TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Tehnoloogiainstituut

Johani Vajakas

Kubernetese võrgukihi tehnoloogiate jõudluse võrdlemine füüsilistel serveritel

Bakalaureusetöö (12 EAP)
Arvutitehnika eriala

Juhendajad:
Ivar Koppel, PhD
Sander Kuusemets, BSc

Tartu 2020

Resümee/Abstract

Kubernetese võrgukihi tehnoloogiate jõudluse võrdlemine füüsilistel serveritel

Konteinerite üks populaarsemaid haldussüsteeme Kubernetes koosneb mitmest seadistatavast alamsüsteemist. Kubernetese seadistuse alguses valitakse võrgutehnoloogia, mis võib olla kõige tähtsam süsteemi osa, sest see tagab ühenduse teenuste vahel. Käesolev töö võrdleb Kubernetese võrgukihtide jõudlust füüsilistel serveritel, et uurida võrgutehnoloogiate võimekust 100 Gbit/s andmekeskuse võrgus. Töö tulemused näitavad, et võrgutehnoloogiad saavutavad üle 50 Gbit/s ning ühenduskiirused võivad vastavalt seadistusest erineda sadu kordi.

Märkõnad: Kubernetes, virtualiseerimine, võrk, konteinerdus

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia

Comparison of Kubernetes network plugins on bare metal servers

One of the more popular container orchestration platforms Kubernetes consists of multiple configurable components. Choosing a network plugin is a critical step during the deployment phase of a cluster because it provides connectivity between containers. The goal of this thesis is to compare Kubernetes cluster network plugins on bare metal servers running on 100 Gbps capable hardware. The results of the thesis show that some of the plugins can achieve over 50 Gbps network throughput and the speeds differ over 100 times, depending of the setup.

Keywords: Kubernetes, virtualization, network, containerization

CERCS: T120 Systems engineering, computer technology

Sisukord

Jooniste loetelu

Tabelite loetelu

Mõisted ja terminid

Konteiner (ingl *container*) on abstraktne andmetüüp.

Sõlm (ingl *node*) on võrgu olem, mis on ühendatud ühe või mitme muu olemiga.

Pod on Kubernetese objekt, mis koosneb ühest või mitmest konteinerist ning jagavad püsimälu- ning võrguressursse. Konteinerivaheline suhtlus toimub kasutades *localhost* aadressi. *Pod*'i konteinerid asetsevad alati samal sõlmel.[1]

Klaster (ingl *cluster*) on rühm lõdvalt seotud, kuid tihedas koostöös arvuteid, mis näiliselt töötab ühe arvutina; rööptöötluse lihtsaim vorm, nt ühisketastega hostide kogum või multitöötluseks ja/või käideldavuse tõstmiseks kokku ühendatud arvutikogum.

API (ingl *application programming interface*) on rakendusliides. Tarkvara rakendamise protokollide, reeglite ja vahendite kogum. Lihtsustab programmeerijate tööd, sest võimaldab sisestada arvutiprogrammidesse eelnevalt valmis tehtud tarkvara.

Tunneldamine on meetod kahe võrgu ühendamiseks kolmanda kaudu, nii et ühendatud liiklus on täielikult isoleeritud muust liiklusest kolmandas võrgus.

VxLAN (ingl *Virtual Extensible LAN*) on tunneldamisel kasutatav protokoll.

IP-in-IP on tunneldamisel kasutatav protokoll.

BPF (ingl *Berkeley Packet Filter*) on tehnoloogia, millega on võimalik efektiivselt võrguliiklust filtreerida.

Geneve - (ingl *Generic Network Virtualization Encapsulation*) on tunneldamisel kasutatav protokoll, milles kapseldatakse kaader UDP paketti.[2]

IPSec (ingl *Internet Protocol Security*) on andmeturbe standard võrgu- või paketitöötluskihi tasemel.

BGP (ingl *Border Gateway Protocol*) on välislüüsiprotokoll marsruutimis- ja ulatuvusteabe vahetuseks.

JSON (ingl *JavaScript object notation*) - andmevahetusvorming.

YAML (ingl *YAML Ain't Markup Language*) on inimloetav andmete jadastuse keel.

Mastaapima - muutma mingi suuruse esitust, väljendades ta teistes ühikutes, nii et ta muutumispiirkond siirduks etteantud vahemikku.

Volitustõend (ingl *token*) on objekt, mis tõendab õigust sooritada mingit toimingut.

NAT (ingl *Network Address Translation*) on marsruutimise tehnoloogia, mis võimaldab välisvõrgu eest peita sisemisi aadresse.

[1] https://kubernetes.io/docs/concepts/workloads/pods/pod/
[2] https://datatracker.ietf.org/doc/draft-ietf-nvo3-geneve/

1. Sissejuhatus

Konteinerduse populaarsuse tõus on toonud kaasa andmekeskustes konteinerite arvu kasvu ning vajaduse neid ühtselt hallata. Üks populaarsemaid haldussüsteeme on Kubernetes, mis koosneb mitmest seadistatavast alamsüsteemist. Kubernetese seadistuse alguses valitakse võrgutehnoloogia, mis võib olla kõige tähtsam süsteemi osa, sest see tagab ühenduse teenuste vahel.

Käesolev töö võrdleb Kubernetese võrgukihtide jõudlust füüsilistel serveritel, et uurida võrgutehnoloogiate võimekust 100 Gbit/s andmekeskuse võrgus. Eelnevad tööd on võrrelnud süsteemide võrgukiirusi kuni 10 Gbit/s võimaldavatel ühendustel [1] [2] või uurinud ainult üht võrgutehnoloogiat [3]. Võrgutehnoloogiate TCP ühenduste läbilaskevõime langus 1500 baidise kaadri suurusega võrreldes füüsiliste serverite võimekusega on Narūnas Kapočius tehtud töös [1] märgatav (~25-50%) ning ei ole teada, kas kiiremates võrkudes on näha sarnast langust.

Töö sisu on jaotatud nelja ossa. Peatükis 2 tuuakse ülevaade Kubernetese ülesehitusest, kasutatavast riistvarast ja süsteemide seadistustest. 3. peatükis antakse ülevaade vaadeldavatest võrgutehnoloogiatest. Peatükis 4 kirjeldatakse kasutatavad testprogrammid ning testimise metoodika ning 5. peatükis analüüsitakse saadud tulemusi.

2. Kubernetese kirjeldus

Peatükk annab ülevaate Kubernetese süsteemi ülesehitusest ning komponentidest kasutades Kubernetese dokumentatsiooni [4]. Seejärel tuuakse välja töös kasutatava riistvara omadused ning klastri seadistamise protseduur.

2.1 Ülesehitus

Kubernetes on klastersüsteem, mis koosneb sõlmedest. Klastris on vähemalt üks juhtsõlm (ingl *master node*) ja üks arvutussõlm (ingl *worker node*). Arvutussõlm majutab rakenduste *pod'*e. Juhtsõlm haldab arvutussõlme(de)s olevaid *pod'*e ning majutab klastri töö haldamiseks vajalike juhttasandi teenuseid.

Juhttasandi teenused:

- *Kube-apiserver* – Kubernetese juhtsüsteemi API teenus.
- *Etcd* – klastri objektide andmebaas.
- Kube-scheduler – määrab *pod*'ile sõlme vastavalt *pod*'i ressursinõuetele.
- *Kube-controller-manager* – jälgib klastri sõlmede kättesaadavust, kontrollib käivitatud *pod*'ide arvu vastavust soovitule ning haldab uute nimeruumide loomisel vaikekontode ja API ligipääsu võtmete seadistamist.
- *Cloud-controller-manager* – ühendab Kubernetese API väliste pilveteenuste süsteemidega.

Juhttasandi teenused töötavad ainult juhtsõlmel. Järgnevad teenused töötavad juht- ning arvutussõlmedel:

- *Kubelet* – kontrollib ja haldab Kubernetese poolt tekitatud konteinerite olekut.
- *Kube-proxy* – haldab sõlmedel töötavate *pod'*idele seadistatud *Service* objektide võrguühendusi.
- *Container runtime* – tarkvara, mis tagab konteinerite kasutamiseks vastava keskkonna.

Lisaks loetletud juht- ja arvutussõlme komponentidele on võimalik seadistada mitmeid klastrit toetavaid teenuseid. Võrguteenus kategoriseeritakse toetavate teenuste alla.

2.2 Objektid

Kubernetese klastrit seadistatakse kasutades objekte, mis defineerivad klastri hetkelist ning soovitud olekut. Igal objektil on selleks kaks väärtust: staatus, mis kirjeldab objekti hetkelist seisu, ja spetsifikatsioon, mis kirjeldab objekti soovitud seadistust.

Objektid liigituvad olemuselt kaheks:

- Baasobjektid:
    - *Pod* – ühest või enamast konteinerist koosnev üksus.
    - *Service* – seadistab ühel või mitmel *pod*'il seadistatud teenuse ühenduspunkti.
    - *Volume* – seob sõlmel oleva kausta *pod*'i süsteemi kaustaga.

    - *Namespace* – virtuaalne klastri keskkond.
- Teenusobjektid:
    - *Deployment* – kirjeldab *pod*'ide ja *ReplicaSet'*ide soovitud olekut.
    - *DaemonSet* – kirjeldab *pod*'i oleku, mis seadistatakse kõikidele (või mingile grupile) sõlmedele.
    - *StatefulSet* – kirjeldab *pod*'ide oleku, mis vajavad püsivaid ja unikaalseid domeeninimesid, püsitalletusega mälu, järjestatud mastaapimist või järjestatud ja hallatud uuendamist.
    - *ReplicaSet* – kirjeldab *pod*'ide soovitud olekut.
    - *Job* – seadistab üks või mitu *pod'*i ning kontrollib, et vajalik arv seadistatud *pod*'e lõpetab edukalt.

Objekte luuakse vastava JSON tüüpi päringuga API poole, kus kirjeldatakse nende soovitud olekut ning baasinfot. Töös kasutan objektide loomiseks kubectl programmi ning YAML formaadis kirjutatud faile. Kubectl tõlgendab faili sisu vastavateks JSON tüüpi päringuteks.

API päring autenditakse kasutades API serverit, mida on võimalik seadistada erinevate autentimise viisidega. Töös kasutatakse klastri tekitamisel volitustõendi põhist autentimist. Vaadeldavad võrgutehnoloogiad seadistavad teenuse tööks vajalikud rollipõhised ligipääsud, mis koosnevad järgnevatest objektidest:

- *Role* – seob klastri nimeruumi, ressursi (objekt, millele antakse õigused) ning õigused.
- *RoleBinding* – seob *role* tüüpi objekti kasutaja või grupiga.

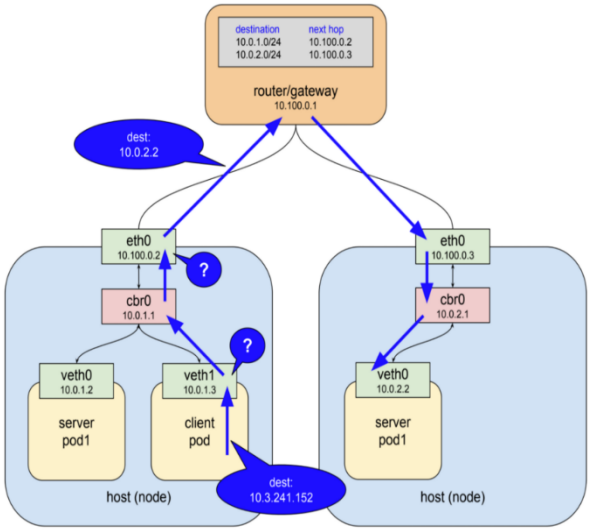
Ligipääsude seadistused on kirjeldatud võrgutehnoloogia algseadistusfailis.

2.3 Võrgukiht

Võrgukiht haldab klastri sees olevaid võrguühendusi ning võimaldab seadistada klastris seadistatud teenustele ligipääsu väljastpoolt klastrit. Kõikide võrgutehnoloogiate juures kehtivad järgnevad reeglid:

- Üks *pod* saab suhelda teiste *pod'*idega kõikidel sõlmedel NAT-ita.
- Sõlmel töötavad teenused saavad suhelda kõikide samal sõlmel olevate *pod*'idega.

Klastrisiseseks ühenduseks antakse *pod*'ile klastri võrguvahemikust unikaalne võrguaadress, mis on klastri piires nähtav igast sõlmest NAT-i kasutamata. *Pod*'is olevad konteinerid suhtlevad üksteisega kasutades localhost aadressi. *Pod*'i konteinerid jagavad ühist pordivahemiku on vaja kontrollida, et kaks konteinerit ei kasutaks sama porti. Joonis 1 näitab, milline on ühendus kahe *pod*'i vahel kahe sõlmega klastris, kui klient ja server on erinevatel sõlmedel.



Joonis 1. Paketi tee kahe sõlme vahel, mis asuvad eri *node*'del[3]

Klastris seadistatud *pod*'il töötavale teenusele ligipääsemiseks väljastpoolt klastrit seadistatakse *Service* objekt. Objektiga määratakse ära teenuse väline võrguaadress, pordid ning millised *pod*'id on objektiga seotud. Ühendused suunatakse vajalikele *pod*'idele kasutades proksi lahendust kasutades *kube-proxy* teenust või välist koormusjaoturit. Mõndadel võrgukihi implementatsioonidel on loodud eraldiseisev koormusjaotur, mis täidab *kube-proxy* ülesandeid.

Pod'ide võrguliikluse piiramiseks kasutatakse Kubernetese klastris objekti *NetworkPolicy*. Objekti kasutamiseks on vaja seadistada võrgutehnoloogia, mis toetab võrguliikluse piiramist. Kui *pod*'ile ei rakendu ühtegi *NetworkPolicy* objekti, on võimalik kasutada *pod*'iga suhtlemiseks kõiki porte. *NetworkPolicy* rakendamisel keelatakse kõik ühendused, mis ei ole objektis lubatud.

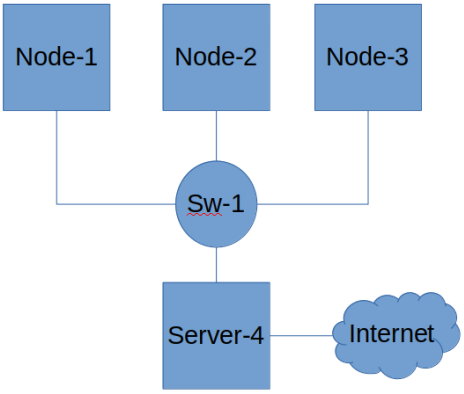
2.4 Klastri seadistamine

Käesolevas töös seadistan Kubernetese klastri ühe juhtsõlme ja kahe arvutussõlmega. Lisaks on kasutusel server, mis ei kuulu klastrisse ning pakub klastri võrgule ühendust Internetiga. Serverid on ühendatud kohtvõrku kasutades kommutaatorit.

2.4.1 Riistvara seadistus

Testimise jaoks on kasutusel kaks Supermicro 6028TP-DNCR serverit, igal kaks lehtserverit, ja Mellanox SN2100-CB2F kommutaator. Seadmete spetsifikatsioonid on kirjas lisas 1. Iga serveri alamsüsteem on ühendatud kommutaatoriga kasutades 100 Gbit/s kiirust võimaldavat AOC-S100G-M2C võrgukaarti ning 100GBASE-CR4 tüüpi liidest. Kommutaatori ja serverite liidesed on seadistatud kasutama 1500 baidist kaadri suurust (MTU'd). Ühenduste skeem on toodud joonisel 2.

