.3. GENEVE technologija

GENEVE yra naujas virtualiųjų tinklų protokolas ir vis dar yra kuriamas kaip bendras technologijų milžinių „Microsoft“, „VmWare“, „Intel“ ir „RedHat“ sprendimas virtualiesiems tinklams įgyvendinti (Schmaus, 2017; Davie, 2014). Ši technologija buvo pradėta kurti siekiant išspręsti VXLAN ir NVGRE technologijų lankstumo trūkumą (Gross, Ganga ir Sridhar, 2018). GENEVE kadro struktūra yra labai panaši į VXLAN kadrus tik GENEVE kadrų atveju yra galimybė naudoti kintamo ilgio pasirinkimų laukus GENEVE antraštėje (žr. 1.4 pav.). Šie pasirinkimų laukai suteikia papildomo lankstumo tinklo aparatinės ir programinės įrangos gamintojams pridėti papildomų funkcionalumų. GENEVE protokolas taip pat palaiko iki 16 milijonų virtualiųjų tinklų, kur kiekvienas tinklas yra identifikuojamas GENEVE antraštėje esančiu 24 bitų VNI lauku. Virtualiojo tinklo kadrai kaip ir VXLAN atveju yra perduodami fiziniu tinklu UDP protokolu inkapsuliuojant virtualiuosius kadrus į UDP datagramas. Verta pabrėžti, kad nepaisant GENEVE protokolo privalumų, protokolas vis dar yra kūrimo stadijoje ir iki protokolo naudojimo pradžios produkcijos aplinkose dar gali užtrukti nemažai laiko (VMGuru, 2015).



1.4 pav. Fizinio kadro struktūra naudojant GENEVE technologiją (DC Lessons, 2019)

1.1.4. IPIP technologija

IPIP yra pakankamai sena technologija, pasiūlyta 1995 m. (Simpson, 1995). Esminis prieš tai aptartų VXLAN, NVGRE ar GENEVE technologijų ir IPIP skirtumas yra tas, kad IPIP atveju virtualiojo tinklo IP paketai yra inkapsuliuojami į fizinio tinklo IP paketus. Iš viso yra išskiriamos trys IPIP technologijos versijos (PacketLife, 2012):

- *ipip* – numatytoji IPIP versija, inkapsuliuojanti virtualiojo tinklo IPv4 paketus į fizinio tinklo IPv4 paketus.
- *ipv6* – IPIP versija, inkapsuliuojanti virtualiojo tinklo IPv6 paketus į fizinio tinklo IPv6 paketus.
- *ipv6ip* – IPIP versija, inkapsuliuojanti virtualiojo tinklo IPv6 paketus į fizinio tinklo IPv4 paketus.

Dažniausiai yra naudojama numatytoji IPIP versija, o kitos versijos yra naudojamos rečiau, tam tikriems specifiniams techniniams uždaviniams spręsti. Įdomu tai, kad numatytosios IPIP versijos atveju yra pridedama tik 20 baitų duomenų, kurie yra papildoma IP paketo antraštė, o tai yra mažiau papildomų tarnybinių duomenų lyginant su VXLAN, NVGRE ar GENEVE 32 baitų antraštėmis.

1.2. Kiti tinklų virtualizacijos sprendimai

NFV taip pat yra tinklų virtualizacijos šaka, kurios pagrindinis tikslas yra sumažinti kompiuterinių ir mobiliųjų tinklų operatorių kaštus, išleidžiamus projektuoti tinklams ir pirkti specializuotą aparatinę tinklo įrangą, tokią kaip RAN mazgai, užkardos, CDN, kraštiniai maršruto parinkikliai ir pan. (Chiosi, 2012). NFV atveju yra taikomi standartiniai IT virtualizacijos sprendimai virtualizuojant anksčiau minėtų tinklo įrenginių funkcijas ir jas perkeliant į standartinius didelio našumo serverius ar tinklo perjungiklius (Chiosi, 2012). Tokiu būdu galima sutaupyti ženklią kapitalo išlaidų dalį, be to, tinklo projektavimo procesas tampa paprastesnis ir tinklo architektūra gali būti dažniau keičiama, kadangi gerokai sutrumpėja kapitalo išlaidų atsipirkimo laikas. Kita vertus, būtina pabrėžti, kad kaip ir standartinės IT resursų virtualizacijos atveju galimi tam tikri našumo praradimai, dėl to taikant NFV yra būtina turėti efektyvią virtualizuotųjų resursų valdymo platformą (Chiosi, 2012).

Verta trumpai aptarti SR-IOV tinklo prievadų virtualizacijos technologiją. SR-IOV technologija suteikia galimybę virtualiosioms mašinoms efektyviai dalintis ta pačia fizine tinklo plokšte. Kitaip tariant, ši technologija leidžia sistemos prižiūryklei ar virtualiajai mašinai matyti vieną fizinį PCIe įrenginį kaip kelis atskirus PCIe įrenginius (Lowe, 2020). SR-IOV technologija šios koncepcijos įgyvendinimui apibrėžia PF ir VF funkcijas. PF funkcijos turi visas fizinio PCIe įrenginio savybes ir gali būti išsamiai konfigūruojamos ir valdomos kaip įprastas PCIe įrenginys. VF funkcijos

turi tik dalį PCIe fizinio įrenginio savybių ir negali būti išsamiai konfigūruojamos – iš esmės VF funkcijos gali tik išsiųsti ir priimti duomenis. Priklausomai nuo tinklo plokštės savybių kiekvienas fizinės tinklo plokštės prievadas teoriškai gali turėti iki 256 VF funkcijų (virtualiųjų prievadų). Praktiškai dėl reikalingų kitų serverio resursų rezervavimo kiekviena fizinė sąsaja dažniausiai gali turėti iki 64 virtualiųjų prievadų (Lowe, 2020). Bendrai SR-IOV ar kita I/O virtualizacijos technologija leidžia sutaupyti fizinių resursų, kadangi viename serveryje gali veikti daugiau virtualiųjų mašinų (Hanson, 2009). Taip pat I/O virtualizacijos technologija suteikia papildomo lankstumo pridedant ar išjungiant virtualiųjų mašinų tinklo sąsajas ir palengvina virtualiųjų mašinų migraciją iš vienos sistemos prižiūryklės į kitą ar suteikia galimybę lengviau pritaikyti tinklo QoS reikalavimus specifinėms virtualiosioms mašinoms (Hanson, 2009).

1.3. „Kubernetes“ ir konteinerių tinklo sąsaja

Su tinklo virtualizacija yra itin glaudžiai susijusios ir konteinerių technologijos, todėl šiame poskyryje yra apžvelgiama ,,Kubernetes“ konteinerių orkestratoriaus veikimo koncepcija, privalumai ir pritaikomumas, apžvelgiamas ,,Kubernetes“ tinklo modelis ir pagrindinės jo savybės. Taip pat glaustai yra palyginami pagrindiniai konteinerių tinklo sąsajų CNI sprendimai, įgyvendinantys virtualiuosius tinklus ir taikomi ,,Kubernetes“ atveju.

1.3.1. „Kubernetes“ konteinerių orkestratorius

,,Kubernetes“ yra 2014 metais „Google“ korporacijos pristatyta atvirojo kodo konteinerių automatizuoto valdymo sistema – orkestratorius. Konteinerių orkestratorius yra atsakingas už šias sritis (Vennam, 2019):

1. Programų diegimas (angl. *deployment*) – darbinių mazgų (angl. *worker nodes)*, kuriuose veiks konteinerizuotos programos parengimas ir priežiūra.
2. Efektyvus resursų panaudojimas (angl. *scaling*) – naudojant tvarkaraščius turimi resursai paskirstomi taip, kad būtų efektyviai naudojami.
3. Tinklo resursų priežiūra / paskirstymas – atlieka mikropaslaugų adresavimą ir padaro jas prieinamas kitoms mikropaslaugoms.
4. Įžvalga (angl. *insight*) – automatinis mikropaslaugų veikimo tikrinimas ir neveikiančių paslaugų keitimas naujomis ir visos sistemos kaip jungiojo tinklo stebėjimas.

,,Kubernetes“ yra paremtas valdančiojo (angl. *master*) ir darbinių mazgų principu. Valdantysis mazgas per tvarkaraščius ir administratoriaus komandas valdo darbinius mazgus ir ,,Kubernetes“ objektus, esančius telkinyje (angl. *cluster*). ,,Kubernetes“ atveju galima išskirti kelis pagrindinius objektus. Pats mažiausias loginis vienetas, kuris gali būti įdiegtas į „Kubernetes“ telkinį, yra vadinamas ankštimi (angl. *pod*) (Vennam, 2019; Kubernetes, 2020). Ankštis yra tolygi procesui, veikiančiam telkinyje. Ankštyje yra inkapsuliuojami programos konteineriai, kurių gali būti vienas ar

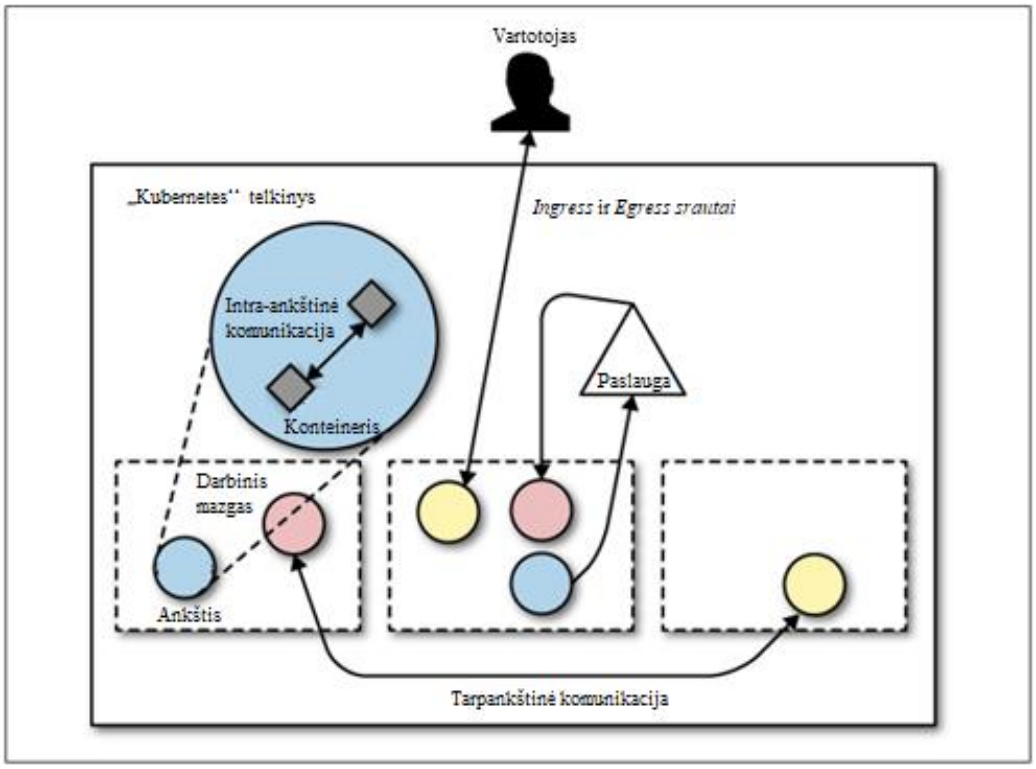
keli. Kitas ,,Kubernetes“ objektas – paslaugos (angl. *services*). Paslaugos yra abstrakcija, kuri susieja kelias grupes ankščių ir nusako jų prieigos taisykles. Paslaugos abstrakcija yra reikalinga, kadangi ankštys gali greitai keistis (būti išjungiamos, atsirasti naujų) ir vartotojui reikia suteikti vieną nesikeičiantį prieigos tašką prie programų veikiančių ankštyse.

Dar vienas svarbus ,,Kubernetes“ objektas yra saugykla (angl. *volume*), kuri suteikia vietą konteinerių sukuriamai informacijai saugoti. Saugyklos pagrindinis privalumas yra tas, kad net ir konteinerio perkrovimo metu saugykla, priklausanti ankščiai, išliks ir joje saugoma informacija liks nepakeista. Įdomu tai, kad ,,Kubernetes“ resursų paskirstymas galimas naudojant vardų erdves (angl. *namespaces*) ir yra naudojamas aplinkose, kur yra daug vartotojų ar norima atskirti aplinkas į kūrimo, testavimo ar produkcijos.

1.3.2. ,,Kubernetes“ tinklo modelis

,,Kubernetes“ tinklo modelis nesuteikia konkretaus tinklo įgyvendinimo sprendimo, tačiau nusako reikalavimus, kuriuos turi atitikti ,,Kubernetes“ tinklo sprendimas. ,,Kubernetes“ tinklo modelis nusako šiuos reikalavimus (Hausenblas, 2018; Sookocheff, 2018):

1. Ankštys privalo komunikuoti su visomis kitomis ankštimis, esančiomis tame pačiame ,,Kubernetes“ telkinyje nenaudojant NAT.
2. Visi darbiniai mazgai privalo komunikuoti su ankštimis, esančiomis telkinyje (ir atvirkščiai) nenaudojant NAT.
3. IP adresas, kurį turi ankštis, yra nekintantis ir mazgai tinkle jį mato vienodą.



1.5 pav. ,,Kubernetes“ tinklo modelio srautų tipai (Hausenblas, 2018)

Atsižvelgiant į anksčiau įvardytus „Kubernetes“ tinklo modelio reikalavimus tinkle būtina užtikrinti šių srautų įgyvendinimą (žr. 1.5 pav.) (Hausenblas, 2018; Sookocheff, 2018):

a) Konteineris–konteineris ryšio įgyvendinimas.

b) Ankštis–ankštis ryšio įgyvendinimas:

I. Kai abi ankštys yra tame pačiame darbiniame mazge.

II. Kai abi ankštys yra skirtinguose dabiniuose mazguose.

c) Ankštis–paslauga ryšio įgyvendinimas:

I. Ankštis → paslauga ryšio įgyvendinimas.

II. Paslauga → ankštis ryšio įgyvendinimas.

d) Internetas–paslauga ryšio įgyvendinimas:

I. Įeinančio srauto (angl. *Ingress*) įgyvendinimas.

II. Išeinančio srauto (angl. *Egress*) įgyvendinimas.

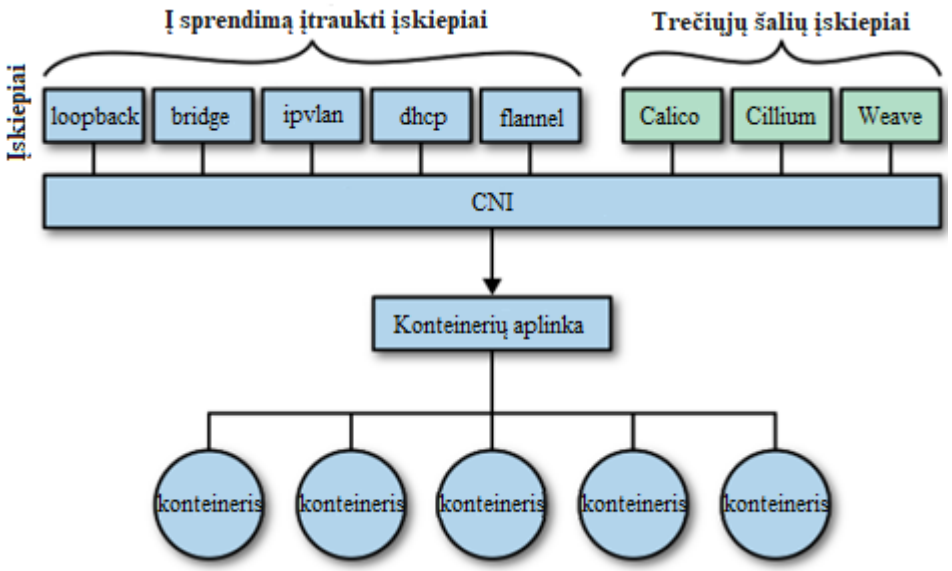
1.3.3 „Kubernetes“ CNI įskiepių apžvalga.

CNI yra *de facto* konteinerių orkestratorių, tokių kaip ,,Kubernetes“, ,,Mesos“ ir pan., virtualiųjų tinklų įgyvendinimo standartas (Hausenblas, 2018). CNI yra parašytas „Go“ programavimo kalba ir yra paremtas įskiepių struktūra (žr. 1.6 pav.). Tai reiškia, kad CNI yra sudarytas iš bibliotekų ir specifikacijų, nusakančių, kaip kurti naujus įskiepius, kurie geba konfigūruoti ,,Linux“ konteinerių tinklo sąsajas (Hausenblas, 2018). Dalis įskiepių yra įmontuoti į patį CNI, o kiti įskiepiai yra sukurti trečiųjų šalių. Visi šie įskiepiai yra atsakingi už tam tikrą tinklo konfigūracijos dalį ir sudaro CNI, kurio pagrindinė užduotis yra įgyvendinti konteinerių tarpusavio ryšį. CNI atlieka IPAM funkcijas – geba konteinerius prijungti / išjungti iš tinklo, priskirti IP adresus, sukonfigūruoti reikalingus maršrutus. Taip pat CNI užtikrina, kad konteineriai būtų pasiekiami ir kad būtų išsaugomi tinklo resursai, kai konteineris yra sunaikinamas (Hausenblas, 2018). Verta pabrėžti tai, kad kiekvieną kartą paleidus konteinerį dalis tinklo sąsajų konfigūracijos yra atliekama paties konteinerio, o kita dalis atliekama CNI. Kadangi CNI yra paremtas įskiepiais, kiekvienas įskiepis gali kitaip išspręsti IP adresavimo, maršrutų konfigūravimo ir jų paskirstymo problemas, todėl vertinga panagrinėti CNI įskiepius ir palyginti, kaip kiekvieno įskiepio atveju yra sprendžiami šie uždaviniai.

Vienas pirmųjų tinklo įskiepių, kuris buvo pradėtas naudoti ,,Kubernetes“ atveju, yra ,,Flannel“ (Zeng, Wang ir Zhang, 2017). Kiekvieną kartą į tinklą pridėjus naują mazgą / konteinerį „flanneld“ ankštis atlieka šiuos veiksmus (Lowe, 2020):

1. Mazgui yra priskiriamas IP adresas naudojant ,,etcd“ paskirstytųjų raktas / reikšmė saugyklą.
2. Yra sukuriama ir mazgui priskiriama virtualioji tinklo tilto sąsaja, vadinama ,,docker0“ tiltu.
3. Yra nustatomas numatytasis mazgas, kuriam bus persiunčiami paketai.

,,Flannel“ yra L2 OSI lygmens sprendimas, kadangi yra naudojama VXLAN technologija (Zeng, Wang ir Zhang, 2017). Todėl ,,Flannel“ gali pasižymėti mažesniu našumu lyginant su kitais L3 lygmens įskiepiais. ,,Flannel“ gali inkapsuliuoti įvairius aukštesnio lygmens protokolus, tokius kaip TCP, UDP, AWS VPC ir GCE. Taip pat ,,Flannel“ IPAM sprendimas yra paremtas IP adresų ir maršrutų planavimu. Maršrutų parinkimas yra atliekamas naudojant ,,iptables“ (CNI įskiepių palyginimas, 2018). ,,Flannel“ yra priskiriamas prie nelanksčių tinklo sprendimų, kadangi yra labai priklausomas nuo naudojamų aukštesnio lygmens protokolų ir yra patariama ,,Flannel“ įskiepį naudoti sprendimuose, kuriuose tinklo lankstumas nėra prioritetas (Zeng, Wang ir Zhang, 2017). Dėl šių ,,Flannel“ trūkumų ir tinklo funkcijų trūkumo dažnai ,,Flannel“ yra naudojamas kartu su kitu tinklo įskiepiu ,,Calico“ ir toks sprendimas yra vadinamas „Canal“ (Machine Zone, 2016; CNI įskiepių palyginimas, 2018).



1.6 pav. Apibendrinta CNI struktūra (Hausenblas, 2018)

,,Calico“ yra L3 OSI lygmens didelio našumo CNI įskiepis paremtas BGP protokolu ir iš esmės kurtas naudojimui duomenų centruose (Zeng, Wang ir Zhang, 2017). Kiekviename mazge yra naudojamas virtualusis maršruto parinkiklis ,,vrouter“, per kurį yra persiunčiami IP paketai. Pats maršruto parinkimas yra atliekamas naudojant ,,iptables“ arba ,,kubeproxy“ (CNI įskiepių palyginimas, 2018). ,,Calico“ našumą užtikrina saugodamas ir atnaujindamas dinamines maršruto lenteles kiekviename mazge, o maršrutai tinkle yra paskirstomi nenaudojant NAT ir tunelių, o naudojant BGP protokolą. Esminis ,,Calico“ trūkumas yra tas, kad kiekvienam mazgui tinkle yra priskiriamas atskiras IP adresas, dėl to maršruto lentelių dydžiai labai išauga ir dėl šių priežasčių ,,Calico“ gali būti ne itin saugus ir lengvai perkeliamas sprendimas (Zeng, Wang ir Zhang, 2017). Įdomu tai, kad ,,Calico“ atveju rekomenduojamas maksimalus mazgų skaičius tinkle yra 5000 (CNI įskiepių palyginimas, 2018). ,,Calico“ CNI įskiepis yra itin paplitęs ir yra naudojamas tokių debesijos

paslaugų tiekėjų kaip ,,Google“ ir ,,Amazon“ (Zeng, Wang ir Zhang, 2017). Verta paminėti, kad ,,Calico“ įskiepis gali veikti trimis režimais: IPIP, VXLAN ir įprastuoju IP režimu (Project Calico, 2020). Priklausomai nuo režimo yra naudojama skirtinga paketų inkapsuliacija. IPIP ir VXLAN režimus rekomenduojama naudoti tinkluose, kurių tinklų adresavimas nėra kontroliuojamas ir tinklai gali persidengti, pavyzdžiui, AWS atveju sujungus kelis VPC (Project Calico, 2020).

Be dviejų jau aptartų vienų iš pagrindinių CNI įskiepių taip pat dar yra naudojami ,,Weave“, ,,Cilium“ ir ,,Contiv“ ir ,,Kube-router“ įskiepiai (Acreman, 2018).

,,Weave“ CNI įskiepis suteikia šifravimo funkcionalumą, kurio nėra nei ,,Flannel“, nei ,,Calico“ atveju. Taip pat ,,Weave“ gali veikti daliniu jungiuoju režimu. Kadangi ,,Weave“ veikia jungiojo tinklo režimu, kai informacija apie tinklą yra saugoma kiekviename mazge, užtenka turėti sujungimą tik su vienu mazgu ir automatiškai visi kiti tinklo mazgai tampa pasiekiami. Jungiojo tinklo režimas yra ,,Weave“ privalumas, bet kartu ir trūkumas, kadangi didėjant mazgų skaičiui tinkle didėja tarnybinės informacijos kiekis perduodamas tarp tinklo mazgų (Acreman, 2018). Yra teigiama, kad maksimalus mazgų skaičius ,,Weave“ atveju yra 500 mazgų (CNI įskiepių palyginimas, 2018). ,,Weave“ atveju maršruto parinkimas taip pat yra paremtas ,,iptables“ ir ,,kubeproxy“.

,,Cilium“ iš kitų CNI įskiepių išsiskiria tuo, kad yra orientuotas į saugumą ir gali taikyti saugumo taisykles kiekvienai programai atskirai ir veikti kaip programų lygmens užkarda. Maršrutų parinkimas yra atliekamas naudojant BPF, kuris yra spartesnis už ,,iptables“ ir ,,kubeproxy“. Šio įskiepio privalumas yra tas, kad jis veikia jungiojo tinklo režimu, kurį yra lengviau sukonfigūruoti nei BGP. Pagrindiniai šio įskiepio trūkumai yra tokie, kad mazgai turi naudoti naujausiąją ,,Linux“ kernelio versiją ir didesniems tinklams patartina naudoti BGP protokolą (CNI įskiepių palyginimas, 2018, Acreman, 2018).

,,Contiv“ CNI įskiepis turi didžiąją dalį anksčiau minėtų įskiepių funkcionalumų, tačiau šį įskiepį išskirianti funkcija yra ta, kad kelioms ankštims galima priskirti tuos pačius IP adresus ir įskiepis yra gerai suderinamas su ,,Cisco“ infrastruktūra ACI. „Contiv“ atveju maršruto parinkimas yra vykdomas naudojant ,,iptables“, paketai yra inkapsuliuojami naudojant VXLAN žymes, o maršrutai yra paskirstomi naudojant BGP protokolą (CNI įskiepių palyginimas, 2018; Acreman, 2018).

2. ANALOGIŠKŲ TYRIMŲ APŽVALGA

Šiame skyriuje yra apžvelgiami jau atlikti panašūs ,,Kubernetes“ CNI įskiepių ir ,,Linux“ tinklo sąsajų jungimo našumo tyrimai.

Trijų populiarių ,,Kubernetes“ tinklo sprendimų našumas buvo nagrinėjamas Kinijos mokslininkų (Zeng, Wang ir Zhang, 2017). Straipsnio autoriai CNI įskiepių tyrimų svarbą grindžia tuo, kad konteinerių technologijos yra atsiradusios palyginti neseniai ir egzistuoja daug CNI įskiepių, kurie sprendžia tą pačią problemą skirtingais būdais. Autoriai pabrėžia, kad tik žinant ir palyginus kiekvieno įskiepio savybes ir našumą galima efektyviai panaudoti turimus tinklo resursus. Autoriai nagrinėjamame straipsnyje lygina ,,Kubernetes“ CNI įskiepių ,,Flannel“, ,,Calico“ ir ,,Swarm Overlay“ tinklo sprendimų vidutinį pralaidumą ir vėlinimą fiziniame 1 Gbit/s teorinės spartos tinkle. ,,Kubernetes“ tinklo sprendimų našumas buvo tiriamas TCP ir UDP protokolų atveju. Straipsnyje pristatomi rezultatai rodo, kad faktinis vidutinis fizinės terpės pralaidumas TCP protokolo atveju yra 919,7 Mbit/s, o atitinkamai ,,Calico“, ,,Flannel“ ir ,,Swarm Overlay“ įskiepių pasiektas vidutinis pralaidumas yra 919 Mbit/s, 815,9 Mbit/s, ir 887,6Mbit/s. Matyti, kad ,,Calico“ vidutinis pralaidumas yra labai artimas fizinės terpės pralaidumui, o ,,Flannel“ ir ,,Swarm Overlay“ įskiepių naudojančių VXLAN technologiją pralaidumas yra pastebimai mažesnis nei fizinės terpės ar ,,Calico“ atveju. Straipsnyje tirtų CNI įskiepių vidutinis paketų vėlinimas yra labai panašus ir yra apie 0,1 ms didesnis nei fizinės terpės atveju.

Anksčiau aptartas straipsnis yra įdomus tuo, kad yra lyginamas ne tik CNI įskiepių pralaidumas bet ir vidutinis paketų vėlinimas. Kita vertus, iš straipsnyje pateikiamos informacijos nėra aiški matavimams naudoto tyrimų stendo konfigūracija, tyrimo metodika, atliktų matavimų skaičius ir pan. Taip pat šio magistro darbo autorius nori pabrėžti, kad nors nagrinėtame straipsnyje ir yra pateikiami CNI įskiepių vidutinio pralaidumo rezultatai, UDP protokolo atveju toks palyginimas yra netikslus, kadangi dėl UDP protokolo specifikos nėra patikimo būdo išmatuoti kanalo pralaidumą.

,,Kubernetes“ CNI įskiepių palyginimo svarba taip pat yra pabrėžiama Pietų Korėjos mokslininkų atliktame CNI įskiepių palyginimo tyrime (Park ir Yang, 2018). Autoriai nagrinėjo ,,OpenStack“ ir ,,Kubernetes“ CNI įskiepių tinklo našumą virtualiajame ,,OpenStack“ tinkle. Tyrimo stendas buvo sudarytas iš vieno fizinio serverio, turinčio 64 GB RAM, ,,Xeon(R) E5-2697“ CPU ir serveris buvo prijungtas prie 10 Gbit/s SFP+ tinklo. Šiame fiziniame serveryje veikė keturios virtualiosios mašinos. Pirmoji virtualioji mašina buvo naudojama kaip ,,OpenStack“ ir ,,Kubernetes“ valdantysis mazgas, o likusios virtualiosios mašinos kaip ,,Kubernetes“ darbiniai mazgai. Tyrimo metu buvo lyginami ,,Flannel“ įskiepio VXLAN ir UDP režimai ir ,,OpenStack“ ,,Kuryr“ tinklo įskiepio pralaidumas. Pralaidumas buvo tiriamas naudojant ,,iperf“ tinklo tyrimo įrankį, tačiau įdomu tai, kad CNI įskiepių pralaidumas buvo matuojamas tarp pirmame ir trečiame ,,Kubernetes“

darbiniame mazge esančių ankščių ir jų paketai buvo persiunčiami per antrame „Kubernetes“ darbiniame mazge veikiančią maršruto parinkiklio ankštį. Autorių atlikti tyrimai rodo, kad tirtų 64 B, 512 B, 8192 B ir 16384 B MTU paketų dydžių atveju „Kuryr“ OVS tinklas yra našesnis nei „Flannel“ VXLAN režimas. Ryškus pralaidumo skirtumas yra pastebimas nuo 512 B MTU paketų dydžio ir atitinkamai pralaidumas yra mažesnis 20 %, 27,27 % ir 8 %. Straipsnio autoriai „Flannel“ įskiepio pralaidumo atsilikimą grindžia skaičiavimo perviršiu, kurį lemia įskiepio architektūra ir jos nulemti dažni kernelio ir vartotojo erdvės keitimai. Verta pastebėti, kad nors straipsnyje ir nėra pabrėžiamas tyrimams atlikti naudojamas transportinio lygmens protokolas, tačiau galima numanyti, kad yra naudojamas TCP protokolas.

