**VILNIAUS UNIVERSITETAS**
**FIZIKOS FAKULTETAS**
**RADIOFIZIKOS KATEDRA**

Mindaugas Giedraitis

# INFORMACIJOS APIE TOPOLOGIJĄ HIBRIDINIUOSE TINKLUOSE SURINKIMAS PRITAIKANT SNMP

Magistrantūros studijų baigiamasis darbas

(studijų programa – TELEKOMUNIKACIJŲ FIZIKA IR ELEKTRONIKA)

| | |
|---|---|
| Studentas | M. Giedraitis |
| Darbo vadovas | Doc. Feliksas Kuliešius |
| Recenzentas | Doc. Arūnas Maršalka |
| Katedros vedėjas | Prof. Jūras Banys |

Vilnius 2017

# Turinys

# Įvadas

Kompiuterių tinklai šiuo metu yra pagrindinis komunikacijos ir informacijos sklaidos kanalas. Individualūs asmenys, verslas ir vyriausybės naudojasi jais įvairioms paslaugoms teikti ir gauti, laisvalaikiui bei darbui. Dėl plataus ir intensyvaus naudojimo, tinklams keliami labai aukšti reikalavimai saugumui, patikimumui ir našumui. Tinklams išsiplėtus, tradicinėmis priemonėmis šiuos reikalavimus yra vis sunkiau įgyvendinti: tinklas tampa sunkiai valdomas ir stebimas, problemos sprendžiamos neefektyviai.

Viena iš technologijų, tinkamų spręsti šias bei kitas problemas, yra vis daugiau pripažinimo sulaukiantys SDN (angl. *software defined networks* – programiškai apibrėžiami tinklai). Esminis šios technologijos bruožas (ir pagrindinis skirtumas lyginant su tradiciniais kompiuterių tinklais) – centralizuotas valdymas. Duomenų persiuntimo ir tinklo valdymo mechanizmai yra atskiriami taip pasiekiant didesnį tinklo funkcionalumą bei atpiginant duomenų persiuntimo įrenginius.

Įprastai dėl finansinių priežasčių ir norint išlaikyti veikiantį tinklą pertvarkos proceso metu, nauja technologija į jį įvedama palaipsniui [1, 2]. Vienu metu naujais įrenginiais pakeičiami tik keletas senų. Tinklas, sudarytas iš tradicinių tinklo įrenginių (angl. *legacy equipment*) ir SDN prietaisų, yra vadinamas hibridiniu (angl. *hybrid network*).

Turint hibridinį tinklą, siekiama integruoti tradicinius įrenginius su SDN ir turėti centralizuotą viso tinklo stebėjimą ir valdymą. Tam reikia žinoti bendrą dinamiškai kintančią tinklo topologiją bei galėti SDN valdikliu/kontroleriu (angl. *controller*) konfigūruoti tradicinius tinklo įrenginius. Šiame darbe buvo siekiama išspręsti pirmąją problemą.

Šio baigiamojo darbo tikslas – pritaikyti SNMP protokolą ir SDN kontrolerio galimybes centralizuotam hibridinio tinklo topologijos informacijos surinkimui. Jam pasiekti iškelti šie uždaviniai:

- išnagrinėti tinklo topologijos nustatymo būdus ir protokolus tradiciniuose ir OpenFlow tinkluose;
- išanalizuoti SNMP protokolo struktūrą ir sintaksę;
- nustatyti topologijos informacijai gauti reikalingus objektų identifikatorius (OID);
- parašyti ir išbandyti programą, galinčią centralizuotai surinkti hibridinio tinklo topologijos informaciją ir atvaizduoti topologiją grafiškai.

# Hibridinių tinklų topologijos nustatymo problematikos apžvalga

Hibridinių tinklų topologijos nustatymas iš esmės susideda iš kelių pagrindinių problemų. Visų pirma, tradicinių tinklų architektūra iš esmės skiriasi nuo SDN. Tradicinis tinklas yra decentralizuotas, todėl informacija apie kiekvieną tinklo įrenginį saugoma tik jame pačiame. SDN tinklas sudarytas priešingai, t.y. informacija yra kaupiama centralizuotai tinklą valdančiame kontroleryje. Norint valdyti hibridinį tinklą, visa topologijos informacija turėtų būti renkama centralizuotai.

Kadangi tradicinis tinklas yra decentralizuotas [3], nėra standartinio būdo nustatyti visai topologijos informacijai centralizuotai. Kad šis tikslas būtų pasiektas, reikia naudotis jau egzistuojančiais tinklo protokolais, kurie renka tokio pobūdžio informaciją, ją suagreguoti ir surinkti į vieną tašką. Tam galima panaudoti išorinės sąsajos (angl. *exterior gateway*) maršrutizavimo protokolą BGP arba jungties būsenos (angl. *link state*) maršrutizavimo protokolus: OSPF [2] ir IS-IS.

Tarpusavio jungčių informaciją pateikia ir tik tam skirti jungties lygmens (angl. *link layer*) protokolai: LLDP (angl. *link layer discovery protocol* – jungties lygmens aptikimo protokolas, žr. 1.2.1 skyrių) ir CDP (angl. *Cisco discovery protocol* – Cisco aptikimo protokolas, žr. 1.2.2 skyrių). Būtent jie šiame darbe pasirinkti topologijos informacijai gauti, kadangi maršrutizavimo protokolų naudojimas gali trukdyti vieningam tinklo politikos (angl. *policy*) įgyvendinimui.

Visgi, vien pastaraisiais protokolais centralizuotai informacijos surinkti negalima. Jie veikia tik vienos antrojo lygmens (L2) jungties ribose ir informaciją saugo tik tame įrenginyje, kurio jungčiai priklauso (t.y. įrenginys turi informaciją tik apie savo sąsajas su kitais įrenginiais). Norint surinkti informaciją centralizuotai, reikalingas papildomas mechanizmas, surenkantis informaciją iš kiekvieno tradicinio tinklo įrenginio. Tai galima pasiekti pasinaudojant SNMP [4] (angl. *simple network management protocol* – paprastas tinklo valdymo protokolas) protokolu (žr. 1.2.3 skyrių). LLDP (bei CDP) pateikiamą informaciją SNMP pagalba galima surinkti į centralizuotą SNMP serverį (angl. *manager*).

Antroji problema yra SDN ir tradicinių tinklo įrenginių topologijos informacijos sujungimas į bendrą viso tinklo topologijos vaizdą. Pagrindiniai iššūkiai yra dviejų skirtingų tipo informacijos (gautos iš SNMP ir iš SDN kontrolerio) transformavimas į vienodą formatą ir jos apjungimas atsižvelgiant į tai, jog vienu būtu gauta topologijos informacija gali būti ne tik nepilna, bet ir neteisinga (pvz. jungtys tarp mazgų, tarp kurių iš tikrųjų yra kitų įrenginių ar jungtys). Tam tikslui pasiekti reikalingas naujas topologijos informacijos apdorojimo algoritmas (žr. 2.3 skyrių).