[3] https://kubernetes.io/blog/2018/07/18/11-ways-not-to-get-hacked/



Joonis 2. Seadmete omavaheline ühendusskeem

Serveritele on paigaldatud operatsioonisüsteem Ubuntu 18.04 kerneliga 4.15.0-55.

2.4.2 Kubernetese seadistamine

Serverite algseadistamiseks paigaldakse vajalikud programmid kasutades programmi Ansible ning Kubernetese paigalduse seadistust [4] punktides 2.1 kuni 2.3. Välja jäeti punktis 2.1 kirjeldatud kasutaja *vagrant* gruppi lisamine ning muudeti punktis 2.1 Ubuntu versioon *bionic*'uks, sest ei kasutata Vagrant süsteemi ning kasutatakse Ubuntu versiooni 18.04. Antud töö kirjutamise hetkel ei ole Kubernetese programmide hoidlat Ubuntu 18.04 süsteemile (koodnimi *Bionic Beaver*) ning kasutatakse Kubernetese Ubuntu 16.04 (koodnimi *Xenial Xerus*) hoidlat. Kasutatud docker-ce, docker-ce-cli, containerd.io, kubelet, kubeadm, kubectl programmi versioonid on toodud tabelis 1.

Tabel 1. Kasutatud programmide versioonid

Programm	**Versioon**
Docker-ce	19.03.8
Docker-ce-cli	19.03.8
Containerd.io	1.2.13
Kubelet	1.17.3
Kubeadm	1.17.3
Kubectl	1.17.3

Klastri algseadistamisel valitakse *pod*'ide võrguvahemikuks 10.100.0.0/16 käsuga *kubeadm init --apiserver-advertise-address="192.168.1.11" --apiserver-cert-extra-sans="192.168.1.11" --node-name node-1 --pod-network-cidr=10.100.0.0/16*. Kui

[4] https://kubernetes.io/blog/2019/03/15/kubernetes-setup-using-ansible-and-vagrant/

võrgutehnoloogia juhendis täpsustatakse kasutatav *pod*'ide võrguvahemik, seadistatakse *--pod-network-cidr* vastavalt soovitule. Kasutatud võrguaadressi vahemikud on toodud tabelis 2. Klastri sõlmede ühendamiseks kasutatakse järgnevat juhtsõlme käsu väljundit:

kubeadm token create --print-join-command

Pärast sõlmede liitmist seadistatakse võrgutehnoloogia kasutades kubectl programmi. Klastri seadistuse korrektsust kontrollitakse käsuga *kubectl get nodes*, mille väljundis peab iga sõlme olek olema *Ready*.

3. Vaadeldavad võrgutehnoloogiad

Võrreldavad võrgutehnoloogiad valiti Kubernetese poolt toodud nimekirja [5] ning eelnevates töödes [1] [2] esinenud võrgutehnoloogiate seast. Töös keskendutakse võrgutehnoloogiatele, mis on kasutatavad füüsilistel serveritel, on tasuta kättesaadavad ning ei nõua spetsiifiliste omadustega riistvara või lisatarkvara kasutamist. Nimekiri koos algseadistusfaili lingiga on toodud tabelis 2.

Tabel 2. Võrreldavate võrgutehnoloogiate nimekiri

Nimi	**Link algseadistusfailile**	**Kasutatud *pod*'ide võrguvahemik**	**Märkus**
Flannel	https://raw.githubusercontent.com/coreos/flannel/v0.12.0/Documentation/kube-flannel.yml	10.244.0.0/16	Kasutati VxLAN kapseldusega ning kapselduseta seadistust. Versioon v0.12.0.
Project Calico	https://docs.projectcalico.org/manifests/calico.yaml	10.100.0.0/16	Kasutati IP-in-IP ja VxLAN kapseldusega ning kapselduseta seadistust. Versioon v3.13.1.
Canal	https://docs.projectcalico.org/manifests/canal.yaml	10.244.0.0/16	Kasutati VxLAN kapseldusega ning kapselduseta seadistust. Canal versioon, mis seadistab Calico versiooni v3.13.2 ja Flannel versiooni v0.11.0.
Cilium	https://raw.githubusercontent.com/cilium/cilium/v1.7.2/install/kubernetes/quick-install.yaml	10.217.0.0/16	Kasutati VxLAN ja Geneve kapseldusega ning kapselduseta seadistust. Kapselduseta seadistusel kasutati sõlmede vahel staatilist marsruutimist. Versioon v.1.7.2.

[5] https://kubernetes.io/docs/concepts/cluster-administration/networking/#how-to-implement-the-kubernetes-networking-model

Kube-router	https://raw.githubusercontent.com/cloudnativelabs/kube-router/v0.4.0/daemonset/kubeadm-kuberouter-all-features.yaml	10.100.0.0/16	Kasutati vaikimisi seadistust. Versioon v0.4.0.
Weave Net from Weaveworks	https://cloud.weave.works/k8s/net?k8s-version=Q2xpZW50IFZlcnNpb246IHZlcnNpb24uSW5mb3tNY-WpvcjoiMSI-sIE1pbm9yOiIxNyIsIEd-pdFZlcnNpb246InYxLjE3LjMiLCBHaXRDb21taXQ6IjA2YWQ5NjBiZmQwM2IzOWM4MzEwYWFmOT-JkMWU3YzEyY2U2MTgyMTMiLCBHaXRUcmVlU3RhdGU6ImNsZWFuIiwgQn-VpbGREYXRlOiIyMDIw	10.32.0.0/12	Kasutati vaikimisi ning krüpteeritud seadistust. Vaikimisi seadistusel kasutati *Fast Datapath* ühenduse tüüpi ning krüpteeritud ühendusel *sleeve* tüüpi. Versioon 2.6.2.

Võrgutehnoloogiate seadistamisel kasutati süsteemi kodulehel olevat teavet ning juhendeid. Järgnevates alapunktides kirjeldatakse võrgutehnoloogiaid vastavalt nende dokumentatsioonile.

3.1 Flannel

Flannel [5] on väheste lisavõimalustega võrgutehnoloogia, mis toetab Linuxi ja Windowsi operatsioonisüsteemi kasutatavat Kubernetese sõlmesid. Süsteemi algseadistamiseks on võimalik kasutada kubectl utiliiti ja Kubespray [6] või KOPS [7] lahendust. Flannel on mõeldud ainult *pod*'ide vahelise transpordikihi seadistamiseks ning võrgupiirangute jaoks on vaja kasutada lisatehnoloogiat, nt Calico. Flanneli algseadistamisel tekitatakse *DaemonSet* tüüpi *pod*, mis juhib sõlmede vahelist võrguliiklust. Transpordikihina on toetatud kapseldamine kasutades VxLAN tehnoloogiat või kapseldamata marsruutimine kohtvõrgus. Vaikimisi kasutatakse VxLAN kapseldamist. Flanneli seadistamiseks kapseldamata marsruutimisele on vaja algseadistusfailis asendada märksõna *vxlan* sõnaga *host-gw*. Flannel ei toeta kirjutamise hetkel IPv6 protokolli[6].

3.2 Cilium

Cillium [8] on võrgutehnoloogia, mis on võimeline filtreerima võrguliiklust OSI mudeli 3., 4. ja osaliselt 7. kihi reeglitega. Toetatud on ainult Linuxi operatsioonisüsteemil töötavad sõlmed. Süsteemi saab algseadistada kasutades kubectl utiliiti ja Kubespray [6] või KOPS [7] lahendust. Cilium toetab IPv4 ja IPv6 protokolli ning kasutab BPF tehnoloogiat, kui sõlme kernel seda toetab. Transpordikihina on toetatud kapseldamine kasutades VxLAN ja Geneve tehnoloogiat ning sõlmedede vaheline marsruutimine ilma kapselduseta. Vaikimisi

[6] https://github.com/coreos/flannel/issues/248

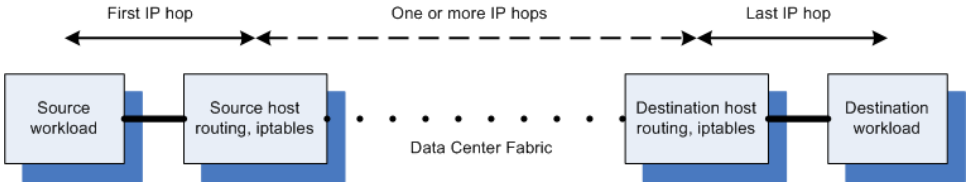
kasutatakse VxLAN kapseldamist. Kapselduseta ülesehitusel on vaja seadistada sõlmede vaheline marsruutimine. Võrguliiklust on võimalik krüpteerida kasutades IPsec-i. Ciliumi algseadistamisel luuakse *DaemonSet* tüüpi *pod*, mis juhib sõlmede võrguliiklust ning *Deployment* tüüpi *pod*, mis haldab võrgutehnoloogia tööd. *DaemonSet* tüüpi *Pod*'is käivitatakse *cilium-agent*, mis haldab sõlmel BPF-i jaoks tehtavaid programme. Võrgutehnoloogia toetab meetrika saatmist kasutades Prometheust ja dünaamilist marsruutimist kasutades BGP protokolli läbi *kube-router* või BIRD lahenduse.

3.3 Project Calico

Project Calico [9] toetab vabavaraliselt Linux operatsioonisüsteemil töötavat Kubernetese sõlmesid. Süsteemi on võimalik seadistada kohapealsele Kubernetese või Openstack klastrile ning Amazoni, Microsofti või Google'i pilvelahendusele. Seadistamiseks saab kasutada kubectl utiliiti ja Kubespray [6] või KOPS [7] lahendust. Transpordikihina on toetatud IP-in-IP ja VxLAN kapseldamine ning sõlmede vaheline marsruutimine ilma kapselduseta. Vaikimisi seadistatakse võrk kasutades IP-in-IP kapseldust. Project Calico koosneb järgmistest komponentidest:

- Felix – agent, mis seadistatakse igale sõlmele, millel on konteinerid.
    - Kontollib sõlme liideseid, ruutingutabelit, haldab võrguühenduste pääsuloendit.
- *Orchestrator plugin* – vahendab klastris kasutatava kontrolleri käsud Calico API käskudeks.
- Etcd – andmebaas süsteemi info hoidmiseks ning komponentide vaheliseks andmete vahetuseks.
- BIRD – dünaamilise marsruutimise agent kasutades BGP-d.

Võrguliiklus läbib alati sõlme, kus on võimalik liiklust piirata OSI mudeli 3. ja 4. kihi reeglitega kasutades iptables programmi. 5.-7. kihi reeglite seadistamine vajab Istio teenuse seadistamist. Võrguliikluse piirang, mis tehakse kahe klastris seadistatud süsteemi vahel, seadistatakse kõikidele sõlmedele, millel piirangus käsitletud süsteem jookseb. Selline lähenemine vähendab ebavajalikku liiklust võrgus, sest liiklus, mis ei ole lubatud, piiratakse ühenduse tekitaja sõlmel. Joonis 3 illustreerib kahe *pod*'i vahelist liiklust.



Joonis 3. Project Calico *pod*'ide vaheline liiklus [7]

Project Calico seadistamisel tekitatakse *DaemonSet* tüüpi *pod*, mis juhib võrguliiklust ning *Deployment* tüüpi *pod*, mis jälgib Kubernetese klastri sündmusi ning juhib võrgutehnoloogia tööd. Võrgutehnoloogia seire jaoks on vaja seadistada Felix saatma meetrikat Prometheuse süsteemi.

[7] https://docs.projectcalico.org/reference/architecture/data-path

3.4 Canal

Võrgutehnoloogia, kus on paigaldatud Flannel võrgutehnoloogia sõlmede vaheliseks ühenduseks koos Project Calico tehnoloogiaga võrguliikluse piiramiseks [8]. Klastri sõlmedele kehtivad mõlema süsteemi nõuded. Seadistamiseks on võimalik kasutada kubectl, KOPS [7] ja Kubespray [6] utiliiti.

3.5 Kube-router

Kube-router [10] võrgutehnoloogia on toetatud Linux operatsioonisüsteemil. Seadistamiseks on võimalik kasutada kubectl, KOPS [7], Kubespray [6] või bootkube utiliiti. Seadistamisel tekitatakse *DaemonSet* tüüpi *pod*, mis juhib võrguliiklust kasutades iptables reegleid. Võrguliiklust ei kapseldata ning piiranguid on võimalik teha *pod*'idele sisenevas ja väljuvas suunas. Kube-router toetab dünaamilist marsruutimist klastri sees ning marsruutimisteabe jagamist klastri väliste marsruuteritega kasutades autentimiseta või MD5 autentimisega BGP protokolli.

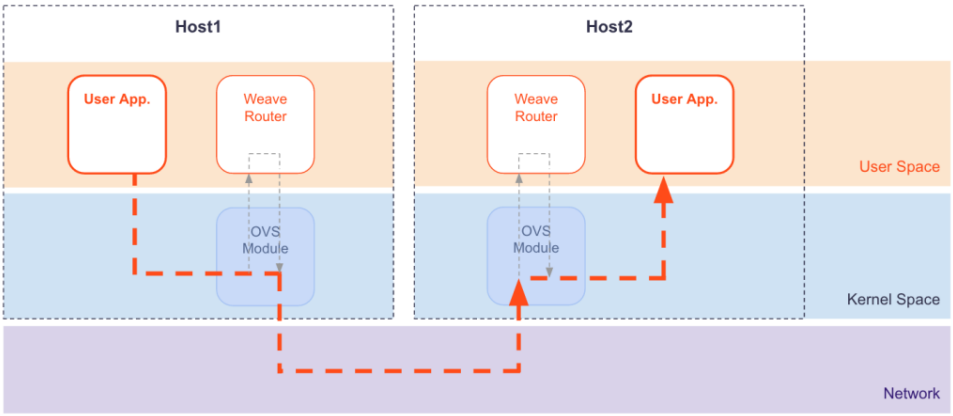
3.6 Weave Net from Weaveworks

Weave Net [11] võrgutehnoloogia toetab Linux operatsioonisüsteemi kasutavat Kubernetes klastrit. Süsteemi saab algseadistada kasutades KOPS [7], Kubespray [6] või kubectl utiliiti. Algseadistamisel tekitatakse *DaemonSet* tüüpi *pod*, mis juhib võrguliiklust. Weave Net kasutab erinevatel sõlmeldel olevate *pod*'ide vahel pakettide marsruutimiseks kahte moodust:

- *Fast Datapath* – sõlmest väljuvaid pakette töödeldakse kerneli tasemel seadistades sõlme kernelit läbi *Open vSwitch*i mooduli.
- *Sleeve* – sõlmest väljuv võrguliiklus töödeldakse kasutaja tasemel ning kapseldatakse UDP pakettidesse. Ühte paketti mahutatakse maksimaalne arv kaadreid.

Vaikimisi kasutatakse Fast Datapath ühendust ning tüübi valik tehakse ühenduse loomisel. *Sleeve* valitakse juhul, kui *Fast Datapath*'i ei saa kasutada. Joonis 4 visualiseerib paketi liikumist kahe *pod*'i vahel, mis asuvad eri sõlmel, kasutades *Fast Datapath* ühenduse tüüpi. *Sleeve* tüübi puhul läbitakse mõlemas sõlmes (joonisel "Host1" ja "Host2") lisaks *Weave Router*'i *pod* (hall joon). Loodud ühendusi ning kasutusel olevat ühenduse tüüpe näeb käsuga *kubectl exec -n kube-system <weave_net_pod_nimi> -c weave -- /home/weave/weave --local status connections*.