Iki šiol išsamiausias „Kubernetes“ CNI įskiepių našumo tyrimas yra pristatomas Ducastelio publikacijoje (Ducastel, 2018) ir papildomi to paties autoriaus rezultatai pristatomi papildyto straipsnio (Ducastel, 2019) publikacijoje. Šiose publikacijose yra nagrinėjamas šešių dažniausiai naudojamų ir „Kubernetes“ oficialioje dokumentacijoje rekomenduojamų (žr. „Kubernetes“ dokumentaciją) CNI įskiepių našumas. Tyrimas buvo atliekamas fiziniame tinkle, kurį sudarė trys vienodi „SuperMicro“ fiziniai serveriai tarpusavyje sujungti naudojant „SuperMicro“ 10 Gbit/s spartos perjungiklį ir DAC SFP+ sujungimus tarp serverių ir tinklo perjungiklio. Serveriuose veikė „Ubuntu“ 18.04 LTS OS ir buvo naudojamos „Kubernetes“ 1.14 ir „Docker“ 18.09.2 versijos. Vienas serveris atliko „Kubernetes“ valdančiojo mazgo funkcijas, o kiti du serveriai „Kubernetes“ darbinių mazgų funkcijas. Pralaidumo tyrimai buvo atliekami nekeičiant kliento ir serverio darbinių mazgų ir tai buvo pasiekta naudojant „Kubernetes“ „NodeSelector“ žymenis. Šis tyrimas yra įdomus tuo, kad CNI įskiepių našumas buvo tiriamas esant 1500 B ir 9000 B MTU dydžio TCP, UDP, HTTP ir SSH protokolų atveju, tačiau publikacijoje yra pateikiami tik 9000 B MTU rezultatai. TCP ir UDP atveju pralaidumo nustatymui buvo naudojamas „Linux“ „iperf3“ įrankis, HTTP našumas buvo nustatytas pasitelkiant „Nginx“ serverį ir „curl“ klientą, FTP protokolo atveju buvo naudojamas „vsftpd“ serveris ir taip pat „curl“ klientas, o SSH protokolo našumas buvo tiriamas naudojant „OpenSSH“ klientą ir serverį. Kiekvieno protokolo atveju buvo atliekami bent trys matavimai ir pralaidumo rezultatai yra pateikiami kaip tų matavimų vidurkis. Verta pabrėžti, kad publikacijoje nėra pateikiama matavimų trukmė. TCP protokolo atveju, esant 9000 B paketų dydžiui, pagal našumą tirti įskiepiai išsidėstė šia tvarka: „Calico“, „Flannel“, „Canal“, „Weave“, „Kube-router“, „Cillium“, o įskiepių pralaidumas lyginant su fizine terpe buvo atitinkamai mažesnis 0,65 %, 0,89 %, 0,91 %, 1 %, 1,41 % ir 2,26 %. Nepaisant minėtame tyrime pristatomo išsamaus CNI įskiepių palyginimo įvairių protokolų atveju, šio magistrinio darbo autoriaus manymu, šie rezultatai yra tinkami įvertinti CNI įskiepių našumą tik specifiniu atveju, kai yra naudojami „Jumbo“ kadrai. Todėl norint visapusiškai įvertinti CNI įskiepių našumą yra būtina įtraukti ir mažesnius MTU dydžius.

Labai panašiame tyrime į ką tik apžvelgtąjį buvo nagrinėjamas „Kubernetes“ CNI įskiepių našumas virtualizuotoje duomenų centro aplinkoje (Kapočius, 2019). Tyrimo metu buvo nustatyta, kad „VmWare“ virtualiųjų mašinų atveju CNI įskiepių TCP protokolo pralaidumas sumažėja nuo ~1 % iki ~36 %, esant 1500 B MTU dydžio paketams ir ~0,25 % iki ~15 %, kai naudojami 9000 B paketai.

Jau aptarti tyrimai nagrinėja įprastus ,,Kubernetes“ CNI įskiepius, kurie yra paremti standartiniu ,,Linux“ kernelio tinklo dėklu, tačiau specifiniais atvejais siekiant itin mažo vėlinimo ir didelės spartos standartinis ,,Linux“ kernelio tinklo dėklas nėra pritaikytas tokiems dideliems našumo poreikiams (Kernelio ir vartojo erdvės tinklo sprendimų aptarimas). Tokiu atveju dažniausiai yra naudojami kernelio aplenkimo sprendimai perkeliant tinklo dėklo funkcijas iš kernelio erdvės į vartotojo erdvę ir yra naudojami tokie sprendimai kaip RDMA, DPDK, ,,Netmap“, ,,Snabbswitch“, ,,PF_RING“ ir pan. (*Cloudflare* tinklaraštis). RDMA ir DPDK sprendimai ,,Kubernetes“ CNI atveju buvo nagrinėjami Budapešto technologijos ir ekonomikos universiteto mokslininkų (Gehberger, 2018). Straipsnio autoriai teigia, kad debesų technologijų pritaikymas itin mažo vėlinimo sprendimams atvertų kelią naujoms debesų technologijų panaudojimo galimybėms gamybos industrijos automatizacijos valdymui ir pan. RDMA ir DPDK sprendimai buvo tiriami naudojant įmonės ,,Intel“ sukurtą SR-IOV ,,Kubernetes“ CNI įskiepį (,,Intel“ SR-IOV įskiepio repozitorija). ,,Kubernetes“ telkinys buvo sukonfigūruotas naudojant du fizinius serverius, turinčius ,,Intel Xeon E5-2670 v3“ CPU, 64GB RAM ir ,,Mellanox ConnectX-4 40GbE“ tinklo plokštes. Tyrimai buvo atlikti naudojant ,,Ubuntu“ 16.04 OS su 4.13 versijos kerneliu ir 1.9 ,,Kubernetes“ versija. Fiziniai serveriai buvo sujungti naudojant ,,Mellanox 1710 40GbE“ perjungiklį. Verta pabrėžti, kad SR-IOV CNI įskiepio naudojimui yra būtina turėti tam pritaikytas tinklo plokštes. Tyrimo autoriai nustatė, kad iki 65536 B dydžio paketų RDMA be DPDK sprendimai pasižymi panašiu vėlinimu, tačiau toliau didėjant paketų dydžiams RDMA sprendimo vėlinimas yra palyginamai mažesnis nei DPDK atveju. Taip pat tyrimo rezultatai rodo, kad RDMA ir DPDK sprendimai vėlinimo ir CPU sąnaudų atžvilgiu yra našesni nei standartinis ,,Linux“ tinklo dėklas.

Dar vienas didelio našumo ir mažo vėlinimo ,,Kubernetes“ tinklo sprendimas yra pasiūlytas Italijos ir JAV bendrame projekte, pavadinimu ,,Polycube“ (Miano et. al., 2019). ,,Polycube“ yra atvirojo kodo programinės įrangos karkasas, skirtas didelio našumo kernelio erdvės tinklo funkcijoms įgyvendinti. ,,Polycube“ karkasas yra paremtas eBPF, suteikiančiu galimybę kurti paketų apdorojimo grandines. Autorių siūlomas ,,Polycube“ sprendimas, priešingai nei SR-IOV CNI įskiepio atveju, veikia tik kernelio erdvėje. Autoriai pabrėžia, kad nors vartotojo erdvės tinklo sprendimai suteikia pastebimus vėlinimo ir pralaidumo patobulinimus, tačiau rezervuojant CPU resursus yra atimami programoms skirti resursai ir visas tinklo dėklas privalo būti perkurtas ir perkeltas į vartotojo erdvę. Tyrimo autoriai pritaikė ,,Polycube“ karkasą ,,Kubernetes“ CNI įskiepio sukūrimui, kurį išbandė

fiziniame tinkle. ,,Kubernetes“ telkinys buvo sudarytas iš trijų fizinių serverių, turinčių ,,Intel Xeon E3-1245v5 @3.50 GHz“ CPU, dviejų prievadų ,,Intel XL710 40GBps“ tinklo plokštes ir serveriai buvo sujungti taškas–taškas sujungimu. Tinklo pralaidumas buvo tiriamas naudojant ,,iperf3“ įrankį su numatytaisiais nustatymais ir tyrimui naudojant TCP protokolą. Pralaidumas buvo tiriamas tiek ir tarp ,,Kubernetes“ ankščių, veikiančių tame pačiame darbiniame mazge, tiek ir ankščių, veikiančių skirtinguose darbiniuose mazguose. Autorių atliktas tyrimas rodo, kad jų sukurtas ,,Polycube“ sprendimas įprastus ,,Kubernetes“ CNI įskiepius lenkia 15–20 % didesniu pralaidumu. Ankščių skirtinguose darbiniuose mazguose atveju didžiausias pralaidumas buvo pasiektas ,,Polycube“ įskiepio, o mažiau našesni mažėjimo tvarka buvo šie įprasti ,,Kubernetes“ CNI įskiepiai: ,,Kube-router“, ,,Romana“, ,,Weave“, ,,Cillium“, ,,Calico“. Iš straipsnyje pateikiamų rezultatų tikėtina, kad ,,Calico“ CNI įskiepis buvo tiriamas numatytuoju IPIP režimu.

Kadangi šiame magistro darbe CNI įskiepių našumas yra nagrinėjamas ir ,,Linux“ tinklo sąsajų jungimo atveju, yra verta apžvelgti ,,Linux“ tinklo sąsajų našumo tyrimus. Atlikus literatūros šaltinių paiešką pastebėta, kad nėra daug ir išsamių ,,Linux“ tinklo sąsajų jungimo tyrimų ir jau atlikti tyrimai nagrinėja senesnes ,,Linux“ tinklo sąsajų ,,bonding“ tvarkykles, o ne naujesnes ,,teamd“ tvarkykles. ,,Linux“ ,,bonding“ tvarkyklių našumas buvo nagrinėjamas (Aust, Kim, Davis, Yamaguchi ir Obana, 2006). Straipsnyje pristatomas ,,RedHat“ ,,Centos 4“ OS, naudojančios 2.6.13 kernelio versiją, ir ,,bonding“ tvarkyklės 2.6.3 versijos tyrimas. Nors šis tyrimas atliktas seniai, tačiau tyrimas vertas apžvalgos dėl jo metu naudotos metodikos. Tyrimas buvo atliekamas naudojant du fizinius serverius, turinčius po tris 1 Gbit/s spartos tinklo plokštes. Serveriai sujungti tiesiogiai CAT6 kategorijos tinklo kabeliais. Pralaidumas buvo tiriamas naudojant ,,iperf“ ,,Linux“ įrankį ir buvo tiriamas UDP protokolas ,,RoundRobin“ ,,bonding“ tvarkyklės režimu. Tyrimo rezultatai parodė, kad ,,RoundRobin“ režimo ir UDP protokolo atveju galima pasiekti didesnę suminę spartą negu yra pasiekiama atskirų sąsajų atveju. Kita vertus, tyrimo autoriai užsimena, kad didinant siuntimo spartą yra pastebima didėjanti paketų priėmimo tvarkos sklaida, o tai TCP protokolo atveju kaip tik nepadidintų, o sumažintų suminę kanalo perdavimo spartą.

Išsamus ,,Linux“ tinklo sąsajų jungimo tyrimas yra pristatomas vokiečių mokslininkų tyrime (Meier, Schmelzer ir Suchodoletz, 2012). Tyrimo autoriai pabrėžia, kad tinklo sąsajų jungimas yra tinkamas sprendimas padidinti tinklo pralaidumą ir patikimumą, kai nėra galimybės atnaujinti tinklo infrastruktūros. Straipsnyje yra pristatomas ,,Linux“ ,,bonding“ tvarkyklių visų 6 režimų tyrimas taikant kliento ir serverio architektūrą. Pralaidumo tyrimui srautas buvo generuojamas naudojant DNBD3 ,,Linux“ įrankį, kuris veikia kaip blokinis ,,Linux“ įrenginys. Klientas taikydamas ,,Linux“ komandą ,,dd“ skaitė duomenis iš serveryje veikiančio minėto blokinio įrenginio ir informaciją rašė į ,,/dev/null“ įrenginį. Tyrimo metu buvo nagrinėtos dvi galimos tinklo sąsajų jungimo konfigūracijos. Pirmojoje konfigūracijoje klientas buvo prijungtas prie perjungiklio 10 Gbit/s spartos

optiniu kanalu, o serveris prijungtas prie perjungiklio 10x1 Gbit/s spartos optiniais kanalais ir juos sujungiant į vieną loginį kanalą taikant „Linux“ „bonding“ tvarkyklę. Šios konfigūracijos našumas yra lyginamas su bazine konfigūracija, kai ir klientas ir serveris prie perjungiklio yra prijungti 10 Gbit/s spartos kanalais. Tyrimo rezultatai rodo, kad bazinės konfigūracijos atveju didžiausias kanalo pralaidumas 9600 Mbit/s pasiekiamas klientui naudojant 4-ias „dd“ kliento instancijas, o tinklo sąsajų jungimo atveju, kai yra naudojamas nulinis „bonding“ tvarkyklės režimas, maksimalus kanalo pralaidumas yra pasisiekiamas jau esant dviem „dd“ instancijoms ir yra pastebimai mažesnis – 2880 Mbit/s. Antruoju konfigūracijos atveju serveris prie perjungiklio buvo prijungtas naudojant 10x1 Gbit/s spartos kanalus, o klientai, kurių buvo 61, prijungti prie kito perjungiklio taip pat 1 Gbit/s spartos kanalais. Minėti perjungikliai tarpusavyje buvo sujungti 10 Gbit/s spartos kanalu. Šia konfigūracija buvo tiriami visi šeši „bonding“ tvarkyklės režimai. Tyrimo rezultatai rodo, kad tik penkto ir šešto „bonding“ tvarkyklės režimų didinant klientų skaičių suminė sparta yra beveik lygi bazinės konfigūracijos suminei spartai. Autoriai teigia, kad „Linux“ tinklo sąsajų jungimas taikant „bonding“ tvarkyklę yra naudingas sprendimas, kai tinkle veikia didelis klientų skaičius. Taip pat taikant sąsajų jungimą didėja procesoriaus apkrova ir ji yra pastebimai didesnė, esant nuliniam „bonding“ tvarkyklės režimui, o kitų režimu atveju galima pastebėti kiek padidėjusią procesoriaus apkrovą, tačiau ji yra nedidelė.

Išsamiausią tinklo sąsajų jungimo tyrimą pristato Tailando mokslininkai (Rattanaopas ir Boonchuay, 2017). Straipsnyje yra pristatomas tyrimas, kurio metu buvo lyginamas „Linux“ „bonding“ ir „teaming“ tvarkyklių našumo KVM virtualizuotoje aplinkoje. Tyrimo stendas buvo sudarytas iš dviejų fizinių serverių, tarpusavyje sujungtų „HP 1810“ tinklo perjungikliu. Kiekvienas fizinis serveris turėjo po tris tinklo sąsajas, iš kurių dvi buvo „Intel 82576EB“ 1 Gbit/s spartos tinklo sąsaja ir „RTL8111“ tinklo sąsaja. Fiziniuose serveriuose buvo įdiegta „Centos“ 7.2 OS ir įdiegtas KVM virtualizacijos dėklas, kurį sudarė „Quemu“ 1.5.3 ir „Libvirt“ 1.2.8 programiniai paketai. Tinklo pralaidumas buvo matuojamas dešimt kartų po keturiadešimt sekundžių „Linux“ „iperf3“ įrankiu. Buvo tiriami trys atvejai. Pirmuoju atveju buvo naudojama viena „iperf“ serverio virtualioji mašina ir trys kliento virtualiosios mašinos, antruoju ir trečiuoju atveju buvo naudojamos dvi „iperf“ serverio virtualiosios mašinos ir trys klientų virtualiosios mašinos, tačiau antruoju atveju nebuvo taikomi srauto apribojimai pagal nustatytas QoS sąlygas o trečiuoju atveju buvo nurodyta, kad pirmasis „iperf“ serveris turi užimti 80 % esamo pralaidumo, o antrasis likusius 20 % pralaidumo. Taip pat kiekvienu atveju buvo tiriama, kaip keičiasi suminis kanalo pralaidumas didinant virtualiosioms mašinoms priskirtų procesoriaus gijų skaičių, priskiriant 3, 45 ir 90 gijų. Šio magistro darbo atveju įdomiausias yra pirmasis tyrimo atvejis, kai buvo naudojama viena „iperf“ serverio virtualioji mašina ir jai taikomas tinklo sąsajų jungimas. Iš pateikiamų rezultatų matyti, kad suminė sparta tiek „bonding“, tiek „teaming“ tvarkyklių atveju yra tokia pati dviejų tinklo sąsajų jungimo

atveju. Pavyzdžiui, naudojant 90 gijų „bonding“ ir „teaming“ tvarkyklių atveju suminė kanalo sparta atitinkamai yra 1840 ir 1874 Mbit/s, o trijų sąsajų jungimo atveju yra 2573 ir 2570 Mbit/s. Verta pastebėti, kad tyrimo rezultatai nėra lyginami su praktine fizinio tinklo sparta, tačiau tikėtina, kad didinant virtualiosioms mašinoms priskirtų gijų skaičių suminė jungtųjų sąsajų sparta artėtų prie maksimalios kanalo spartos.

Be jau aptartųjų „Kubernetes“ CNI įskiepių ir „Linux“ sąsajų jungimo tvarkyklių našumo tyrimų, literatūroje taip pat yra pastebimas akademinės ir verslo bendruomenės susidomėjimas „Kubernetes“ įvairių savybių našumo tyrimais. Pavyzdžiui, viename tyrime (Ferreira ir Sinnott, 2019) yra lyginamas „Kubernetes“ kaip paslaugos sprendimų našumas tinklo ir resursų panaudojimo atžvilgiu AWS, „Azure“ ir GCP paslaugų atveju. Straipsnyje yra lyginamas EKS, AKS ir GKE paslaugų našumas tarpusavyje ir su savarankiškai diegtais „Kubernetes“ telkiniais minėtuose debesijos paslaugų tiekėjų platformose. Taip pat yra lyginamas įvairių konteinerių variklių našumas, tokių kaip „Docker“ ir „rkt“ „Kubernetes“ aplinkoje (Xien, Wang ir Wang, 2017) ar „Kubernetes“ gebėjimas užtikrinti programų pasiekiamumą įvairių „Kubernetes“ komponentų gedimo ar sutrikimo atveju (Vayghan, Saied, Toeroe ir Khendek, 2018) ir taip pat tiriamas „Kubernetes“ kaip orkestratoriaus našumas valdant didelius telkinius (Lei, Liao, Jiang, Yang ir Li, 2019; Medel, Rana, Banares ir Arronategui, 2016).

Išnagrinėjus akademinės ir IT kūrėjų bendruomenės naujausias publikacijas „Kubernetes“ tinklo našumo ir Linux tinklo sąsajų jungimo temomis suformuluotini šie apibendrinimai:

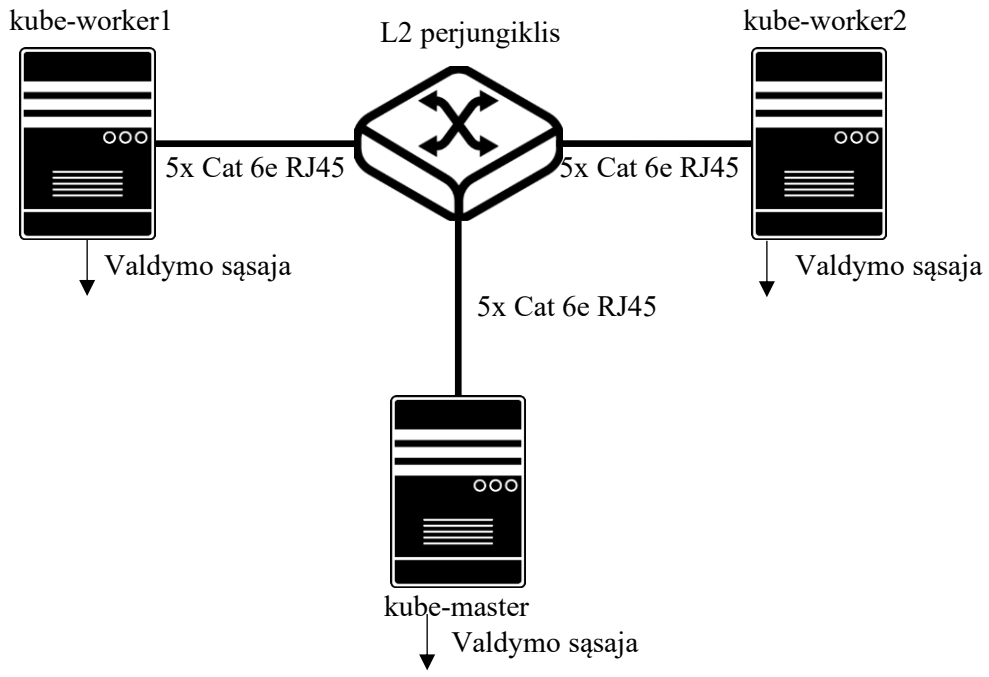
1. „Kubernetes“ konteinerių orkestratorius yra naujas sprendimas, tačiau jau dabar itin plačiai naudojamas ir tiriamas tiek akademinėje, tiek verslo ICT kūrėjų bendruomenėje. Tikėtina, kad per ateinančius kelerius metus ICT bendruomenės susidomėjimas ir poreikis tirti „Kubernetes“ savybes augs.
2. „Kubernetes“ CNI įskiepių našumas yra neblogai išnagrinėta tema, tačiau esami tyrimai nenagrinėjo įvairių paketų dydžio įtakos CNI įskiepių našumui, taip pat daugelio tyrimų atveju iki galo nėra aiški tyrimų metodika, galutiniam rezultatui galinti turėti svarios įtakos.
3. Literatūroje nebuvo aptikta publikacijų, nagrinėjančių atskirų CNI įskiepių savybes, nors daugelis CNI įskiepių, tokių kaip „Calico“ ir „Flannel“, turi kelis veikimo režimus, kurie, tikėtina, pasižymi skirtingomis našumo savybėmis.
4. „Linux“ sąsajų jungimo sprendimai taip pat yra gerai išnagrinėti, tačiau nebuvo rasta publikacijų, nagrinėjančių „teamd“ tvarkyklių našumą fizinėje terpėje ir jungiant daugiau nei tris fizines sąsajas „teamd“ tvarkykle.

3. TYRIMO METODIKOS SUDARYMAS

Šiame skyriuje yra pristatoma tyrimui atlikti naudojama techninė ir programinė įranga ir pristatomas tyrimų metu naudojamas matavimų stendas.

3.1. Matavimų stendo kūrimas

„Kubernetes“ CNI įskiepiai yra tiriami privačiame duomenų centre naudojant tris fizinius serverius ir vieną dedikuotą L2 perjungiklį. Du serveriai yra skirti „Kubernetes“ darbiniams mazgams, o vienas serveris „Kubernetes“ valdančiajam mazgui (žr. 3.1 pav.). „Kubernetes“ darbiniai serveriai yra identiškų specifikacijų serveriai, o „Kubernetes“ valdantysis mazgas turi 16 GB daugiau darbinės atminties ir procesorių taktinis dažnis yra šiek tiek mažesnis (žr. 3.1 lentelę). Visi trys serveriai turi tris tinklo plokštes, turinčias po dvi sąsajas – iš viso šešias tinklo sąsajas. Dviejose tinklo plokštėse veikia „Broadcom NetXtreme II BCM5709“ lustas, o vienoje „Intel 82571EB/82571GB“. Šių rūšių tinklo plokščių maksimalus palaikomas pralaidumas yra 1 Gbit/s. Kiekviename serveryje viena tinklo sąsaja yra naudojama serverį prijungti prie valdymo tinklo, per kurį serverius galima valdyti nuotoliniu būdu. Kitos penkios kiekvieno serverio tinklo sąsajos yra prijungtos prie „HP ProCurve 2810 24G“ perjungiklio naudojant Cat 6e kategorijos laidus. Šie fiziniai sujungimai yra naudojami „Kubernetes“ duomenų tinklui sukurti. Serverių sąsajų MAC adresų ir perjungiklio prievadų numerių jungimo išsidėstymas yra pateiktas 3.2 lentelėje. Abi tinklo plokštės prie serverio yra prijungtos PCIe v1.0a sąsaja. Visų tinklo sąsajų maksimali duomenų perdavimo tinklu sparta yra 1 Gbit/s.



3.1 pav. Matavimų stendo blokinė diagrama

3.1 lentelė. Tyrime naudojamų fizinių serverių charakteristikos

Parametras	kube-master	kube-worker1 / kube-vorker2
Modelis	HP Proliant DL380 G3	
CPU	8 x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz	8 x Intel(R) Xeon(R) CPU E5606 @ 2.13GHz
RAM	32 GB	16GB
Disko tipas / talpa	HDD / 500GB	

3.2 lentelė. Serverių sąsajų ir perjungiklio prievadų jungimo išsidėstymas

Serverio vardas	Valdymo sąsajos IP adresas	Sąsajos vardas	Sąsajos MAC adresas	Perjungtuvo prievado numeris	Lustas
kube-worker1	10.128.16.1	enp3s0f0	2c:76:8a:a9:8b:92	-	,,Broadcom“
		enp3s0f1	2c:76:8a:a9:8b:94	1	
		enp4s0f0	2c:76:8a:a9:8b:96	2	
		enp4s0f1	2c:76:8a:a9:8b:98	3	
		ens2f0	00:26:55:db:e7:fe	16	,,Intel“
		ens2f1	00:26:55:db:e7:ff	14	
kube-master	10.128.16.2	enp3s0f0	00:9c:02:a1:32:6c	-	,,Broadcom“
		enp3s0f1	00:9c:02:a1:32:6e	4	
		enp4s0f0	00:9c:02:a1:32:70	5	
		enp4s0f1	00:9c:02:a1:32:72	6	
		ens2f0	00:26:55:db:f5:3e	18	,,Intel“
		ens2f1	00:26:55:db:f5:3f	22	
kube-worker2	10.128.16.3	enp3s0f0	68:b5:99:71:67:46	-	,,Broadcom“
		enp3s0f1	68:b5:99:71:67:48	7	
		enp4s0f0	68:b5:99:71:67:4a	10	
		enp4s0f1	68:b5:99:71:67:4c	8	
		ens2f0	e8:39:35:14:a0:82	20	,,Intel“
		ens2f1	e8:39:35:14:a0:83	24	

3.2. Tyrimo metu naudojama programinė įranga

3.2.1. Operacinė sistema ir programiniai paketai

Visuose tyrimo metu naudojamuose serveriuose buvo įdiegta ,,CentoOS 7.7.1908“ operacinė sistema ir ,,Chef Workstation 0.11.21“ IaaC kūrimo įrankių rinkinys. ,,Chef“ buvo naudojamas sukurti scenarijams (angl. *scripts*), kurie automatizuoja programinių paketų įrašymą, serverių vardų nustatymą, prideda ,,Bash“ profilį ir SSH viešuosius raktus, kad prie serverių būtų galima prisijungti nuotoliniu būdu naudojant SSH raktus. Visi sukurti „Chef“ skriptai yra įkelti į viešąją ,,GitHub“ repozitoriją (Kapočius, 2020), o naudojami programiniai paketai nurodyti 3.3 lentelėje. ,,Chef“ scenarijus sudaro numatytasis ir kiekvienam serveriui skirtas ,,Ruby“ scenarijus. Numatytajame scenarijuje yra aprašyti veiksmai, programiniai paketai ir failai, kurie yra įrašomi visuose trijuose

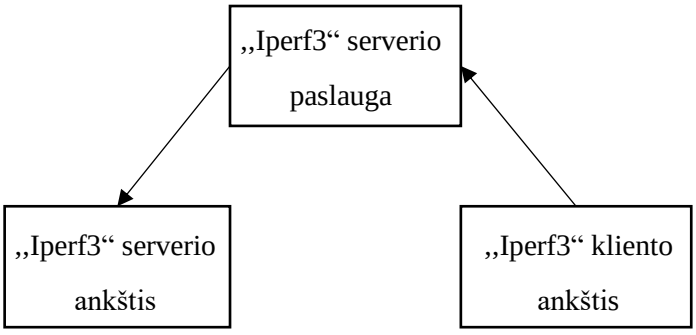
serveriuose. Kiekvieno serverio specifinis kodas, kaip pavyzdžiui, serverio vardo nustatymas, yra aprašytas atskirame, kiekvienam serveriui skirtame „Ruby“ scenarijuje.