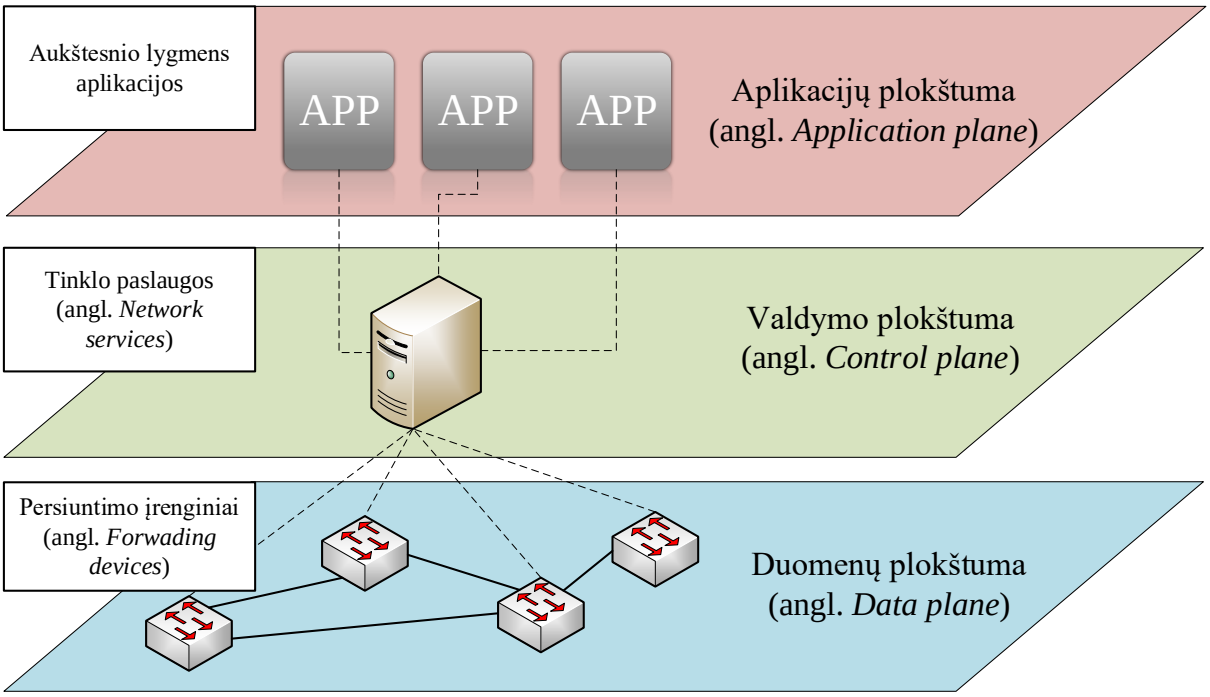
Visai tai įgyvendinus, galima gauti bendrą hibridinio tinklo topologijos vaizdą.

## Programiškai apibrėžiami tinklai

Bendra SDN architektūra (1 pav.), kuri nepriklauso nuo ją įgyvendinančio protokolo, yra pagrįsta dviem principais:

1) Duomenų, valdymo ir aplikacijų plokštumų (angl. *planes*) atskyrimu per standartizuotas sąsajas (angl. *interfaces*).
2) Centralizuotu tinklo valdymu per vieną loginį tašką (siekiant padidinti sistemos saugą, našumą ir kitus parametrus, dažnai naudojama paskirstyta architektūra, logiškai veikianti kaip vienas valdymo elementas) – kontrolerį.

Aplikacijų plokštumos programinė įranga (aplikacijos, paslaugos, etc.) kitas plokštumas traktuoja kaip vieną loginį objektą, kadangi valdymo plokštuma abstrahuoja tinklo įrenginius ir suteikia galimybę formuoti politiką nesigilinant į konkrečią tinklo struktūrą.




1 pav. Plokštumų atskyrimo principas SDN.

Duomenų plokštumą sudaro persiuntimo įrenginiai, kurių pagrindinė paskirtis yra persiųsti duomenų paketus. Valdymo plokštumoje realizuota tinklo protokolų logika. Ja remiantis formuojami valdymo plokštumos nurodymai įgalina tinklo protokolų veikimą persiuntimo lygmenyje. Aplikacijų plokštuma skirta aukštesnio lygmens sprendimams įgyvendinti: tinklo politikai, orkestravimui ar debesų kompiuterijai. Šis funkcijų išskirstymas yra pagrindinis SDN technologijos skirtumas nuo tradicinių tinklų.

## OpenFlow

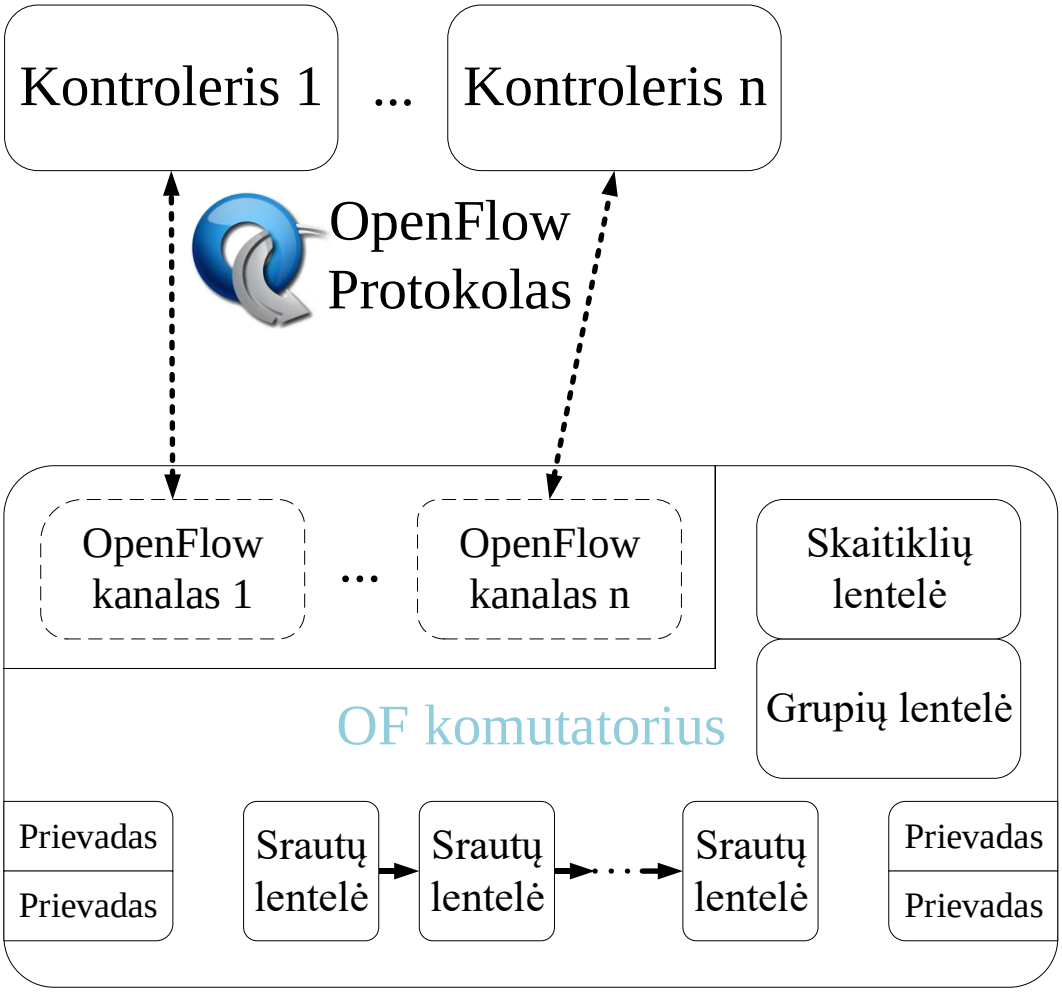
OpenFlow (OF) yra viena iš populiariausių ir labiausiai išvystytų programiškai apibrėžiamų tinklų technologijų. OpenFlow standartas [5] aprašo du pagrindinius šios technologijos aspektus:

OF komutatoriaus veikimo principą (žr. 1.1.2 ir 1.1.3 skyrių) bei komunikacijos protokolą (sąsają) tarp kontrolerio ir komutatoriaus (žr. 1.1.4 skyrių).

## OpenFlow komutatorius

Openflow komutatoriai skirstomi į tris grupes:

- Hibridiniai įrenginiai – tradicinio tinklo įrenginiai, pavyzdžiui komutatoriai ar maršrutizatoriai, kurie palaiko ir OpenFlow.
- OpenFlow komutatoriai, kurie palaiko tik OF protokolą. Šie įrenginiai atlieka tik persiuntimo funkciją, o sprendimus priima pagal gautus ir įrašytus srautų įrašus, kuriuos formuoja kontroleris. Jie veikia tik aparatiniame lygmenyje.
- Virtualūs OF komutatoriai. Jie veikia virtualizacijos platformose ir atlieka vienų iš anksčiau aprašytų komutatorių funkcijas.




2 pav. Apibendrinta OF komutatoriaus sandara.

Apibendrinta OpenFlow komutatoriaus sandara ir jo sąsaja su kontroleriu yra pavaizduota 2 paveiksle. Pagrindinės ją sudarančios dalys yra šios:

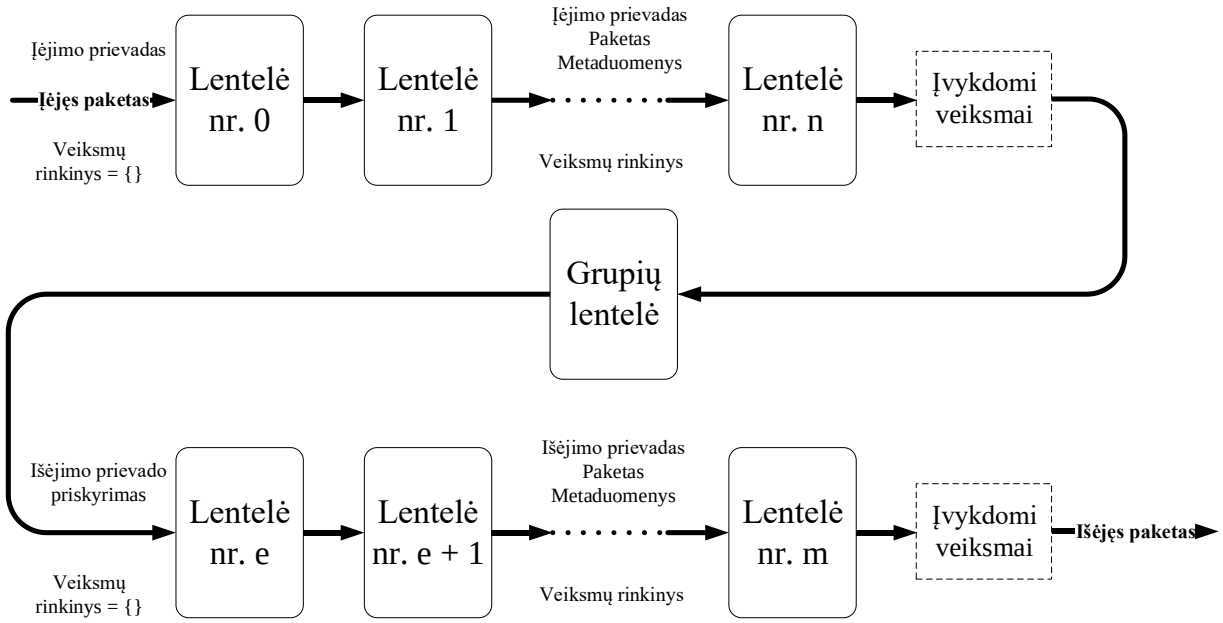
- Valdymo kanalas su vienu ar daugiau OpenFlow kanalų, prijungtų prie vieno ar kelių kontrolerių. OpenFlow kanale veikia OF protokolas.

- Tinklo prievadai, jungiantys komutatorių su kitais tinklo įrenginiais.
- Darbinis kanalas (angl. *pipeline*), kurį sudaro viena ar kelios srautų lentelės, pagal kurių įrašus nustatomas tolesnis paketo persiuntimo kelias.
- Grupių lentelė, naudojama papildomiems persiuntimo veiksmams.
- Skaitiklių lentelė, skirta įvairios statistikos rinkimui.

Kiekvienas komponentas detaliau aptariamas tolesniuose skyriuose

## OpenFlow lentelės

OpenFlow komutatoriuje kiekvienas paketas apdorojamas pagal srauto įrašus (angl. *flow entry*) srautų ir grupių lentelėse (atitinkamai *flow table* ir *group table*). Įėjusio paketo antraštės (angl *headers*) yra tikrinamos pagal šiuos įrašus ir radus atitinkantį įrašą, pritaikomi jame nurodyti veiksmai. Taip paketas gali būti apdorotas pagal vieną ar kelis įrašus ir išsiųstas per kažkokį prievadą arba atmestas. Visas procesas vyksta darbiniame kanale ir yra pavaizduotas 3 paveiksle.