[8] https://docs.projectcalico.org/getting-started/kubernetes/flannel/



Joonis 4. Võrguliiklus kahe *pod*'i vahel kasutades Weave Net võrgutehnoloogiat[9]

Weave Net toetab võrguliikluse krüpteerimist. Krüpteerimiseks seadistatakse *WEAVE_PASSWORD* muutuja ning seadistuse rakendumist näeb käsuga *kubectl exec -n kube-system <weave_net_pod_nimi> -c weave -- /home/weave/weave --local status | grep Encryption*. Võrgutehnoloogia toetab multisaadet ning vajadusel saab selle blokeerida.

[9] https://www.weave.works/docs/net/latest/concepts/fastdp-how-it-works/

4. Testimise metoodika

Eelnevad artiklid [1] [2] Kubernetese võrgutehnoloogiate testimise kohta on võrrelnud TCP, UDP, HTTP, FTP ja SSH ühenduste mahtu. TCP ja UDP ühenduste testid näitavad süsteemi üldist võimet suhelda võrgutasemel vastava OSI neljanda taseme protokolliga. HTTP, FTP ja SSH ühendusi on testitud, kuna need näitavad levinute süsteemide võimekust. Töös keskendutakse sünteetilistele testidele kasutades programmi iperf ja qperf. Lisaks testitakse HTTP serveri ühenduste mahtuKubernetese API) kasutavad HTTP liiklust.

Selles töös võrreldakse võrgutehnoloogiaid järgnevate parameetrite alusel:

- *Pod'*ide vaheliste TCP ja UDP ühenduse kiirus
- *Pod'*ide vaheliste TCP ja UDP ühenduse latentsus
- *Pod'*ide vaheliste HTTP ühenduste maht
- Serveri koormatus ühenduse kiiruse ja HTTP testide ajal
- Võrgutehnoloogia algseadistamisel tekitatud *pod'*ide mälukasutus

Protokollide ja nende testimiseks kasutatud programmide nimekiri on toodud tabelis 3.

Tabel 3. Testprogrammide nimekiri

Testitav parameeter	**Programm**		**Ühik**
	Serveri pool	**Kliendi pool**	
TCP ja UDP ühenduse kiirus	Iperf versioon 2.0.10 Qperf versioon 0.4.10		Bitti sekundis (b/s); suurem on parem
TCP ja UDP ühenduse latentsus	Qperf versioon 0.4.10		Mikrosekundit (us); väiksem on parem
HTTP ühenduste maht	NGINX versioon 1.14.0	AB (ApacheBench) versioon 2.3 (programmi apache-utils versioon 2.4.29)	Ühendusi sekundis; suurem on parem
Süsteemi koormatus	Linux serveri *cat /proc/loadavg* väljund		"1" on võrdne ühe protsessori lõime kasutusega; väiksem on parem
Võrgutehnoloogia *pod*'ide mälukasutus	Metrics Server		Megabaiti; väiksem on parem

Iperf ja qperf programmi paralleelne kasutus on vajalik, sest mõlemal programmil on puudujäägid. Testides leiti, et qperf programmi maksimaalne läbilase on ligikaudu 20 Gbit/s. Seadistades programm iperf võtmega *-P* töötama mitme protsessiga saavutati samal süsteemi seadistustel üle 70 Gbit/s. Programm iperf ei tagasta alati UDP testidel latentsuse tulemusi ning seega ei saa seda programmi kasutada UDP latentsuse mõõtmiseks.

Serverite koormuse visualiseerimiseks testide läbiviimise ajal seadistati klastri välisele serverile programmid Elasticsearch ja Kibana ning igale sõlmele Metricbeat programm. Metricbeat seadistati raporteerima Elasticsearch serverisse 5-sekundilise intervalliga järgmisi andmeid[10]:

- *Cpu* – protsessori kasutus
- *Load* – protsessori koormuse jooksev keskmine 1, 5 ja 15 minuti kohta
- *Memory* – mälukasutus
- *Network* – võrgu liideste kasutus
- *Process* – protsesside ressursikasutus
- *Socket_summary* – avatud soklite kasutus

Metricbeat kasutab kogutud andmete kohaselt jõude oleval serveril keskmiselt 5-7% ühe lõime ressursist (0,05-0,07 ühikut *cat /proc/loadavg* väljundis), mis ei ole süsteemi koormatuse mõõtmistulemustest maha arvestatud.

Serveri koormatust mõõdeti kasutades füüsilise serveri *cat /proc/loadavg* väljundi 1 minuti väärtust. Testid, mille lõpus salvestati serveri koormus, on toodud tabelis 4.

Tabel 4. Testprogrammide käivitamiseks kasutatud käsud

Test	**Käsk**	**Testi pikkuse valimise moodus**
Iperf TCP ühenduskiiruse ja latentsuse mõõtmine	Iperf –c <server ip> -P <P väärtus> -f m –i 10 –t 180 –m -e	Võtme "-t" väärtus määrab testi pikkuse.
Iperf UDP ühenduskiiruse ja latentsuse mõõtmine	Iperf –c <server ip> -u –P <P väärtus> -f m –i 10 –t 180 –b 100G -e	Võtme "-t" väärtus määrab testi pikkuse.
HTTP ühenduste maht	ab –k -n 10000000 –c 350 –H "Accept-Encoding: gzip, deflate" http://<nginx_server_ip>/	Võti "-n" valitakse piisavalt suur.
Qperf TCP ühenduskiiruse mõõtmine	Qperf <server ip> -ub –e 2 –t 180 –vv tcp_bw	Võtme "-t" väärtus määrab testi pikkuse.

[10] https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-module-system.html

Qperf UDP ühenduskiiruse mõõtmine	Qperf <server ip> -ub –e 2 –t 180 –vv udp_bw	Võtme "-t" väärtus määrab testi pikkuse.

Nimetatud teste käitati vähemalt 70 sekundit. Serveri keskmiseks koormuseks võib lugeda *cat /proc/loadavg* 1 minuti tulemus, sest testi kestvus on vähemalt 60 sekundit.

Testide läbiviimisel Kubernetese klastris, mille ülesehitus on kirjeldatud peatükis 2.4, seadistatakse võrgutehnoloogia vastavalt peatükis 3 esitatud seadistuse failile. Käsuga *kubectl get pods –A –o wide* kontrollitakse, et võrgutehnoloogia *pod'*id oleksid tekitatud ning olekuga *Running*. Seejärel seadistatakse Metrics Server programm. Käskude *kubectl get pods –A –o wide*, mis näitab *pod*'ide nimesid ja sõlmesid, millel need jooksevad, ning *kubectl top pods –n kube-system*, mis näitab *pod*'ide ressursikasutust, leitakse igal sõlmel käimas olevate *pod*'ide mälukasutus.

Tekitatud klastris seadistati programmide iperf, qperf ja NGINX *pod'*id igale sõlmele kasutades tabelis 5 toodud konteinereid. *Pod'*id seadistati *DaemonSet* tüüpi objektina kirjeldades *spec.template.tolerations.key:node-role.kubernetes.io/master* võtme *effect* väärtuseks *NoSchedule*, mis seadistab *pod'*i ka juhtsõlmele. Iga *pod*'i paari vahel viidi läbi testimine mõlemat pidi. Kontrolliti, et ei testitaks *pod*'iga samal sõlmel olevat serverit võrreldes testprogrammi käivitava *pod*''i ja serveri *pod*'i sõlme võrguaadressi. Sõlmede võrguaadress leiti käsuga *kubectl get pod <pod'i_nimi> -o jsonpath='{.status.hostIP}'*. Väärtus *<pod'i_nimi>* asendati testprogrammi käivitava või serveri *pod*'i nimega. Testi alustamiseks kasutati juhtsõlmes käsku *kube*bectl *exec –it <pod'i_nimi> -- <testi_käsk>*, mille tulemusena edastati *<testi_käsk>* vastavale *pod'*ile. Selleks, et testimise tulemus oleks võimalikult täpne, käivitati klastris üks testprogramm korraga. Samade sisendväärtustega testi korrati kolm korda. Tulemused keskmistati vastavalt testprogrammi sisendväärtustele.

Tabel 5. Testimisel kasutatud konteinerid

Programm	**Konteiner**	**Veebilink**
Iperf	mindthecap/iperf	https://hub.docker.com/repository/docker/mindthecap/iperf
Qperf	mindthecap/qperf	https://hub.docker.com/repository/docker/mindthecap/qperf
NGINX	mindthecap/nginx	https://hub.docker.com/repository/docker/mindthecap/nginx

Füüsiliste serverite võimekuse baasi leidmiseks testiti füüsilisi servereid Kubernetese klastrit tekitamata. Igale serverile seadistati programmid iperf, qperf ja NGINX. Servereid testiti paari kaupa, käivitades üks testprogrammi korraga, sarnaselt klastris testimisele. Testimine viidi läbi serveri paaril mõlemat pidi ja korrati kolm korda iga testi sisendväärtuse korral. Tulemused keskmistati vastavalt testi sisendväärtustele.

Kõikide testide tulemused kirjutati faili, millele lisati testi alustamise ja lõpetamise aeg. Serveri koormuseks loeti testi lõpus füüsiliste serverite *cat /proc/loadavg* väljund, mis lisati tulemuste faili. Järgnevad alampunktid kirjeldavad kasutatud programme.

4.1 Iperf

Iperf [11] programm töötab klient-server süsteemina, kus üks pool on suhtluse algataja (klient) ja teine pool vastab alustatud sessioonile. Programmi seadistatakse serveriks võtmega *-s* ning käivitatakse kliendina võtmega *-c*, millele järgneb serveri võrguaadress. Programm testib ühenduse mahtu ja parameetreid saates andmeid kliendilt serverile. Testimisel kasutatud käsud on toodud tabelis 6.

Tabel 6. Programmi iperf käivitamise käsud

Test	**Kasutuskoht (server/klient)**	**Käsk**
TCP ühenduskiiruse ja latentsuse mõõtmine	Server	Iperf –s –p 5001 -e
	Klient	Iperf –c <server ip> -P <P väärtus> -f m –i 10 –t 180 –m -e
UDP ühenduskiiruse ja latentsuse mõõtmine	Server	Iperf –s –u –b
	Klient	Iperf –c <server ip> -u –P <P väärtus> -f m –i 10 –t 180 –b 100G -e

Töös kasutatud programmi võtmed on kirjas tabelis 7. Testi kestvuseks seadistati 180 sekundit, et oleks võimalik näha süsteemi koormatust kasutades seiresüsteemi Metricbeat ja Elasticsearch.

[11] http://manpages.ubuntu.com/manpages/bionic/man1/iperf.1.html

Tabel 7. Programmi iperf võtmed

Iperf võti	**Võtme väärtus**	**Kasutuskoht (server/klient)**	**Selgitus**
-c	-	Klient	Programmi kliendi versioonina käivitamine
-P	1,2,4,6,8,10	Klient	Paralleelselt mitme lõime käivitamine kliendi versioonis
-i	10	Klient	Tulemuste raporteerimise sagedus
-t	180	Klient	Testi kestvus
-m	-	Klient	TCP ühenduse maksimaalse segmendi suuruse raporteerimine
-u	-	Klient/server	UDP testi valimine
-e	-	Klient	Testi lisaparameetrite näitamine (TCP testil lisab RTT välja ning UDP testil latentsuse välja)
-b	100G	Klient	UDP ühenduse testimisel maksimaalse andmemahu seadistus (vaikimisi 1 Mbit/s)
-f	m	Klient	Tulemuste formaatimine Mbit/s kujule, suurusühikute vahe on 1000 (1 K = 1000, 1M = 1000000)
-s	-	Server	Programmi server versioonina käivitamine
-p	5001	Server	Port, millel teenus kuulab (vaikimisi 5001)

Testi väljundis loetakse ühenduse kiiruseks 1 lõime käivitamisel ajavahemiku 0.00-180.00 *Bandwidth* väärtus ning RTT väli ühenduse latentsuseks. Mitme lõime käivitamisel loetakse ühenduse kiiruseks välja *Bandwidth* ajavahemiku 0.00-180.00 summaarne väärtus ning ühenduse latentsuseks välja RTT sama ajavahemiku tulemuste keskmine väärtus. UDP testidel ei tagastanud programm RTT väärtust.

4.2 NGINX ja AB

Töös testitakse vaikeseadistustes veebiserveri NGINX võimekust programmiga AB [12]. Server programmiks vailiti NGINX, sest võrreldes Apache'ga on süsteem võimeline teenindama rohkem ühendusi [13] ning on laialdaselt kasutusel [14]. Testimiseks käivitati programm AB tabelis 8 nimetatud võtmete ja väärtustega.

Tabel 8. Testi AB võtmed

Võti	**Võtme väärtus**	**Selgitus**	**Põhjus**
-k	-	HTTP sessiooni kasutatakse mitme järgneva päringu tegemiseks	Veebibrauserid kasutavad tavaliselt sama sessiooni. Seadistati, et testimine oleks mingil määral võrreldav reaalse HTTP liiklusega [12].
-n	10000000	Testis tehtavate päringute arv	
-c	350	Üheaegselt tehtavate päringute arv	
-H	Accept-Encoding: gzip, deflate	Päringu päise lisaväärtused	Tihendab päringu vastuses olevaid andmeid. Seadistati, et testimine oleks mingil määral võrreldav reaalse HTTP liiklusega [15] [12].

Programm käivitatakse järgmiselt:

ab –k -n 10000000 –c 350 –H "Accept-Encoding: gzip, deflate" http://<nginx_server_ip>/

Osa *<nginx_server_ip>* vastab igale testitavale NGINX serveri võrguaadressile. Võtme *-c* väärtus valiti suvaliselt ning katsetes nähti, et see tõstab serveri koormust.

HTTP ühenduse mahuks loeti testi väljundi *Requests per second* väärtus. Lisaks salvestati *Time per request* väärtus, mis näitab, kui kaua aega kulub keskmiselt 350 pöördumisele.

[12] http://manpages.ubuntu.com/manpages/bionic/man1/ab.1.html
[13] https://www.semanticscholar.org/paper/Web-Server-Performance-of-Apache-and-Nginx%3A-A-Kunda-Chihana/2f884c2bb3ccda21f3b9ad370d5975d7eb90e74b
[14] https://w3techs.com/technologies/overview/web_server
[15] https://en.wikipedia.org/wiki/HTTP_compression

4.3 Qperf

Qperf [16] on klient-server tüüpi programm. Tehtavad testid ja tulemuste formaadi määrab ära kliendi poolel seadistatud parameetrid. Serveri seadistamiseks käivitatakse programm ilma parameetriteta. Testi tulemustes tagastatakse kliendi ja serveri protsessori ressursikasutus kasutades võtit *-vv*. Testimisel kasutatud võtmed ja väärtused on toodud tabelis 9 ning kasutatud käsud on toodud tabelis 10.

Tabel 9. Testi qperf võtmed

Võti	**Väärtus**	**Selgitus**
-t	180 (ühenduse kiiruse testil), 30 (latentsuse testil)	Testi kestvus.
-vv	-	Väljastab testi kohta detailse informatsiooni.
-e	2	Testi tulemuste tüvenumbrite arv
-ub	-	Kasutatakse ühikuna bitte.
Tcp_lat	-	Testi tüüp; tehakse TCP ühenduse latentsuse test
Tcp_bw	-	Testi tüüp; tehakse TCP ühenduse kiiruse test
Udp_lat	-	Testi tüüp; tehakse UDP ühenduse latentsuse test
Udp_bw	-	Testi tüüp; tehakse UDP ühenduse kiiruse test

Tabel 10. Testi qperf käivitamise käsud

Test	**Kasutuskoht (server/klient)**	**Käsk**
Kõik testid	Server	Qperf
TCP ühenduskiiruse mõõtmine	Klient	Qperf <server ip> -ub –e 2 –t 180 –vv tcp_bw
UDP ühenduskiiruse mõõtmine	Klient	Qperf <server ip> -ub –e 2 –t 180 –vv udp_bw

[16] http://manpages.ubuntu.com/manpages/bionic/man1/qperf.1.html

TCP latentsuse mõõtmine	Klient	Qperf <server ip> -ub –e 2 –t 30 –vv tcp_lat
UDP latentsuse mõõtmine	Klient	Qperf <server ip> -ub –e 2 –t 30 –vv udp_lat

Ühenduskiiruseks loeti TCP testil väljundi *bw* tulemuse väärtus, UDP testil väljundi *send_bw* tulemuse väärtus ning latentsuse testidel *latency* tulemuse väärtus.