3.3 lentelė. Tyrime naudojami ir tyrimo stendo serveriuose įdiegti programiniai paketai

Programinis paketas	**Paskirtis**
vim	Teksto redaktorius
binutils	Dvejetainių įrankių paketas
pciutils	Įrankių rinkinys, skirtas dirbti su PCI magistrale
tree	Medžio tipo direktorijų atvaizdavimas
git	Paskirstyta išeities kodo versijų kontrolės sistema
dstat	Disko ir tinklo I/O stebėjimo įrankis
iperf3	Tinklo našumo tyrimo įrankis
sysstat	Linux našumo stebėjimo įrankių paketas. Darbe naudojamas *sar* įrankis
tcpdump	Tinklo paketų analizatorius
nc	Tinklo įrankis, skirtas skaityti ir rašyti į TCP ar UDP sujungimus

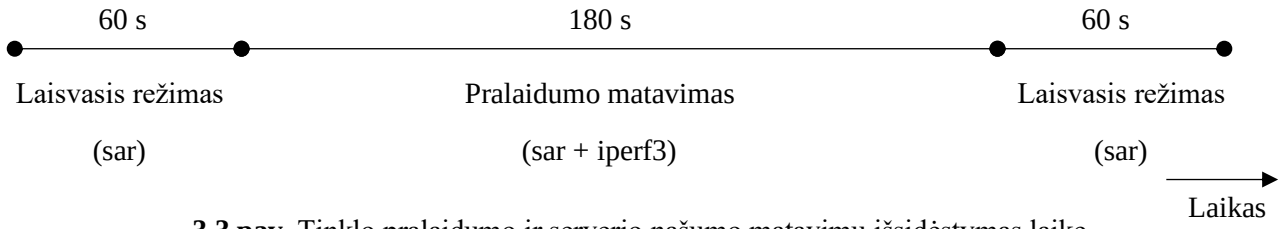
3.2.2. Pralaidumo ir našumo matavimo įrankiai

Tinklo pralaidumui įvertinti naudojamas ,,iperf3“ programinis paketas. Vienas ,,Kubernetes“ darbinis mazgas visada yra parinktas kaip ,,iperf3“ klientas, o kitas ,,Kubernetes“ mazgas kaip ,,iperf3“ serveris. „Kubernetes“ atveju viename darbiniame mazge visada veikia ,,iperf3“ serverio ankštis, o kitame darbiniame mazge veikia ,,iperf3“ kliento ankštis. „Kubernetes“ ankščių ir paslaugos konfigūraciniai YAML failai yra pateikti A priede. Pagal ,,iperf3“ numatytuosius nustatymus klientas visada įkelia duomenis, o serveris juos atsisiunčia iš kliento (žr. ,,iperf3“ dokumentaciją). „Iperf3“ kliento ankštis „iperf3“ serverio ankštį pasiekia naudodamas „iperf3” serverio ,,Kubernetes“ paslaugą (žr. 3.2 pav.).



3.2 pav. ,,Iperf3“ serverio ir kliento ryšio loginė diagrama

Mazgo našumo statistiniai CPU, RAM pertraukčių dažnumo ir tinklo pralaidumo duomenys kas sekundę yra matuojami su ,,sar“ ,,Linux“ įrankiu. Norint įvertinti serverio OS naudojamus resursus, serveriui veikiant laisvuoju režimu, prieš ir po kiekvieno pralaidumo matavimo taip pat yra matuojami serverio našumo statistiniai duomenys. Nuspręsta, kad laisvasis serverio režimas užtrunka 60 s, o pralaidumo matavimas 180 s (žr. 3.3 pav.).



3.3 pav. Tinklo pralaidumo ir serverio našumo matavimų išsidėstymas laike

Siekiant automatizuoti pralaidumo ir serverių našumo matavimo procesą buvo sukurti ,,Bash“ scenarijai ,,iperf3“ serverį ir klientą leidžiantiems mazgams. Scenarijai paremti 3.3 pav. pavaizduotu matavimų išsidėstymu laike. Scenarijų išeities kodai yra įkelti į viešąją ,,GitHub“ repozitoriją (Kapočius, 2020) ir pateikti B ir C prieduose. ,,Iperf3“ serverį paleidžiančiam scenarijui kaip argumentą reikia paduoti norimos naudoti tinklo sąsajos IPv4 adresą. ,,Iperf3“ klientą paleidžiančiam scenarijui taip pat reikia nurodyti norimos naudoti tinklo sąsajos IPv4 adresą ir ,,iperf3“ serverio IPv4 adresą. Tiek serverio, tiek kliento scenarijai nurodytoje direktorijoje išsaugo ,,sar“ raportų rezultatus ir ,,iperf3“ rezultatus.

„Sar“ raportų rezultatai yra saugomi dvejetainiu formatu ir gali būti nuskaitomi naudojantis „sadf“ įrankio komandine eilute. ,,Sadf“ įrankis ,,sar“ rezultatus gali pavaizduoti įvairiais formatais, tokiais kaip CSV, XML ar tekstiniu formatu, kurį patogiausia apdoroti naudojant ,,Linux“ ,,awk“ įrankį. Taip pat ,,sadf“ įrankį patogu naudoti rašant rezultatų apdorojimo ,,Bash“ ar ,,Python“ scenarijus, todėl ir nuspręsta naudoti „sadf“ įrankį. Rezultatų apdorojimo ir „Chef“ „Linux“ tvarkaraščių kūrimo scenarijai yra pateikti D ir E prieduose.

3.2.3. ,,Kubernetes“ ir „Docker“ nustatymai

,,Kubernetes“ telkinys anksčiau apžvelgtame tyrimo stende buvo įdiegtas naudojantis ,,kubeadm“ programiniu paketu ir sekant instrukcijas, esančias oficialiame ,,Kubernetes“ tinklalapyje (žr. ,,Kubernetes“ dokumentaciją). Prieš diegiant ,,Kubernetes“ telkinį buvo įsitikinta, kad matavimų stendas atitinka ,,Kubernetes“ aplinkai keliamus reikalavimus (žr. 3.4 lentelę).

Kad būtų įdiegtas ,,Kubernetes“ telkinys visuose stendo serveriuose buvo išjungtas SWAP, įdiegta ,,Docker-CE“ 19.03.4 konteinerių vykdymo aplinka, „kubeadm“ 1.17.3 programinis paketas ir „kubelet“ 1.17 paprogramė. Atlikus šiuos veiksmus ,,Kubernetes“ valdančiajame mazge komanda „kubeadm init“ buvo inicializuotas „Kubernetes“ telkinys. Inicializavus telkinį buvo sukuriamas

„Kubernetes“ tinklo diegimas. Kiekvienam CNI įskiepiui žingsniai, kuriuos reikia atlikti, gali skirtis ir yra aprašyti „Kubernetes“ oficialioje dokumentacijoje (žr. įskiepių diegimo instrukcijos). Įdiegus CNI įskiepį kiekviename „Kubernetes“ darbiniame mazge yra paleidžiama komanda „kubeadm join“, kuriai yra nurodomas „Kubernetes“ valdančiojo serverio IPv4 adresas, identifikacijos ir autentifikacijos žetonai, ir tokiu būdu prie „Kubernetes“ telkinio yra pridedami darbiniai mazgai.

3.4 lentelė. „Kubernetes“ telkiniui keliami techniniai reikalavimai

Reikalavimai	
OS	„CentOS“ 7
Atmintis	Min. 2 GB RAM
	Išjungtas SWAP
CPU	Min. 2 CPU
Tinklas	Visiškas sujungimas tarp telkinio serverių
	Unikalūs sąsajų MAC adresai ir „product_uuid“
	„Firewalld“ išjungtas arba atidaryti tam tikri prievadai

3.2.4. Tinklo sąsajų jungimo nustatymai

Tinklo sąsajų jungimas buvo įgyvendintas „Linux“ „teamd“ tvarkyklėmis. Tyrimo metu buvo pasirinkta naudoti „teamd“ tvarkykles vietoje senesnių „bonding“ tvarkyklių. Yra teigiama, kad „teamd“ sąsajų jungimo tvarkyklės pasižymi tokiu pat geru našumu kaip ir „bonding“ tvarkyklės, tačiau suteikia daugiau lankstumo konfigūravimui ir sąsajų stebėjimui. Taip pat „teamd“ tvarkyklės turi galimybę išsamiai konfigūruoti paketų maišos funkcijas, kurios yra naudojamos srautų tolygiam paskirstymui (Pirko, 2014).

Sąsajų jungimas buvo sukonfigūruotas naudojant „Linux“ tinklo sąsajų scenarijus, saugomus „/etc/sysconfig/network-scripts“ direktorijoje. Fizinės sąsajos naudotuose serveriuose buvo jungiamos į vieną loginę tinklo sąsają, pavadinimu „team“. Loginės tinklo sąsajos konfigūravimo scenarijus yra pateiktas 3.4 pav., o vienos iš fizinių sąsajų konfigūravimo scenarijus 3.5 pav.

Tyrimo metu buvo naudojamas aktyvus „loadbalance“ „teamd“ tvarkyklių režimas, kuris pagal paketų maišos funkcijas, atsižvelgiant į esamą fizinių tinklo sąsajų apkrovimą, paskirsto paketus. Tyrimo metu buvo išbandytos kelios išsiunčiamų paketų maišos funkcijos konfigūracijos į maišos skaičiavimą įtraukiant:

- VLAN žymas ir siuntėjo ir gavėjo L4 protokolo prievadų numerius;
- Siuntėjo ir gavėjo L4 protokolo prievadų numerius ir IP adresus;
- Siuntėjo ir gavėjo MAC ir IP adresus;
- Siuntėjo ir gavėjo MAC adresus ir L4 protokolo prievadų numerius.

Išsiaiškinta, kad didžiausia suminė sparta, naudojant kelias TCP sesijas tiriamo srauto atveju, yra pasiekiama į maišos funkciją įtraukiant VLAN žymos informaciją ir TCP siuntėjo ir gavėjo prievadų numerius. Taip nustatyta, kad taikant „teamd“ sąsajų jungimą vienos TCP sesijos maksimali duomenų perdavimo sparta neviršija vienos fizinės sąsajos teorinės spartos. Todėl tolesniuose magistro darbo skyriuose apžvelgiamų tyrimų metu yra nagrinėjamas suminis kanalo pralaidumas, kai buvo naudojamos kelios TCP sesijos. Pavyzdžiui, dviejų fizinių tinklo sąsajų jungimo atveju buvo naudojami du „iperf3“ serveriai ir klientai. Viename fiziniame serveryje veikė du „iperf3“ serveriai, o kitame fiziniame serveryje du „iperf3“ klientai.

```
DEVICE=team
DEVICETYPE=Team
ONBOOT=yes
BOOTPROTO=none
NM_CONTROLLED=no
IPADDR=172.16.0.2
PREFIX=24
TEAM_CONFIG='{"runner": {"name": "loadbalance", "tx_hash": ["vlan", "l4"],
"tx_balancer": {"name": "basic"}}}'
```

3.4 pav. „Team“ loginės tinklo sąsajos konfigūravimo scenarijaus išeities kodas

```
DEVICE=ens2f0
DEVICETYPE=TeamPort
ONBOOT=yes
TEAM_MASTER=team
NM_CONTROLLED=no
TEAM_PORT_CONFIG='{"prio": 100}'
```

3.5 pav. Fizinės sąsajos pridėjimo prie loginės tinklo sąsajos „team“ konfigūravimo scenarijaus išeities kodas

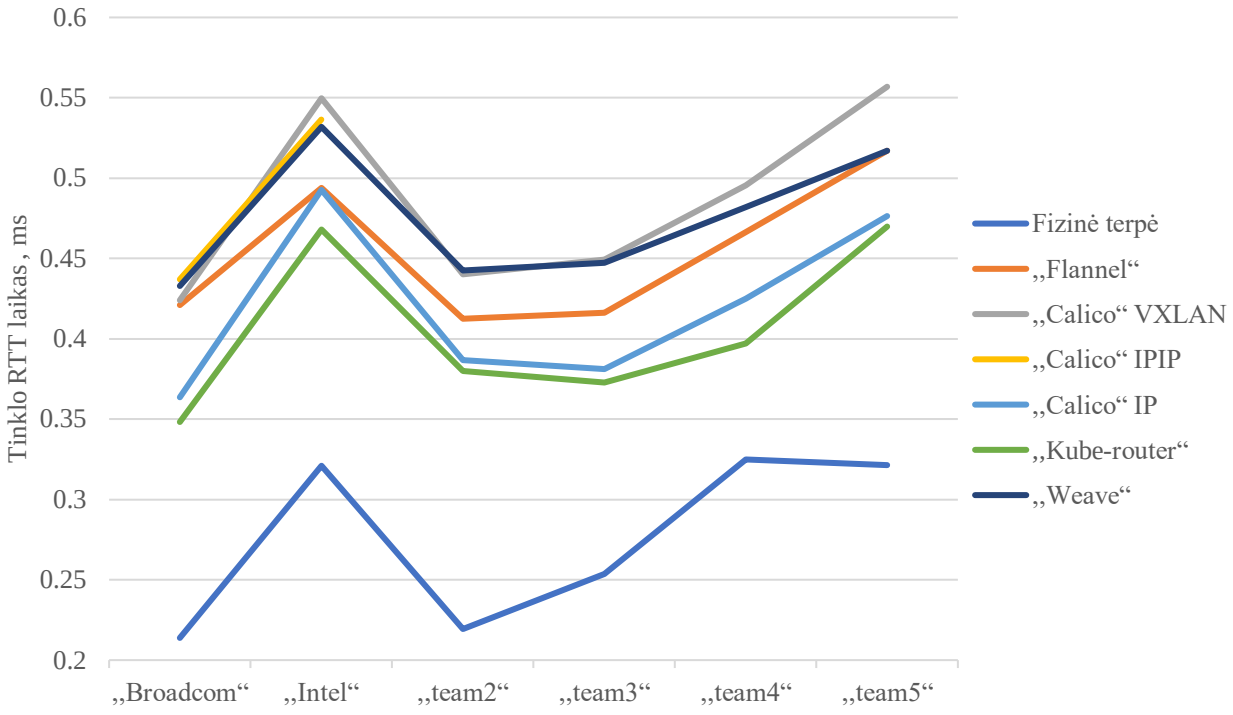
4. „KUBERNETES“ TINKLO ĮSKIEPIŲ TYRIMAS

Šiame skyriuje yra apžvelgiami CNCF keturių rekomenduojamų „Kubernetes“ tinklo įskiepių: „Flannel“, „Calico“, „Kube-router“ ir „Weave“ našumo tyrimo rezultatai. „Calico“ atveju buvo ištirti visi trys šio įskiepio veikimo režimai: IPIP, VXLAN ir įprastasis IP režimas. Visi minėti CNI įskiepiai buvo tiriami pagal keturis scenarijus. Pirmuoju atveju buvo lyginamas „ping“ ICMP paketų vidutinis laikas RTT, kurį paketai užtrunka tinkle. Antrasis scenarijus buvo skirtas palyginti CNI įskiepių našumą, kai visi tyrimo stendo serveriai naudojo vieną fizinę sąsają. Trečiuoju scenarijumi buvo tiriamas suminis vidutinis TCP protokolo pralaidumas, kai visuose serveriuose buvo jungiama nuo dviejų iki penkių tinklo sąsajų. Ketvirtuoju atveju CNI įskiepių našumas buvo tiriamas taip pat naudojant vieną fizinę sąsają, tačiau išjungus visų serverių tinklo sąsajų TCP išsiuntimo / priėmimo GSO mechanizmus. Visų tyrimo metu nagrinėtų scenarijų rezultatai yra lyginami su fizinės terpės rezultatais. Taip pat antrojo–ketvirtojo tyrimų scenarijų atvejais buvo vertinamas ne tik vidutinis ir suminis TCP protokolo pralaidumas, bet ir vidutinis OS kontekstų keitimo skaičius per sekundę ir vidutinė CPU apkrova.

4.1. CNI įskiepių vėlinimo tyrimo rezultatai

Tirtų CNI įskiepių vidutinis „ping“ ICMP paketų vėlinimas tinkle, esant skirtingam skaičiui agreguotų tinklo sąsajų, yra pateikiamas 5.1 pav. Iš pateiktų rezultatų matyti, kad „Intel“ tinklo sąsajos pasižymi ~0,1 ms mažesniu vėlinimu nei taip pat tyrimo metu naudotos „Broadcom“ tinklo sąsajos. Taip pat visi CNI įskiepiai prideda bent ~0,2 ms papildomą vėlinimą lyginant su fiziniu tinklu. Kita vertus, IP CNI sprendimų kaip „Kube-router“ ir „Calico IP“ pridedamas vėlinimas yra pastebimai mažesnis nei užklotojo tinklo sprendimų. Toks įskiepių pridedamo vėlinimo skirtumas gali būti paaiškintas tuo, kad užklotojo tinklo sprendimai pasižymi didesniu skaičiavimo perviršiu, kurį lemia papildomos VXLAN ar IPIP antraštės. Taip pat įdomu, tai, kad „Kube-router“ CNI įskiepio vidutinis RTT yra ~1–5 % mažesnis lyginant su „Calico IP“ RTT. Iš užklotojo tinklo sprendimų mažiausiu vėlinimu pasižymėjo „Flannel“ įskiepis, aplenkęs „Calico“ VXLAN ir „Weave“ CNI įskiepius, nors visi šie įskiepiai ir naudoja tą pačią užklotojo tinklo technologiją.

Be to, vėlinimo tyrimo rezultatai rodo, kad didėjant jungtųjų tinklo sąsajų skaičiui vidutinis kanalo vėlinimas taip pat didėja. Tai gali būti paaiškinta tuo, kad kuo daugiau tinklo sąsajų yra jungiama, tuo „teamd“ sąsajų jungimo tvarkyklės „loadbalance“ algoritmas bando tolygiau paskirstyti paketus tarp jungtųjų tinklo sąsajų. Todėl kelias, kuriuo keliauja paketai, yra keičiamas dažnai ir dėl to yra pastebimas padidėjęs kanalo vėlinimas.



4.1 pav. CNI įskiepių vidutinis „ping“ ICMP paketų RTT laikas, esant skirtingam jungtųjų tinklo sąsajų skaičiui, kur „team2“–„team5“ yra jungiamos 2–5 fizinėmis tinklo sąsajomis

4.2. CNI įskiepių našumo tyrimo rezultatai naudojant vieną fizinę tinklo sąsają

4.2.1. TCP protokolo vidutinio pralaidumo tyrimo rezultatai

CNI įskiepių vidutinio TCP protokolo pralaidumo tyrimo rezultatai, esant įvairiems MSS dydžiams, kai buvo naudojama viena fizinė „Intel“ tinklo sąsaja, yra pateikti 4.1. lentelėje. „Broadcom“ tinklo sąsajos TCP pralaidumo rezultatai lyginant su „Intel“ tinklo sąsaja yra ~40 % blogesni esant mažiausiam 76 B MSS paketų dydžiui, o visų didesnių paketų atveju rezultatai yra prastesni nuo ~0,2 % iki ~3,2 % (žr. 4.2. lentelę). Iš pateiktų CNI įskiepių TCP protokolo pralaidumo rezultatų (žr. 4.1. lentelę) matyti, kad IP CNI įskiepių „Calico IP“ ir „Kube-router“ pralaidumas visame tirtame MSS ruože beveik lygus fizinės terpės pralaidumui. Verta pastebėti, kad „Calico“ VXLAN režimo vidutinis TCP protokolo pralaidumas iš visų tirtų CNI įskiepių buvo mažiausias, esant mažiems paketų dydžiams. Iš užklotojo tinklo sprendimų „Flannel“ vidutinis pralaidumas buvo geriausias visame tirtame MSS ruože. Kita vertus, esant didesniems nei 1448 B paketams visų užklotojo tinklo sprendimų, TCP protokolo pralaidumas buvo vienodas. Todėl galima teigti, kad paketų dydžiams artėjant prie „Jumbo“ kadrų dydžio, skirtumas tarp CNI sprendimų pasiekiamo vidutinio TCP protokolo našumo ir fizinės terpės našumo sparčiai mažėja ir naudojant „Jumbo“ kadrus skirtumas yra toks mažas, kad jo galima nepaisyti. Tačiau „Jumbo“ kadrų naudojimas nėra visada tinkamas sprendimas, kadangi didžioji dalis Internete naudojamų HTTP protokolu paremtų programų naudoja 460–1448 B dydžio IP paketus (Center for Applied Internet Data Analysis, 2008)

4.1 lentelė. CNI įskiepių TCP protokolo vidutinis pralaidumas, esant įvairiems MSS dydžiams, kai yra naudojama viena „Intel“ tinklo sąsaja, kur raudona spalva pažymėti blogiausi stebėti pralaidumo rezultatai, žalia – geriausi

Tinklo realizavimas	MSS dydis, B								
	76	**204**	**460**	**972**	**1448**	**1996**	**4044**	**8140**	**8948**
Fizinė terpė	457	694	836	915	941	957	978	989	990
„Calico IP“	458	694	836	915	939	957	978	989	990
„Calico IPIP“	326	649	807	898	927	948	974	987	988
„Calico VXLAN“	99	253	595	874	907	935	967	983	985
„Flannel“	159	449	767	874	909	935	967	983	985
„Kube-router“	458	694	836	915	942	957	978	989	990
„Weave“	122	346	766	874	904	935	967	983	985

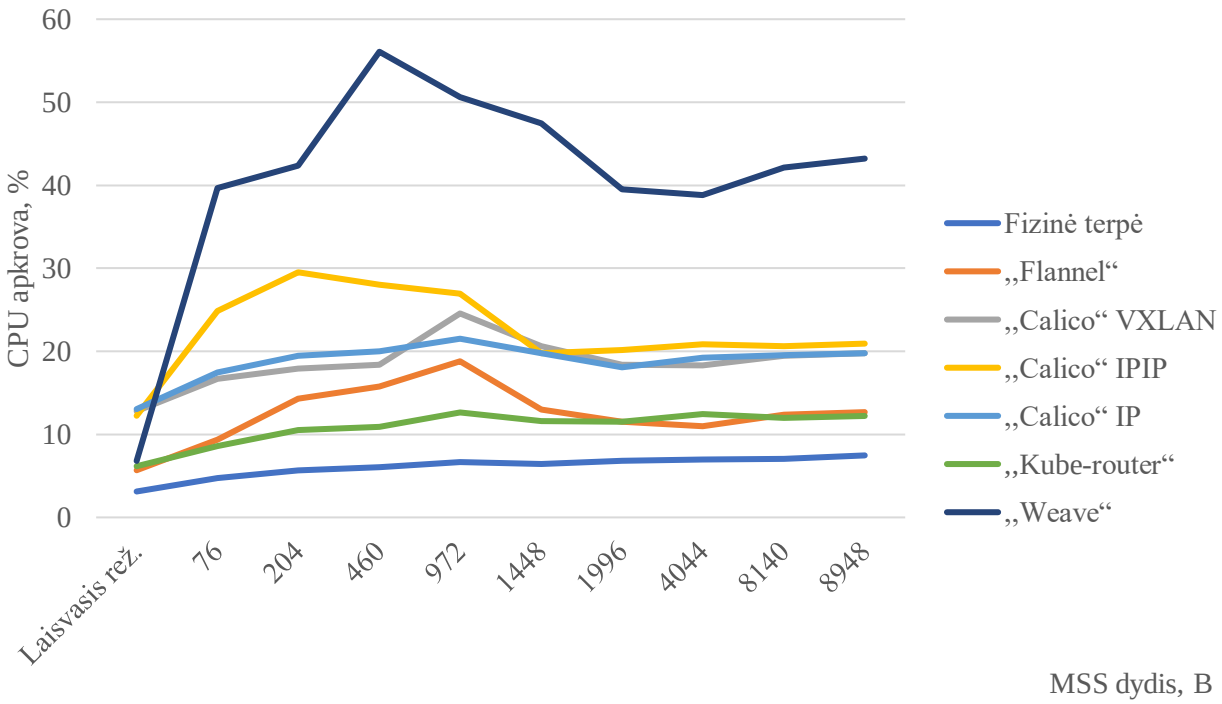
4.2 lentelė. „Intel“ ir „Broadcom“ fizinių sąsajų TCP protokolo vidutinio pralaidumo palyginimas, esant skirtingiems paketų MSS dydžiams

„Ethernet“ lustas	MSS dydis, B								
	76	**204**	**460**	**972**	**1448**	**1996**	**4044**	**8140**	**8948**
„Broadcom“, Mb/s	274	671,4	827,2	910,2	935,4	955	969,2	960,2	970
„Intel“, Mb/s	457	693,4	836	915	941	957	978	989	990
„Broadcom atsilikimas“, %	40,04	3,17	1,05	0,52	0,60	0,21	0,90	2,91	2,02

4.2.2. Vidutinės CPU apkrovos tyrimo rezultatai

Apibendrinti CNI įskiepių vidutinės vieno CPU apkrovos rezultatai, esant skirtingiems MSS dydžiams, kai yra naudojama viena fizinė „Intel“ sąsaja, yra pateikti 4.2 pav. CPU apkrovos tyrimo metu CPU apkrova buvo matuota iperf3“ klientą leidžiančiame mazge. CPU apkrovos tyrimo rezultatai rodo, kad visų tirtų CNI įskiepių ir fizinės terpės atveju CPU apkrova auga iki 972 B paketų dydžio, o po to laipsniškai mažėja. Tokią CPU apkrovos pokyčio tendenciją galima paaiškinti tuo, kad didesnių nei 972 B paketų atveju TCP GSO mechanizmas pradeda duoti matomų rezultatų ir efektyviau sumažina CPU apkrovą nei esant mažam paketų dydžiui. Taip pat matoma, kad mažiausią CPU apkrovą visų tirtų MSS dydžių atveju lėmė „Kube-router“ CNI įskiepis. „Kube-router“ sukurta CPU apkrova buvo nuo 64 % iki 98 % didesnė nei fizinės terpės tyrimo metu stebėta CPU apkrova. Didžiausią CPU apkrovą visame tirtame MSS ruože sukūrė „Weave“ CNI įskiepis. „Weave“ sukurta CPU apkrova buvo nuo 118 % iki 832 % didesnė nei fizinės terpės tyrimo atveju. „Flannel“ lemiama CPU apkrova visame tirtame MSS ruože yra labai artima „Kube-router“ CPU apkrovai, nors ir „Flannel“ CNI įskiepis yra paremtas užklotojo tinklo sprendimų VXLAN technologija, o „Kube-router“ CNI įskiepis priskiriamas prie IP sprendimų. Taip pat įdomu tai, kad „Calico“ visų tirtų

režimų atveju CPU apkrova yra panaši, tačiau galima išskirti „Calico IPIP“ režimą, kurio sukurta CPU apkrova buvo pastebimai didesnė paketams, mažesniems nei 972 B. Kita vertus, „Calico VXLAN“ ir „Calico IP“ režimų sukuriama CPU apkrova yra labai artima, nors šiuose sprendimuose yra naudojimas visiškai skirtingos tinklo technologijos. Todėl galima teigti, kad CNI įskiepių atveju CPU apkrovą ne visada lemia naudojama tinklo technologija, o paties CNI sprendimo programinio kodo efektyvumas.



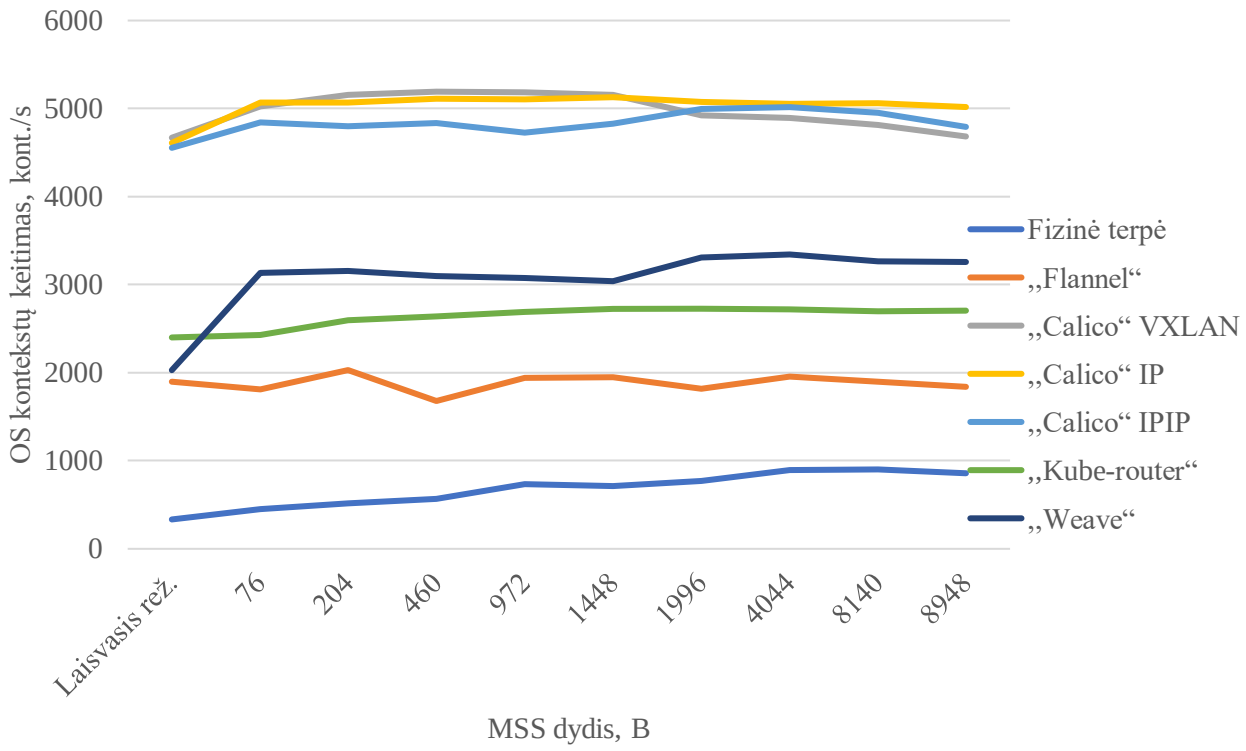
4.2 pav. CNI įskiepių vidutinė vieno CPU apkrova, esant įvairiems MSS dydžiams, kai naudojama viena „Intel“ tinklo sąsaja

4.2.3. Vidutinio OS kontekstų keitimo tyrimo rezultatai

Kontekstų keitimas (angl. *context switching*) yra procesas, kurio metu yra išsaugojama kompiuterio proceso ar gijos būsena tam, kad vėliau procesas ar gija galėtų būti atstatyti ir būtų pratęstas jų vykdymas. Tai leidžia keliems procesams dalintis vieno CPU resursais, o tai yra esminis daugiaprocesės OS reikalavimas. Dažniausiai kontekstų keitimas vyksta, kai keli procesai yra vykdomi vienu metu arba kai yra gaunamos pertrauktys. Pertrauktys dažniausiai yra gaunamos, kada procesoriui yra būtina nuskaityti duomenis iš disko, tinklo plokštės ar pan. Verta pabrėžti, kad kontekstų keitimas reikalauja papildomų skaičiavimo išteklių ir dažnesnis kontekstų keitimas reiškia didesnį sistemos našumo sumažėjimą (Tanenbaum ir Herbert, 2014).