3 pav. OpenFlow komutatoriaus paketo apdorojimas darbiniame kanale [5].

Komutatoriuje yra viena ar kelios srautų lentelės, kurios sunumeruotos nuo 0. Pirmiausia paketo antraštę atitinkančio įrašo ieškoma pirmoje lentelėje, o jį radus, paketas pagal jį apdorojamas. Jei vis dėlto atitinkantis paketą įrašas nerandamas, taikomas paskutinis įrašas (nulinio prioriteto), kuris yra tinkamas bet kokiam paketui ir gali atmesti paketą arba persiųsti jį kontroleriui. Toks įrašas vadinamas neatitinkančios lentelės srauto įrašu (angl. *table-miss flow entry*).

## Srautų lentelė

Komutatoriuje gali būti viena ar kelios srautų lentelės. Jas sudaro srauto įrašai, nuo kurių priklauso paketo apdorojimas. Tikrinimas pradedama pirmoje (nr. 0) lentelėje aukščiausio prioriteto

įrašu. Jei paketas jo neatitinka, tikrinamas kitas įrašas ir taip toliau, kol randamas tinkamas. Bendra srauto įrašo struktūra aprašyta 1 lentelėje.

1 lentelė. Srauto įrašo struktūra [6, 7].

| **Lauko pavadinimas** | **Paaiškinimas** |
|---|---|
| Tikrinimo laukai | Aprašomi paketo antraštės laukai, pagal kuriuos tikrinamas paketo atitikimas. Literatūroje išskiriami tokie laukai (versija 1.0):<br>1. Įėjimo prievadas<br>2. VLAN identifikatorius<br>3. VLAN identifikatoriaus prioritetas (angl. *Class of service* - CoS)<br>4. Šaltinio MAC adresas<br>5. Paskirties MAC adresas<br>6. Ethernet paketo tipas<br>7. Šaltinio IP adresas<br>8. Paskirties IP adresas<br>9. IP protokolo tipas<br>10. IP prioritetas (angl. *Type of service* – ToS)<br>11. Šaltinio TCP/UDP prievadas<br>12. Paskirties TCP/UDP prievadas<br>Nuo 1.1 versijos atsirado galimybė pridėti ir tikrinti metaduomenų lauką, taip perduodant informaciją tarp srautų lentelių.<br>Nuo 1.2 versijos galima aprašyti TLV laukus (angl. *type-length-value*), taip užtikrinant bet kokio tinklo protokolo palaikymą. |
| Prioritetas | Pagal prioritetą nustatomas srauto įrašų tikrinimo eiliškumas – didesnis skaičius reiškia pirmenybę. Įrašas su prioritetu 0 gali būti taikomas visiems paketams (angl. *wildcard*), t.y. kai paketas neatitinka nė vieno srauto įrašo. |
| Skaitliukai (angl. *counters*) | Statistikos skaitliukai, atnaujinami, kai paketas yra apdorojamas pagal srauto įrašą. |
| Instrukcijos | Instrukcijos modifikuoja veiksmų rinkinį, kuris turi būti atliktas su paketu:<br>• Generuoja pranešimą kontroleriui<br>• Pritaiko veiksmus iškart<br>• Išvalo veiksmų rinkinį<br>• Įrašo veiksmus į veiksmų rinkinį<br>• Įrašo metaduomenis<br>• Persiunčia paketą į kitą srautų lentelę |
| Laikmatis | Maksimalus srauto galiojimo laikas arba neaktyvumo laikas sekundėmis, kuriam praėjus srautas ištrinamas iš lentelės. Reikšmė 0 reiškia, jog srauto įrašas nenustoja galioti. |
| Slapukai (angl. *cookies*) | Papildoma srauto įrašo informacija grąžinama kontroleriui. |

Radus paketą atitinkantį srauto įrašą paketui pritaikomas instrukcijų rinkinys, kuris aprašytas srauto įraše. Pagrindinis instrukcijų tikslas yra įrašyti ar pakeisti veiksmus paketui skirtame

veiksmų rinkinyje. Svarbiausi veiksmai pateikti 2 lentelėje. Dažniausiai naudojami veiksmai yra atmetimas ir persiuntimas į prievadą. Galimų OpenFlow komutatoriaus prievadų sąrašas pateiktas 3 lentelėje. Paprastai skiriamos 3 prievadų rūšys: fiziniai, loginiai ir rezervuoti.

2 lentelė. Pagrindiniai veiksmai OpenFlow veiksmų rinkinyje [5].

| **Veiksmas** | **Paaiškinimas** |
|---|---|
| DROP | Atmesti. |
| OUTPUT *prievado_nr* | Persiųsti paketą į vieną iš prievadų. |
| SET-FIELD *lauko_tipas reikšmė* | Modifikuoti paketo antraštę. |
| SET-QUEUE *eilės_nr* | Siųsti paketą į vieną iš QoS eilių. |
| COPY-FIELD *šaltinio_lauko_tipas paskirties_lauko_tipas* | Kopijuoti paketo antraštės lauką į kitą antraštės lauką, ar paketo registrą. |

3 lentelė. Galimi OpenFlow komutatoriaus prievadai [5].

| **Prievadas** | **Paaiškinimas** |
|---|---|
| Fizinis prievadas | Prievadas, kuris yra tiesiogiai susietas su konkrečiu aparatiniu prievadu (pvz. FastEthernet 0/1). |
| Loginis prievadas | Prievadas, kuris yra susietas su aukštesnio lygmens abstrakcija, aprašyta ne OF metodais hibridiniame komutatoriuje (pvz. tuneliai arba agreguoti prievadai). |
| ALL | Rezervuotas prievadas, susietas su visais, paketą galinčiais išsiųsti fiziniais prievadais. |
| CONTROLLER | Sąsaja su kontroleriu ar kontroleriais. |
| TABLE | Komutatoriaus darbinis kanalas (pirma srautų lentelė). |
| IN_PORT | Paketo įvesties prievadas. |
| ANY | Speciali reikšmė, naudojama OF pranešimuose, kai nenorima nurodyti prievado (nėra naudojamas nei kaip įvesties, nei kaip išvesties prievadas). |
| UNSET | Reikšmė, nurodanti, kad prievadas nebuvo priskirtas. |
| LOCAL | Prievadas, skirtas nutolusių įrenginių bendravimui su komutatoriumi ir jo valdymui. |
| NORMAL | Prievadas, galimas tik hibridiniame įrenginyje, skirtas persiųsti paketą į tradicinio tinklo įrenginio darbinį kanalą. |
| FLOOD | Prievadas, galimas tik hibridiniame įrenginyje, skirtas persiųsti paketą į visus fizinius prievadus, atsižvelgiant į tradicinių protokolų (pvz. STP – angl. *spanning tree protocol*) blokavimus. |

## Grupių lentelė

Grupių lentelė naudojama papildomiems sudėtingesniems persiuntimo veiksmams, pavyzdžiui, grupiniam transliavimui (angl. *multicast*) atlikti. Ji sudaryta iš grupės įrašų (angl. *group entry*). Grupės įrašo struktūra pateikta 4 lentelėje.