4.4 Metrics Server

Kubernetes klastris seadistatud *pod'*ide mälukasutuse leidmiseks kasutatakse töös Metrics Server [17] lahendust, mis kogub andmeid klastris töötavate sõlmede ja *pod'*ide protsessori ja mälu ressursikasutuse kohta. *Pod'*ide ressursikasutust näeb käsuga *kubectl top pods –n kube-system*. Võti *-n* väärtusega *kube-system* valib nimeruumi *kube-system*, kuhu tekitatakse vaikimisi võrgutehnoloogia *pod'*id. Tekitatud *pod*'ide tuvastamiseks uuriti võrgutehnoloogia algseadistamisel programm kubectl väljundit, mis väljastab loodud objektide nimed, ning leiti vastavalt väljundile loodud *DaemonSet*, *ReplicaSet*, *StatefulSet* või *Deployment* tüüpi objektide *pod*'id. Mälukasutuseks loeti võrgutehnoloogiate algseadistamisel tekitatud *pod*'ide summeeritud mälukasutus. Töös kasutatakse Metrics Server versiooni 0.3.6.

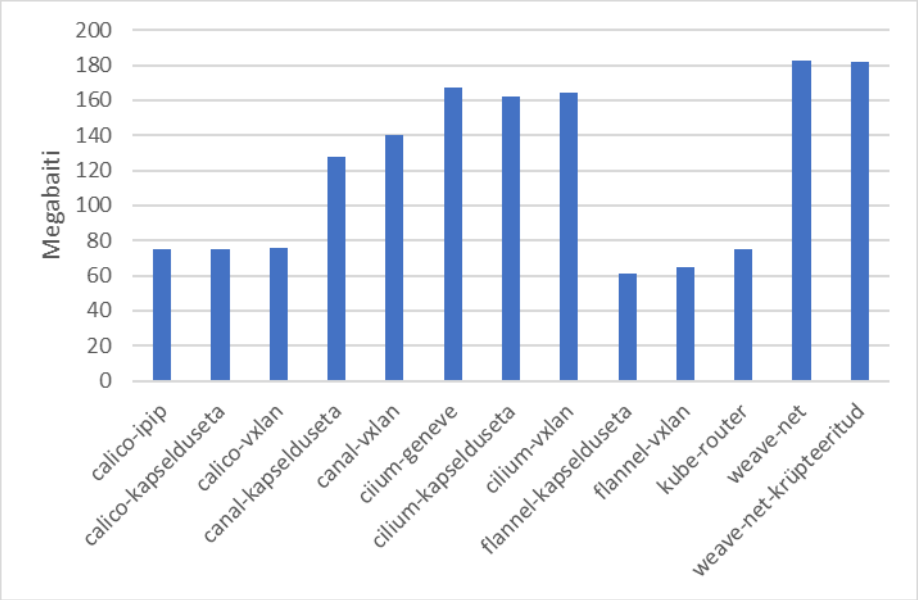
[17] https://github.com/kubernetes-sigs/metrics-server

5. Tulemused ja analüüs

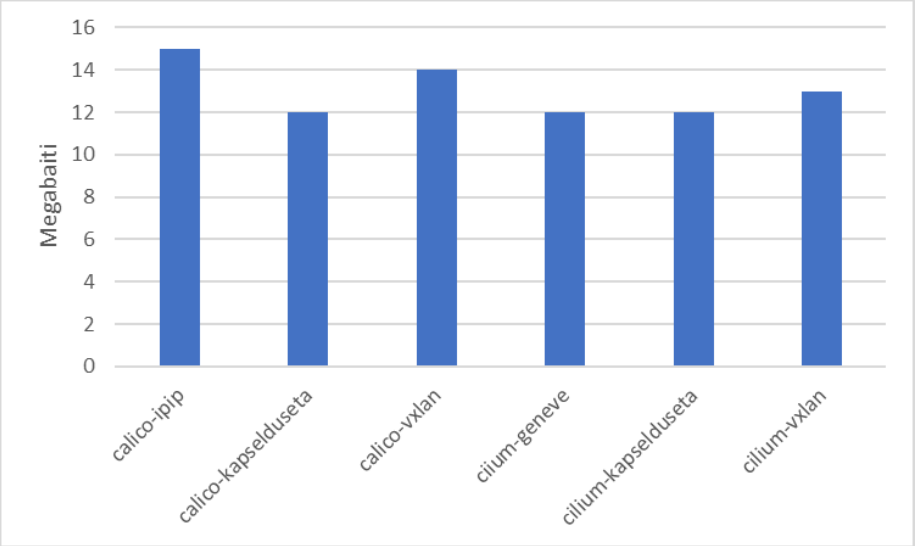
Peatükis antakse ülevaade töös käsitletud võrgutehnoloogiate testide tulemustest. Analüüs on jagatud viide alampeatükki vastavalt vaadeldavate parameetrite alusel.

5.1 Võrgutehnoloogia tekitatud *pod*'ide mälukasutus

Võrgutehnoloogiate seadistamisel tekitatud *pod*'ide mälukasutus on toodud joonisel 5. Sama tehnoloogia eri seadistuste vahel on suurim erinevus Canal tehnoloogial, mille kapselduseta seadistus vajab 12 MB vähem mälu. Kõige rohkem vajab mälu Weave Net tehnoloogia ning jagades tulemuse võrdselt kolme sõlme peale, kasutatakse igal sõlmel ligikaudu 60 MB ehk alla 0,7% testimisel kasutatud serverite mälust. Calico ja Cilium tehnoloogiad tekitasid lisaks *DaemonSet* objektile *Deployment* tüüpi objekti, mille mälukasutus on toodud joonisel 6.



Joonis 5. Võrgutehnoloogiate *pod*'ide mälukasutus



Joonis 6. Lisa *pod*'i mälukasutus

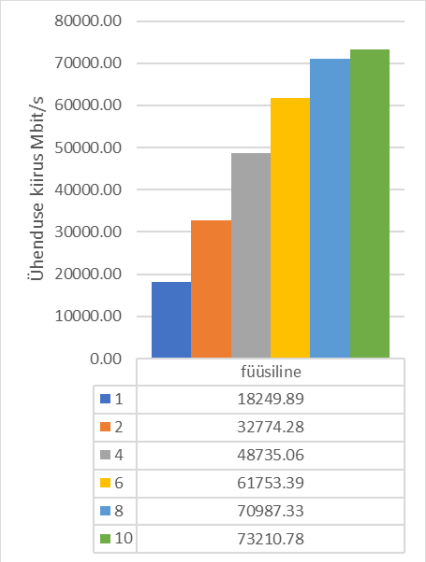
Deployment tüüpi *pod*'ide mälukasutus jäi vahemiku 12-15 MB, mis töös kasutatavatel serveritel on kogu kasutatavast mälust alla 0,02%.

Võrgutehnoloogiate *pod*'id kasutavad piisavalt vähe mälu võrreldes serverite mälu hulgaga, et eelistada ühte tehnoloogiat teisele. Ühel sõlmel kasutatava mälu hulk erineb vaadeldavatel tehnoloogiatel maksimaalselt ~40 MB, mis on Kubernetese minimaalsest soovitatavast 2 GB-st [18] ~2%.

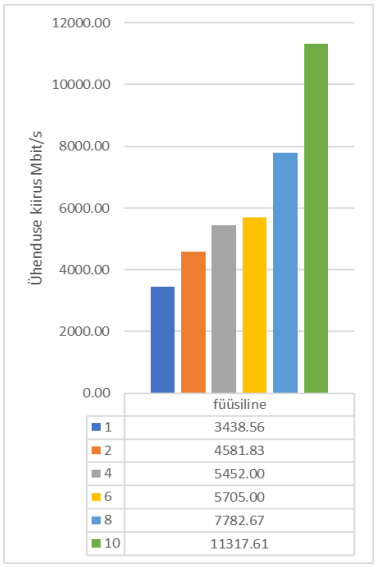
5.2 *Pod*'ide vaheliste TCP ja UDP ühenduste kiirus

Ühenduskiiruse testimisel käivitati erinevate parameetritega teste, mille tulemused on toodud lisas 2 ja 3. Serverite baasvõimekuse leidmiseks sooritati iperf test füüsilistel serveritel, mille tulemused sõltuvalt võtme *-P* sisendist on toodud joonistel 7 ja 8.

[18] https://v1-17.docs.kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

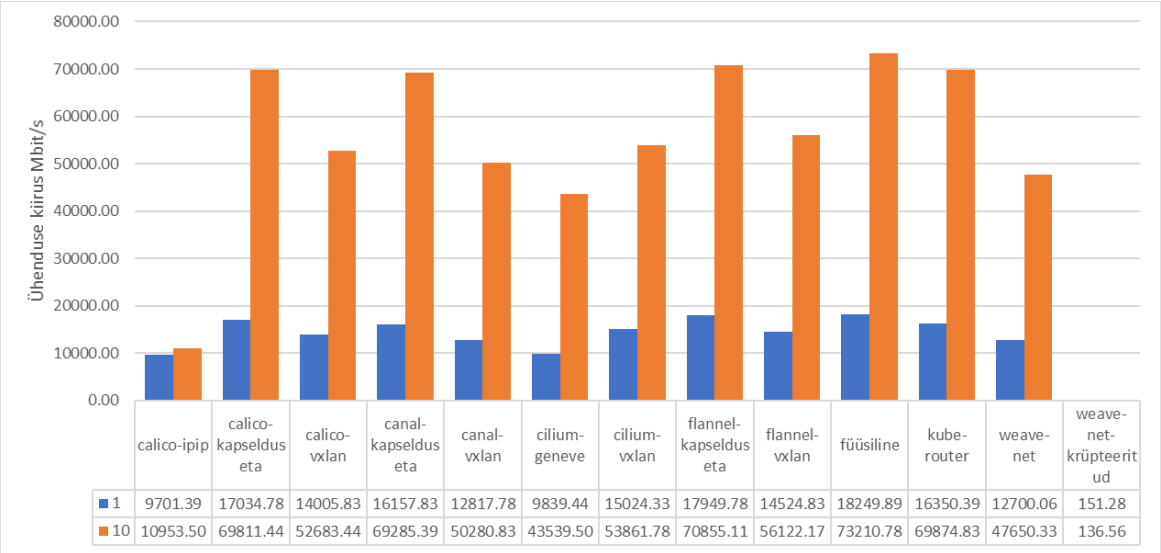


Joonis 7. Programmi iperf TCP ühenduskiiruse testi tulemused füüsilistel serveritel vastavalt võtme -*P* väärtusele



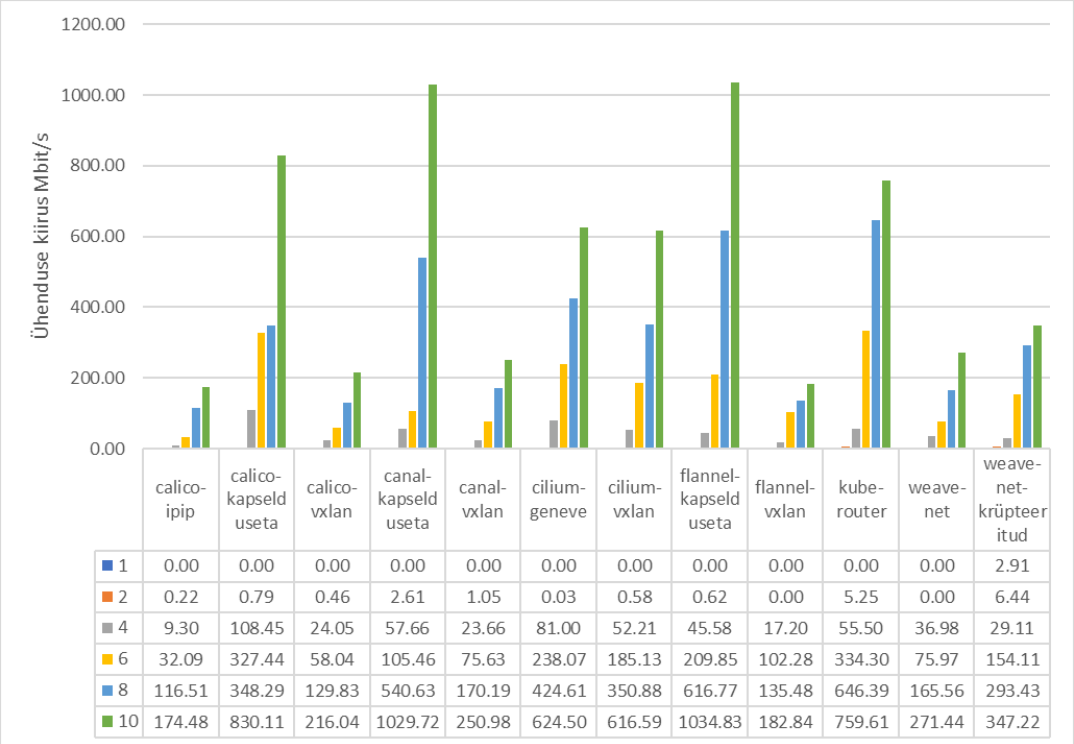
Joonis 8. Programmi iperf UDP ühenduskiiruse testi tulemused füüsilistel serveritel vastavalt võtme -*P* väärtustele

Joonistelt 7 ja 8 on näha, et ühenduskiirus kasvab võtme -*P* väärtuse (programmi lõimede arv) suurenemisel. Sarnane trend on näha enamustel võrgutehnoloogiatel, v.a Calico IP-in-IP ning Weave Net tehnoloogia krüpteeritud seadistuse korral (lisa 2 joonis 24). Nende kahe võrgutehnoloogia seadistustel ei tõuse ühenduskiirus sarnaselt teistele, mis võib olla põhjustatud Calico puhul võrgutehnoloogia ülesehitusest ning Weave Net seadistusel liikluse krüpteerimisest. Kõikide võrgutehnoloogiate TCP ühenduskiiruse testide, v.a krüpteeritud Weave Net seadistus, tulemused on joonisel 9 kümne lõime kasutamise korral üle 10000 Mbit/s. Kõige aeglasem on krüpteeritud Weave Net seadistuse järel Calico IP-in-IP seadistus ning kiireim on Flannel'i kapselduseta seadistus (tulemusega 70855,11 Mbit/s), mis oli võrreldes füüsilise serveriga ~3,2 % aeglasem. TCP ühenduskiiruste tulemused ja standardhälbed on kujutatud lisas 2 joonisel 24. Kapseldust kasutatavatest seadistustest on 1 lõime kasutamise korral kiireim Cilium VxLAN, 10 korral Flannel võrgutehnoloogia VxLAN kapseldusega, mis on vastavalt 17,7% ja 23,3% aeglasemad, kui füüsilised serverid.