CNI įskiepių vidutinis OS kontekstų kitimo tyrimo rezultatai, kai buvo naudojama viena „Intel“ fizinė sąsaja yra pateikti 4.3. pav. Visame tirtame MSS ruože visais CNI įskiepių ir fizinės

terpės atvejais vidutinis kontekstų keitimas didėjant paketų dydžiui kinta nežymiai. Laisvojo režimo atveju, kai „Kubernetes“ telkinyje veikia tik sisteminės ankštys, mažiausias vidutinis kontekstų keitimas pastebėtas „Flannel“, „Weave“ ir „Kube-router“ CNI įskiepių atveju. Didžiausias laisvojo režimo vidutinis kontekstų keitimas buvo „Calico“ visų trijų tirtų režimų atveju. „Flannel“ vidutinis kontekstų keitimas taip pat buvo mažiausias visame tirtame MSS ruože ir buvo nuo ~110 % iki 304 % didesnis lyginant su fizinės terpės rezultatais. Verta atkreipti dėmesį į „Weave“ CNI įskiepio rezultatus. Nors „Weave“ vidutinis OS kontekstų keitimas laisvojo režimo metu yra tik ~7 % didesnis nei „Flannel“ atveju, tačiau pralaidumo tyrimo metu „Weave“ vidutinis OS kontekstų keitimas buvo didesnis nuo 56 % iki 85 %. Tai rodo, kad vykdant duomenų siuntimą „Weave“ CNI įskiepis reikalauja pastebimai daugiau procesų ir resursų nei „Flannel“ įskiepis. Įdomu tai, kad „Calico IP“ ir „Kube-router“ CNI įskiepiai TCP protokolo vidutinio pralaidumo tyrimo metu pasiekė pralaidumą, artimą fizinės terpės pralaidumui, tačiau abiejų CNI įskiepių vidutinis OS kontekstų keitimas gerokai skiriasi. „Calico IP“ vidutinis OS kontekstų keitimas buvo nuo 85 % iki 109 % didesnis lyginant su „Kube-router“. Šie rezultatai paaiškina stebėtą didesnę „Calico IP“ CPU apkrovą, aptartą 4.2.2. poskyryje.



4.3 pav. CNI įskiepių vidutinis OS kontekstų keitimas, kai yra naudojama viena „Intel“ tinklo sąsaja

4.3. CNI įskiepių našumo tyrimo naudojant jungtąsias tinklo sąsajas rezultatai

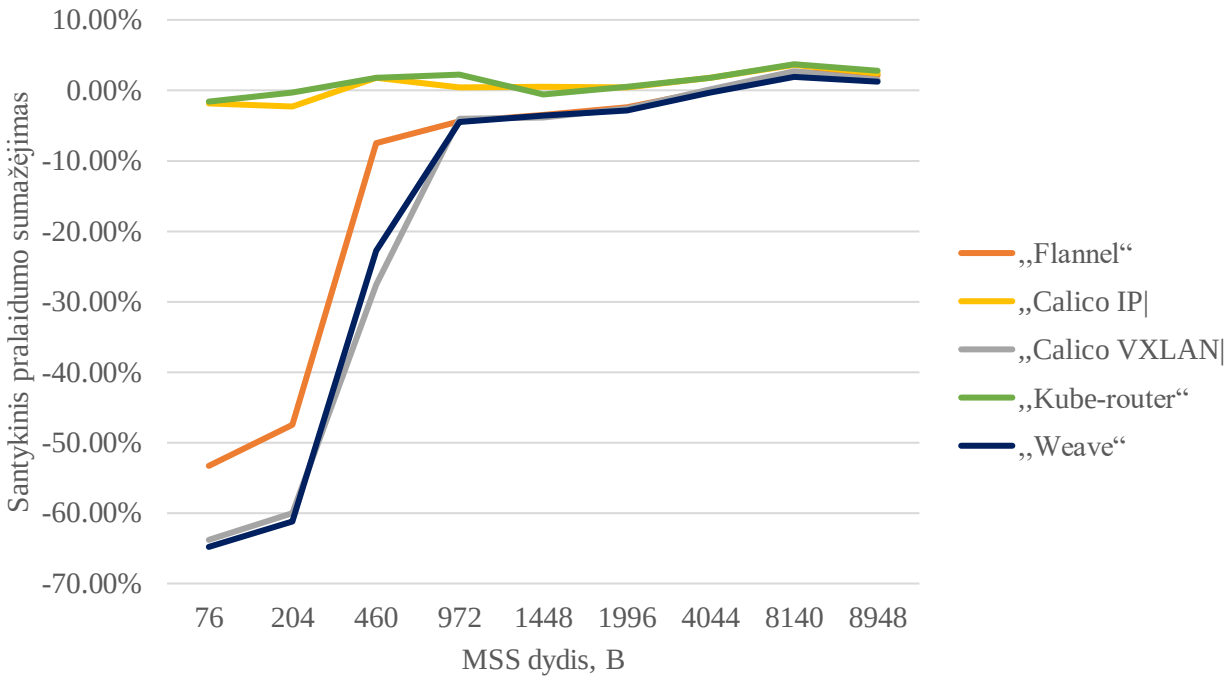
CNI įskiepių našumas taip pat buvo lyginamas jungiant nuo dviejų iki penkių tinklo sąsajų. Tinklo sąsajos buvo jungiamos naudojant „Linux“ „teamd“ tvarkykles. Jungiant dvi ir tris tinklo sąsajas buvo naudojamos tik „Broadcom“ fizinės tinklo sąsajos.

4.3.1. Bendro vidutinio TCP pralaidumo tyrimo rezultatai

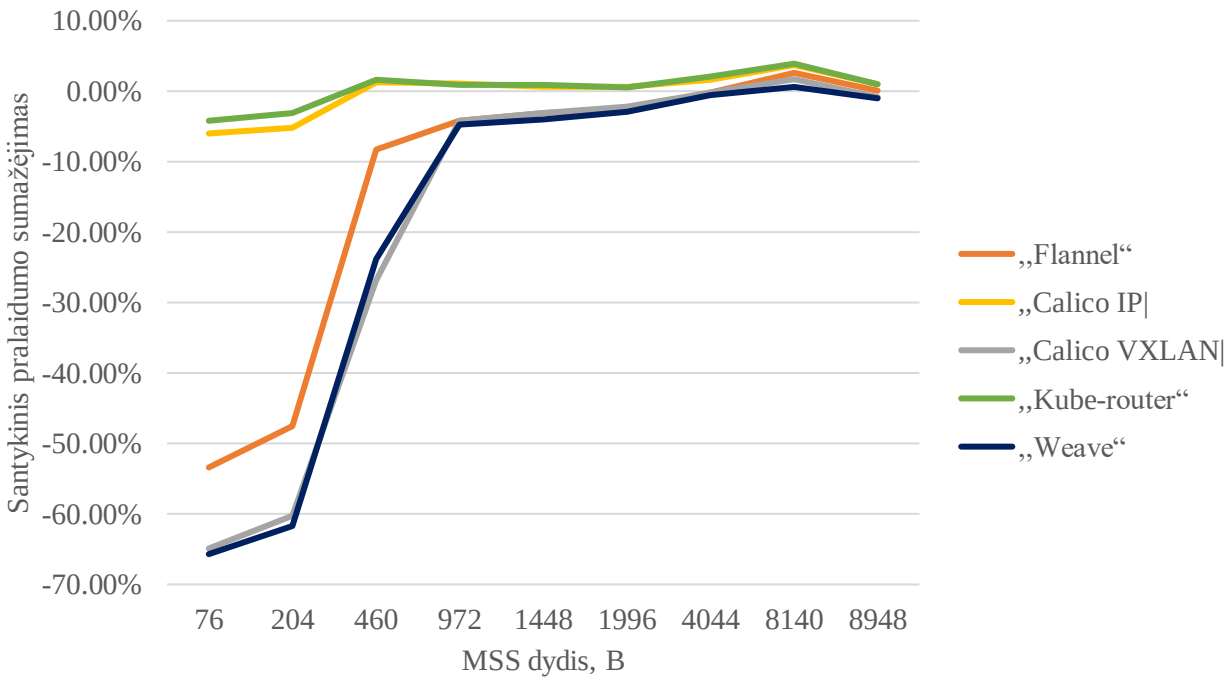
Tyrimų metu CNI įskiepių TCP protokolo bendras vidutinis pralaidumas jungiant nuo dviejų iki penkių tinklo sąsajų buvo lyginamas su fizinės terpės rezultatais. CNI įskiepių bendro vidutinio TCP protokolo santykinio pralaidumo sumažėjimo rezultatai, esant įvairiems MSS dydžiams, lyginant su fizine terpe, yra pateikti 4.4–4.7 pav. Fizinės terpės TCP protokolo vidutinio pralaidumo rezultatai, esant skirtingam jungtųjų tinklo sąsajų skaičiui, pateikti F priede.

Iš pateiktų rezultatų matyti, kad užklotojo tinklo sprendimų „Flannel“, „Calico“ VXLAN / IP ir „Weave“ TCP pralaidumas, esant mažiausiems MSS dydžiams, yra stipriai (nuo ~48 % iki ~73 %) mažesnis lyginant su fizinės terpės rezultatais. Kita vertus, kaip ir vienos tinklo sąsajos atveju, pristatytu 4.2.1 poskyryje, užklotojo tinklo sprendimų TCP protokolo pralaidumas tampa labai artimas IP sprendimų pralaidumui, kai MSS dydis yra didesnis nei 972 B. Verta pastebėti, kad didėjant jungtųjų tinklo sąsajų skaičiui, visais tirtais atvejais, kai MSS dydis buvo mažesnis nei 972 B, santykinis pralaidumo sumažėjimas tik didėjo. Pavyzdžiui, esant 76 B MSS dydžio paketams „Flannel“ CNI įskiepio pralaidumas nuo fizinės terpės atsiliko 53,3 %, kai buvo jungtos dvi tinklo sąsajos, o kai buvo jungtos penkios tinklo sąsajos, „Flannel“ pralaidumas nuo fizinės terpės atsiliko ~60,1 %. IP sprendimų „Calico IP“ ir „Kube-router“ TCP protokolo pralaidumas visame tirtame MSS ruože buvo labai artimas fizinės terpės rezultatams ir nuo 460 B MSS dydžio pralaidumo skirtumas nuo fizinės terpės rezultatų yra toks mažas, kad jo galima nepaisyti.

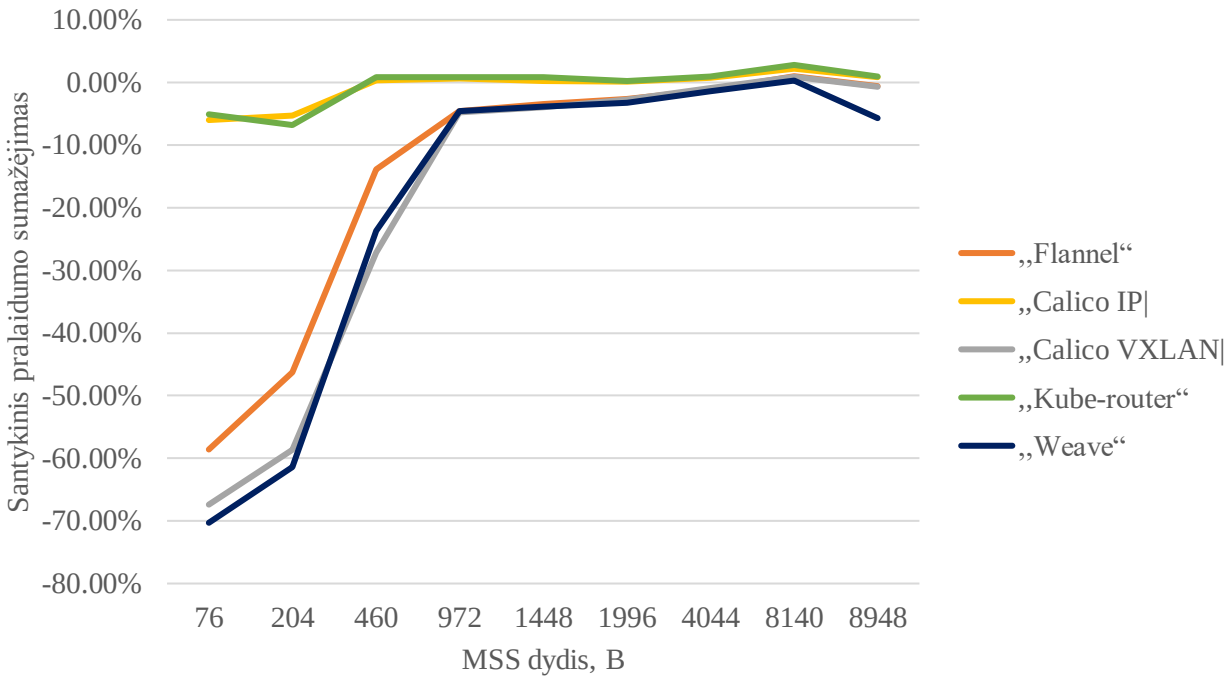
TCP protokolo suminis pralaidumas, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų, yra pateiktas 4.8 pav. Iš pateiktų rezultatų matyti, kad esant 76 B MSS dydžio paketams net ir jungiant penkias tinklo sąsajas tiek išmatuotas fizinės terpės pralaidumas, tiek visų CNI įskiepių pralaidumas neviršija 2 Gbit/s duomenų perdavimo tinklu spartos. Taip pat didėjant jungtųjų sąsajų skaičiui užklotojo tinklo CNI įskiepių pasiekiama suminė duomenų perdavimo sparta vis labiau atsilieka nuo IP sprendimų perdavimo spartos. Kita vertus, esant 1460 ir 8960 B MSS dydžiams TCP protokolo pasiekiama suminė kanalo sparta yra artima teoriniam kanalo pralaidumui visų CNI įskiepių atveju. Didžiausių, 8960 B MSS dydžio, paketų naudojimas padeda pasiekti nuo 0,5 % iki 1 % didesnį suminį kanalo pralaidumą lyginant su 1460 B MSS dydžio maketais visų tirtų CNI įskiepių atveju.



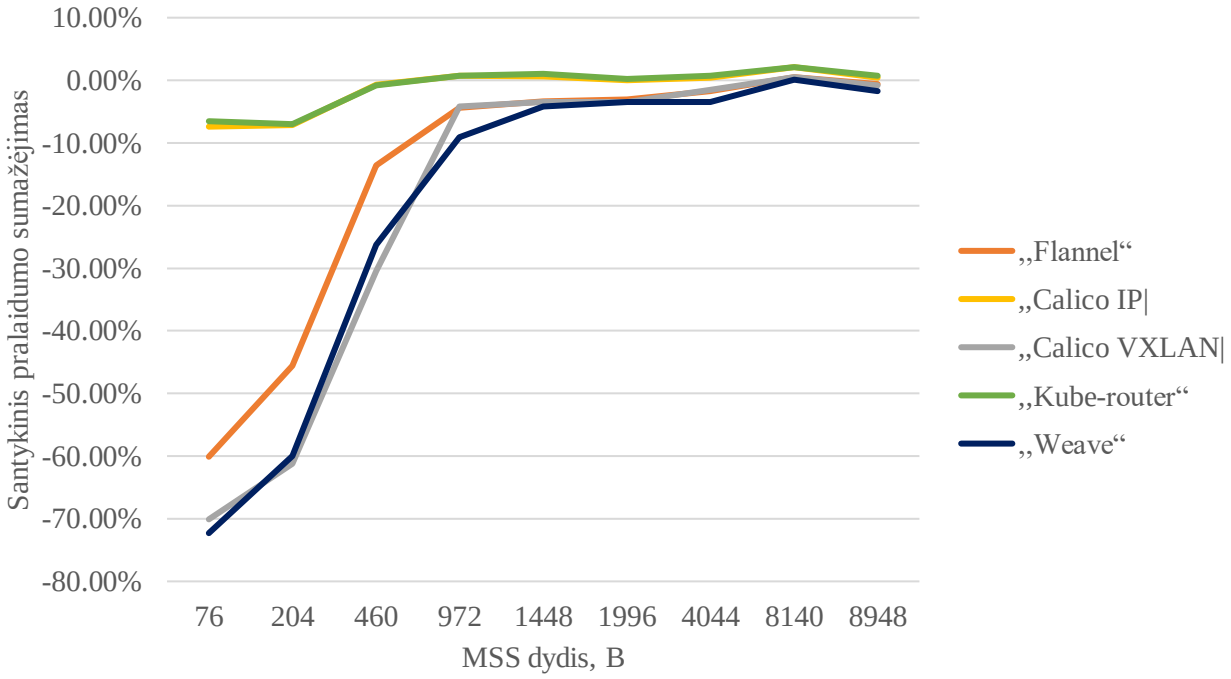
4.4 pav. CNI įskiepių TCP protokolo santykinis suminio pralaidumo sumažėjimas lyginant su fizinės terpės rezultatais, kai yra jungiamos dvi tinklo sąsajos, kurių suminis teorinis pralaidumas yra 2 Gbit/s, o praktiškai išmatuotas fizinis pralaidumas kiekvienam MSS, MSS didėjimo tvarka yra: 548,8; 1353,4; 1642,8; 1816,4; 1876,7; 1904,4; 1920; 1906,6; 1924 Mbit/s



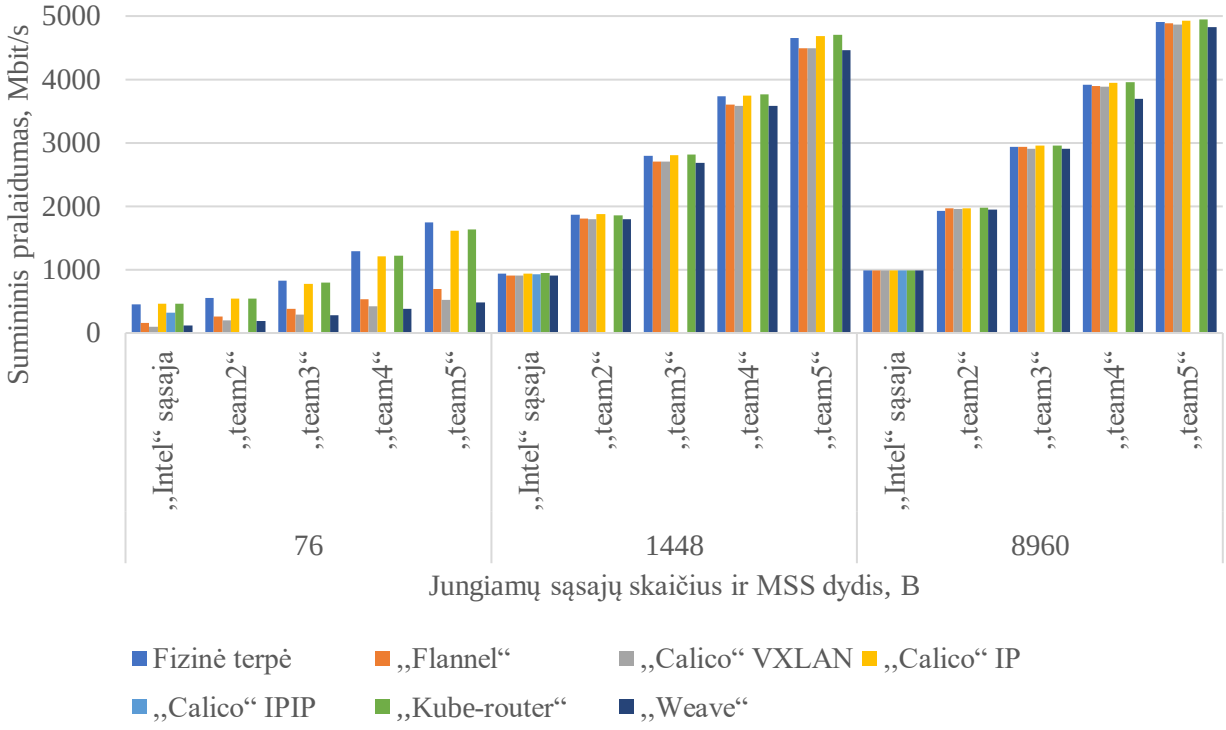
4.5 pav. CNI įskiepių TCP protokolo santykinis suminio pralaidumo sumažėjimas lyginant su fizinės terpės rezultatais, kai yra jungiamos trys tinklo sąsajos, kurių suminis teorinis pralaidumas yra 3 Gbit/s, o praktiškai išmatuotas fizinis pralaidumas kiekvienam MSS, MSS didėjimo tvarka yra: 827,6; 2026,2; 2466,6; 2713,8; 2792,6; 2852,4; 2870,2; 2849,2; 2932 Mbit/s



4.6 pav. CNI įskiepių TCP protokolo santykinis suminio pralaidumo sumažėjimas lyginant su fizinės terpės rezultatais, kai yra jungiamos keturios tinklo sąsajos, kurių suminis teorinis pralaidumas yra 4 Gbit/s, o praktiškai išmatuotas fizinis pralaidumas kiekvienam MSS, MSS didėjimo tvarka yra: 1286,4; 2697,6; 3302,6; 3619,8; 3732,6; 3818,2; 3874,6; 3844; 3914,6 Mbit/s



4.7 pav. CNI įskiepių TCP protokolo santykinis suminio pralaidumo sumažėjimas lyginant su fizinės terpės rezultatais, kai yra jungiamos penkios tinklo sąsajos, kurių suminis teorinis pralaidumas yra 5 Gbit/s, o praktiškai išmatuotas fizinis pralaidumas kiekvienam MSS, MSS didėjimo tvarka yra: 1741,2; 3406,8; 4136,6; 4530,8; 4653,4; 4776,2; 4851,8; 4840,4; 4909,6 Mbit/s

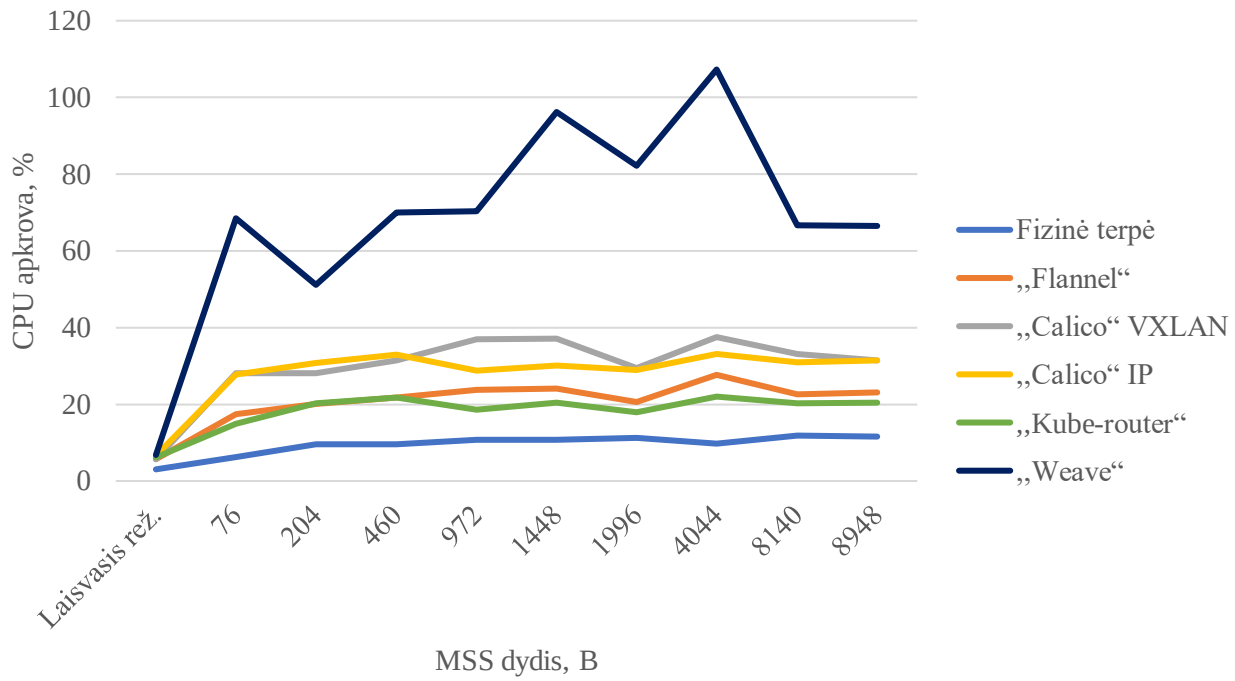


4.8 pav. CNI įskiepių TCP protokolo suminis pralaidumas, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų

4.3.2. CPU apkrovos rezultatai naudojant tinklo sąsajų jungimą

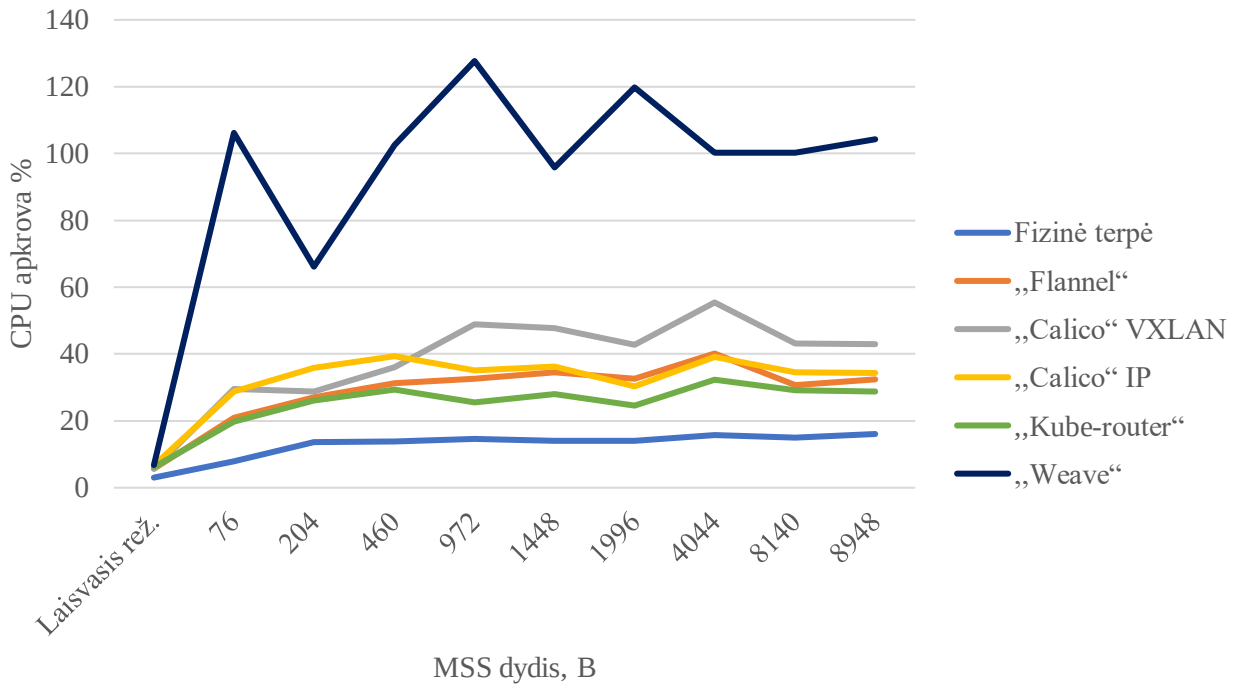
Vidutiniai CPU apkrovos vienam CPU rezultatai naudojant dviejų–penkių fizinių sąsajų jungimą yra pateikti 4.9–4.12 pav. Pateiktuose grafikuose didesnė nei 100 % CPU apkrova reiškia, kad sistema visiškai sunaudoja bent vieno CPU resursus. Pavyzdžiui, 210 % apkrova rodo, kad du fiziniai CPU yra visiškai apkrauti, o trečiasis CPU yra apkrautas 10 %. Kaip ir vienos fizinės sąsajos atveju, mažiausią CPU apkrovą visais tirtais atvejais lėmė „Kube-router“ CNI įskiepis, o didžiausią – „Weave“ įskiepis. Visų tirtų CNI įskiepių atveju, išskyrus „Weave“, net ir jungiant penkias tinklo sąsajas vieno CPU vidutinė apkrova neviršijo 100 %. Tai reiškia, kad teoriškai įmanoma jungti penkias tinklo sąsajas naudojant serverį tik su vienu fiziniu CPU ir nepridedant papildomo pralaidumo degradavimo tirtų CNI įskiepių atveju. Kita vertus, „Weave“ CNI įskiepio atveju, kai yra jungiamos penkios tinklo sąsajos, norint pasiekti maksimalų pralaidumą yra būtina, kad sistema turėtų bent tris fizinius CPU. Iš pateiktų rezultatų taip pat matyti, kad didėjant jungtųjų sąsajų skaičiui išryškėja CPU apkrovos skirtumai tarp „Calico VXLAN“ ir „Calico IP“ CNI įskiepių, kurių nebuvo naudojant tik vieną fizinę tinklo sąsają (žr. 4.2 pav.). Nuo trijų jungtųjų tinklo sąsajų didesnę (nuo 25 % iki 64 %) CPU apkrovą visame tirtame MSS ruože lemia „Calico VXLAN“ įskiepis lyginant su „Calico IP“ CNI įskiepiu. Tai rodo, kad jeigu yra taikomas sąsajų jungimas ir naudojamas „Calico“ CNI įskiepis ir jei architektūrinės galimybės leidžia verčiau yra naudoti „Calico“ IP, o ne VXLAN

režimą. Kaip ir vienos fizinės sąsajos atveju „Flannel“ įskiepio lemiama CPU apkrova visame tirtame MSS ruože ir visų sąsajų jungimo atveju yra labai artima „Calico IP“ režimo sukuriamai CPU apkrovai.