4 lentelė. Grupės įrašo struktūra [5].

| Lauko pavadinimas | Paaiškinimas |
|---|---|
| Grupės identifikatorius | 32 bitų unikalus grupės numeris |
| Grupės tipas | Tipas nusakantis grupės semantiką. |
| Skaitliukai (angl. *counters*) | Statistikos skaitliukai, atnaujinami, kai paketas yra apdorojamas pagal grupės įrašą. |
| Veiksmų rinkinių eilė (angl. *action bucket*) | Sąrašas veiksmų rinkinių ir papildomų parametrų. |

Kitaip nei srauto įrašuose, grupės įrašai turi ne prioritetą, o grupės identifikatorių, nes paketas persiunčiamas į grupių lentelę tik per srauto įraše esančius veiksmus, kurie nukreipia paketą tiesiai į konkretų grupės įrašą.

Grupės įrašas gali būti vieno iš šių grupės tipų:

- Netiesioginis (angl. *indirect*). Paprasčiausia grupė, kuri palaiko vieną veiksmų rinkinių eilę. Ji yra identiška visiškai grupei su viena veiksmų rinkinių eile.
- Visiškas (angl. *all*). Palaiko daug veiksmų rinkinių eilių ir vykdo jas visas. Naudojamas grupiniam transliavimui. Paketas yra klonuojamas kiekvienai eilei.
- Pasirinkimo (angl. *select*). Vykdo vieną eilę iš kelių turimų.
- Greito atsistatymo (angl. *fast failover*). Vykdo pirmą eilę, kuri yra susieta su veikiančiu prievadu.

Veiksmų rinkinių eilėje paprastai yra veiksmas, kuris modifikuoja paketą ir persiuntimo veiksmas. Joje taip pat gali būti nurodymas perduoti paketą į kitą grupę (grupės įrašą), kuris vadinamas grupės veiksmu (angl. *group action*).

Yra galimas ir įrašas be veiksmų eilės. Jei įraše nėra persiuntimo ar grupės veiksmų, paketas yra atmetamas.

## Skaitiklių lentelė

Skaitiklių lentelė (angl. *meter table*) sudaryta iš skaitiklių įrašų (angl. *meter entry*)[1], kurie yra susieti su srauto įrašais. Kiekvienas skaitiklis (įrašas) matuoja jam priskirtų paketų (jie ateina per srauto įrašus) spartą ir gali tą spartą apriboti. Skaitiklių įrašo sandara pateikta 5 lentelėje.

[1] Nepainioti su skaitliukais (angl. *counter*). Skaitliukai skirti statistikos rinkimui, o skaitikliai (angl. *meter*) yra skaitiklių lentelės įrašai, pagal kuriuos yra formuojami paslaugų prioritetai (angl. *quality of service* – QoS).

5 lentelė. Skaitiklių įrašo sandara [5].

| Lauko pavadinimas | Paaiškinimas |
|---|---|
| Skaitiklio identifikatorius | 32 bitų unikalus skaitiklio numeris |
| Skaitiklio diapazonai (angl. *meter bands*) | Diapazonų, kurių kiekviename nurodyta paketų sparta ir veiksmai, kurie turi būti atlikti, jei paketų sparta viršija nurodytąją, sąrašas. |
| Skaitliukai (angl. *counters*) | Statistikos skaitliukai, atnaujinami, kai paketas yra apdorojamas pagal skaitiklių įrašą. |

Skaitikliai yra skirti paslaugų prioritetams (angl. *quality of service* – QoS) tinkle nustatyti. Kuomet paketas, apdorojamas konkretaus skaitiklio įrašo viršija kurį nors skaitiklio diapazoną, jam yra pritaikomi tame diapazone nurodyti veiksmai. Pagrindiniai skaitiklio diapazono laukai pavaizduoti 6 lentelėje.

OpenFlow specifikacijoje [5] nurodomi 2 neprivalomi (angl *optional*) diapazonų tipai:

- Atmetimo (angl. *drop*). Atmeta atitikusius paketus. Gali būti naudojamas nustatant srauto limitą.
- DSCP žyma (angl. *DSCP remark*). Padidina atmetimo prioritetą DSCP[2] lauke paketo IP antraštėje. Leidžia formuoti sudėtingesnę prioritetų politiką tinkle.

Šių tipų OF komutatorius gali ir nepalaikyti, tačiau gali palaikyti ir daugiau. Kontroleris gali siųsti užklausas komutatoriui apie jo palaikomus diapazonų tipus.

6 lentelė. Skaitiklių diapazono sandara [5].

| Lauko pavadinimas | Paaiškinimas |
|---|---|
| Diapazono tipas | Nusako, kaip paketai bus apdoroti (kokie veiksmai pritaikyti). |
| Sparta (angl. *rate*) | Sparta, pagal kurią pasirenkamas diapazonas. Paprastai žemiausia sparta, kurią viršijus pritaikomas diapazonas. |
| Pliūpsnis (angl. *burst*) | Nusako paketų pliūpsnio, kurio sparta gali viršyti diapazone nurodytą spartą, tačiau diapazonas nėra pritaikomas paketams, trukmę. |
| Skaitliukai (angl. *counters*) | Statistikos skaitliukai, atnaujinami, kai paketas yra apdorojamas pagal skaitiklių diapazoną. |
| Papildomi argumentai | Gali būti vienas ar keli papildomi argumentai priklausomai nuo diapazono tipo. |

[2] DSCP (angl. *differentiated services code point*) – paslaugų atskyrimo lauko DS, IP antraštėje pakeitusio pasenusį paslaugos tipo (angl. *type of service* – ToS) lauką, dalis.

## OpenFlow kanalas

OpenFlow kanalas yra sąsaja, kurioje veikia OpenFlow komunikacijos protokolas, ir kuriuo kontroleris konfigūruoja bei valdo OF komutatorių, o komutatorius jam siunčia įvairių įvykių informaciją. Vienas komutatorius gali palaikyti keletą OF kanalų ir būti sujungtas su keliais kontroleriais (1 paveikslas). Bendrai visi OpenFlow kanalai vadinami valdymo kanalu. Paprastai OF kanale veikia TLS (angl. *transport layer security*) šifravimas. Galima ir vien TCP veika.