Joonis 9. Programmi iperf TCP ühenduskiiruse testi tulemused võtme -*P* väärtusega 1 ja 10

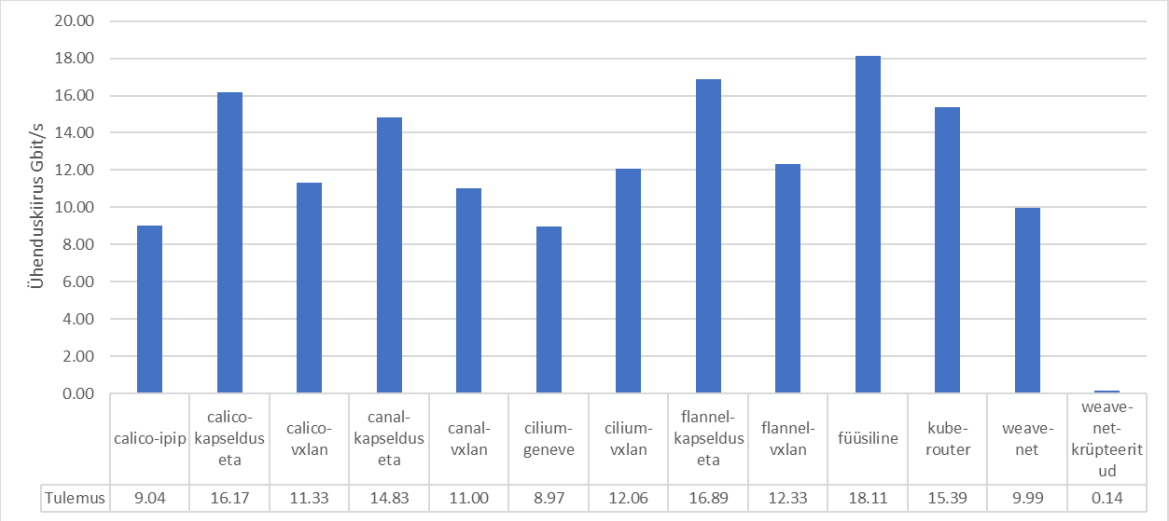
Programmi iperf UDP testide tulemused joonisel 10 on ootamatud, sest kuni 2 paralleelse testi käivitamise korral on võrgukiirus väga väike või puudub. Test ei tagastanud veakoodi, kuid võrreldes programmi qperf tulemustega joonisel 12, erinevad tulemused suurel määral. Ühenduskiirus tõuseb võtme *-P* väärtuse suurenemisel, nagu TCP ühenduskiiruse testil, ning kiiremad võrgutehnoloogiad on 10 lõime kasutamise korral kapseldamata Flannel ja Canal seadistused vastavalt 1034,8 ja 1029,7 Mbit/s. Samas testis on kapseldusega võrgutehnoloogiatest parema tulemusega Cilium tehnoloogia kasutades Geneve ning VxLAN kapseldust vastavalt tulemusega 624,50 ja 616,59 Mbit/s olles järgmisest tehnoloogiast üle kahe korra kiirem. Kõige aeglasem on Calico IP-in-IP seadistus tulemusega 174,48 Mbit/s. UDP võrgukiiruse testil olid füüsiliste serverite tulemused võrreldes võrgutehnoloogiate tulemustega liialt erinevad, et panna neid samale graafikule, saavutades 10 lõime kasutamisel 11317,61 Mbit/s, mis on kiireimast Kubernetese võrgutehnoloogia kiirusest üle 10 korra kiirem.. Programmi iperf UDP testide tulemused koos standardhälbega on toodud lisas 2 joonisel 25.



	calico-ipip	calico-kapselduseta	calico-vxlan	canal-kapselduseta	canal-vxlan	cilium-geneve	cilium-vxlan	flannel-kapselduseta	flannel-vxlan	kube-router	weave-net	weave-net-krüpteeritud
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.91
2	0.22	0.79	0.46	2.61	1.05	0.03	0.58	0.62	0.00	5.25	0.00	6.44
4	9.30	108.45	24.05	57.66	23.66	81.00	52.21	45.58	17.20	55.50	36.98	29.11
6	32.09	327.44	58.04	105.46	75.63	238.07	185.13	209.85	102.28	334.30	75.97	154.11
8	116.51	348.29	129.83	540.63	170.19	424.61	350.88	616.77	135.48	646.39	165.56	293.43
10	174.48	830.11	216.04	1029.72	250.98	624.50	616.59	1034.83	182.84	759.61	271.44	347.22

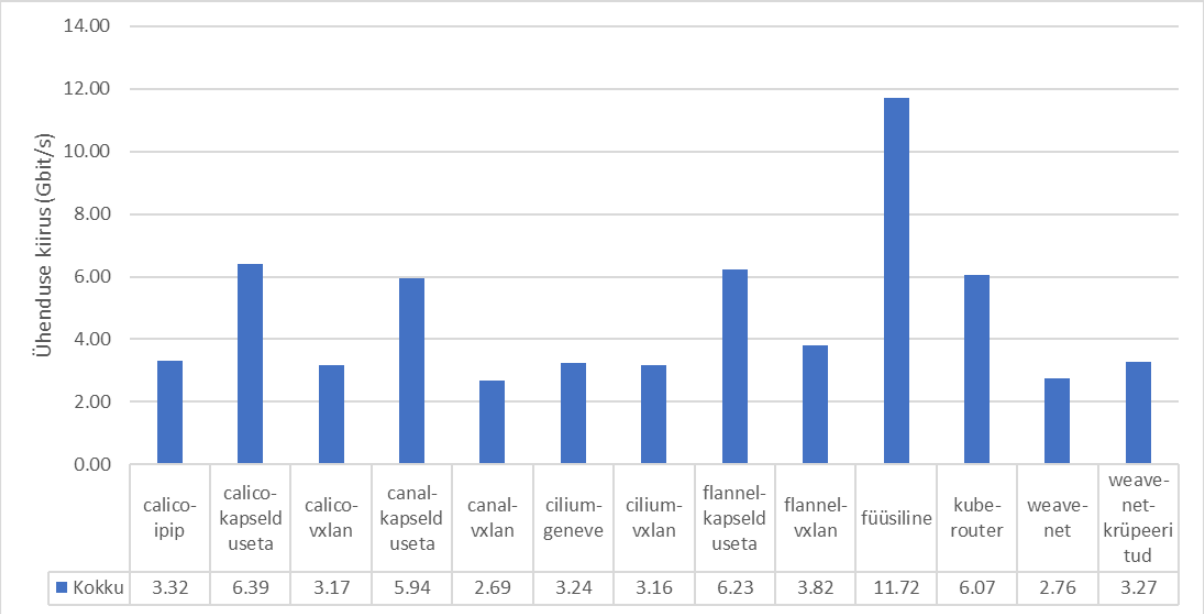
Joonis 10. Programmi iperf UDP testide tulemused vastavalt võtme *-P* väärtusele (v.a füüsiliste serverite tulemus)

Programmi qperf TCP testil on võrgutehnoloogiate järjestus sarnane programmi iperf TCP testi ühe lõime tulemustel. Kahe programmi testide tulemused on sarnased, kui programmil iperf kasutatakse ühte lõime, kuid mitme lõime kasutamisel saavutab iperf 73 Gbit/s, mis on programmi qperf parimast tulemusest ~3 korda kiirem. See näitab, et üle 20 Gbit/s võrgukiiruse maksimaalse läbilaske leidmiseks ei ole kõik programmid sobilikud. Joonisel 11 on näha, et füüsilistel serveritel tehtud test saavutas 18,11 Gbit/s ning kiiremad võrgutehnoloogiad on Flanneli ja Project Calico kapselduseta seadistused, mille tulemused on programmi qperf TCP ühenduse testis vastavalt ~6,7% ja ~10,7% füüsilistest serveritest väiksemad. Kõige aeglasem oli Weave Net krüpteeritud ühendus saavutades 0,14 Gbit/s. Programmi qperf TCP testi tulemused koos standardhälbega on toodud lisas 3 joonisel 26.

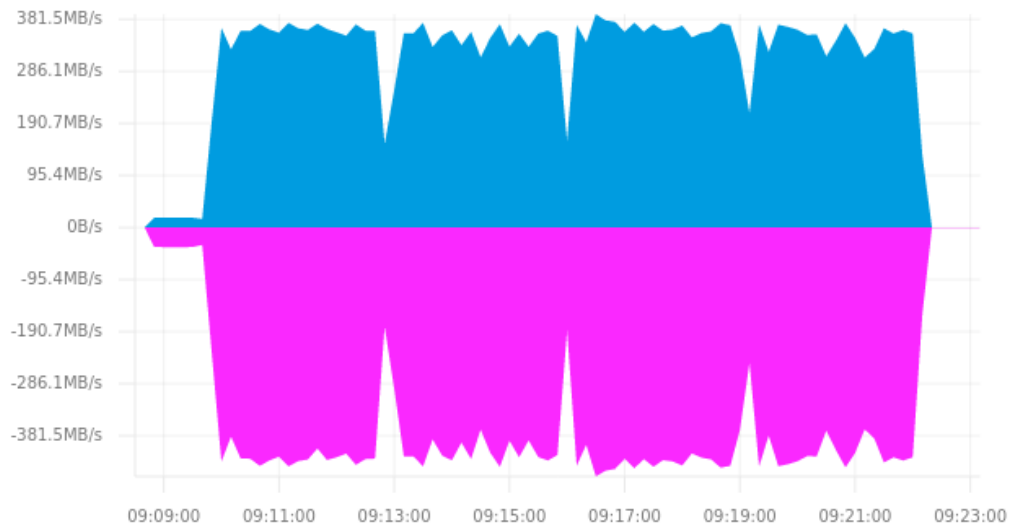


Joonis 11. Programmi qperf TCP ühenduskiiruse testide tulemused

Programmi qperf UDP testi tulemused joonisel 12 näitavad, et võrgutehnoloogia seadistused, mis ei kasuta kapseldust, on teistest kiiremad. Kiireim on Project Calico kapselduseta seadistus saavutades 6,39 Gbit/s olles ~45,5% aeglasem võrreldes füüsilise serveritulemustega. Võrgutehnoloogia Weave Net krüpteeritud seadistusel tekkisid testimisel ühenduse aegumise vead, ning uurides Elasticsearchi salvestatud meetrikat, tehti järjest kolme testi asemel neli (joonis 13). Jooniselt on võimalik välja lugeda nelja testi käivitamist, mis on eraldatud üksteisest võrgukiiruse vähenemisega, sest testide käivitamiste vahel oli mõningane viide. Anomaalne test on esimene, mis andis väljundis veateate „*failed to receive results: timed out*“. Testide käivitamise ja lõppemise ajad, mis salvestati testi tulemustega, kattuvad joonisel nähtava võrguühenduse kasutamisega.



Joonis 12. Programmi qperf UDP ühenduskiiruse testide tulemused



Joonis 13. Programmi qperf UDP ühenduskiiruse testil mõõdetud sõlme võrgukiirus kasutades Weave Net krüpteeritud seadistust

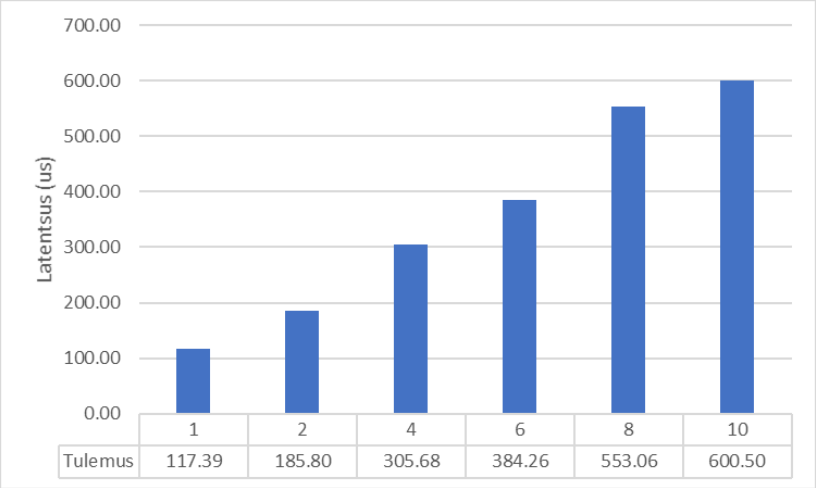
Üldiselt väheneb qperf programmi testis UDP ühenduste kiirus võrreldes füüsiliste serverite tulemusega märgatavalt ning kapseldust kasutavad võrgutehnoloogiate seadistuste tulemused jäävad programmi qperf testis vahemiku 2,69 – 3,82 Gbit/s, millest aeglaseim on Canal'i VxLAN seadistus ning kiireim Flannel VxLAN seadistus.

Ühenduskiiruse testides on kiiremad Flannel ja Calico kapselduseta seadistused ning kapseldusega seadistusel Cilium ja Flannel võrgutehnoloogiad. Kapseldust kasutavad seadistused on aeglasemad sama võrgutehnoloogia kapselduseta seadistusest, sest kapseldamisel lisatakse paketile või kaadrile informatsiooni. TCP ühenduskiiruste testil saavutati tihti suuremaid kiirusi, kui 10Gbit/s, mis näitab, et võrgutehnoloogiate maksimaalne kiirus on suurem, kui Narūnas Kapočius tehtud töös [1]. UDP võrgukiiruste testide tulemused on madalamad võrreldes Alexis Ducasteli tulemustega [2], mis võib olla tingitud erinevate testide kasutamisest. Ka selles töös näitasid iperf ja qperf programmid eri võrgukiirusi, mis võib olla tingitud programmide erisusest.

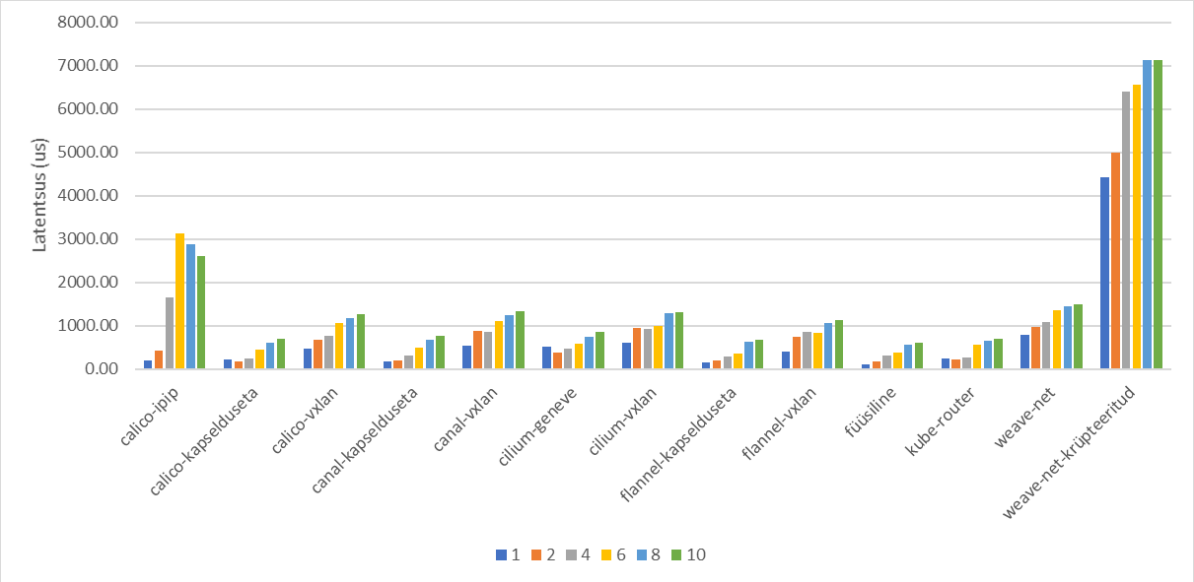
5.3 *Pod*'ide vaheliste TCP ja UDP ühenduste latentsus

Ühenduste latentsust testiti programmidega iperf ja qperf. Võrgutehnoloogiate tulemused ja nende standardhälve on toodud lisas 4.