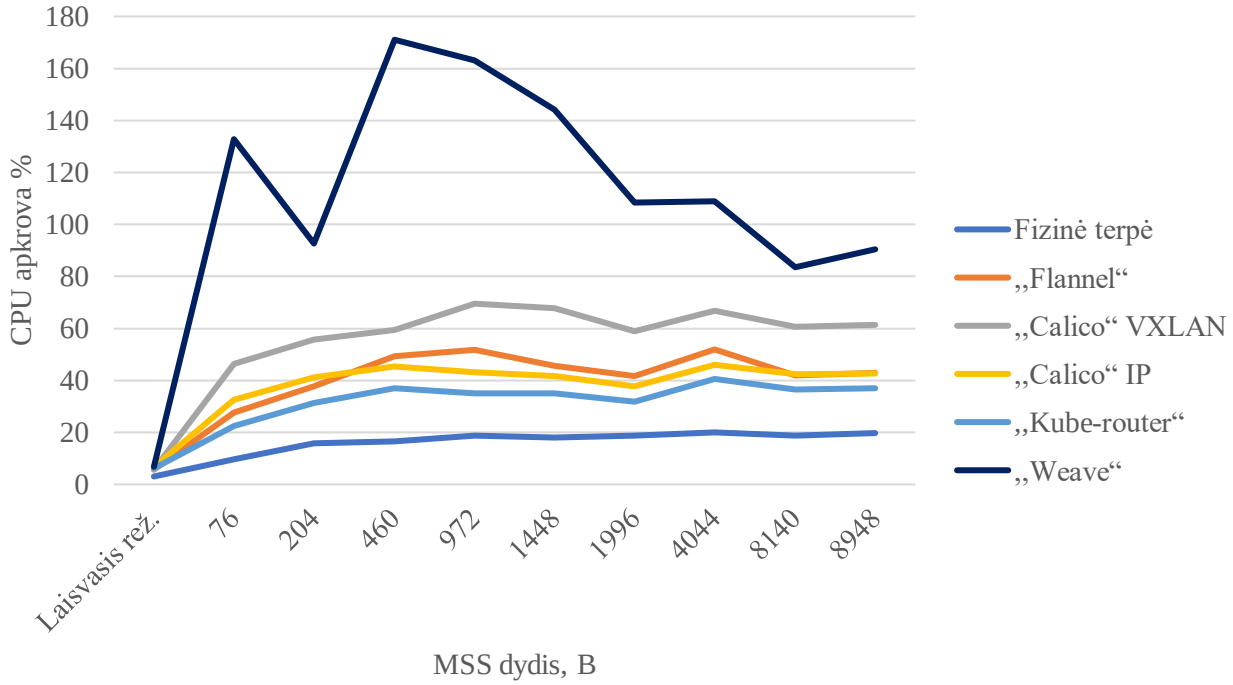


4.9 pav. CNI įskiepių vidutinė vieno CPU apkrova, esant įvairiems MSS dydžiams, kai jungiamos dvi tinklo sąsajos

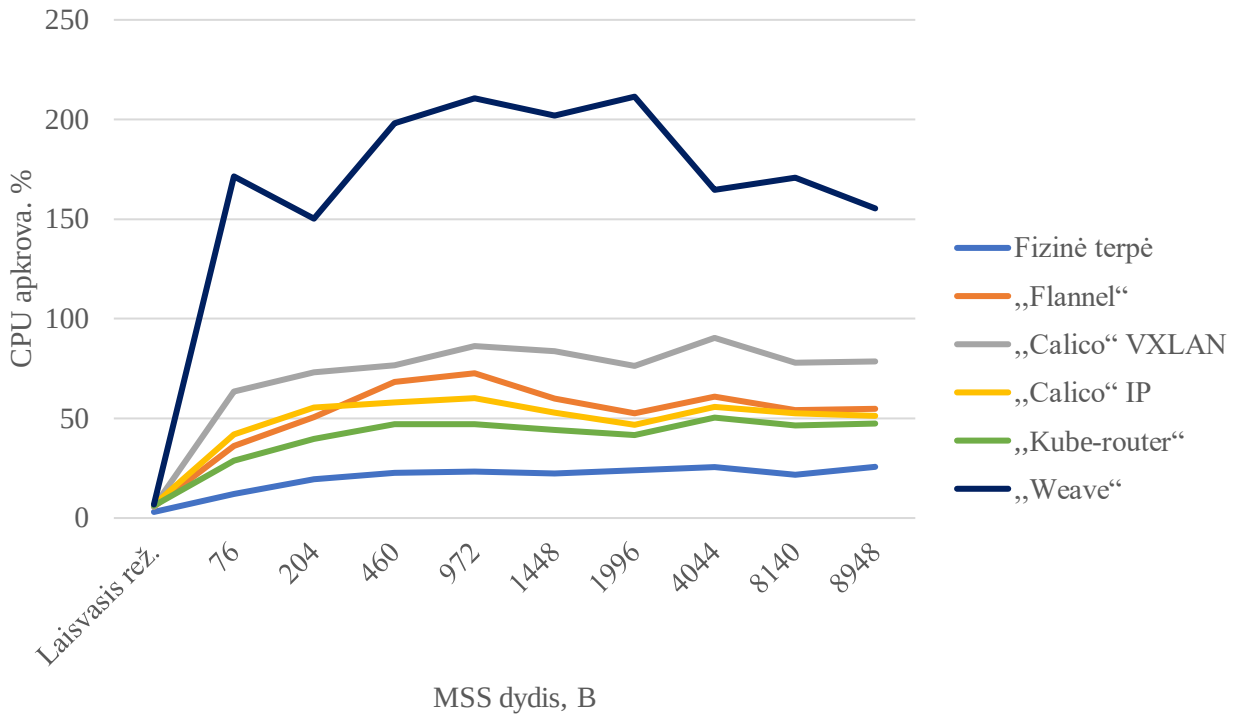
Vidutinė CPU apkrova, esant skirtingiems CNI įskiepiams, 76, 1448 ir 8960 B MSS dydžiams, ir kintant jungtųjų sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų, yra pateikta 4.13 pav. Iš pateiktų rezultatų matyti, kad didėjant jungtųjų tinklo sąsajų skaičiui vidutinė CPU apkrova taip pat didėja. Fizinės terpės atveju CPU apkrova lyginant su vienos tinklo sąsajos generuojama CPU apkrova, išauga ~3 kartus, kai yra jungiamos penkios sąsajos. Didžiausias CPU apkrovos augimas didėjant jungtųjų tinklo sąsajų skaičiui yra pastebimas užklotojo tinklo sprendimų atveju.



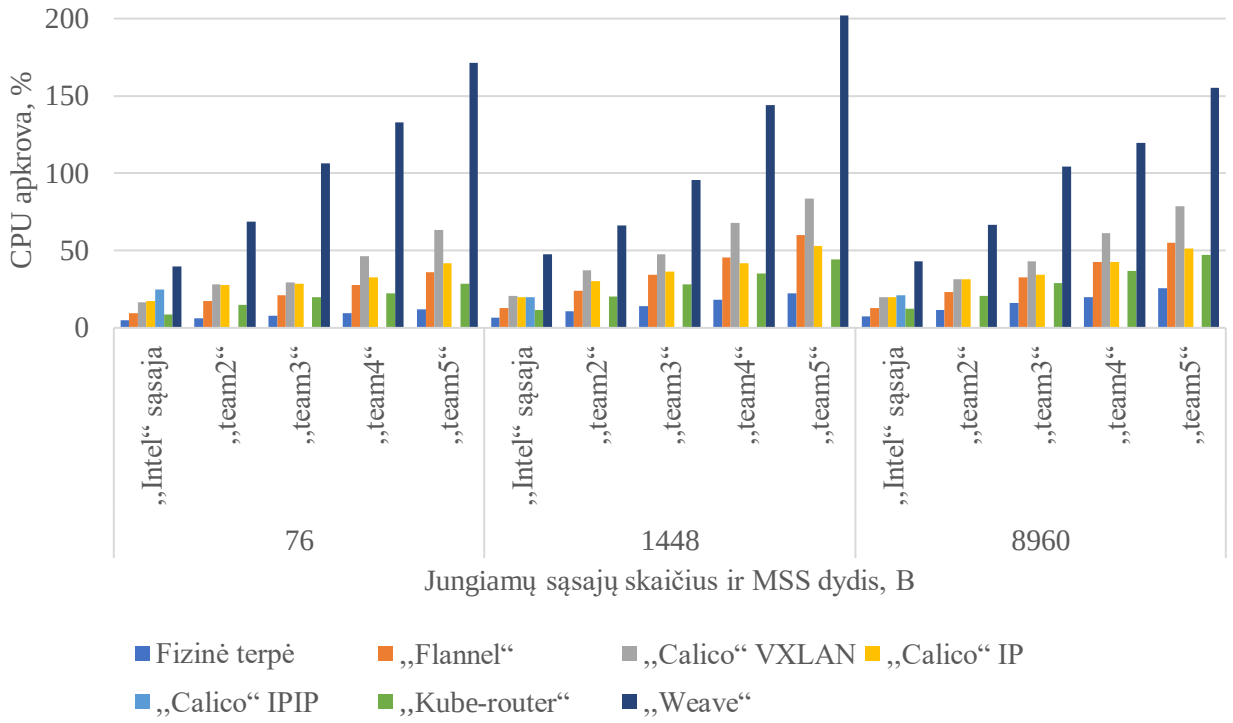
4.10 pav. CNI įskiepių vidutinė vieno CPU apkrova, esant įvairiems MSS dydžiams, kai jungiamos trys tinklo sąsajos



4.11 pav. CNI įskiepių vidutinė vieno CPU apkrova, esant įvairiems MSS dydžiams, kai jungiamos keturios tinklo sąsajos



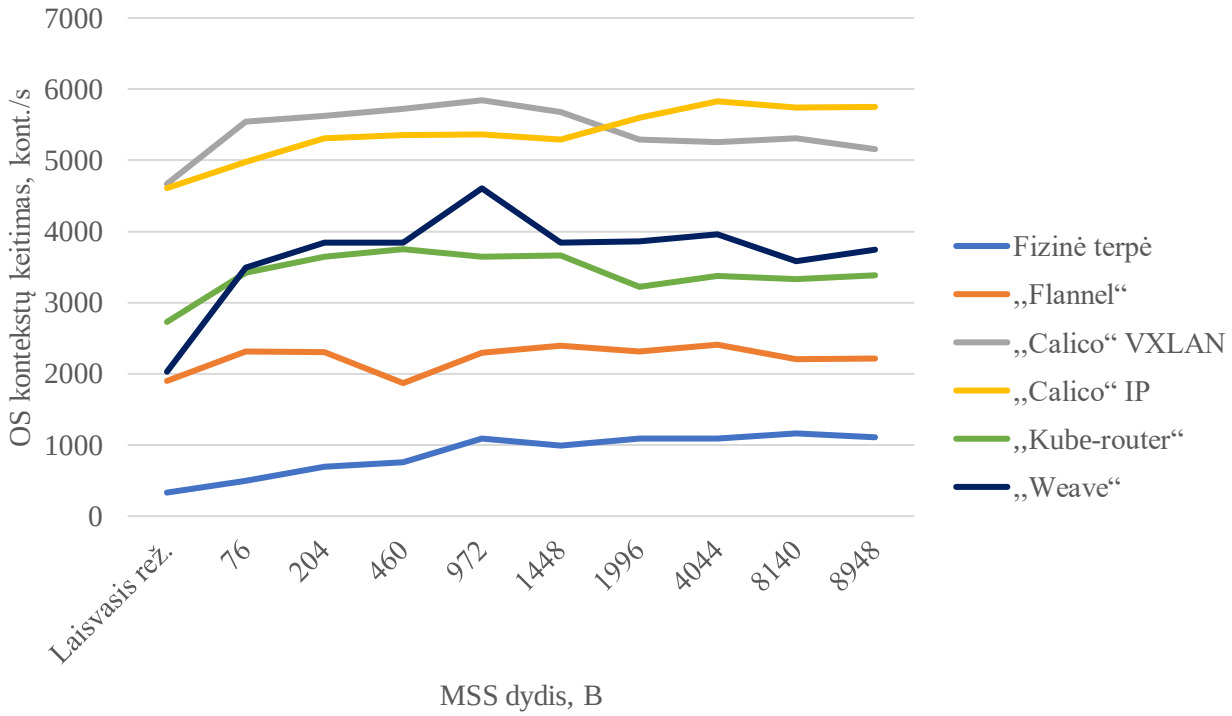
4.12 pav. CNI įskiepių vidutinė vieno CPU apkrova, esant įvairiems MSS dydžiams, kai jungiamos penkios tinklo sąsajos



4.13 pav. CNI įskiepių vidutinė CPU apkrova, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų

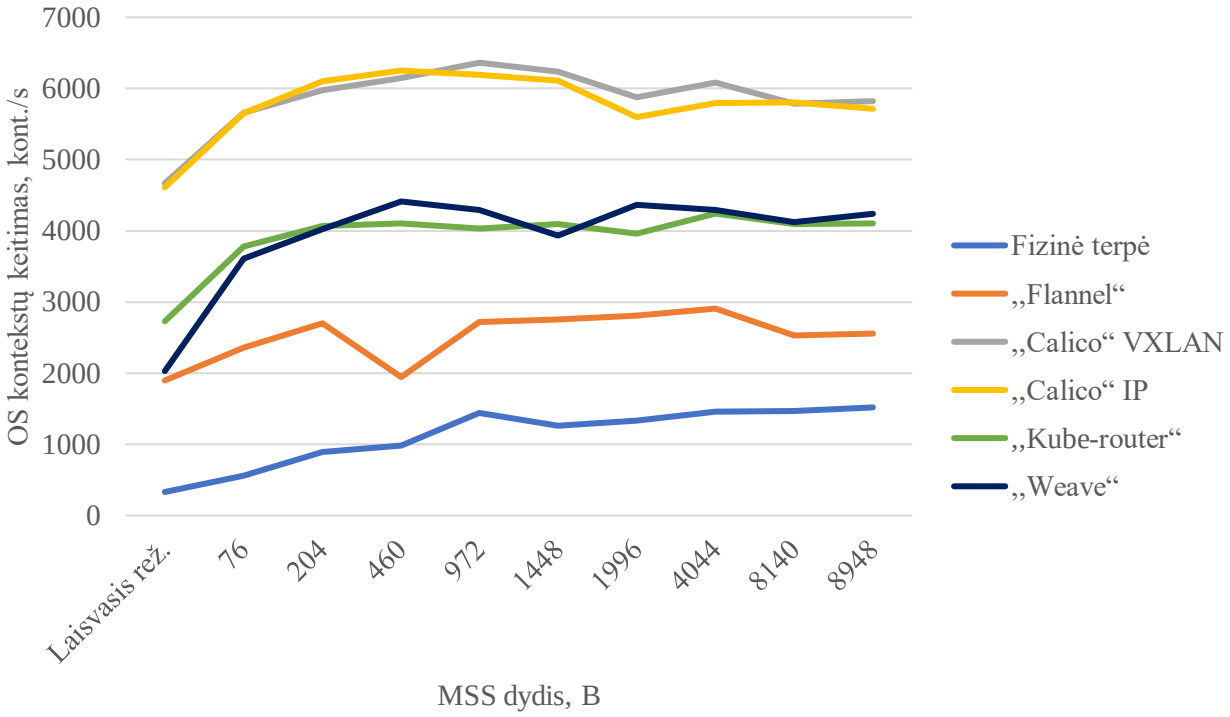
4.3.3. Vidutinio OS kontekstų keitimo taikant tinklo sąsajų jungimą tyrimo rezultatai

Vidutinio OS kontekstų keitimo rezultatai, kai buvo taikomas tinklo sąsajų jungimas, yra pateikti 4.14–4.17 pav. Gauti rezultatai pagal kontekstų keitimo tendenciją didėjant MSS dydžiui yra panašūs į rezultatus, gautus, kai buvo naudojama viena tinklo sąsaja (žr 4.3 pav.). Visais sąsajų jungimo atvejais „Flannel“ pasižymėjo mažiausiu vidutiniu OS kontekstų keitimu. Kaip ir CPU apkrovos tyrimo atveju (žr. 5.4–5.7 pav.) skirtumai tarp „Calico“ IP ir VXLAN režimų išryškėjo, kai buvo jungiamos keturios ir penkios tinklo sąsajos. Šiais atvejais „Calico“ VXLAN vidutinis OS kontekstų keitimas buvo nuo 1 % iki 20 % didesnis. Taip pat „Calico“ VXLAN ir IP režimai pasižymėjo didžiausiu vidutiniu OS kontekstų keitimu visais tirtais atvejais. Įdomu tai, kad visais keturiais tirtais sąsajų jungimo atvejais „Flannel“ vidutinis OS kontekstų keitimas, esant 460 B MSS dydžiui, sumažėdavo nuo ~19 % iki ~31 % lyginant su reikšme, stebėta, esant 204 B MSS dydžiui. Kita vertus, toks pokytis nėra pastebimas kitų CNI įskiepių atveju. Galima daryti prielaidą, kad šį pokytį lemia specifinės „Flannel“ įskiepio savybės. Taip pat tikėtina, kad šis vidutinis OS kontekstų keitimo sumažėjimas yra pastebimas todėl, kad didėjant paketų dydžiui mažėja tinklu persiunčiamų paketų skaičius, todėl mažėja tinklo sąsajų generuojamų pertraukčių. Verta atkreipti dėmesį, kad didėjant jungtųjų tinklo sąsajų skaičiui didėja vidutinis OS kontekstų keitimo skaičius visų tirtų CNI įskiepių atveju. Šis pokytis taip pat yra nulemtas papildomų tinklo sąsajų generuojamų pertraukčių ir papildomų ankščių, veikiančių „Kubernetes“ telkinyje.

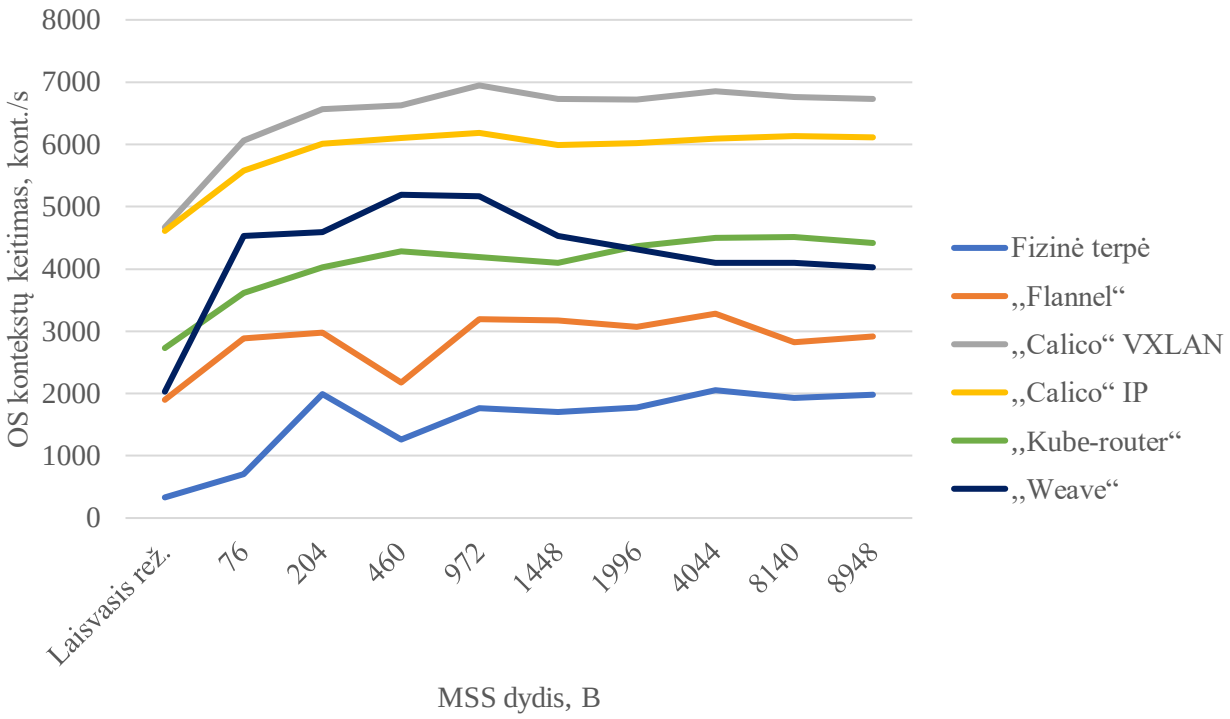


4.14 pav. CNI įskiepių vidutinis OS kontekstų keitimas, kai jungiamos dvi tinklo sąsajos

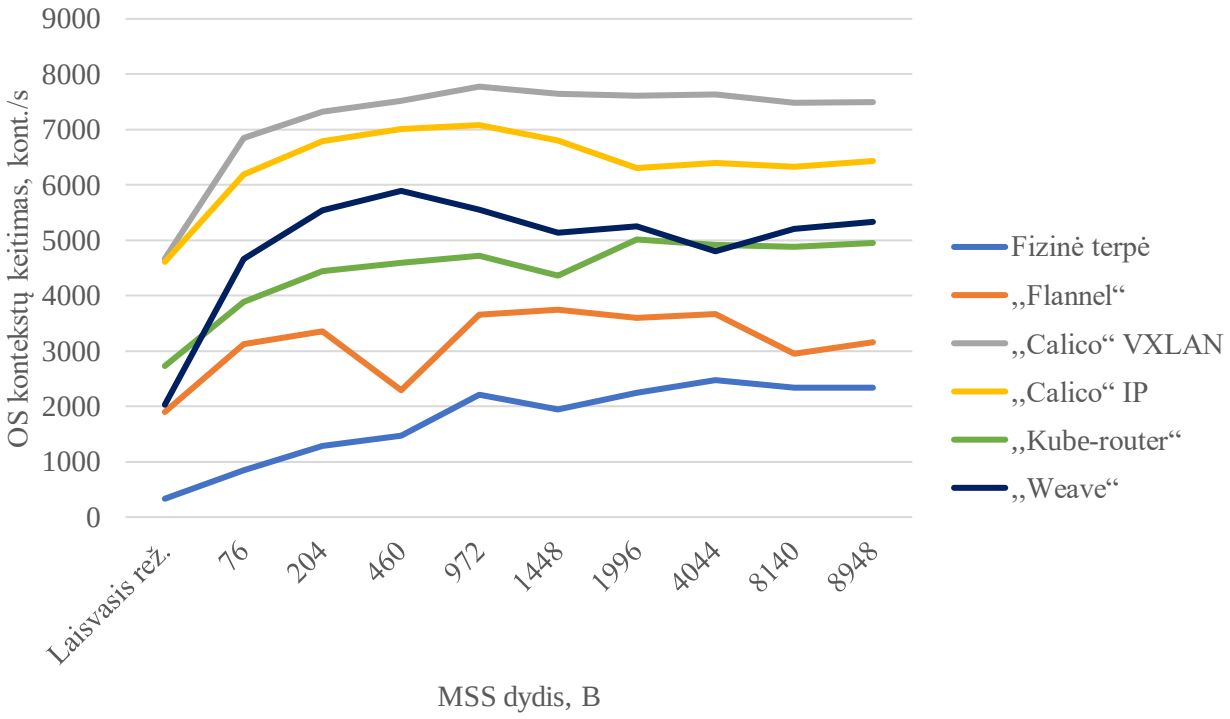
Vidutinio OS kontekstų keitimo rezultatai, esant 76, 1448 ir 8960 B MSS dydžiams ir kintant jungtųjų tinklo sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų, yra pateikti 4.18 pav. Iš pateiktų rezultatų matyti, kad didėjant jungtųjų tinklo sąsajų skaičiui vidutinis OS kontekstų keitimas taip pat didėja. Daugumos tirtų CNI įskiepių atveju šis didėjimas nesiekia dviejų kartų ir gali būti paaiškinamas tuo, kad didėjant jungtųjų sąsajų skaičiui, tinklu perduodamų paketų skaičius taip pat auga, o tai ir sukelia didesnį tinklo plokštės generuojamų pertraukčių skaičių.



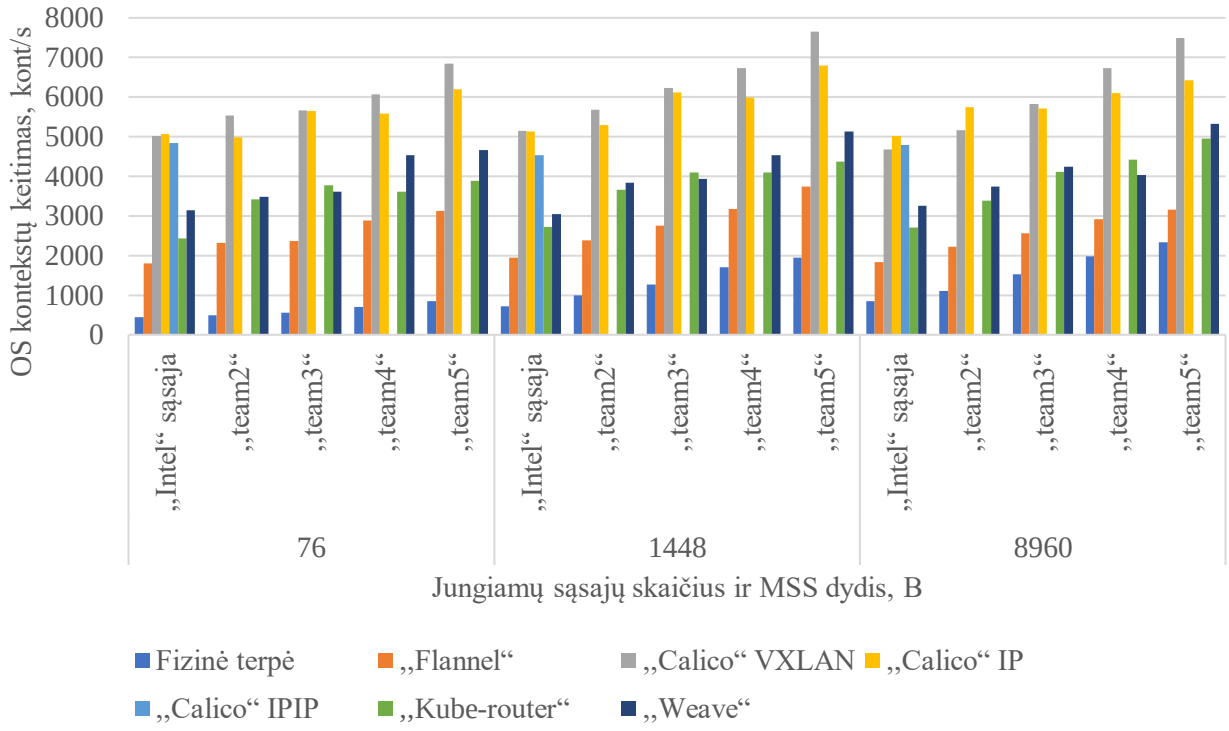
4.15 pav. CNI įskiepių vidutinis OS kontekstų keitimas, kai jungiamos trys tinklo sąsajos



4.16 pav. CNI įskiepių vidutinis OS kontekstų keitimas, kai jungiamos keturios tinklo sąsajos



4.17 pav. CNI įskiepių vidutinis OS kontekstų keitimas, kai jungiamos penkios tinklo sąsajos



4.18 pav. CNI įskiepių vidutinis OS kontekstų keitimas, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų

4.4. CNI įskiepių našumo tyrimo rezultatai išjungus TCP GSO mechanizmus

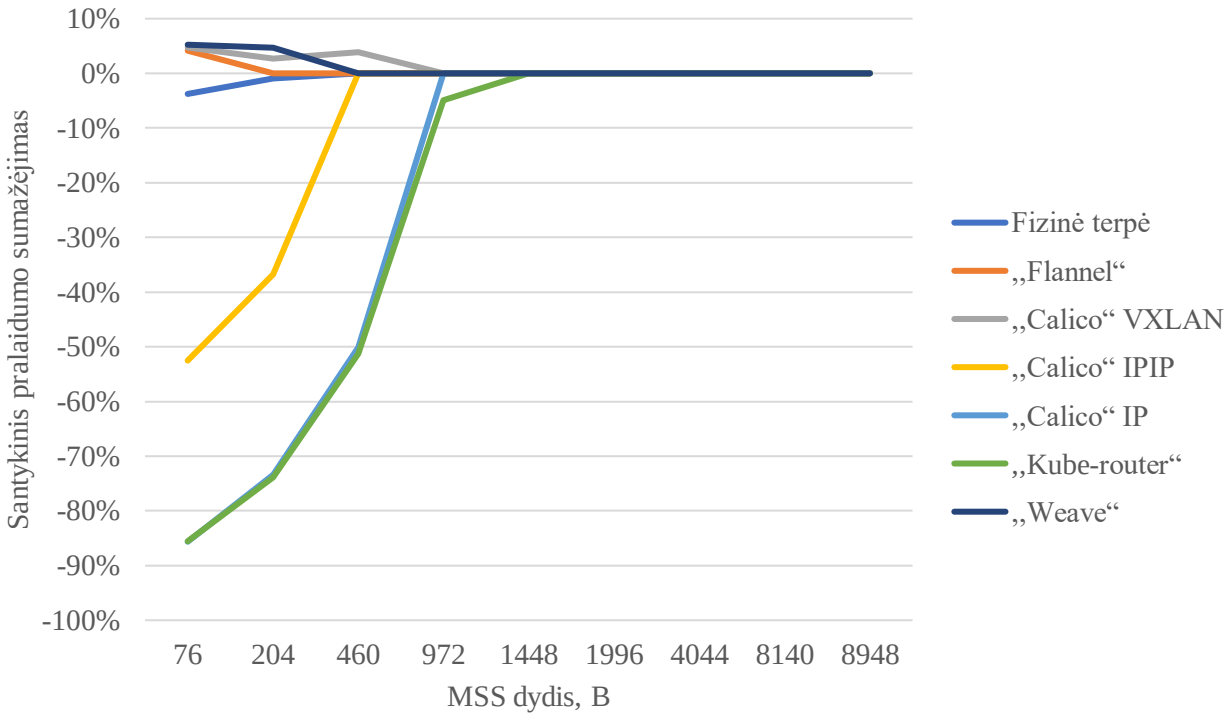
Šiame skyriuje yra apžvelgiami CNI įskiepių našumo tyrimo, kai tinklo sąsajų TCP GSO mechanizmas buvo išjungtas, rezultatai. Šio tyrimo rezultatai skirti išsiaiškinti, kokią įtaką CNI įskiepių našumui turi TCP GSO mechanizmo išjungimas, kai yra naudojami įvairūs MSS dydžiai. Tyrimo rezultatai buvo gauti pakartojus tyrimus, apžvelgtus 4.2 ir 4.3 skyriuose, tačiau išjungus visų tinklo sąsajų TCP GSO mechanizmus.