OpenFlow pranešimai skirstomi į tris grupes:

- Kontrolerio inicijuoti pranešimai (angl. *controller-to-switch*).
- Asinchroniniai pranešimai (angl. *asynchronous*).
- Simetriniai pranešimai (angl. *symmetric*).

## Kontrolerio inicijuoti pranešimai

Kontroleris inicijuoja šiuos pranešimus, kad galėtų sužinoti komutatoriaus būseną, galimybes bei jį valdyti. Jiems reikalingas atsakymas iš komutatoriaus. Šių pranešimų rūšys pateiktos 7 lentelėje.

7 lentelė. Kontrolerio inicijuoti pranešimai [6].

| **Pranešimo pavadinimas** | **Paaiškinimas** |
|---|---|
| Savybės (angl. *feature*) | Kontroleris gali siųsti savybių užklausą, kad sužinotų OF komutatoriaus tapatybę ir galimybes. Komutatorius atsako savybių aprašu. Ši užklausa dažniausiai siunčiama užmezgant OpenFlow kanalą. |
| Konfigūracija (angl. *configuration*) | Kontroleris gali nustatyti arba sužinoti komutatoriaus konfigūracijos parametrus. Komutatorius atsako tik į užklausas. |
| Būsenos modifikavimas (angl. *modify-state*) | Šie pranešimai siunčiami, kad kontroleris galėtų modifikuoti komutatoriaus būseną. Pagrindinis jų tikslas – srauto ir grupės įrašų įrašymas, trynimas ir modifikavimas, bei prievadų būsenų nustatymas |
| Būsenos nuskaitymas (angl. *read-state*) | Būsenos nuskaitymo pranešimai naudojami siekiant sužinoti įvairią informaciją apie komutatorių: konfigūraciją, statistikas. |
| Paketo išsiuntimas (angl. *packet-out*) | Šiais pranešimais kontroleris instruktuoja komutatorių išsiųsti jo suformuotą paketą per kažkurį prievadą. Šiame pranešime būna pats paketas arba jo identifikatorius komutatoriaus buferyje, bei veiksmai, kuriuos reikia atlikti su paketu (tuščias veiksmų sąrašas reiškia, jog paketas bus atmestas). Įprastai šis pranešimas naudojamas išsiųsti paketą, gautą „atėjusio paketo“ pranešime (žr. skyrių „Asinchroniniai pranešimai“), kuomet kontroleris nusprendžia, ką su paketu daryti. |
| Barjeras (angl. *barrier*) | Barjero užklausos ir atsakymai naudojami pranešimų priklausomybių (angl. *dependencies*) tenkinimo tikrinimui bei informacijos apie pabaigtas operacijas gavimui. |

| | |
|---|---|
| Rolė (angl. *role-request*) | Kontroleris naudoja šias užklausas siekiant modifikuoti arba sužinoti OpenFlow kanalo rolę ir savo identifikatorių komutatoriaus sistemoje. Jos naudojamos, kai yra daugiau nei vienas kontroleris. |
| Asinchroninė konfigūracija (angl. *asynchronous-configuration*) | Naudojama siekiant patikrinti, uždėti ar modifikuoti filtrus asinchroniniams pranešimams. Šie pranešimai naudingi esant keliems kontroleriams. |

## Asinchroniniai pranešimai

Asinchroninius pranešimus inicijuoja komutatorius be kontrolerio užklausos. Jų paskirtis yra pranešti kontroleriui apie būsenos komutatoriuje arba kanale pasikeitimus, arba jam perduoti atėjusio paketo valdymą, kuomet paketas neatitiko jokio srauto įrašo. Svarbiausi asinchroniniai pranešimai pateikti 8 lentelėje.

8 lentelė. Asinchroniniai pranešimai [5].

| **Pranešimo pavadinimas** | **Paaiškinimas** |
|---|---|
| Atėjęs paketas (angl. *packet-in*) | Šis pranešimas siunčiamas, kuomet paketas yra persiunčiamas į rezervuotą CONTROLLER prievadą. Kontroleriui perduodamas visas paketas arba, išsaugant jo duomenis komutatoriaus buferyje, jo antraštė ir paketo identifikatorius buferyje. |
| Pašalintas srautas (angl. *flow-removed*) | Informuoja kontrolerį apie ištrintą arba baigusį galioti srauto įrašą, kai srauto įrašas turi atitinkamą šio pranešimo siuntimo reikalaujančią žymę. |
| Prievado būsena (angl. *port-status*) | Pranešimai apie prievado būsenos pakitimus ar konfigūracijos pakitimus. Pvz.: prievado gedimas, prievado įjungimas ar išjungimas. |
| Rolės būsena (angl. *role-status*) | Praneša kontroleriui apie jo rolės pasikeitimus. Jei naujas kontroleris išsirenka save kaip pagrindinį (angl. *master*), komutatorius informuoja apie tai ankstesnį pagrindinį kontrolerį. |
| Kontrolerio būsena (angl. *controller-status*) | Informuoja kontrolerį apie OpenFlow kanalo būsenos pasikeitimus. Pranešimas siunčiamas visiems prijungtiems kontroleriams. |
| Srauto monitorius (angl. *flow-monitor*) | Kontroleris gali apibrėžti monitorius, kurie seka pasikeitimus srautų lentelėse. Šis pranešimas siunčiamas, kai įvyksta pakitimas, atitinkantis nustatytą monitorių. |

## Simetriniai pranešimai

Simetriniai pranešimai yra tokie, kuriuos naudoja tiek kontroleris, tiek komutatorius. Juos inicijuoti gali abi pusės. Jų tipai pateikti 9 lentelėje.