Programmi iperf TCP ühenduse latentsuse tulemused joonisel 14 tõusevad vastavalt võtme *-P* väärtusele. Selline trend võib olla seotud serveri koormuse kasvuga (lisa 6 joonis 32). Serveri protsessor on koormatud teiste protsessidega ning seetõttu võib ühenduse vastus viibida. Erisusena langeb joonisel 15 Project Calico IP-in-IP seadistusel latentsus väärtuse 8 ja 10 korral. Suurima latentsusega on ühe lõime kasutamise korral Weave Net võrgutehnoloogia krüpteeritud seadistus tulemusega 4417,67 µs, mis on füüsilisest serveri testist ~38 korda suurem. Vähima latentsusega on nii 1 ja 10 lõime kasutamisel Flannel kapselduseta seadistus vastavalt 33% ja 12% halvema tulemusega võrreldes füüsilise serveriga, mille tulemus on 117,39 ja 600.50 µs. Kapseldust kasutavatest seadistustest on lisa 4 joonise 28 järgi ühe lõime kasutamisel parima tulemusega Calico IP-in-IP tulemusega 193,56 µs ning 10 lõime korral Cilium võrgutehnoloogia Geneve kapseldusega seadistus tulemusega 864,03 µs.

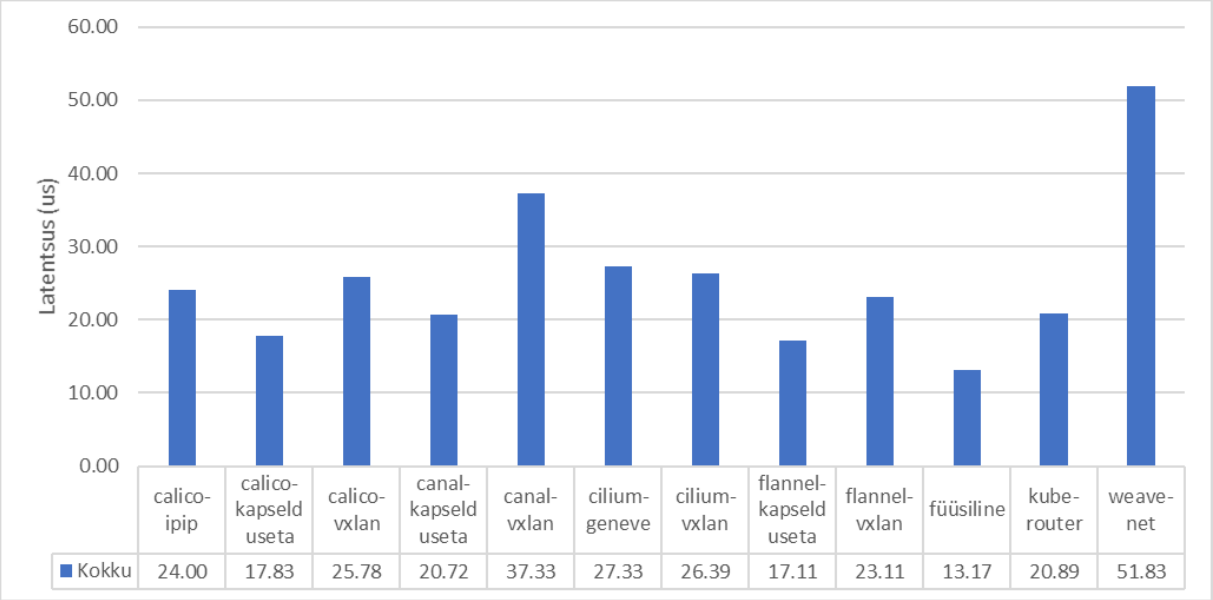


Joonis 14. Programmi iperf TCP ühenduste latentsus füüsilistel serveritel

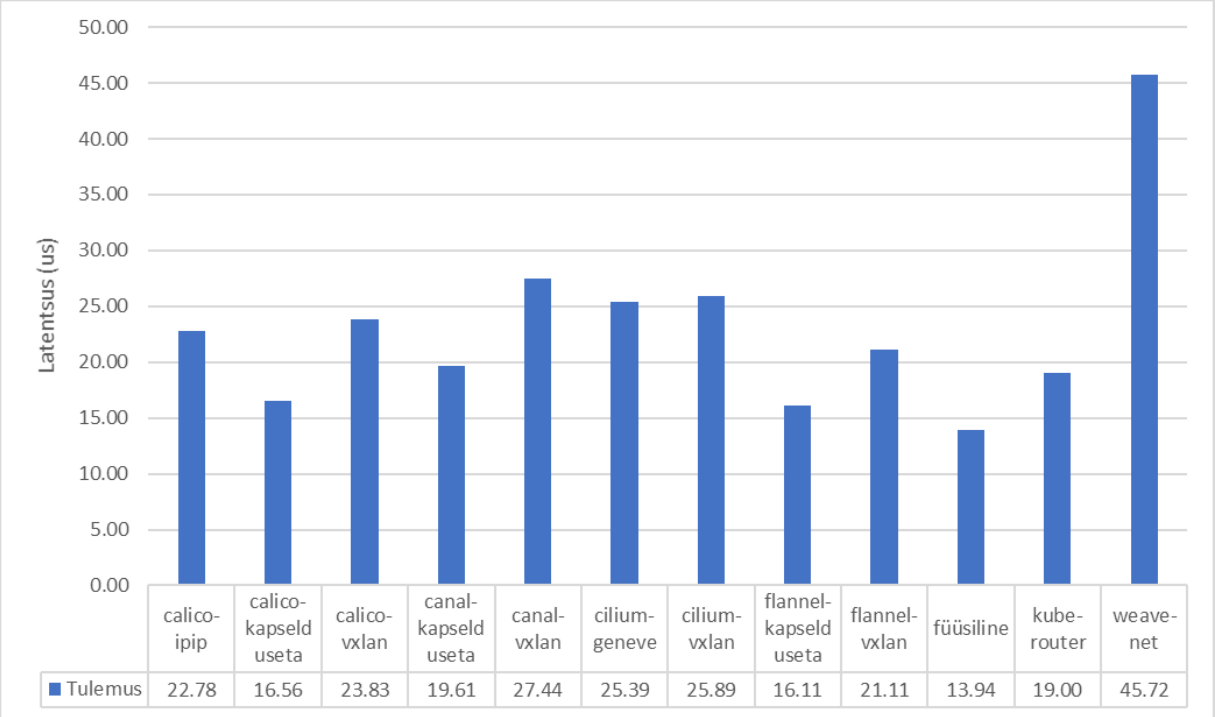


Joonis 15. Programm iperf TCP ühenduse latentsus vastavalt võtme *-P* väärtusele

Kapselduseta seadistused on väiksema latentsusega võrreldes teiste võrgutehnoloogiate seadistustega ka programmi qperf TCP ühenduse latentsuse testis (joonis 16). Parima tulemusega kapselduseta võrgutehnoloogia on Flannel (17,11 µs) ning parim kapseldusega tehnoloogia on Flannel VxLAN seadistus (23,11 µs) saavutades füüsilise serveri tulemusest vastavalt ~30% ja ~75% kõrgema tulemuse. Kõige suurem latentsus on krüpteeritud seadistusega Weave Net tehnoloogial tulemusega 468,11 µs, mille tulemus on toodud lisas 4 joonisel 29 koos teiste võrgutehnoloogiate tulemuste ja standardhälvetega.



Joonis 16. Programm qperf TCP ühenduse latentsus

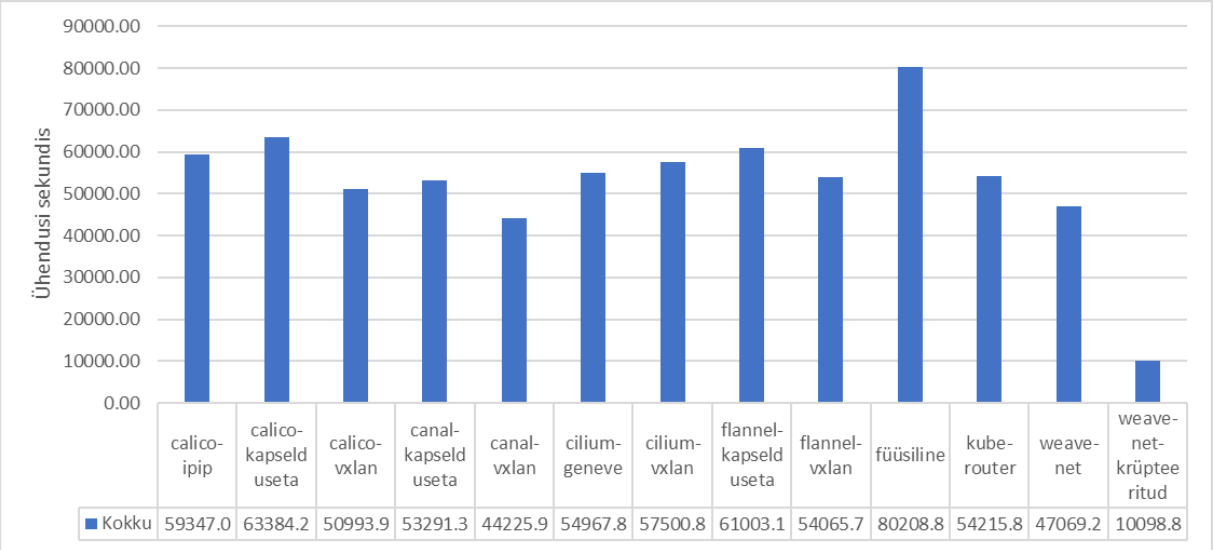


Joonis 17. Programmi qperf UDP ühenduse latentsus

Programmi qperf UDP latentsuse tulemused (joonis 17) näitavad sarnaselt TCP latentsuse tulemustele kapselduseta võrgutehnoloogiate seadistuste madalamat latentsust võrreldes kapseldust kasutavate seadistustega (v.a kapselduseta Canal tehnoloogia). Madalaima latentsusega on kapselduseta Flannel võrgutehnoloogia 16,11 µs, millele järgneb Project Calico kapselduseta seadistus 16,56 mikrosekundiga. Flannel võrgutehnoloogia kasutades VxLAN-i on madalaim kapseldust kasutav võrgutehnoloogia seadistus tulemusega 21,11 µs. Kõige suurema latentsusega on Weave Net krüpteeritud ühendus 451,89 µs latentsusega, mis on füüsiliste serverite tulemusest (13,94 µs) üle 30 korra suurem.

5.4 *Pod*'ide vaheliste HTTP ühenduste maht

HTTP ühenduste mahu testi tulemused joonisel 18 näitavad, et kapselduseta võrgutehnoloogiad saavutavad suurema ühenduse mahu, kui kapseldust kasutavad seadistused; erandina Canal kapselduseta seadistus, mis sai mitmest kapseldust kasutavast seadistusest halvema tulemuse. Parima tulemuse saavutas kapselduseta Project Calico võrgutehnoloogia (63384,2 üheaegset pöördumist), millele järgnes kapselduseta Flannel tehnoloogia tulemusega 61003,1 üheaegset pöördumist (~4% väiksem tulemus). Halvim tulemus on Weave Net krüpteeritud ühenduse seadistusel tulemusega 10098,8, mis on ~6 korda vähem, kui parim võrgutehnoloogia. Võrreldes füüsiliste serverite tulemusega väheneb samaaegsete ühenduste maht minimaalselt ~21%.



Joonis 18. Programmi ab testide tulemused

Testide tulemused koos standardhälbega on toodud lisas 5.

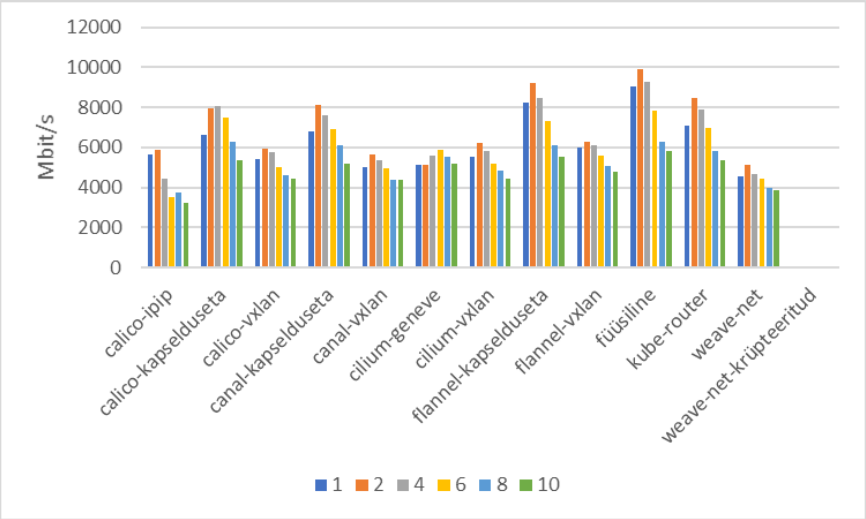
5.5 Serveri koormus

Võrgutehnoloogiate TCP, UDP ja HTTP ühenduste testide ajal mõõdetud keskmine serverite koormus on toodud lisas 6. Koormuse tõus programmi iperf testidel on sarnane ühenduse kiiruse tulemustega, mida kinnitab serveri koormuse ja testi tulemuste vaheline korrelatsioonikordaja 0,88 TCP testi ja 0,78 UDP testi puhul.

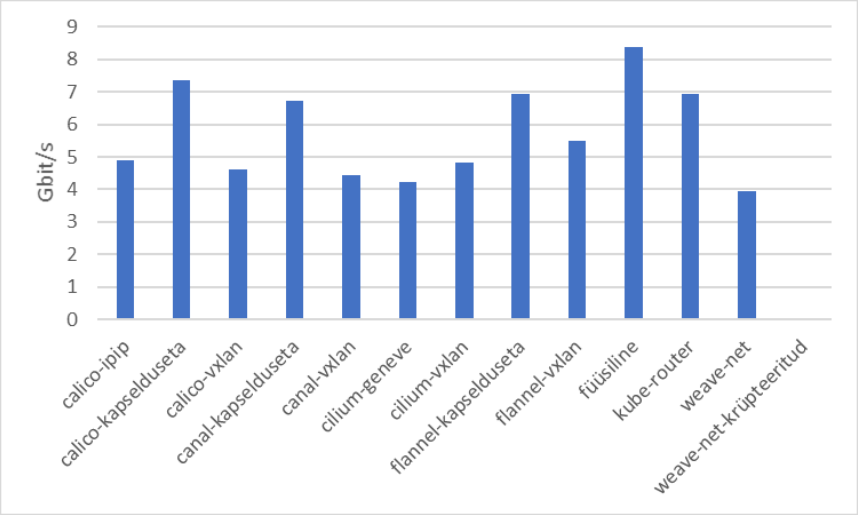
Programmi iperf TCP testi serveri koormus lisas 6 joonisel 32 näitab, et võrgutehnoloogiate kapselduseta seadistused nõudsid võtme *-P* väärtuse 10 korral serveritelt kõige rohkem ressurssi. Mitmel võrgutehnoloogia seadistusel (nt Project Calico ja Canal VxLAN seadistusega) on sama parameetrite juures serveri koormus suurem kui füüsilistel serveritel. Suurenenud koormuse vajadus ja väiksem ühenduskiiruse tulemus näitab, et võrgutehnoloogia seadistus piirab ühenduskiirust ning samas nõuab serverilt lisakoormust.