4.4.1. TCP GSO mechanizmo įtaka CNI įskiepių TCP protokolo pralaidumui

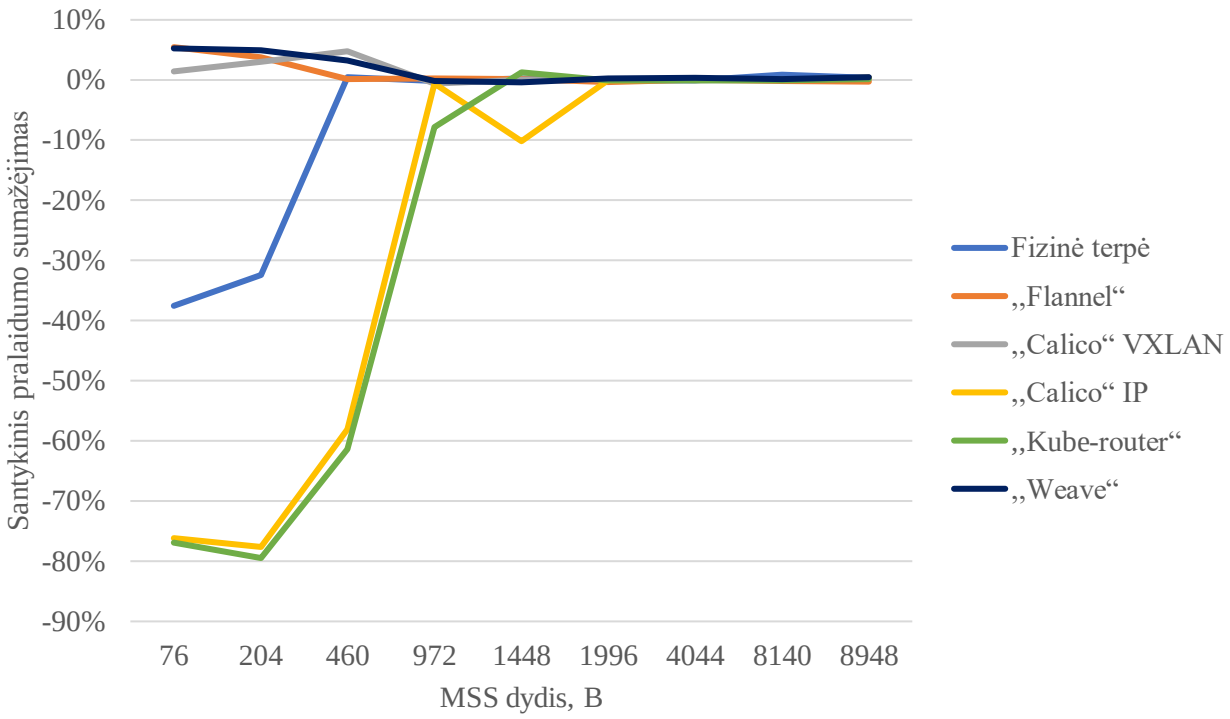
CNI įskiepių TCP protokolo santykinis pralaidumo sumažėjimas, kai buvo naudotos viena–penkios tinklo sąsajos, yra pateiktas 4.19–4.23 pav. Iš pateiktų rezultatų matyti, kad TCP GSO mechanizmo išjungimas įtakos turi tik mažesniems paketų dydžiams. Visais tirtais atvejais siunčiant didesnius nei 1448 B paketus pasiektas toks pat TCP protokolo pralaidumas, nesvarbu, ar TCP GSO mechanizmas buvo įjungtas ar išjungtas. Kita vertus, didėjant jungtųjų tinklo sąsajų skaičiui matomas augantis TCP pralaidumo mažėjimas 1448 B MSS atveju. CNI įskiepių atžvilgiu užklotojo tinklo įskiepių TCP pralaidumas beveik nesumažėjo visais tirtais atvejais, išskyrus „Calico IPIP“ atveju. CNI IP sprendimų, „Calico IP“ ir „Kube-router“, TCP pralaidumas visais tirtais atvejais, esant mažesnėms MSS vertėms, sumažėjo nuo 5 % iki 84 %, tačiau didėjant jungtųjų tinklo sąsajų skaičiui, santykinis TCP pralaidumo sumažėjimas šių įskiepių atveju kito nežymiai. Verta atkreipti dėmesį į „Weave“ CNI įskiepio rezultatus. Šio įskiepio atveju TCP pralaidumo sumažėjimas buvo stebimas

tik tada, kai buvo naudojamos penkios jungtosios tinklo sąsajos. Tokį „Weave“ TCP protokolo pralaidumo sumažėjimą galima susieti su išaugusia CPU apkrova (žr. 4.12 pav.). Penkių sąsajų jungimo atveju „Weave“ CNI įskiepis visiškai sunaudojo dviejų fizinių CPU resursus ir dalį trečiojo CPU resursų. Ši išaugusi CPU apkrova paaiškina santykinį TCP protokolo pralaidumo sumažėjimą, kai yra jungiamos 5 tinklo sąsajos. Tikėtina, kad toliau didėjant jungtųjų tinklo sąsajų skaičiui, „Weave“ TCP protokolo pralaidumas dar labiau mažėtų.

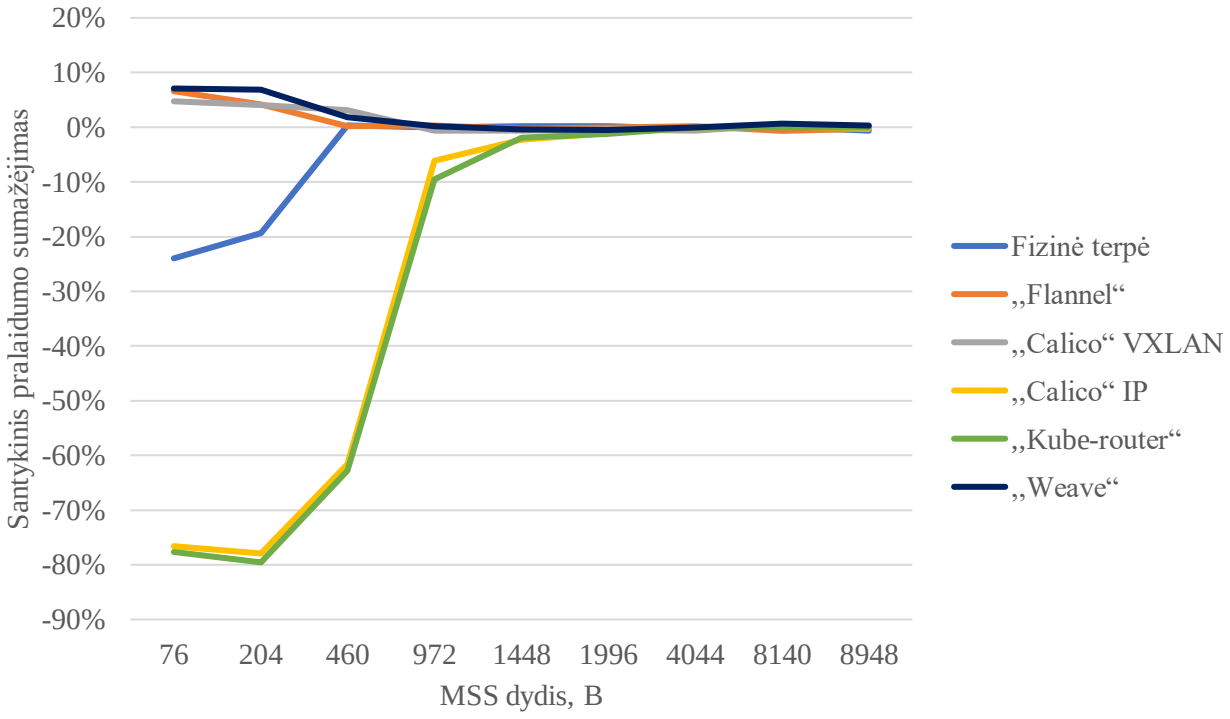
TCP protokolo suminio pralaidumo rezultatai, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų tinklo sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų, kai TCP protokolo GSO mechanizmas yra išjungtas, yra pateikti 4.24 pav. Šie rezultatai yra labai artimi TCP protokolo suminio pralaidumo rezultatams, pateiktiems 4.8 pav., tačiau matomas skirtumas, esant 76 B MSS dydžio paketams, kur fizinės terpės suminis pralaidumas visų tirtų CNI įskiepių pralaidumą viršijo iki dviejų kartų.



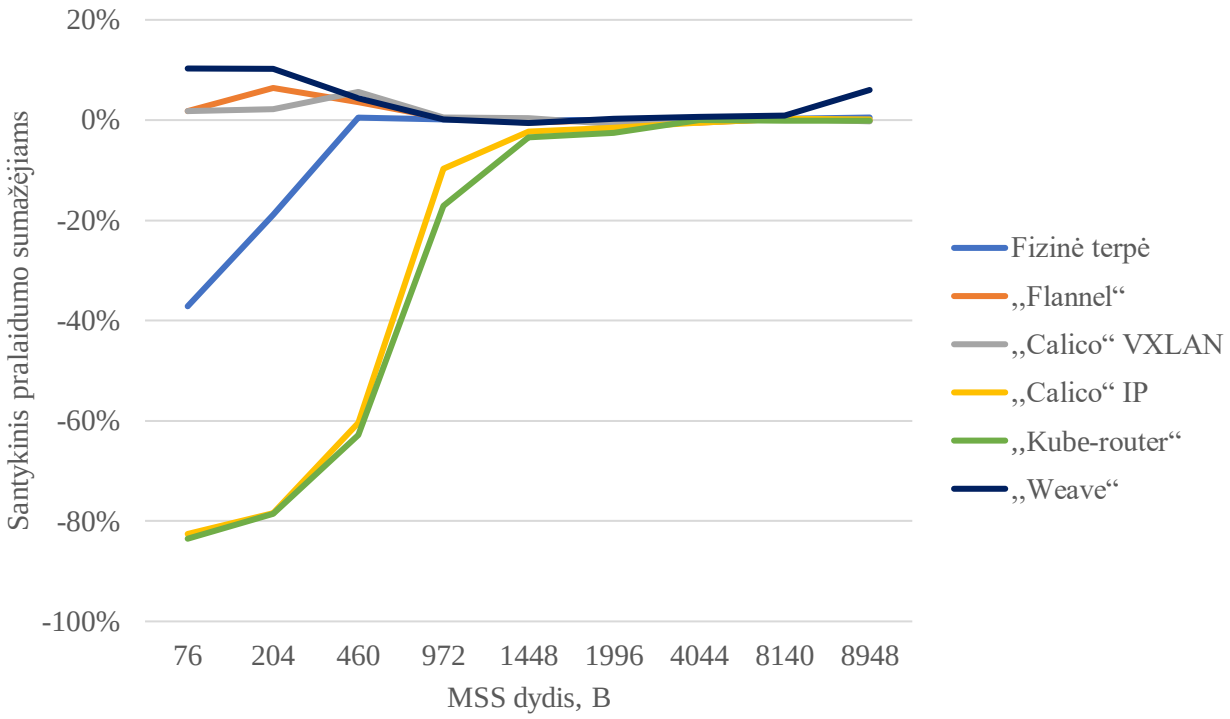
4.19 pav. Santykinis CNI įskiepių TCP protokolo pralaidumo sumažėjimas, esant įvairiems MSS dydžiams, kai naudojama viena „Intel“ sąsaja ir TCP GSO mechanizmas yra išjungtas



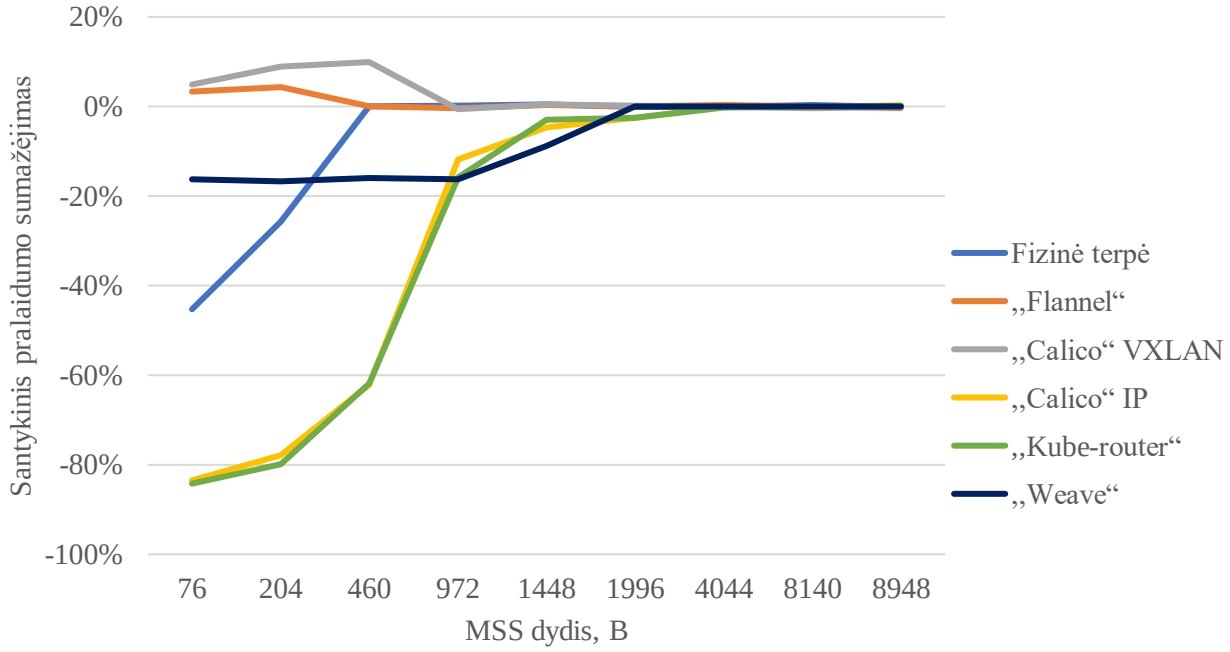
4.20 pav. Santykinis CNI įskiepių TCP protokolo pralaidumo sumažėjimas, esant įvairiems MSS dydžiams, kai naudojamos dvi jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



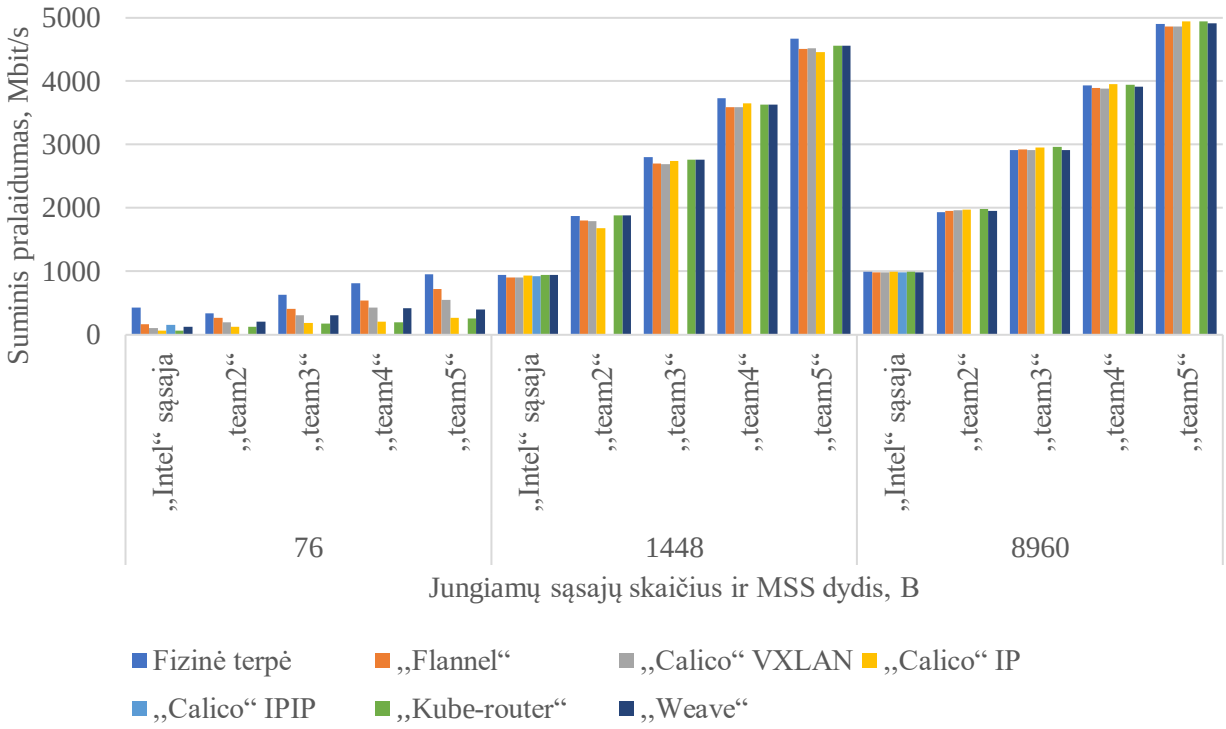
4.21 pav. Santykinis CNI įskiepių TCP protokolo pralaidumo sumažėjimas, esant įvairiems MSS dydžiams, kai naudojamos trys jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



4.22 pav. Santykinis CNI įskiepių TCP protokolo pralaidumo sumažėjimas, esant įvairiems MSS dydžiams, kai naudojamos keturios jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



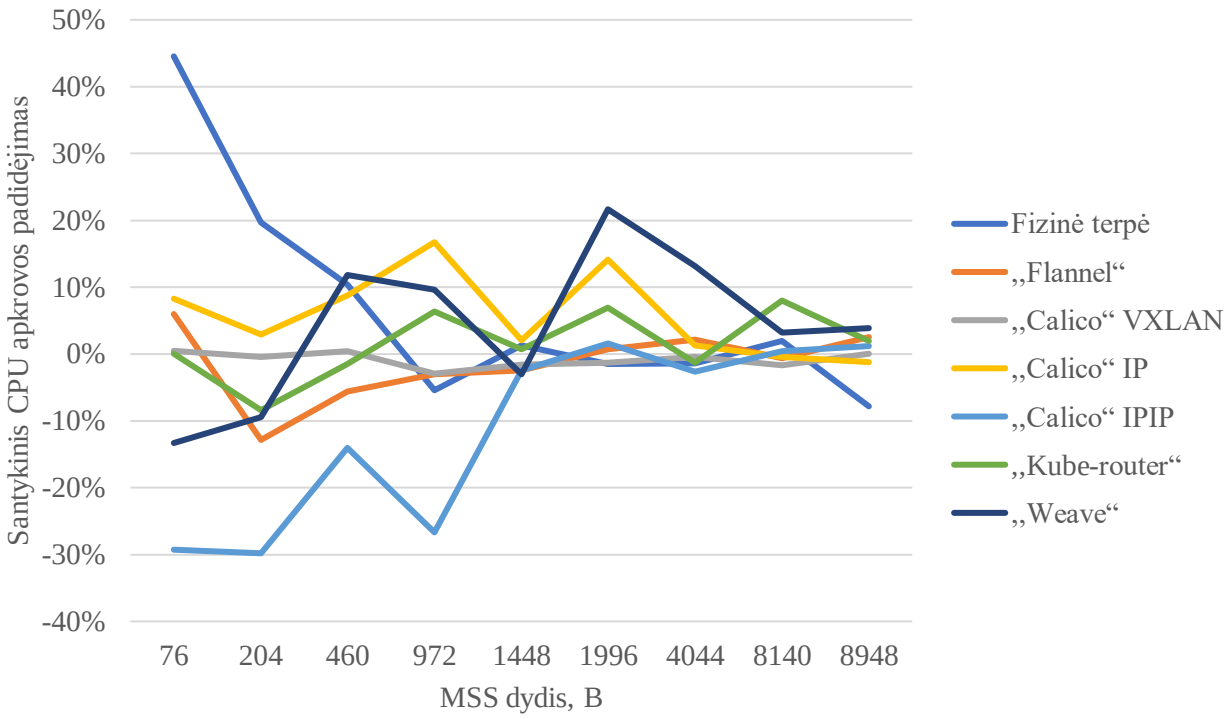
4.23 pav. Santykinis CNI įskiepių TCP protokolo pralaidumo sumažėjimas, esant įvairiems MSS dydžiams, kai naudojamos penkios jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



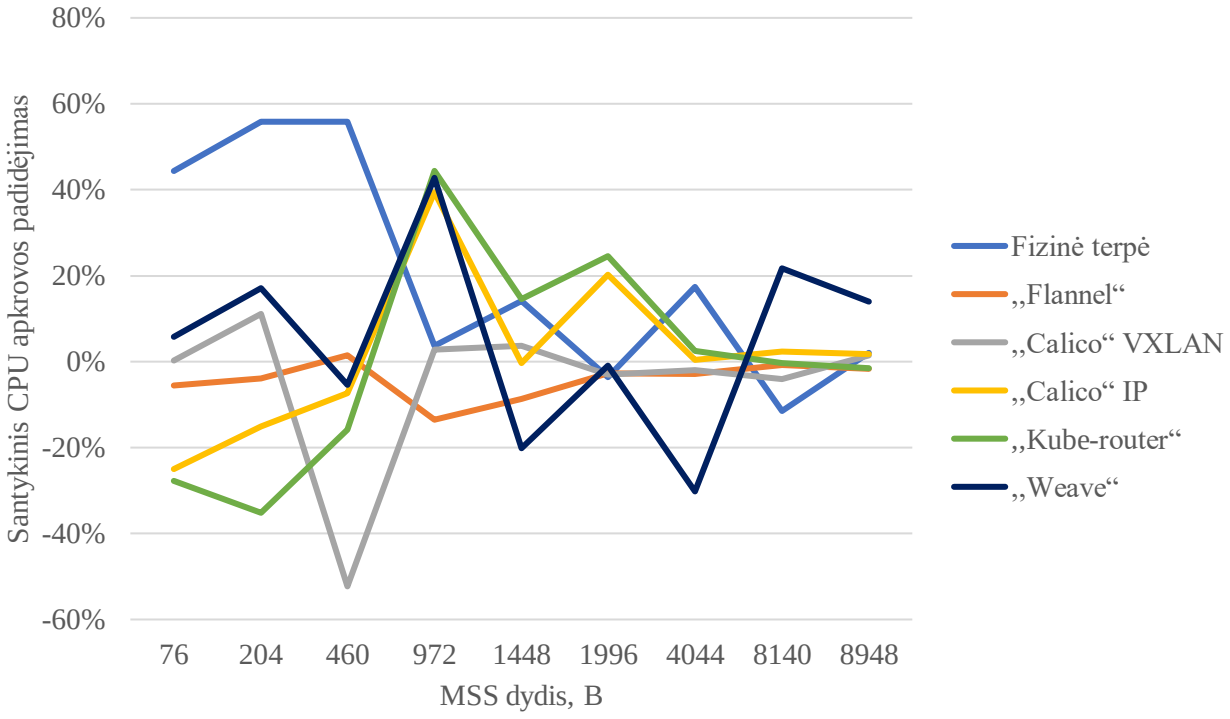
4.24 pav. CNI įskiepių TCP protokolo suminis pralaidumas, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų, kai TCP GSO mechanizmas yra išjungtas

4.4.2. TCP GSO mechanizmo įtaka CPU apkrovai

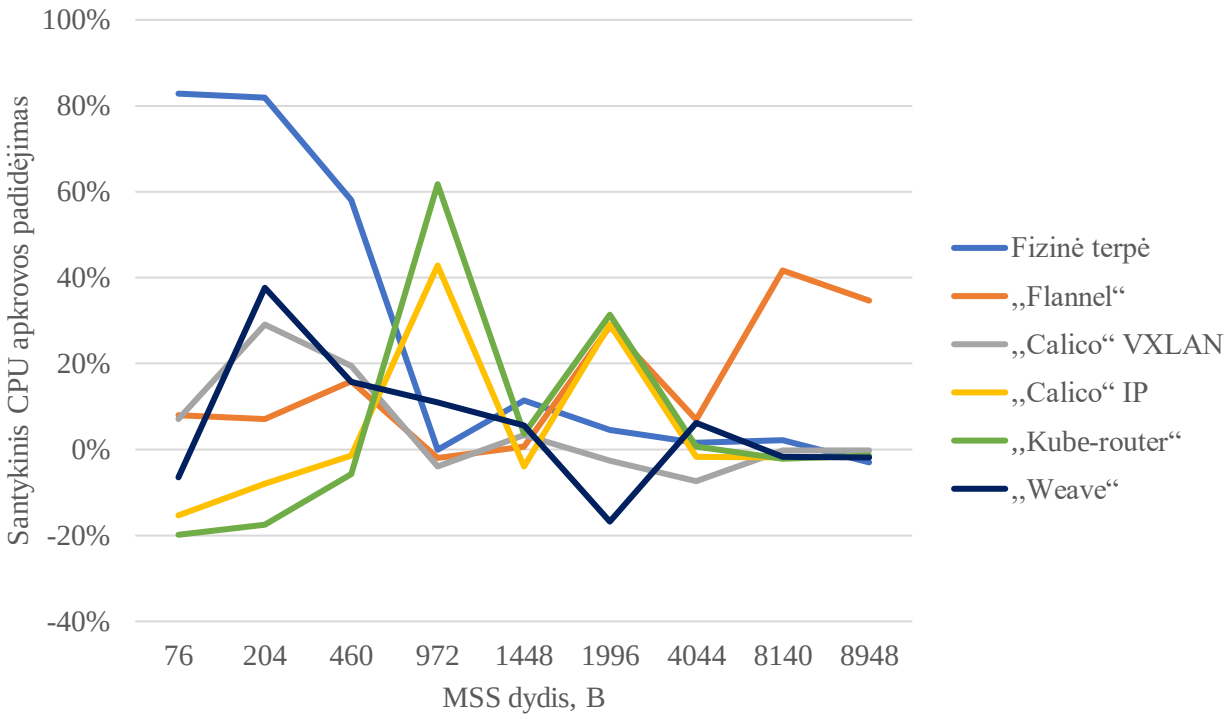
Santykinis vidutinės CPU apkrovos padidėjimas vienam CPU, kai TCP GSO mechanizmas išjungtas ir naudojamos viena–penkios tinklo sąsajos, yra pateiktas 4.25–4.29 pav. Iš rezultatų matyti, kad visų sąsajų jungimo atveju, kai MSS dydis buvo 76–460 B, CPU apkrova fizinės terpės atveju išaugo iki 83 % ir laipsniškai mažėjo didėjant paketų dydžiui. CNI įskiepių atveju CPU apkrovos padidėjimas dažniausiai buvo stebimas, tačiau ne toks žymus kaip fizinės terpės atveju. Įdomu tai, kad fizinės terpės atveju CPU apkrova buvo žymiai didesnė, esant mažesniems paketams, o CNI įskiepių atveju, atvirkščiai, CPU apkrova, esant mažiems paketams, padidėjo nedaug, tačiau labiau padidėjo esant didesnėms MSS reikšmėms. Verta atkreipti dėmesį į visų „Calico“ režimų CPU apkrovos pokytį, esant mažoms 76–460 B MSS reikšmėms. Priešingai nei galima būtų tikėtis, išjungus TCP GSO mechanizmą, minėtais atvejais „Calico“ įskiepių CPU apkrova sumažėjo net iki 30 %. Tokį CPU apkrovos sumažėjimą galima susieti su nuo ~50 % iki 80 % sumažėjusiu pralaidumu. Stipriai sumažėjęs TCP pralaidumas kompensuoja papildomą CPU apkrovą, kuri atsiranda išjungus TCP GSO mechanizmą ir todėl atrodo, kad nenaudojant TCP GSO mechanizmo „Calico“ atveju CPU apkrova sumažėja. Be to, tokia pati CPU apkrovos sumažėjimo tendencija, esant 76–460 B MSS dydžiui, yra stebima ir „Kube-router“ atveju.