9 lentelė. Simetriniai pranešimai [5].

| **Pranešimo pavadinimas** | **Paaiškinimas** |
|---|---|
| Pasveikinimo pranešimai (angl. *hello*) | Šie pranešimai yra siunčiami tarp kontrolerio ir komutatoriaus OpenFlow kanalo sukūrimo metu. |
| Aidas/atsakymas (angl. *echo*) | Naudojami norint patikrinti kitos pusės pasiekiamumą. Naudojant šiuos pranešimus galima pamatuoti kanalo vėlinimą ir pralaidumą. |
| Klaida (angl. *error*) | Klaidos pranešimai informuoja kitą pusę apie sujungimo problemas. Dažniausiai siunčiami iš komutatoriaus kontroleriui, pranešant apie neįvykdytą kontrolerio užklausą. |
| Eksperimentiniai pranešimai (angl. *experimenter*) | Šie pranešimai skirti praplėsti OpenFlow funkcionalumą ateityje, arba kaip standartinis būdas įgyvendinti savo funkcijas. |

## Topologijos informacijos surikimas tradiciniuose tinkluose

Tradicinės architektūros tinkluose nėra numatyto standartinio ir plačiai naudojamo būdo naudojamo detaliai topologijai nustatyti, kadangi sprendimai tinklo įrenginiuose priimami lokaliai ir užtenka turėti informaciją tik apie artimiausią aplinką. Detali tinklo topologija naudojama ne paties tinklo darbui ir jo optimizavimui, bet tinklo administravimui. Vis dėlto, yra pasiūlytas ne vienas būdas, kaip tą padaryti pasinaudojant įvairiaisiais egzistuojančiais protokolais, pavyzdžiui: OSPF [8, 9], BGP [9] ar SNMP [4, 9, 10]. Vieni iš jų skirti tinklo būsenai ar topologijai nustatyti, kiti gali būti maršrutizuojantys protokolai, turintys topologijos nustatymo funkciją. Šiame darbe įrenginių tarpusavio jungčių informacijai gauti pasirinkti LLDP ir CDP protokolai, o surinkti informacijai iš įrenginių pasirinktas SNMP protokolas.

## LLDP

LLDP yra jungties lygmens protokolas skirtas įrenginiams pranešti (angl. *advertise*) apie savo technines galimybes, valdančio įrenginio ar įrenginių adresus bei jungties identifikaciją. Kaimyninių įrenginių informacija, gauta šiuo protokolu yra laikoma MIB duomenų bazėje, kad ją būtų galima pasiekti SNMP protokolu (žr. 1.2.3 skyrių).

Šis protokolas yra vienkryptis. LLDP subjektas (angl. *agent*) gali siųsti ir gauti LLDP informaciją, tačiau šio protokolo standarte nėra numatyto būdo siųsti užklausoms ar patvirtinimams. Dėl šios priežasties yra galimybė vienai pusei tik priimti, o kitai tik siųsti LLDP paketus.

LLDP yra antro lygmens (L2) protokolas, skirtas lokaliems tinklams (angl. *local area network* – LAN) ir standarte [11] yra aprašytas LLDP duomenų perdavimas Ethernet paketuose. Bendra paketo struktūra pateikta 10 lentelėje. Paskirties MAC adresui naudojamas vienas iš trijų protokolui rezervuotų adresų. Ethernet paketo tipas – 0x88CC.

Pats LLDP paketas, dar vadinamas LLDPDU (angl. *LLDP data unit*), yra sudarytas iš TLV laukų. Kiekviename pakete turi būti 4 privalomi TLV laukai ir bet kiek (tačiau neviršijant didžiausio galimo Ethernet paketo dydžio) papildomų neprivalomų TLV laukų. Kiekvienas laukas nusako kažkokią siunčiančios sistemos savybę.

Bet koks (nebūtinai LLDP protokolo) TLV laukas yra sudarytas iš trijų dalių:

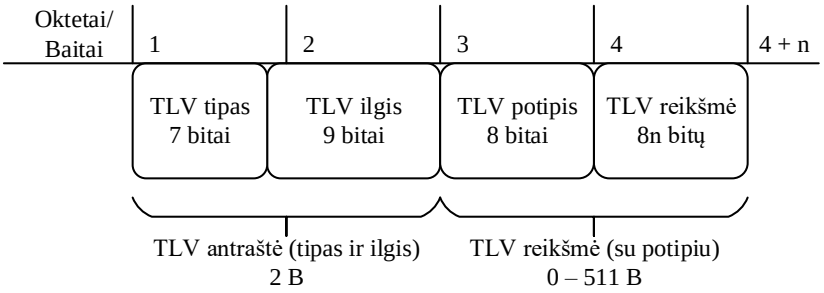
- Tipo. Skaičius, reiškiantis perduodamos informacijos tipą (pvz. prievado ID arba operacinės sistemos pavadinimą).
- Ilgio. Skaičius, nurodantis reikšmės lauko ilgį, skirtas programiniam ieškikliui (angl. *crawler*), kad šis galėtų atskirti, kur vienas TLV laukas baigiasi, o kitas prasideda.
- Reikšmės, kartais dar vadinamos informacine eilute (angl. *information string*). Perduodama reikšmė. Jos ilgis baitais turi sutapti su ilgio lauko reikšme. Reikšmės

lauke 1 baitas gali būti išskirtas reikšmės potipiui nusakyti (pvz. sistemos ID potipis – MAC adresas, reikšmė – pats adresas).

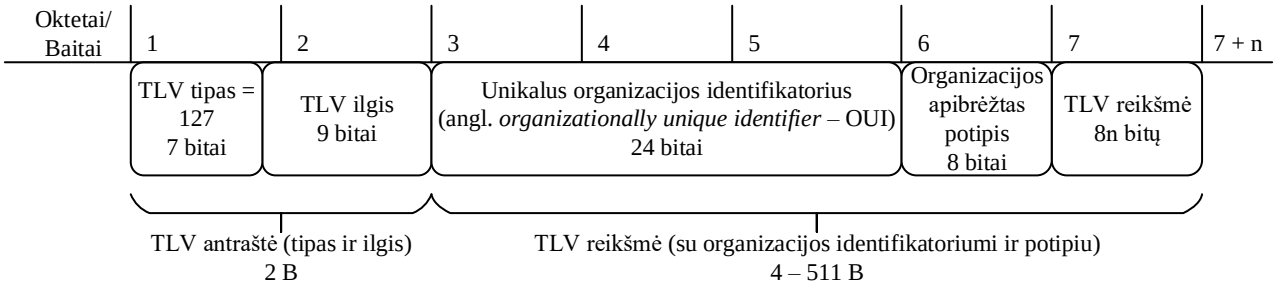
10 lentelė. LLDP paketo struktūra [11].