Selleks, et näha serveri koormust võrreldes testi tulemustega (andmemaht või ühendusi sekundis), kasutatakse testi tulemuse ja serveri koormuse jagatist, mis näitab ühikulise serveri koormuse puhul saavutatud tulemusi.

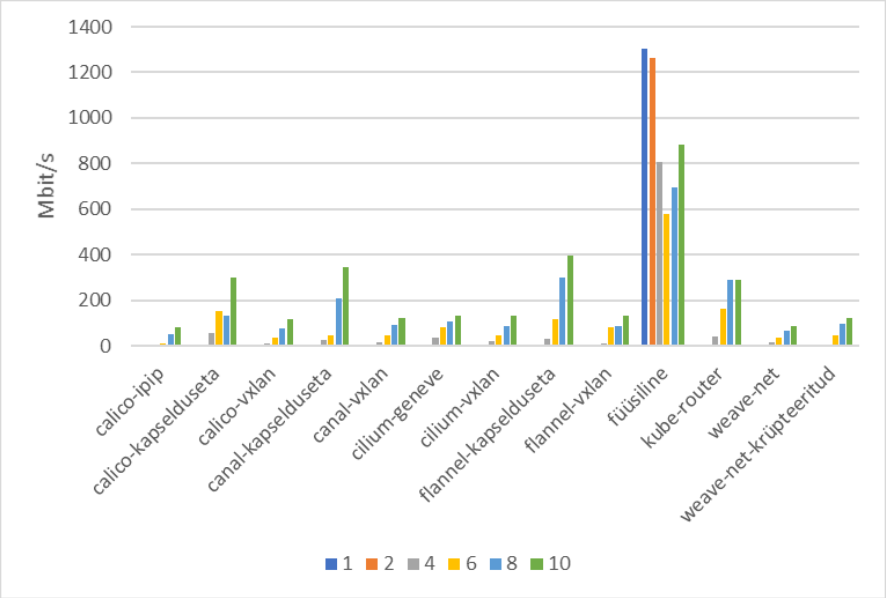
Serveri ühikulisele koormusele vastavad ühenduskiiruse graafikud joonistel 19 - 22 näitavad, et kapseldamata võrgutehnoloogia seadistused on efektiivsemad ressursi kasutamises kui kapseldust kasutavad seadistused. Kõige madalama tulemusega on programmi iperf TCP ühenduskiiruse testil on krüpteeritud seadistusega Weave Net tehnoloogia, mis on ~100 – 200 korda halvem võrreldes teiste tulemustega saavutades normaliseeritud ressursi kasutusel ühenduskiiruse keskmiseks 42 Mbit/s ning ~100 korda halvem programmi qperf TCP ühenduskiiruse testil koormates serverit kõige enam (lisa 6 joonis 34).



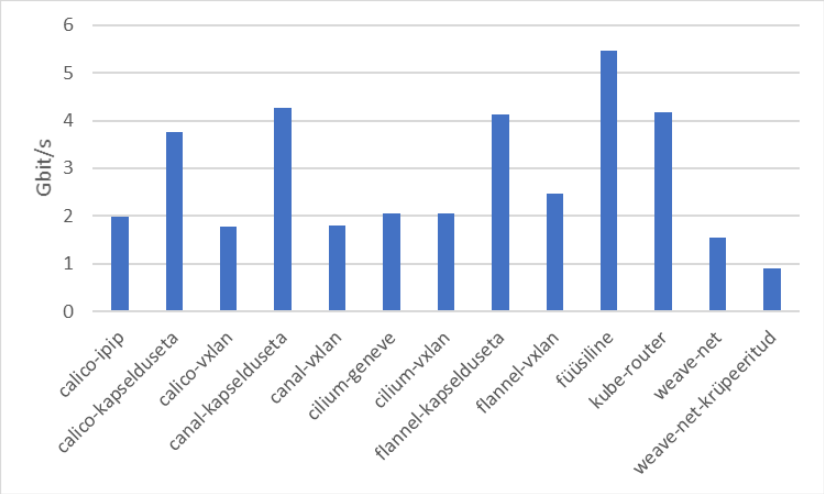
Joonis 19. Serveri koormusele normaliseeritud programmi iperf TCP ühenduskiiruse testi tulemused



Joonis 20. Serveri koormusele normaliseeritud programmi qperf TCP testide tulemused

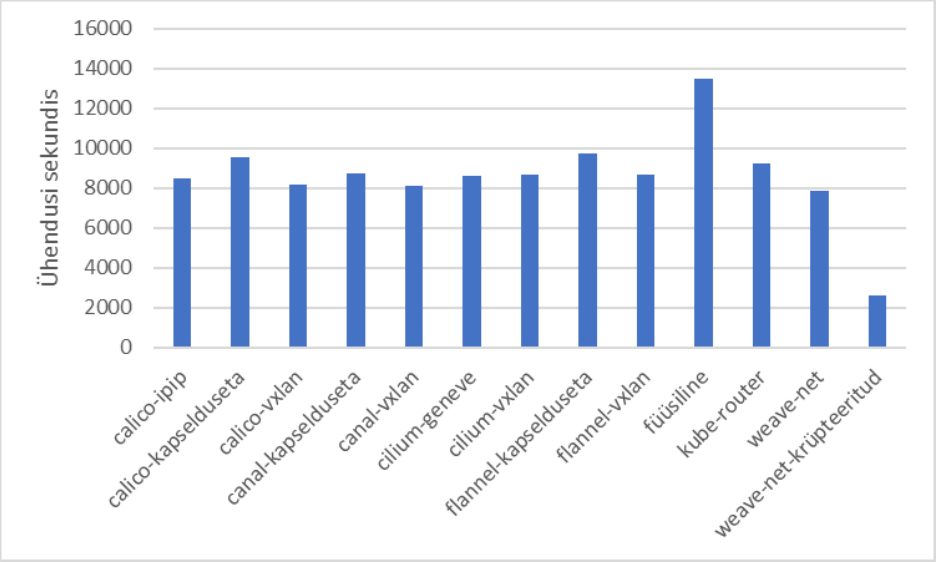


Joonis 21. Serveri koormusele normaliseeritud programmi iperf UDP testide tulemused



Joonis 22. Serveri koormusele normaliseeritud programmi qperf UDP testide tulemused

Serveri koormus UDP testidel on suurim Weave Net võrgutehnoloogia krüpteeritud ühendusel. Erisusena tõuseb Ciliumi võrgutehnoloogia serveri koormus iperf UDP testis koos seadistatud lõimede arvuga (lisa 6 joonis 35), kuid normaliseeritud graafikul (joonis 21) ei ole võrgutehnoloogia teistest palju efektiivsem.



Joonis 23. Serveri koormusele normaliseeritud programmi ab testide tulemused

Normaliseeritud HTTP ühenduse testi tulemused joonisel 23 ei näita nii suurt vahet võrgutehnoloogiatel, kui TCP ühenduse testil (joonis 20). Üldine serveri koormus on lisa 6 joonise 36 järgi suurim Calico IP-in-IP seadistusel olles võrgutehnoloogiate keskmisest 15% kõrgem ning kõige vähem koormas servereid HTTP ühenduse testil krüpteeritud seadistusega Weave Net võrgutehnoloogia.

6. Kokkuvõte

Käesolev töö andis ülevaate valitud Kubernetese klastri võrgutehnoloogiatest. Töö eesmärgiks oli uurida võrgutehnoloogiate võimekust 100 Gbit/s andmekeskuse võrgus. Töös käigus seadistati Kubernetese klaster kolmele füüsilisele serverile, mis koosnes ühest juhtsõlmest ning kahest arvutussõlmest. Võrgutehnoloogiaid võrreldi TCP ja UDP ühenduse kiiruse ja latentsuse, võrgutehnoloogia *pod*'ide mälukasutuse ning HTTP serveri ühenduse mahu põhjal. Lisaks uuriti serverite koormatust ühenduskiiruste ja HTTP serveri ühenduse mahu testide ajal.

Testidest selgus, et võrgutehnoloogiad kasutavad erineval hulgal mälu, kuid võrreldes töös kasutatud serverite mälu hulgaga, ei olnud võrgutehnoloogiate *pod*'ide mälu kasutus märkimisväärne.

Samuti võib testide põhjal väita, et võrreldes kapseldust kasutavaid ning kapselduseta seadistusi, on kiiremad kapselduseta võrgutehnoloogia seadistused. Mitme võrgutehnoloogia ühenduskiirused saavutasid kapseldusega kui kapselduseta seadistusel üle 50 Gbit/s, mis on eelnevate artiklite ja võrdluste tulemustest palju kõrgem ning näitab, et 10 Gbit/s võrgukiiruselt üleminek 100 Gbit/s võrgukiirusele võib tõsta Kubernetese klastris ühenduskiirusi märgatavalt.

Tööd saaks täiendada suurema MTU seadistamisega, mis võib suurendada võrgutehnoloogiate ühenduskiirusi. Veel võiks uurida võrgutehnoloogiate arendatud proksi ja võrguühenduste piiramise võimekust, mis näitaks võrgutehnoloogiate lisaomaduste jõudlust.

Viidatud kirjandus

1. Kapočius N. "Overview of Kubernetes CNI Plugins Performance", 2020 [Võrgumaterjal]. https://journals.vgtu.lt/index.php/MLA/article/view/11454/9757 [kasutatud 25. veebruar 2020].
2. Ducastel A. "Benchmark results of Kubernetes network plugins (CNI) over 10Gbit/s network (Updated: April 2019)", 2019 [Võrgumaterjal]. https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-april-2019-4a9886efe9c4 [kasutatud 25. veebruar 2020].
3. "Performance Analysis of CNI (Container Networking Interface) based Container Network", 2018 [Võrgumaterjal]. https://ieeexplore-ieee-org.ezproxy.utlib.ut.ee/stamp/stamp.jsp?tp=&arnumber=8539382 [kasutatud 25. veebruar 2020].
4. The Kubernetes Authors. "Kubernetes Documentation", 2020 [Võrgumaterjal]. https://v1-17.docs.kubernetes.io/docs/home/ [kasutatud 15. märts 2020].
5. CoreOS. "GitHub - coreos/flannel", 2020 [Võrgumaterjal]. https://github.com/coreos/flannel [kasutatud 30. märts 2020].
6. Google. "Github - kubernetes-sigs/kubespray", 2020 [Võrgumaterjal]. https://github.com/kubernetes-sigs/kubespray [kasutatud 26. aprill 2020].
7. Google. "Networking Overview including CNI ", 2020 [Võrgumaterjal]. https://kops.sigs.k8s.io/networking/ [kasutatud 31. märts 2020].
8. Cilium Authors. "Welcome to Cilium’s documentation!", 2020 [Võrgumaterjal]. https://docs.cilium.io/en/v1.7/ [kasutatud 31. märts 2020].
9. Tigera, Inc. "Project Calico", 2020 [Võrgumaterjal]. https://www.projectcalico.org/ [kasutatud 31. märts 2020].
10. "Kube-Router", [Võrgumaterjal]. https://www.kube-router.io/ [kasutatud 7. aprill 2020].
11. Weaveworks. "Weave Net: Network Containers Across Environments | Weaveworks", 2020 [Võrgumaterjal]. https://www.weave.works/oss/net/ [kasutatud 1. aprill 2020].
12. "Stackoverflow - apache - ab load testing", 2012 [Võrgumaterjal]. https://stackoverflow.com/questions/12732182/ab-load-testing [kasutatud 30. märts 2020].

Lisa 1 Kasutatud riistvara spetsifikatsioonid

Tabel 11. Kasutatud riistvara spetsifikatsioonid

Nimi	**Protsessor**	**Mälu**	**Kirjeldus**	**IP**
Node-1	2 x Intel Xeon E5-2620 v4 @ 2.10 GHz (8 tuuma, 16 lõime)	6 x 16384 MiB Micron DDR4 @ 2133 MHz	Juhtsõlm	192.168.1.11
Node-2	2 x Intel Xeon E5-2620 v4 @ 2.10 GHz (8 tuuma, 16 lõime)	6 x 16384 MiB Micron DDR4 @ 2133 MHz	Arvutussõlm	192.168.1.12
Node-3	2 x Intel Xeon E5-2620 v4 @ 2.10 GHz (8 tuuma, 16 lõime)	6 x 16384 MiB Micron DDR4 @ 2133 MHz	Arvutussõlm	192.168.1.13
Server-4	2 x Intel Xeon E5-2620 v4 @ 2.10 GHz (8 tuuma, 16 lõime)	6 x 16384 MiB Micron DDR4 @ 2133 MHz	Klastriväline server	192.168.1.1
Sw-1	N/A	N/A	Kommutaator	N/A

Lisa 2 Programmi iperf ühenduskiiruse testide tulemused

Võrgutehnoloogia	Võtme -*P* väärtus											
	1	Standard-hälve	2	Standard-hälve	4	Standard-hälve	6	Standard-hälve	8	Standard-hälve	10	Standard-hälve
calico-ipip	9701.39	113.15	13318.56	283.48	11941.56	162.06	11324.44	178.63	11139.28	116.65	10953.50	107.81
calico-kapselduseta	17034.78	248.06	27645.61	674.29	41723.78	785.88	57940.67	3642.41	68183.78	3273.91	69811.44	2243.52
calico-vxlan	14005.83	419.48	23833.50	4153.66	36256.78	3327.77	42728.39	4219.90	49156.00	5268.34	52683.44	3890.15
canal-kapselduseta	16157.83	302.92	26484.94	675.71	41371.56	2118.21	54789.94	2737.02	65670.39	3797.43	69285.39	2128.66
canal-vxlan	12817.78	299.00	23077.39	2289.76	34103.28	3399.98	42227.67	5071.72	48767.44	4817.91	50280.83	3761.67
cilium-geneve	9839.44	305.23	14018.83	3290.03	25373.61	1600.67	33144.28	2629.19	38148.17	3122.81	43539.50	3412.17
cilium-vxlan	15024.33	406.11	23390.22	6093.21	37884.50	6026.45	45151.78	5325.90	50102.50	3418.09	53861.78	3040.50
flannel-kapselduseta	17949.78	376.24	29024.83	744.82	44961.89	1658.37	62536.17	3221.03	71782.39	2325.82	70855.11	2344.49
flannel-vxlan	14524.83	559.81	26192.00	1812.35	37265.17	3207.88	47693.33	3240.73	52173.78	4508.02	56122.17	3702.26
füüsiline	18249.89	365.98	32774.28	1095.82	48735.06	2615.48	61753.39	3301.69	70987.33	2832.08	73210.78	1554.12
kube-router	16350.39	405.03	27090.94	884.29	41148.83	517.44	56447.33	4315.48	68129.33	4462.98	69874.83	1909.14
weave-net	12700.06	438.44	21789.61	3367.86	32666.61	5433.21	40267.78	6237.78	44126.50	4460.01	47650.33	2620.70
weave-net-krüpteeritud	151.28	1.49	148.67	1.37	143.39	1.09	140.17	1.20	137.94	1.11	136.56	1.20

Joonis 24. Testi iperf TCP ühenduste keskmine kiirus ja standardhälve vastavalt võtmele -*P*