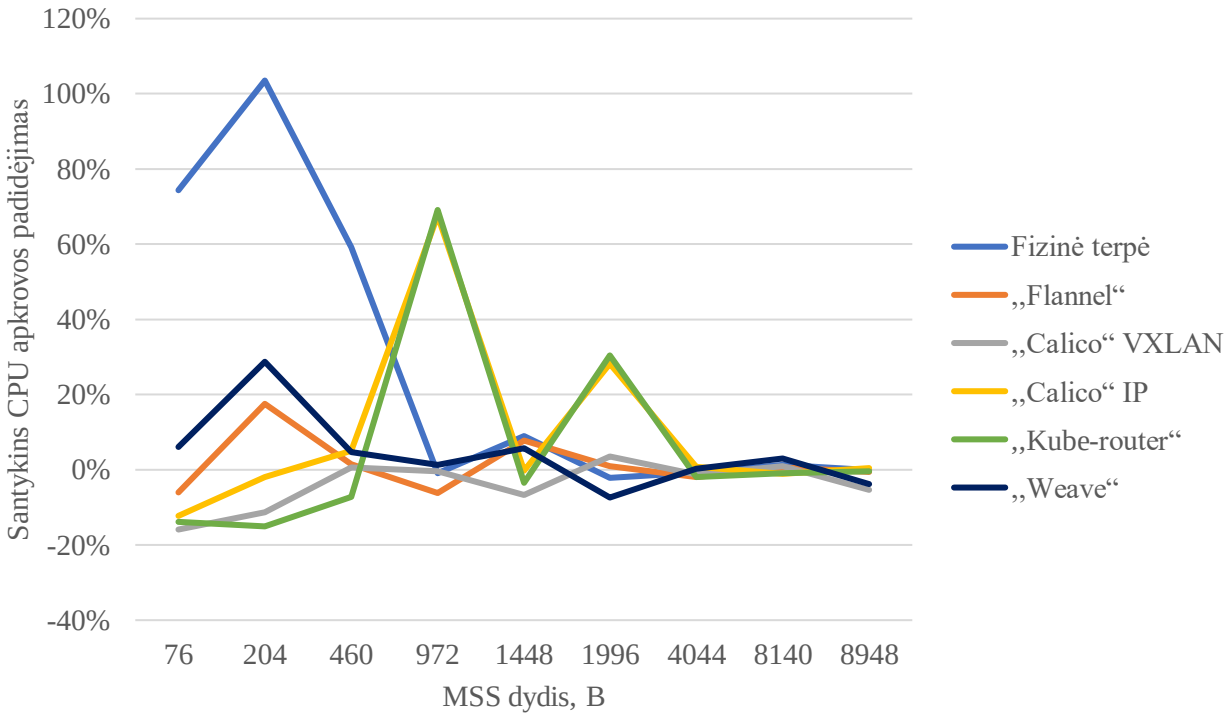
4.25 pav. CNI įskiepių santykinis CPU apkrovos padidėjimas, esant įvairiems MSS dydžiams, kai naudojama viena „Intel“ tinklo sąsaja ir TCP GSO mechanizmas yra išjungtas



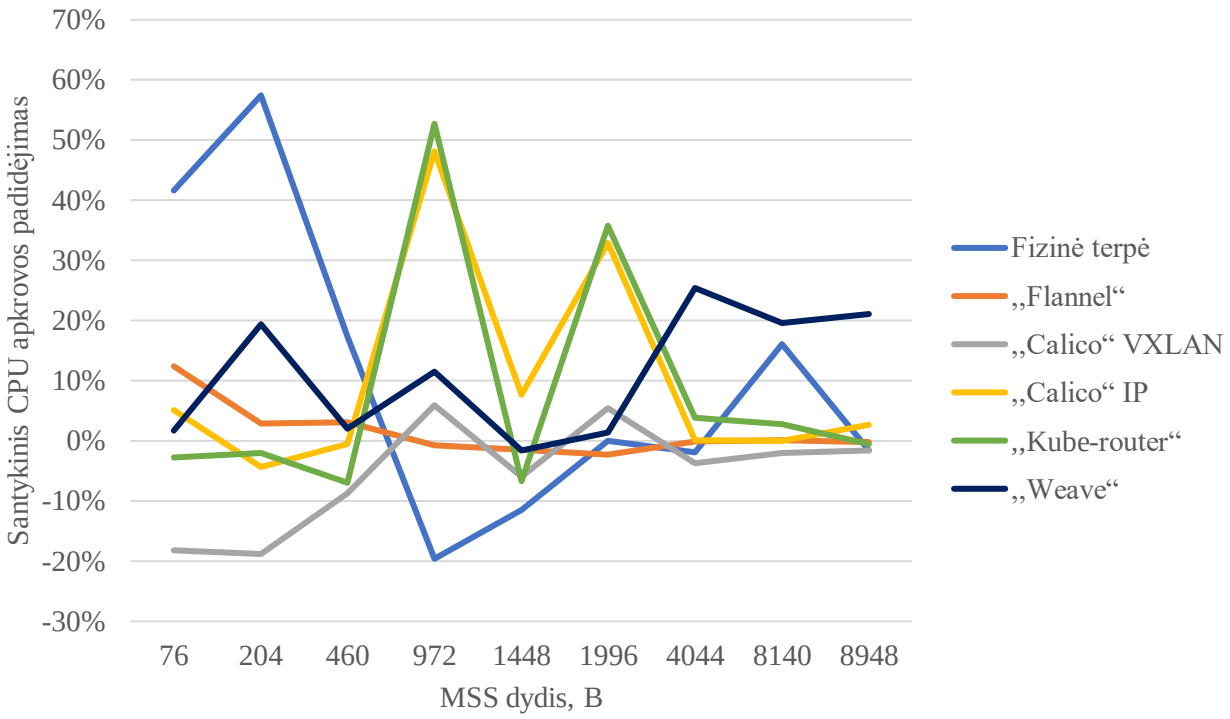
4.26 pav. CNI įskiepių santykinis CPU apkrovos padidėjimas, esant įvairiems MSS dydžiams, kai naudojamos dvi jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



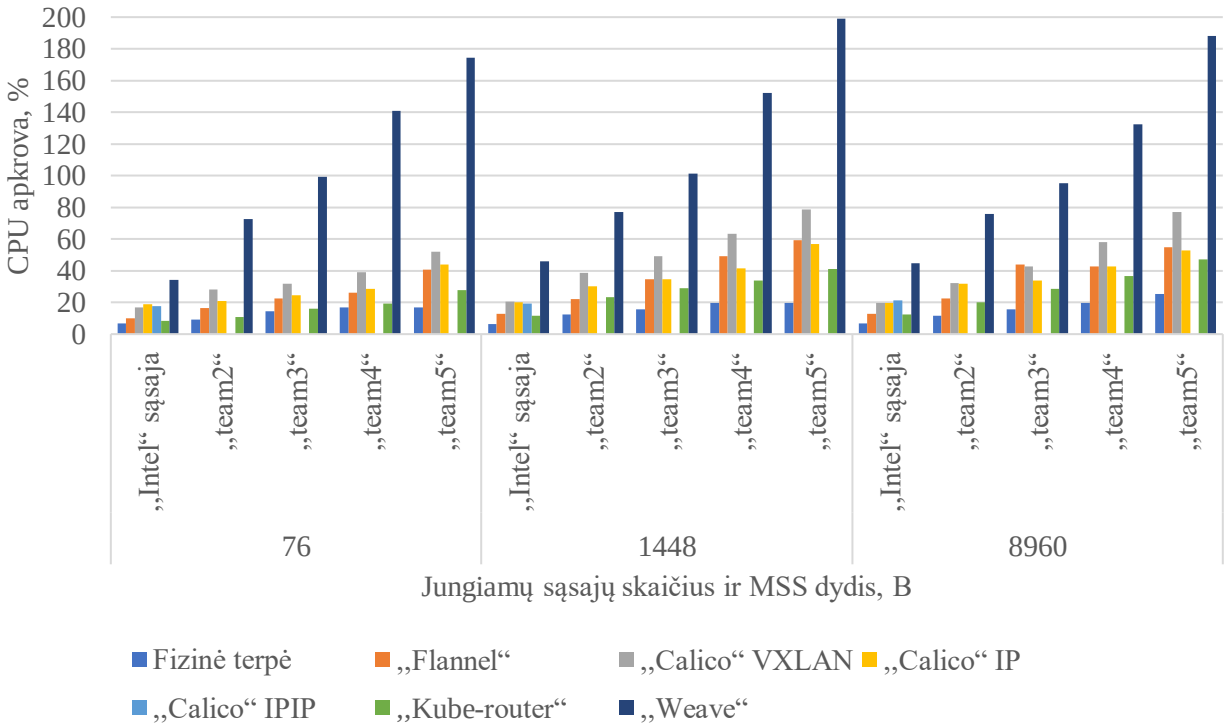
4.27 pav. CNI įskiepių santykinis CPU apkrovos padidėjimas, esant įvairiems MSS dydžiams, kai naudojamos trys jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



4.28 pav. CNI įskiepių santykinis CPU apkrovos padidėjimas, esant įvairiems MSS dydžiams, kai naudojamos keturios jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



4.29 pav. CNI įskiepių santykinis CPU apkrovos padidėjimas, esant įvairiems MSS dydžiams, kai naudojamos 5 jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas

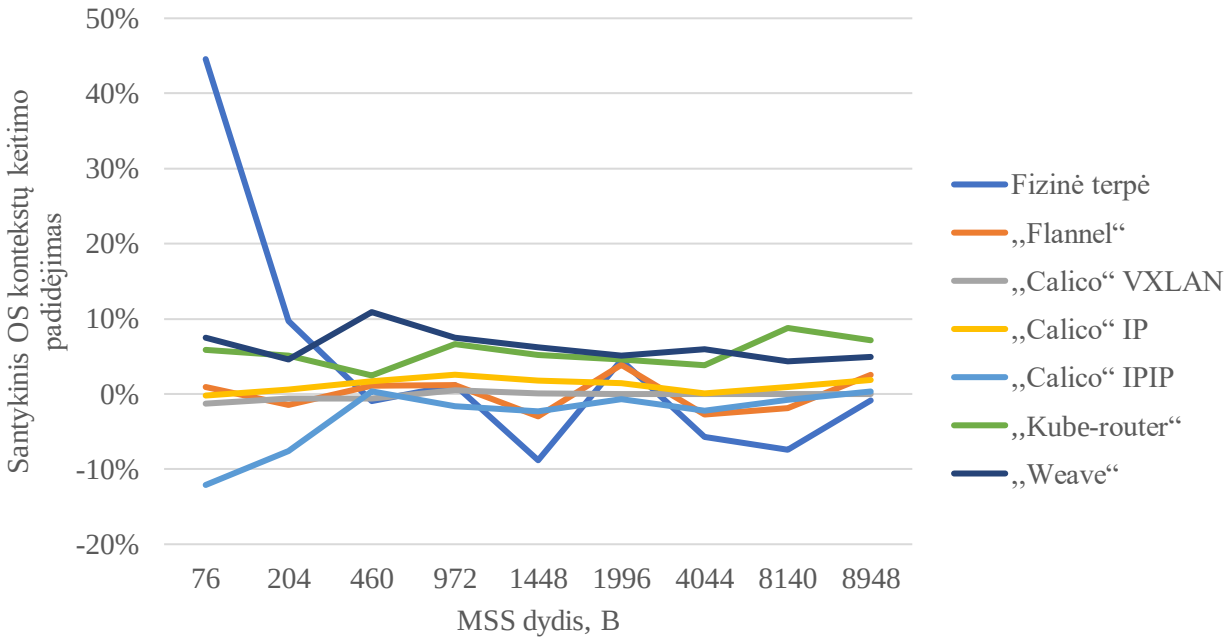


4.30 pav. CNI įskiepių vidutinė CPU apkrova, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų, kai TCP GSO mechanizmas yra išjungtas

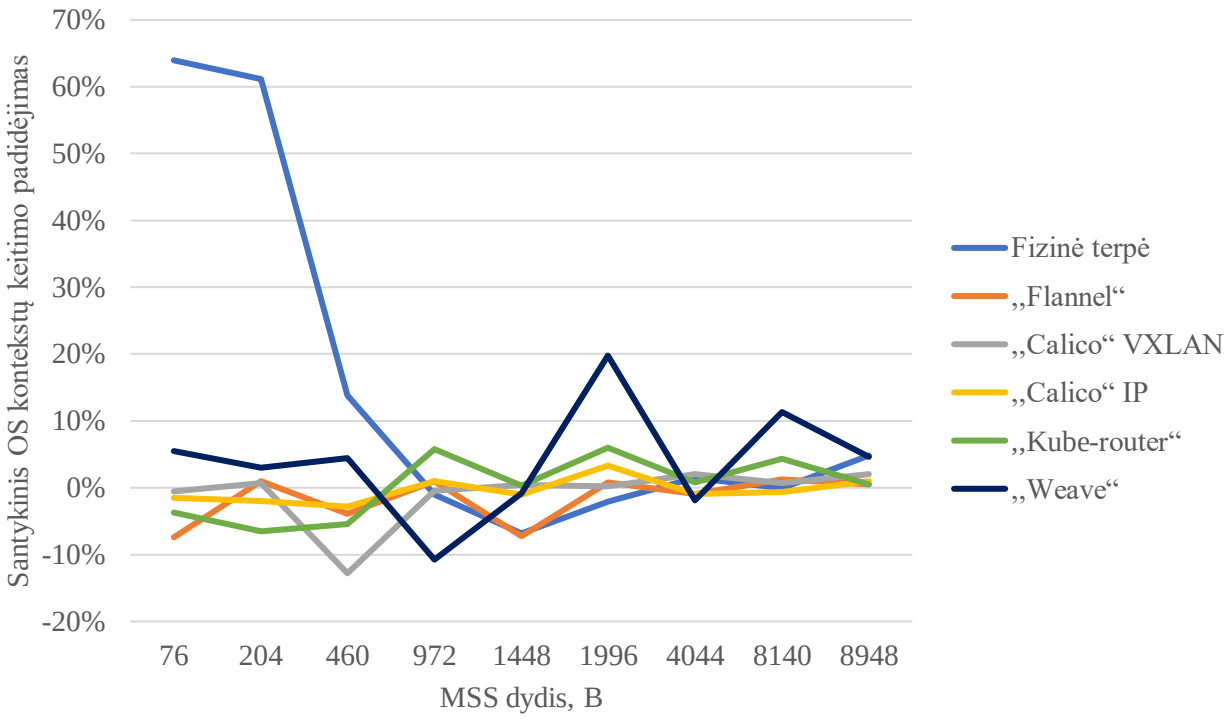
Vidutinės CPU apkrovos skirtingų CNI įskiepių rezultatai, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų tinklo sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų, kai TCP protokolo GSO mechanizmas yra išjungtas, yra pateikti 4.30 pav. Pateiktų rezultatų kitimo pobūdis yra panašus į CPU apkrovos kitimo polinkį, pateiktą 4.13 pav.

4.4.3. TCP GSO mechanizmo įtaka OS kontekstų keitimui

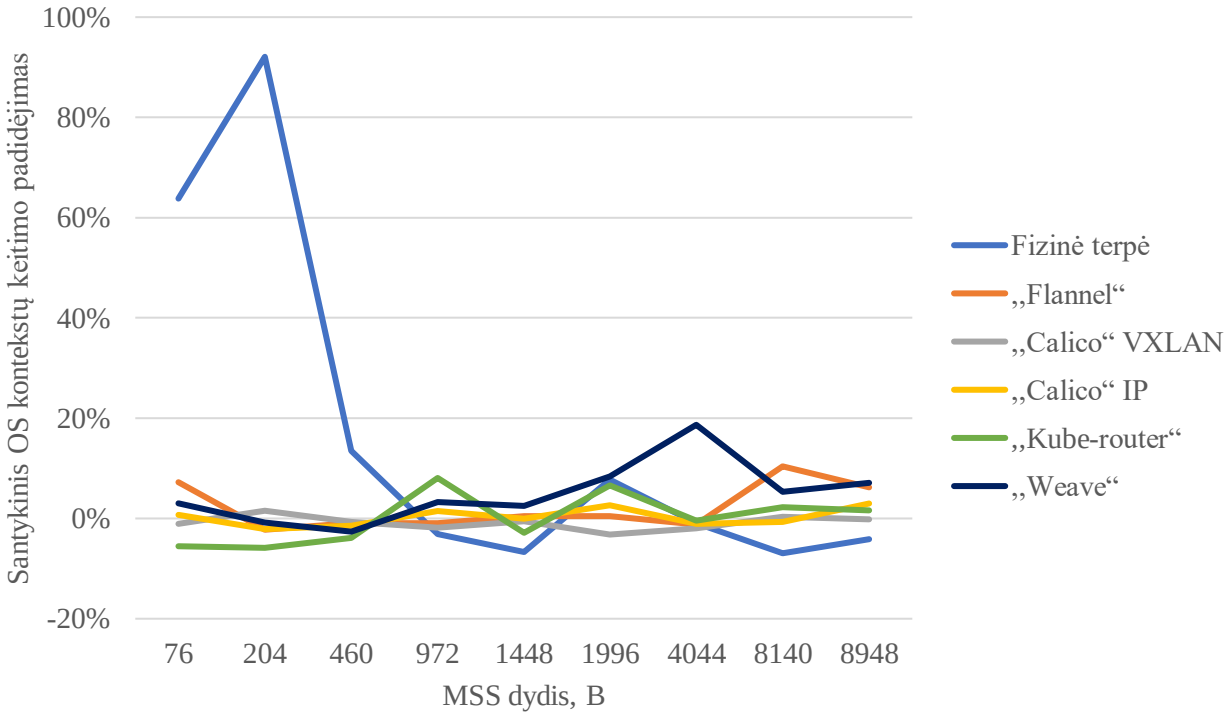
Santykinio vidutinio OS kontekstų keitimo padidėjimo rezultatai tirtų CNI įskiepių atveju, kai buvo naudojamos viena–penkios tinklo sąsajos, yra pateikiami 4.31–4.35 pav. Rezultatai rodo, kad visų sąsajų jungimo atveju, esant 76–460 B MSS dydžiui, OS kontekstų keitimas padidėjo iki ~90 %, tačiau, esant didesnėms reikšmėms, žymesnis padidėjimas nėra pastebimas. Taip pat OS kontekstų keitimas, esant mažesniems MSS dydžiams, yra pastebimas ir daugelio CNI įskiepių atveju. Tai galima paaiškinti tuo, kad išjungus TCP GSO mechanizmą yra išsiunčiama daugiau mažų paketų, todėl yra generuojama daugiau pertraukčių, lemiančių kontekstų keitimą. „Calico IPIP“, „Calico IP“ ir „Kube-router“ atveju, esant mažesnėms MSS reikšmėms, yra stebimas iki kelių procentų OS kontekstų keitimo sumažėjimas. Šį OS kontekstų keitimo sumažėjimą kaip ir CPU tyrimo atveju galima sieti su sumažėjusiu TCP protokolo pralaidumu, dėl to taip pat yra persiunčiama mažiau paketų ir generuojama mažiau pertraukčių. Be to, 4.36 pav. yra pateiktas tirtų CNI įskiepių OS kontekstų keitimas, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų, kai TCP GSO mechanizmas yra išjungtas. Šių rezultatų kitimo polinkis yra labai artimas aptartiems rezultatams, pateiktiems 4.18 pav.



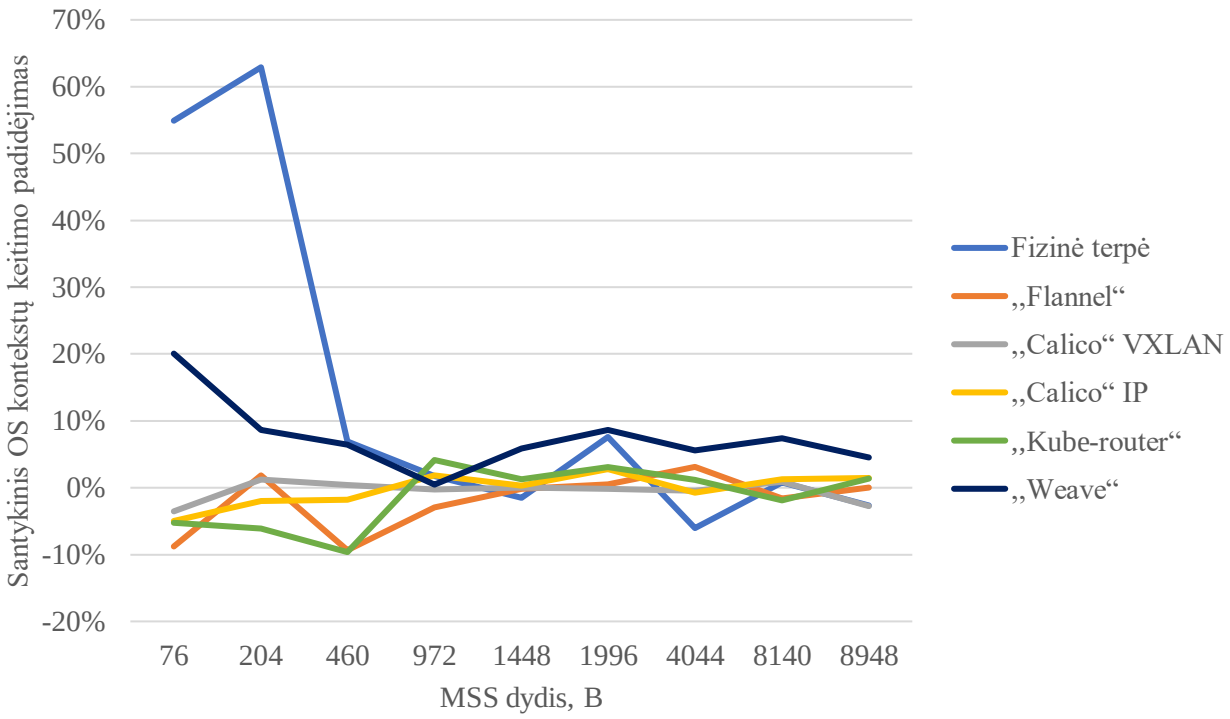
4.31 pav. CNI įskiepių santykinis OS kontekstų keitimo padidėjimas, esant įvairiems MSS dydžiams, kai naudojama viena „Intel“ tinklo sąsaja ir TCP GSO mechanizmas yra išjungtas



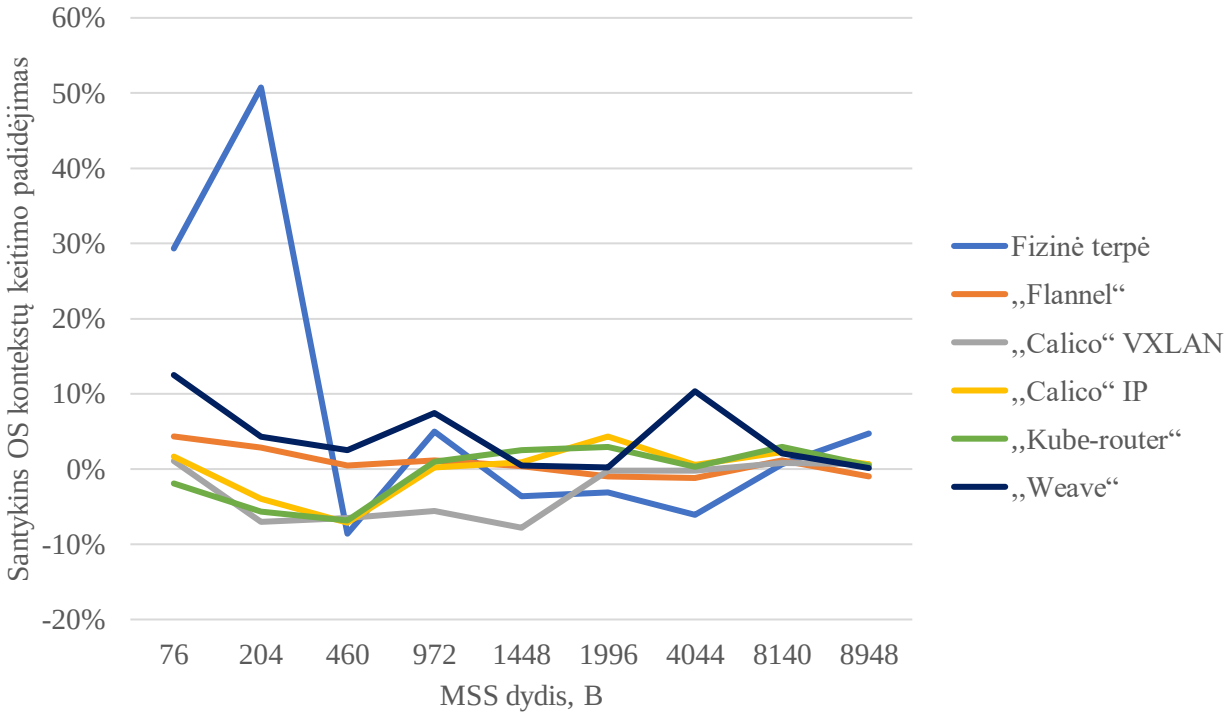
4.32 pav. CNI įskiepių santykinis OS kontekstų keitimo padidėjimas, esant įvairiems MSS dydžiams, kai naudojamos dvi jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



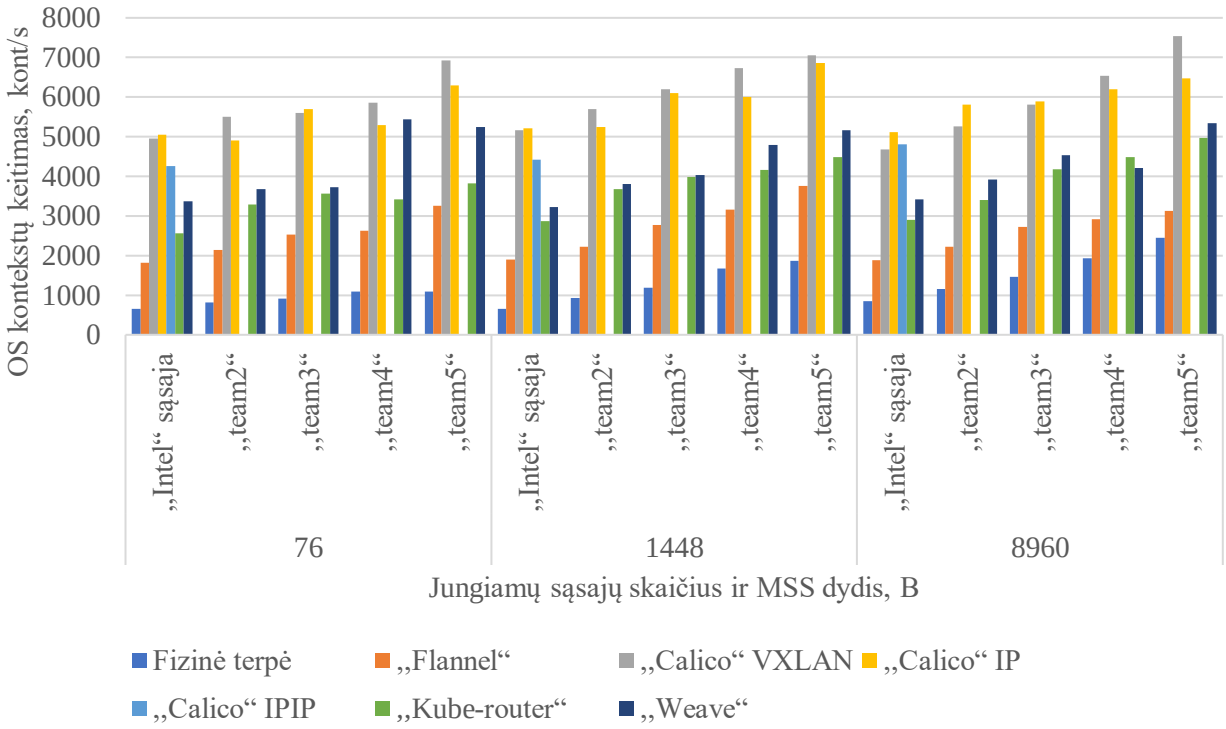
4.33 pav. CNI įskiepių santykinis OS kontekstų keitimo padidėjimas, esant įvairiems MSS dydžiams, kai naudojamos trys jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



4.34 pav. CNI įskiepių santykinis OS kontekstų keitimo padidėjimas, esant įvairiems MSS dydžiams, kai naudojamos keturios jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



4.35 pav. CNI įskiepių santykinis OS kontekstų keitimo padidėjimas, esant įvairiems MSS dydžiams, kai naudojamos penkios jungtosios tinklo sąsajos ir TCP GSO mechanizmas yra išjungtas



4.36 pav. CNI įskiepių vidutinis OS kontekstų keitimas, esant 76, 1448 ir 8960 B MSS dydžiui ir kintant jungtųjų sąsajų skaičiui nuo vienos iki penkių tinklo sąsajų

IŠVADOS

Šiame darbe atliktas „Kubernetes“ keturių CNCF rekomenduojamų CNI įskiepių tyrimas fiziniame 1 Gbit/s spartos duomenų centro tinkle. Buvo nagrinėjami „Flannel“, „Calico“, „Kube-router“ ir „Weave“ CNI įskiepiai. Išmatuotos ir lygintos šios charakteristikos: TCP pralaidumas, vidutinė CPU apkrova, ICMP paketų vėlinimas ir OS kontekstų keitimas. Matavimai buvo atliekami esant skirtingam „Linux“ „teamd“ tvarkykle jungtųjų tinklo sąsajų skaičiui, skirtingiems TCP GSO parametrams ir skirtingiems paketų dydžiams. Atlikus tyrimą suformuluotinos tokios išvados:

1. Visi tirti CNI įskiepiai lyginant su fizine terpe prideda mažiausiai 0,2 ms papildomo vėlinimo. Taip pat užklotojo tinklo technologijomis paremtų CNI įskiepių vėlinimas yra 0,05 ms didesnis lyginant su IP CNI sprendimais. Mažiausiu vėlinimu pasižymėjo „Flannel“ ir „Kube-router“ CNI įskiepiai.
2. Visų tirtų CNI įskiepių TCP protokolo vidutinis pralaidumas yra artimas fizinės terpės pralaidumui, kai yra naudojama viena tinklo sąsaja ir esant didesniam nei 972 B MSS dydžiui, nesvarbu, ar TCP GSO mechanizmas yra įjungtas ar išjungtas.
3. Tinklo sąsajų jungimas prideda papildomą vėlinimą ir jis didėja didinant jungtųjų tinklo sąsajų skaičių, nes „teamd“ tvarkyklės „loadbalance“ algoritmas to paties srauto paketus paskirsto tarp skirtingų tinklo sąsajų.
4. Didėjant jungtųjų tinklo sąsajų skaičiui užklotojo tinklo CNI įskiepių TCP pralaidumo atsilikimas nuo fizinės terpės pralaidumo rezultatų didėja, tačiau IP CNI įskiepių TCP pralaidumui įtakos neturi. Kita vertus, IP CNI įskiepių atveju esant 76–460 B paketams TCP pralaidumas buvo mažesnis (nuo 5 % iki 84 %) lyginant su fizinės terpės rezultatais.
5. Didėjant jungtųjų sąsajų skaičiui yra pastebimas CPU apkrovos didėjimas (nuo dviejų iki trijų kartų) lyginant su viena tinklo sąsaja ir kai yra jungiamos penkios tinklo sąsajos. Tai rodo, kad naudojant sąsajų jungimą „teamd“ tvarkyklėmis galima pasiekti TCP pralaidumą, artimą teoriniam sąsajų pralaidumui, tačiau yra stebima padidėjusi CPU apkrova.
6. TCP GSO mechanizmo išjungimas TCP pralaidumo sumažėjimui turi įtakos tik CNI IP sprendimams, o įtaka užklotojo tinklo CNI įskiepiams nebuvo pastebėta.
7. Visų tirtų CNI įskiepių TCP vidutinis pralaidumas, esant 76–1460 B paketams, yra pastebimai mažesnis (nuo 30 % iki 70 %) nei fizinės terpės, kai yra taikomas tinklo sąsajų jungimas ir yra išjungtas TCP GSO mechanizmas.
8. TCP GSO mechanizmo išjungimas CNI įskiepių atveju, kai yra naudojama viena tinklo sąsaja, lemia iki 20 % CPU apkrovos padidėjimą. ,,Calico“ atveju buvo stebimas CPU apkrovos sumažėjimas iki 30 %, kurį lėmė nuo 50 % iki 80 % sumažėjęs TCP pralaidumas, esant 76–460 B dydžio paketams.