Võrgutehnoloogia	Võtme -*P* väärtus											
	1	Standard-hälve	2	Standard-hälve	4	Standard-hälve	6	Standard-hälve	8	Standard-hälve	10	Standard-hälve
calico-ipip	0.00	0.00	0.22	0.31	9.30	12.32	32.09	24.46	116.51	47.55	174.48	58.44
calico-kapselduseta	0.00	0.00	0.79	3.35	108.45	220.26	327.44	454.21	348.29	224.01	830.11	670.37
calico-vxlan	0.00	0.00	0.46	1.95	24.05	23.04	58.04	38.96	129.83	56.00	216.04	130.42
canal-kapselduseta	0.00	0.00	2.61	9.66	57.66	91.70	105.46	86.31	540.63	611.66	1029.72	653.27
canal-vxlan	0.00	0.00	1.05	2.71	23.66	17.22	75.63	89.23	170.19	154.41	250.98	272.96
cilium-geneve	0.00	0.00	0.03	0.13	81.00	191.79	238.07	248.72	424.61	439.57	624.50	507.58
cilium-vxlan	0.00	0.00	0.58	1.67	52.21	75.17	185.13	268.60	350.88	310.95	616.59	422.18
flannel-kapselduseta	0.00	0.00	0.62	2.62	45.58	48.41	209.85	167.48	616.77	428.62	1034.83	859.42
flannel-vxlan	0.00	0.00	0.00	0.00	17.20	19.00	102.28	68.91	135.48	39.84	182.84	55.54
füüsiline	3438.56	127.12	4581.83	1587.59	5452.00	2121.08	5705.00	3070.00	7782.67	3252.60	11317.61	2867.98
kube-router	0.00	0.00	5.25	8.37	55.50	52.54	334.30	246.89	646.39	513.90	759.61	367.01
weave-net	0.00	0.00	0.00	0.00	36.98	47.07	75.97	57.38	165.56	164.20	271.44	159.28
weave-net-krüpteeritud	2.91	2.83	6.44	7.77	29.11	19.16	154.11	105.92	293.43	151.30	347.22	121.18

Joonis 25. Testi iperf UDP ühenduste keskmine kiirus ja standardhälve vastavalt võtmele -*P*

Lisa 3 Programmi qperf ühenduskiiruse testide tulemused

Võrgutehnoloogia	Kiirus (Gbit/s)	Standard-hälve
calico-ipip	9.04	0.18
calico-kapselduseta	16.17	0.38
calico-vxlan	11.33	0.49
canal-kapselduseta	14.83	0.51
canal-vxlan	11.00	0.00
cilium-geneve	8.97	0.21
cilium-vxlan	12.06	0.24
flannel-kapselduseta	16.89	0.68
flannel-vxlan	12.33	0.49
füüsiline	18.11	0.96
kube-router	15.39	0.50
weave-net	9.99	0.32
weave-net-krüpteeritud	0.14	0.00

Joonis 26. Testi qperf TCP ühenduste keskmine kiirus ja standardhälve

Võrgutehnoloogia	Kiirus (Gbit/s)	Standard-hälve
calico-ipip	3.32	0.24
calico-kapselduseta	6.39	0.46
calico-vxlan	3.17	0.42
canal-kapselduseta	5.94	0.49
canal-vxlan	2.69	0.17
cilium-geneve	3.24	0.82
cilium-vxlan	3.16	0.76
flannel-kapselduseta	6.23	0.45
flannel-vxlan	3.82	0.46
füüsiline	11.72	0.67
kube-router	6.07	0.42
weave-net	2.76	0.45

Joonis 27. Testi qperf UDP ühenduste keskmine kiirus ja standardhälve

Lisa 4 Latentsuse testide tulemused

Võrgutehnoloogia	Võtme *-P* väärtus											
	1	Standard-hälve	2	Standard-hälve	4	Standard-hälve	6	Standard-hälve	8	Standard-hälve	10	Standard-hälve
calico-ipip	193.56	107.09	430.79	252.83	1660.17	701.92	3131.29	957.43	2876.90	734.24	2606.87	868.27
calico-kapselduseta	223.78	125.40	181.81	62.73	238.40	116.41	453.19	217.35	606.74	182.48	698.29	235.15
calico-vxlan	480.56	255.75	676.21	187.74	758.33	227.17	1062.72	195.50	1169.67	340.32	1255.81	269.25
canal-kapselduseta	178.33	35.20	196.65	13.94	310.17	153.57	486.77	203.01	681.07	182.63	763.87	177.60
canal-vxlan	545.11	292.19	887.22	278.37	847.32	224.75	1103.46	361.81	1250.72	294.61	1325.90	249.83
cilium-geneve	515.22	543.85	376.10	331.29	467.46	247.47	583.99	272.93	733.47	240.42	864.03	261.35
cilium-vxlan	600.28	190.82	943.22	295.56	932.01	311.92	993.67	245.35	1284.98	310.79	1304.03	295.90
flannel-kapselduseta	155.83	20.83	203.35	72.63	291.97	144.80	361.19	146.36	637.58	155.66	673.58	181.26
flannel-vxlan	413.33	247.03	752.57	208.10	862.92	273.90	832.96	226.19	1062.40	208.02	1129.57	223.98
füüsiline	117.39	10.31	185.80	62.94	305.68	154.91	384.26	133.79	553.06	197.86	600.50	238.94
kube-router	232.67	194.07	214.27	68.03	259.60	104.82	563.11	284.90	658.47	120.86	694.64	259.59
weave-net	796.06	77.24	972.45	226.82	1085.11	303.63	1367.69	394.28	1446.33	237.10	1493.61	267.78
weave-net-krüpteeritud	4417.67	1271.48	4982.67	1012.94	6396.56	1137.23	6549.30	654.60	7133.23	787.10	7134.09	714.51

Joonis 28. Testi iperf TCP ühenduste latentsus ja standardhälve vastavalt võtmele *-P*

Võrgutehnoloogia	Latentsus (us)	Standard-hälve
calico-ipip	24.00	0.00
calico-kapselduseta	17.83	0.38
calico-vxlan	25.78	0.55
canal-kapselduseta	20.72	0.46
canal-vxlan	37.33	6.37
cilium-geneve	27.33	4.74
cilium-vxlan	26.39	3.91
flannel-kapselduseta	17.11	0.32
flannel-vxlan	23.11	0.32
füüsiline	13.17	0.51
kube-router	20.89	0.47
weave-net	51.83	0.79
weave-net-krüpteeritud	468.11	10.52

Joonis 29. Testi qperf TCP ühenduste latentsus ja standardälve

Võrgutehnoloogia	Latentsus (us)	Standard-hälve
calico-ipip	22.78	0.43
calico-kapselduseta	16.56	0.51
calico-vxlan	23.83	0.71
canal-kapselduseta	19.61	0.50
canal-vxlan	27.44	3.18
cilium-geneve	25.39	2.99
cilium-vxlan	25.89	3.68
flannel-kapselduseta	16.11	0.32
flannel-vxlan	21.11	0.32
füüsiline	13.94	0.87
kube-router	19.00	0.00
weave-net	45.72	1.07
weave-net-krüpteeritud	451.89	10.78

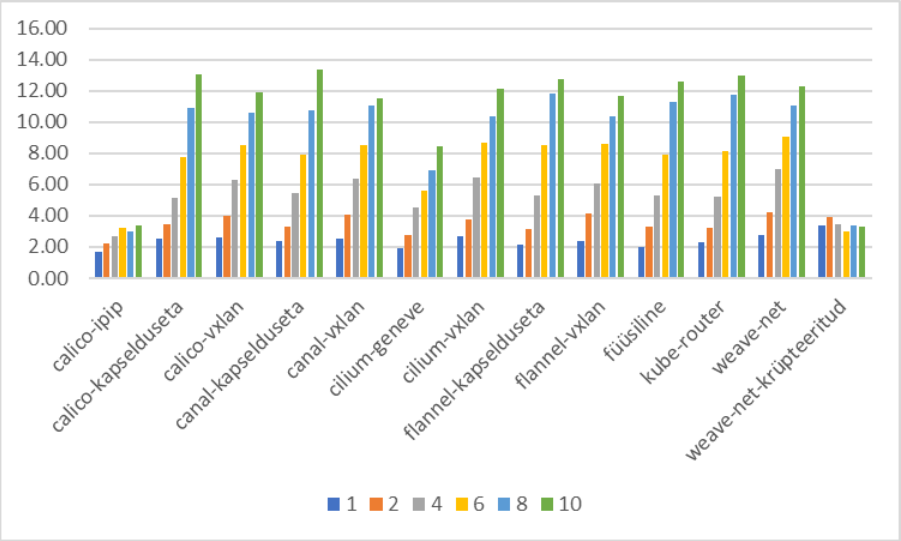
Joonis 30. Testi qperf UDP ühenduste latentsus ja standardhälve

Lisa 5 HTTP ühenduste testide tulemused

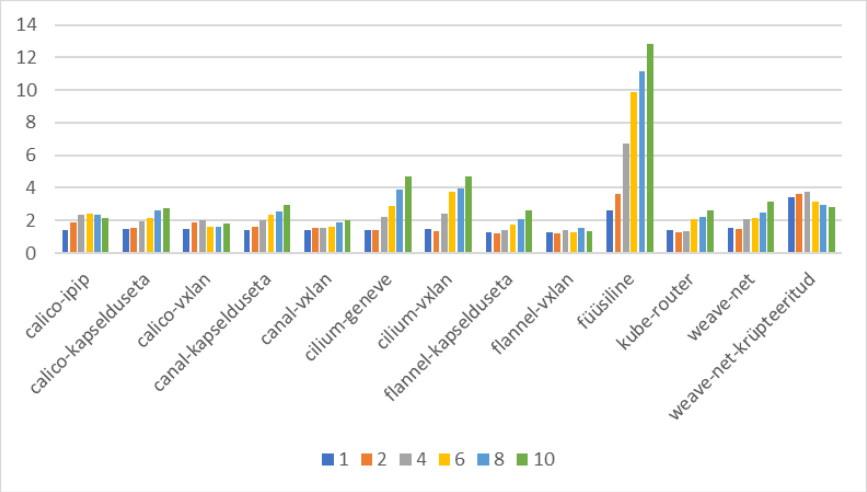
Võrgutehnoloogia	Ühendust sekundis	Standard-hälve
calico-ipip	59347.09	5526.06
calico-kapselduseta	63384.26	8481.54
calico-vxlan	50993.99	8626.01
canal-kapselduseta	53291.34	6606.77
canal-vxlan	44225.91	6913.36
cilium-geneve	54967.88	8257.85
cilium-vxlan	57500.89	9221.10
flannel-kapselduseta	61003.19	4784.54
flannel-vxlan	54065.74	8383.64
füüsiline	80208.87	9013.60
kube-router	54215.86	5982.27
weave-net	47069.28	8659.16
weave-net-krüpteeritud	10098.83	78.78

Joonis 31. HTTP ühenduste mahu testi tulemused ja standardhälve

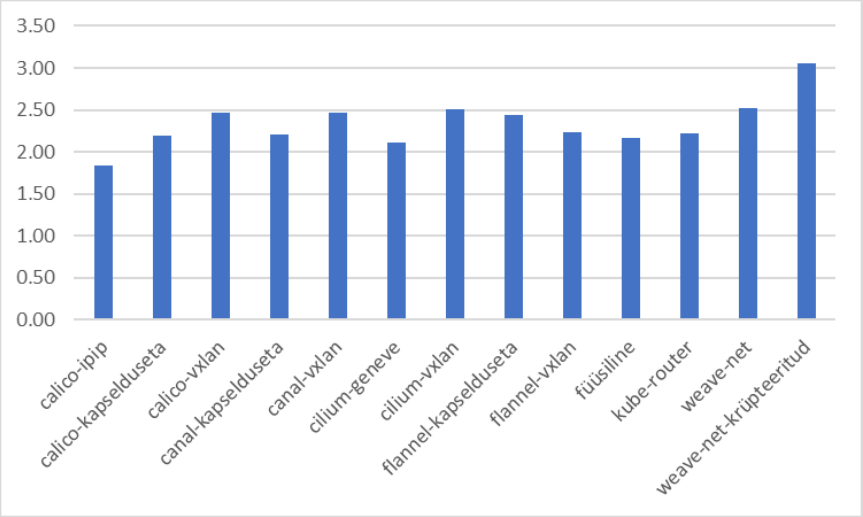
Lisa 6 Vaadeldud serverite koormus



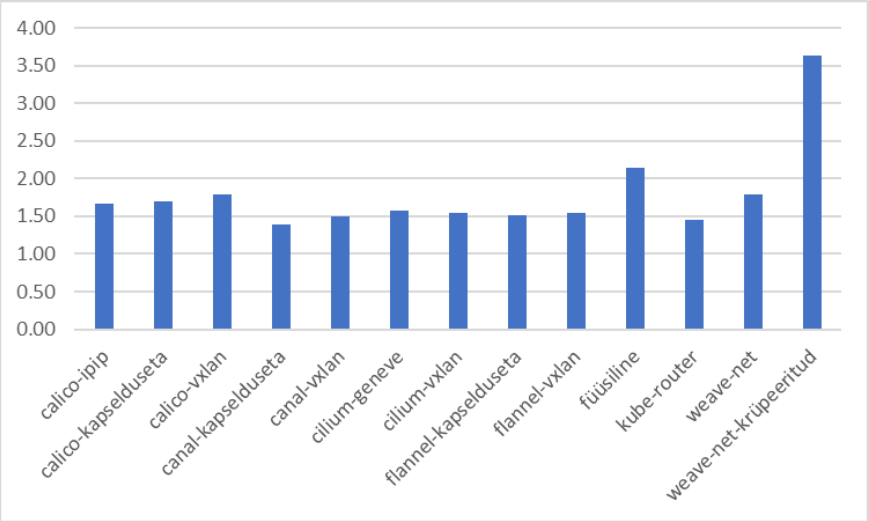
Joonis 32. Programmi iperf TCP testide serverite keskmine koormus vastavalt võtmele *-P*



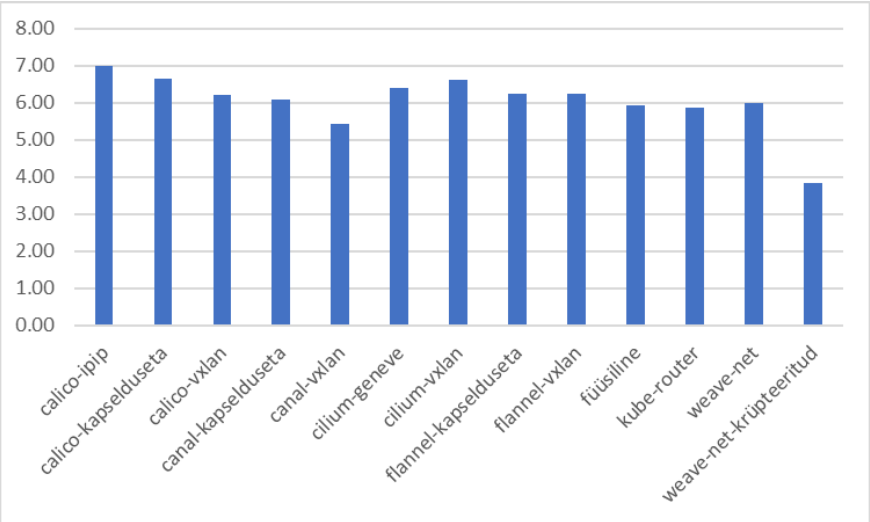
Joonis 33. Programmi iperf UDP testide serverite keskmine koormus vastavalt võtmele *-P*



Joonis 34. Programmi qperf TCP testide serverite keskmine koormus



Joonis 35. Programmi qperf UDP testide serverite keskmine koormus



Joonis 36. Programmi ab testide serverite keskmine koormus

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Johani Vajakas**

(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
Kubernetese võrgukihi tehnoloogiate jõudlu-se võrdlemine füüsilistel serveritel,
(lõputöö pealkiri)

mille juhendajad on Ivar Koppel ja Sander Kuusemets

(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Johani Vajakas

18.05.2020