9. TCP GSO mechanizmo išjungimas lemia OS kontekstų keitimų padidėjimą (iki 90 %), esant 76–460 B MSS dydžio paketams, tačiau prie didesnių MSS reikšmių OS kontekstų keitimo padidėjimas nėra pastebimas.
10. Iš visų „Calico“ įskiepio režimų IP režimas buvo našiausias, todėl jeigu tinklo architektūra leidžia, yra rekomenduojama naudoti šį „Calico“ režimą. Jeigu yra būtina taikyti užklotojo tinklo sprendimą ir norima naudoti „Calico“ CNI įskiepį, verčiau yra pasirinkti IPIP režimą vietoje VXLAN, nes IPIP režimo TCP pralaidumas, esant 76–460 B dydžio paketams, buvo nuo 36 % iki 230 % didesnis, o vėlinimas vidutiniškai 0,05 ms mažesnis nei VXLAN režimo.
11. TCP pralaidumo atžvilgiu iš užklotojo tinklo CNI įskiepių našiausias buvo „Flannel“, o iš IP CNI įskiepių grupės našiausias buvo „Kube-router“ įskiepis.
12. Įrodyta, kad taikant tinklo sąsajų jungimą, galima padidinti suminį kanalo pralaidumą, kuris visų CNI įskiepių atveju, esant didesniems nei 972 B paketams, buvo artimas fizinės terpės pralaidumui. Būtina pabrėžti, kad norint efektyviai išnaudoti kanalą, yra būtina parinkti tinkamus „teamd“ tvarkyklės parametrus, kurie priklauso nuo tinklo srauto tipo.
13. Visais tirtais atvejais didėjant paketų dydžiui TCP pralaidumas taip pat didėjo. Didžiausias pralaidumas yra pasiekiamas naudojant 8960 B dydžio paketus.
14. Remiantis tyrimo rezultatais buvo publikuoti du moksliniai straipsniai.

LITERATŪROS IR INFORMACIJOS ŠALTINIAI

Acreman, S. (2018*). CNI plugins comparison*. Prieiga per internetą: https://docs.google.com/spreadsheets/d/1qCOlor16Wp5mHd6MQxB5gUEQILnijyDLIExEpqmee2k/edit#gid=0

Acreman, S. (2018). *Kubernetes Network Plugins*. Prieiga per internetą: https://kubedex.com/kubernetes-network-plugins/

Aust, S., Kim, J., Davis, P., Yamaguchi, A. ir Obana, S. (2006, lapkritis). *Evaluation of Linux Bonding Features*. Pranešimas konferencijoje: International Conference on Communication Technology, Guilin, Kinija.

Center for Applied Internet Data Analysis. (2008). *Packet size distribution comparison between Internet links in 1998 and 2008*. Prieiga per internetą: https://www.caida.org/research/traffic-analysis/pkt_size_distribution/graphs.xml

Chiosi, M. (2012). *Network Functions Virtualisation*. Prieiga per internetą: https://portal.etsi.org/NFV/NFV_White_Paper.pdf

Creating a single control-plane cluster with kubeadm. (n.d.). Prieiga per internetą: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

Das, S. (2017). *SDN dilemma: Linux kernel networking vs. kernel bypass*. Prieiga per internetą: https://www.infoworld.com/article/3189664/sdn-dilemma-linux-kernel-networking-vs-kernel-bypass.html

Davie, B. (2014). *Geneve, VXLAN, and Network Virtualization Encapsulations*. Prieiga per internetą: https://octo.vmware.com/geneve-vxlan-network-virtualization-encapsulations/

DC Lessons (2019). *VXLAN*. Prieiga per internetą: https://www.dclessons.com/category/courses/vxlan

DPDK & SR-IOV CNI Plugin. (n. d.). Prieiga per internetą: https://github.com/intel/sriov-cni

Ducastel A. (2018). *Benchmark results of Kubernetes network plugins (CNI) over 10Gbit/s network*. Prieiga per internetą: https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-april-2019-4a9886efe9c4

Ducastel A. (2019). *Benchmark results of Kubernetes network plugins (CNI) over 10Gbit/s network (Updated April 2019)*. Prieiga per internetą: https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-36475925a560

Eclavea, M. ir Allen, J. (2020). *Overlaying Virtualized Layer 2 Networks Over Layer 3 Networks*. Prieiga per internetą: https://www.apricot.net/apricot2013/assets/virtualized_l2_over_l3_et-final_matt-2_1361954087.pdf

Ferreira, P. A. ir Sinnott, O. R. (2019, Gruodis). *A Performance Evaluation of Containers runnning on Managed Kubernetes Services*. Pranešimas konferencijoje: *2019* IEEE International Conference on Cloud Computing Technology and Science (CloudCom*)*, Sidnėjus, Australija.

Fiber Cabling Solution (2018). *NVGRE vs VXLAN: What's the Difference?*. Prieiga per internetą: http://www.fiber-optic-cable-sale.com/nvgre-vs-vxlan-network-virtualization-difference.html

Garg, P. ir Wang,Y. (2015). *NVGRE: Network Virtualization Using Generic Routing Encapsulation*. Prieiga per internetą: https://tools.ietf.org/html/rfc7637

Gehberger, D., Balla, D., Maliosz, M. ir Simon, C. (2018, rugsėjis). *Performance Evaluation of Low Latency Communication Alternatices in a Containerized Cloud Environment*. Pranešimas konferencijoje: 11th International Conference on Cloud Computing (CLOUD), San Franciskas, JAV.

Gozani, M. (2016). *Network virtualization for dummies*. JAV: John Wiley & Sons, Inc.

Gross, J., Ganga, I. ir Sridhar, T. (2018*). Geneve: Generic Network Virtualization Encapsulation*. Prieiga per internetą: https://tools.ietf.org/html/draft-ietf-nvo3-geneve-06

Hanson, S. (2009). *Strategies to Optimize Virtual Machine Connectivity with Xsigo Virtual I/O*. Prieiga per internetą: https://www.dell.com/downloads/global/power/ps3q09-20090423-Xsigo.pdf

Hausenblas, M. (2018). *Container networking. From Docker to Kubernetes.* O'Reilly. 62 p. ISBN 978-1-492-03681-4.

Huang, D. ir Wu, H. (2017). *Mobile Cloud Computing – Foundations and Service Models*. JAV: Elsevier Inc.

Installing kubeadm. (n. d.). Prieiga per internetą: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

Installing addons. (n. d.). Prieiga per internetą: https://kubernetes.io/docs/concepts/cluster-administration/addons/

iPerf – The ultimate speed test tool for TCP, UDP and SCTP. (n. d.). Prieiga per internetą: https://iperf.fr/iperf-doc.php

Kapočius, N. (2019). *Chef scenarijų repozitorija*. Prieiga per internetą: https://github.com/Narunas-K/server-setup

Kapočius, N. (2020). Overview of Kubernetes CNI Plugins Performance. *Science – Future of Lithuania*, 12(1), 1–5. https://doi.org/10.3846/mla.2020.11454

Kubernetes Concepts. (n. d.). Prieiga per internetą: https://kubernetes.io/docs/concepts/

Lei, Q., Liao, W., Jiang, Y., Yang, M. ir Li, H. (2019, balandis). *Performance and Scalability Testing Strategy Based on Kubemark*. Pranešimas konferencijoje: IEEE 4th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), Chengdu, Kinija.

Lowe, S. (2020). *Using Paw to Launch an EC2 Instance via API Calls*. Prieiga per internetą: https://blog.scottlowe.org/2009/12/02/what-is-sr-iov/

Machine Zone, (2016). *Networking solutions for Kubernetes*. Prieiga per internetą: https://machinezone.github.io/research/networking-solutions-for-kubernetes/

Majkowski, M. (2016). *Why we use the Linux kernel's TCP stack*. Prieiga per internetą: https://blog.cloudflare.com/why-we-use-the-linux-kernels-tcp-stack/

Medel, V., Rana, O., Banares, A. J. ir Arronategui, U. (2017, Kovas). *Modelling Perfrmance & Resource Management in Kubernetes*. Pranešimas konferencijoje: 9th International Conference on Utility and Cloud Computing (UCC), Šanchajus, Kinija

Meier, K., Schmelzer, S. ir Suchodoletz, D. (2012*). Evaluation of Network Bonding Performance in Client/Server Scenarios*. Hochleistungreschnen in Baden-Wurttemberg – Ausgewahlte Aktivitaten im bwGRiD 2012 (221–234). Vokietija

Miano, S. et. al. (2019). *A Service-Agnostic Software Framework for Fast and Efficient In-Kernel Network Services*. Pranešimas konferencijoje: ACM/IEEE Symposium on Architecture for Networking and Communications Systems (ANCS), Kembridžas, JK.

PacketLife. (2012). *GRE vs IPIP Tunneling*. Prieiga per internetą: https://packetlife.net/blog/2012/feb/27/gre-vs-ipip-tunneling/

Park, V. ir Yang H. (2018, lapkritis). *Performance Analysis of CNI (Container Networking Interface)*. Pranešimas konferencijoje: International Conference of ICT convergence, Jeju ,Pietų Korėja.

Pirko, J. (2014). *If You Like Bonding, You Will Love Teaming*. Prieiga per internetą: https://www.redhat.com/en/blog/if-you-bonding-you-will-love-teaming

Project Calico. (2020*). Connecting workloads across network that you do not control*. Prieiga per internetą: https://docs.projectcalico.org/v3.8/networking/vxlan-ipip

Rattanaopas, K. ir Boonchuay K. (2017, spalis). *A high Performance Network Virtualization Achitecture with Bandwidth Guarantees*. Pranešimas konferencijoje: 5th International Electrical Engineering Congress (iEECON), Pattaya ,Tailandas.

Schmaus, B. (2017). *What is Geneve?*. Prieiga per internetą: https://www.redhat.com/en/blog/what-geneve

Simpson, W. (1995). *IP in IP Tunneling*. Prieiga per internetą: https://tools.ietf.org/html/rfc1853

Sookocheff, K. (2018). *Understating Kubernetes Networking Model*. Prieiga per internetą: https://sookocheff.com/post/kubernetes/understanding-kubernetes-networking-model/

Tanenbaum, A. S., Bos, H. (2014). *Modern Operating Systems (4th ed.).* Pearson. 1101 p. ISBN 978-0133591620.

Vayghan, A., Saied, A. M., Toeroe, M. ir Khendek, F. (2018). *Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned*. Pranešimas konferencijoje: 11th International Conference on Cloud Computing, San Franciskas, JAV.

Vennam, S. (2019). *Konteinerių orkestratorių veikimo apžvalga ir funkcijos*. Prieiga per internetą: https://www.ibm.com/cloud/blog/new-builders/container-orchestration-explained.

Vennam, S. (2019). *Kubernetes veikimo apžvalga*. Prieiga per internetą: https://www.ibm.com/cloud/learn/kubernetes

VM Ware, (2020). *Virtualization.* Prieiga per internetą: http://www.vmware.com/virtualization/history.html

VMGuru, (2015). *The Future of Virtual Networking fromVMworld – Part 2*. Prieiga per internetą: https://vmguru.com/2015/10/future-virtual-networking-vmworld-part-2/

Xien, X., Wang, P. ir Wang, Q. (2017). *The Performance Analysis of Docker and Rkt Based on Kubernetes*. Pranešimas konferencijoje: 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, Guilin, Kinija.

Zeng, H., Wang, B. ir Zhang, W. (2017). *Measurement and Evaluation for Docker Container Networking*. Pranešimas konferencijoje: International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Nanjing, Kinija.

PRIEDAI

A priedas. „Iperf3“ ankščių „Kubernetes“ konfigūraciniai failai

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: iperf3-client1
  labels:
    instance: first
    app: iperf
spec:
  containers:
  - name: iperf3-client1
    image: narunask/networkstatic-iperf
    imagePullPolicy: IfNotPresent
    command: ['/bin/bash', '-c', 'sleep infinity']
  nodeSelector:
    name: worker3
```

A priedas 1 pav. „Iperf3“ kliento ankšties konfigūracinis failas

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: iperf3-server1
  labels:
    instance: first
    app: iperf
    type: server
spec:
  containers:
  - name: iperf3-server1
    image: narunask/networkstatic-iperf
    imagePullPolicy: IfNotPresent
    args: ['-s']
  nodeSelector:
    name: worker1
```

A priedas 2 pav. „Iperf3“ serverio ankšties konfigūracinis failas

```yaml
apiVersion: v1
kind: Service
metadata:
  name: iperf3-server1
spec:
  selector:
    instance: first
    type: server
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5201
```

A priedas 3 pav. „Iperf3“ serverio paslaugos konfigūracinis failas

B priedas. „Iperf3“ serverio „bash“ scenarijus

```
#!/bin/sh
# By Narunas Kapocius
# 2020 01 05
# Iperf3 client run with sar logging script

# Read arguments
interface_ip=$1
if [ -z $interface_ip ]
then
  echo "Please provide inteface parameter. Example:"
  echo "./iperf-server-run.sh enp3s0f0"
  echo "Exiting.."
  exit
fi

# Variables
todays_date=`date +%Y%m%d-%H%M%S`
host=`hostname -f`

sar_results_dir="/tmp/phys-tcp-throughput-test/sar-results"
sar_results_file_name=$host-$todays_date-iperfserver-sar.results

sar_logs_dir="/tmp/phys-tcp-throughput-test/sar-logs"
sar_logs_file_name=$host-$todays_date-iperfserver-sar.ou

iperf_results_dir="/tmp/phys-tcp-throughput-test/iperf-results"
iperf_results_file_name=$host-$todays_date-iperfserver-iperf.results

idle_timer_before_measurement=5
performance_measurement_timer=5
idle_timer_after_measurement=5

# Script
echo "Starting CNI/Network performance measurement run"
echo $todays_date
echo $host

# Sar logging part
nohup sar -A 1 480 -o $sar_results_dir/$sar_results_file_name >
$sar_logs_dir/$sar_logs_file_name 2>&1 &
echo "sar results are stored in $sar_results_dir/$sar_results_file_name"
echo "sar run log is stored in $sar_logs_dir/$host-$todays_date-iperfserver-sar.out"

echo "going to sleep for $idle_timer_before_measurement seconds"
sleep $idle_timer_before_measurement

# Iperf3 logging part
echo "Starting iperf3 server"
echo "iperf3 results are stored in $iperf_results_dir/$iperf_results_file_name"
iperf3 -s -i $interface_ip -D --logfile $iperf_results_dir/$iperf_results_file_name

echo "going to sleep for $performance_measurement_timer seconds"
sleep $performance_measurement_timer

# Kill iper3 server
echo "killing iperf3 server which PID is: $(pidof iperf3)"
if [ -n $(pidof iperf3) ]
then
  kill -9 $(pidof iperf3)
```

```
else
  echo "there is no such a process as iperf3"
fi

echo "going to sleep for $idle_timer_after_measurement seconds"
sleep $idle_timer_after_measurement

# Kill sar
echo "killing sar process which PID is: $(pidof sar)"
if [ -n $(pidof sar) ]
then
  kill -9 $(pidof sar)
else
  echo "there is no such a process as sar"
fi

sleep 5
# Finish script run
echo "This run results can be found:"
echo "    sar results: $sar_results_dir/$sar_results_file_name"
echo "    iperf3 results: $iperf_results_dir/$iperf_results_file_name"
echo "Exiting.."
```

C priedas. „Iperf3“ kliento „bash“ scenarijus

```
#!/bin/sh
# By Narunas Kapocius
# 2020 01 05
# Iperf3 client run with sar logging script

# Read arguments
interface_ip=$1
server_ip=$2
if [ -z $interface_ip ] && [ -z $server_ip ]
then
  echo "Please provide interface_ip and server_ip parameters. Example"
  echo "./iperf-client-run.sh interface_ip iperf_server_ip"
  echo "Exiting.."
  exit
fi

# Variables
todays_date=`date +%Y%m%d-%H%M%S`
host=`hostname -f`
sar_results_dir="/tmp/phys-tcp-throughput-test/sar-results"
sar_results_file_name=$host-$todays_date-iperf-client-sar.results

sar_logs_dir="/tmp/phys-tcp-throughput-test/sar-logs"
sar_logs_file_name=$host-$todays_date-iperf-client-sar.out

iperf_results_dir="/tmp/phys-tcp-throughput-test/iperf-results"
iperf_results_file_name=$host-$todays_date-iperf-client.results

idle_timer_before_measurement=5
performance_measurement_timer=5
idle_timer_after_measurement=5

iperf_server_port=5201
iperf_loop_counter=0
max_iperf_client_retries=15
# Script
echo "Starting CNI/Network performance measurement run. Client side"
echo $todays_date
echo $host

# Sar logging part
nohup sar -A 1 480 -o $sar_results_dir/$sar_results_file_name >
$sar_logs_dir/$sar_logs_file_name 2>&1 &
echo "sar results are stored in $sar_results_dir/$sar_results_file_name"
echo "sar run log is stored in $sar_logs_dir/$sar_logs_file_name"

echo "going to sleep for $idle_timer_before_measurement seconds"
sleep $idle_timer_before_measurement

# Iperf3 logging part
echo "Starting iperf3 client"
echo "iperf3 results are stored in $iperf_results_dir/$iperf_results_file_name"
until nc -v -z $server_ip $iperf_server_port
do
  echo "waiting for iperf server to come up"
  sleep 1
  if [ $iperf_loop_counter -eq $max_iperf_client_retries ]
  then
    echo "Could not connect to iperf sever even after $iperf_loop_counter retries.
```

```
Aborting.."
    kill -9 $(pidof sar)
    exit
  fi
  iperf_loop_counter=$((iperf_loop_counter+1))
  echo $iperf_loop_counter
done

#start iperf client
iperf3 -c $server_ip -B $interface_ip --logfile
$iperf_results_dir/$iperf_results_file_name
echo "going to sleep for $performance_measurement_timer seconds"
sleep $performance_measurement_timer

# Kill iper3 client
echo "killing iperf3 client which PID is: $(pidof iperf3)"
if [ -n $(pidof iperf3) ]
then
  kill -9 $(pidof iperf3)
else
  echo "there is no such a process as iperf3"
fi

echo "going to sleep for $idle_timer_after_measurement seconds"
sleep $idle_timer_after_measurement

# Kill sar
echo "killing sar process which PID is: $(pidof sar)"
if [ -n $(pidof sar) ]
then
  kill -9 $(pidof sar)
else
  echo "there is no such a process as sar"
fi

sleep 5
# Finish script run
echo "This run results can be found:"
echo "    sar results: $sar_results_dir/$sar_results_file_name"
echo "    iperf3 results: $iperf_results_dir/$iperf_results_file_name"
echo "Exiting.."
```

D priedas. „Iperf3“ rezultatų apdorojimo scenarijus

```
#/bin/sh
# By Narunas Kapocius
# 2020 02 02
# Iperf3 results file parser script

#Variables
current_work_dir=`pwd`

#Read arguments
source_dir=$1
results_file=$2
parse_regex=$3
if [ -z $source_dir ] || [ -z $results_file ] || [ -z $parse_regex ]
then
  echo "Please provide source data directory, results file name to which write
results, and if to use regex to parse teamed interfaces results"
  echo "./iperf-results-parser.sh source-data-dir results-file-name
team|kube|baremetal"
  echo "Exiting.."
  exit
fi

cd "$source_dir"
echo -n  "$(ls > $current_work_dir/filesList.txt)"

files_array=`cat $current_work_dir/filesList.txt`

#clean results file
echo "" > $current_work_dir/$results_file

for file_name in $files_array
do
  if [ $parse_regex == "team" ]
  then
    test_name=`echo "$file_name" | grep -Po "kw\d\w{4}\d_kw\d\w{4}\d_\w{8}-\d_\w{3}-
\d{2,4}_\d-\d{8}-\d{6}"`
  elif [ $parse_regex == "kube" ]
  then
    test_name=`echo "$file_name" | grep -Po "client\d_instance-\d_\w{3}-\d{2,4}_\d-
\d{8}-\d{6}"`
  else
    test_name=`echo "$file_name" | grep -Po "kw\d\w{3}\d_kw\d\w{3}\d_\w{3}-
\d{2,4}_\d-\d{8}-\d{6}"`
  fi
  results=`grep "0.00-120.00" $source_dir/$file_name | grep -Po "\d+ Mbits" | grep -
Po "\d+"`
  results=`echo $results | tr '\n' ' '`
  printf "$test_name $results \\n" >> $current_work_dir/$results_file
done
```

E priedas. „Chef“ „Linux“ tvarkaraščių kūrimo scenarijus

```
###variables
$start_hour = 03
$start_min = 10
$minutes_from_midnight = $start_hour*60+$start_min
#Iperf server interfaces names
$kw1team1='team1'
iperf_server_instances_ports_array = [5000, 5001]
#Interfaces arrays
iperf_server_interfaces_array = [$kw1team1]
iperf_server_interfaces_names_array = ["kw1team1"]
iperf_client_interfaces_names_array = ["kw3team2"]

mss_values_array = [88, 216, 472, 984, 1460]
#mss_values_array = [2008, 4056, 8152, 8960]
#interfaces_number_array = [0, 1, 2, 3, 4]
interfaces_number_array = [0]
number_of_iperf_instances_array = [0, 1]
test_number_array = [0, 1, 2, 3, 4]
$global_test_number = 0
$cron_hour = 0
$cron_min = 0

#Cron creation loop
##Assuming that one measurement is taken every 6 minutes and no more than 240
measurements are taken during a 24hr day
for interface_number in  interfaces_number_array do
  for test_number in test_number_array do
    for mss_value in mss_values_array do
      for instance_number in number_of_iperf_instances_array do
        until $minutes_from_midnight <1440 do
          $minutes_from_midnight = $minutes_from_midnight - 1440
        end
        $cron_hour = $minutes_from_midnight/60.floor
        $cron_min = $minutes_from_midnight-($cron_hour*60)
        if "#{instance_number}" == "0"
          cron
"#{iperf_server_interfaces_names_array[interface_number]}_#{iperf_client_interfaces_nam
es_array[interface_number]}_instance-#{instance_number}_mss-
#{mss_value}_#{test_number}" do
            hour "#$cron_hour"
            minute "#$cron_min"
            user 'narunas'
            command "/home/narunas/repos/iperf3-scripts/iperf-server-run.sh
#{iperf_server_interfaces_array[interface_number]}
#{iperf_server_instances_ports_array[instance_number]}
#{iperf_server_interfaces_names_array[interface_number]}_#{iperf_client_interfaces_name
s_array[interface_number]}_instance-#{instance_number}_mss-#{mss_value}_#{test_number}
> /tmp/phys-tcp-throughput-test/script-
logs/#{iperf_server_interfaces_names_array[interface_number]}_#{iperf_client_interfaces
_names_array[interface_number]}_instance-#{instance_number}_mss-
#{mss_value}_#{test_number} 2>&1"
          end
        else
          cron
"#{iperf_server_interfaces_names_array[interface_number]}_#{iperf_client_interfaces_nam
es_array[interface_number]}_instance-#{instance_number}_mss-
#{mss_value}_#{test_number}" do
            hour "#$cron_hour"
            minute "#$cron_min"
```

```
            user 'narunas'
            command "/home/narunas/repos/iperf3-scripts/iperf-server-run-without-sar.sh
#{iperf_server_interfaces_array[interface_number]}
#{iperf_server_instances_ports_array[instance_number]}
#{iperf_server_interfaces_names_array[interface_number]}_#{iperf_client_interfaces_name
s_array[interface_number]}_instance-#{instance_number}_mss-#{mss_value}_#{test_number}
> /tmp/phys-tcp-throughput-test/script-
logs/#{iperf_server_interfaces_names_array[interface_number]}_#{iperf_client_interfaces
_names_array[interface_number]}_instance-#{instance_number}_mss-
#{mss_value}_#{test_number} 2>&1"
          end
        end
      end
    $global_test_number = $global_test_number + 1
    $minutes_from_midnight = $minutes_from_midnight + 6
    end
  end
end

#After tests are completed cleanup cron
$cron_hour = $minutes_from_midnight/60.floor
$cron_min = $minutes_from_midnight-($cron_hour*60)
cron 'Cleanup cron' do
  hour "#$cron_hour"
  minute "#$cron_min"
  user 'narunas'
  command "/usr/bin/crontab -r "
end
```

F priedas. Fizinės terpės TCP protokolo vidutinio pralaidumo rezultatai

F priedas 1 lentelė. Jungtųjų tinklo sąsajų fizinės terpės TCP protokolo suminio pralaidumo rezultatai, Mbit/s

	MSS dydis, B								
Jungtųjų sąsajų skaičius	**76**	**204**	**460**	**972**	**1448**	**1996**	**4044**	**8140**	**8948**
„team2“	548,8	1353,4	1642,8	1816,4	1867,6	1904,4	1920	1906,6	1924
„team3“	827,6	2026,2	2466,6	2713,8	2792,6	2852,4	2870,2	2849,2	2932
„team4“	1286,4	2697,6	3302,6	3619,8	3732,6	3818,2	3874,6	3844	3914,6
„team5“	1741,2	3406,8	4136,6	4530,8	4653,4	4776,2	4851,8	4840,4	4909,6

G priedas. Dalyvavimo tarptautinėje konferencijoje „eStream 2020“ sertifikatas

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS

Open International Conference "Electrical, Electronic and Information Sciences"

eStream 2020

CERTIFICATE

Narūnas Kapočius

Delivered an Oral Presentation

PERFORMANCE STUDIES OF KUBERNETES NETWORK SOLUTIONS

Chair of Science Committee Prof. Dr Dalius Navakauskas

Section Chair Prof. dr Algirdas Baškys