| **Laukas** | **Ilgis** | **Reikšmė/Paaiškinimas** |
|---|---|---|
| Preambulė | 7B | 01010101 01010101 01010101 01010101 01010101 01010101 01010101 |
| Kadro pradžios skirtukas (angl. *frame start delimiter*) | 1B | 01010111 |
| Paskirties MAC adresas | 6B | 01:80:C2:00:00:0E, 01:80:C2:00:00:03 arba 01:80:C2:00:00:00 |
| Šaltinio MAC adresas | 6B | Konkretus šaltinio prievado adresas |
| Paketo ilgis/tipas | 2B | 0x88CC |
| Sistemos ID (angl. *chassis ID*) TLV | $x_1$B | Tipas = 1. Privalomas. Reikšmė nurodo sistemos ID formatą ir patį ID. |
| Prievado ID (angl. *port ID*) TLV | $x_2$B | Tipas = 2. Privalomas. Reikšmė nurodo prievado ID formatą ir patį ID. |
| Gyvavimo trukmės (angl. *tme to live* - TTL) TLV | 4B | Tipas = 3. Privalomas. Nurodo informacijos galiojimo trukmę sekundėmis. |
| Papildomi neprivalomi TLV | $x_3$B | 0 arba daugiau neprivalomų TLV. |
| LLDPDU pabaigos TLV | 2B | Tipas = 0, ilgis = 0. Privalomas. Reiškia LLDPDU pabaigą. |
| FCS (angl. *frame check sequence*) | 4B | Paketo integralumui užtikrinti naudojama duomenų suma. |

Yra skiriami 2 TLV tipai: įprasti (4 paveikslas) ir organizacijų apibrėžti (angl. *organizationally specific*) (5 paveikslas). Organizacijų apibrėžti TLV turi savo atskirą rezervuotą tipą – 127, bei papildomą lauką prieš TLV potipį, kuriame nurodomas unikalus organizacijai priskirtas 3 baitų identifikatorius.




4 pav. Įprasto LLDP TLV struktūra.




5 pav. Organizacijos apibrėžto LLDP TLV struktūra.

LLDP paprastai surenkama tokia informacija:

- Sistemos pavadinimas ir aprašas.
- Prievado pavadinimas ir aprašas.
- Virtualaus vietinio tinklo (VLAN) pavadinimas.
- Valdymo IP adresas.
- Sistemos galimybės (komutavimas, maršrutizavimas ir t.t.).
- Fizinio lygmens informacija (pvz. MAC adresas).
- PoE (angl. *power over Ethernet*) informacija.
- Sąsajų agregavimo informacija.

Visa surinkta informacija (TLV reikšmės) yra saugomos MIB duomenų bazėje, kur ja galima pasiekti SNMP protokolu.

## CDP

CDP yra kompanijos Cisco sukurtas LLDP atitikmuo. Jo veikimo principas yra analogiškas LLDP – t.y. CDP siunčia periodinius skelbimus apie įrenginio galimybes ir kitą informaciją savo kaimyniniams įrenginiams. CDP paketo sandara yra pateikta 11 lentelėje. CDP protokolo informacija taip pat perduodama TLV laukais. Pagrindinė priežastis, kodėl šiame darbe numatyta galimybė pasinaudoti CDP surinkta informacija yra tai, jog kai kurie tradiciniai tinklo įrenginiai gali nepalaikyti LLDP, tačiau palaikyti CDP.

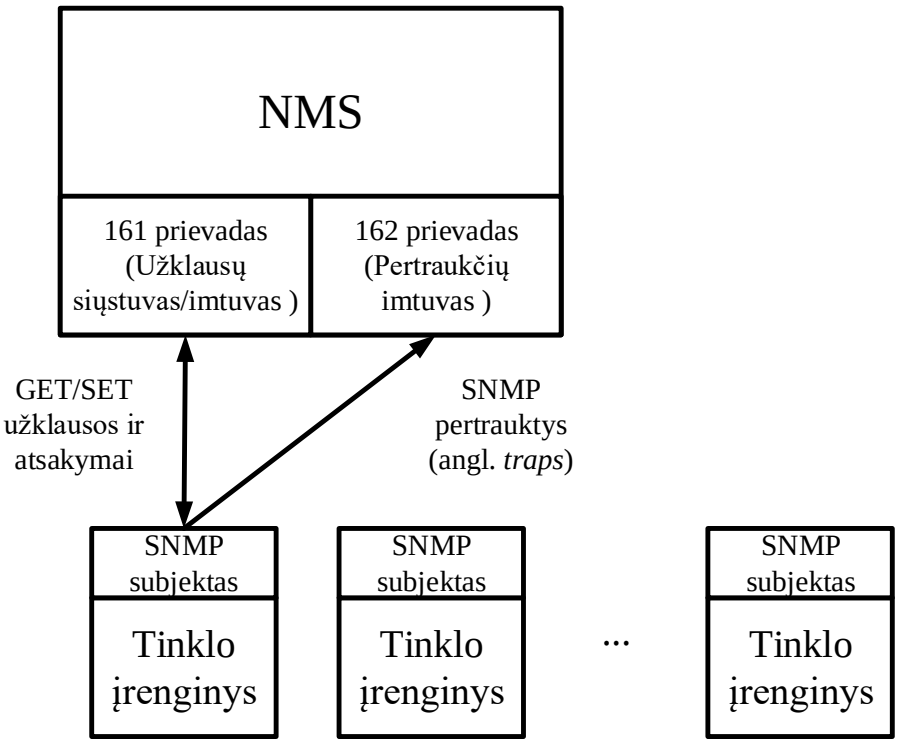
11 lentelė. CDP paketo struktūra.

| Laukas | Ilgis | Reikšmė/Paaiškinimas |
|---|---|---|
| Preambulė | 7B | 01010101 01010101 01010101 01010101 01010101 01010101 01010101 |
| Kadro pradžios skirtukas (angl. *frame start delimiter*) | 1B | 01010111 |
| Paskirties MAC adresas | 6B | 01:00:0C:CC:CC:CC. CDP/VTP grupinio transliavimo adresas. |
| Šaltinio MAC adresas | 6B | Konkretus šaltinio prievado adresas |
| Paketo ilgis/tipas | 2B | Nurodomas ilgis (CDP neturi rezervuoto Ethernet paketo tipo). |
| LLC antraštė | 3B | 0xAAAA03 |
| Organizacijos ID | 3B | 0x00000C. Kompanijos Cisco kodas |
| HDLC protokolo tipas | 2B | 0x2000. CDP protokolo identifikatorius. |
| Informacija | $x_1$B | Perduodama informacija TLV laukų pavidalu. |
| FCS (angl. *frame check sequence*) | 4B | Paketo integralumui užtikrinti naudojama duomenų suma. |

## SNMP

SNMP – IETF (angl *Internet Engineering Task Force*) standartizuotas protokolas tinklo įrenginių informacijai rinkti bei organizuoti ir įrenginiams valdyti. Šis protokolas veikia aplikacijų lygmenyje TCP/IP protokolų steke. Bendra aplikacijos struktūra yra pagrįsta kliento – serverio modeliu. Tinklo įrenginiai, kuriuose veikia SNMP savyje turi SNMP subjektą (angl. *SNMP agent*) – tai yra programinė įranga, įgalinanti SNMP veikimą valdomuose (angl. *managed*) įrenginiuose. Jie yra sujungti su SNMP serveriu dar vadinamu tinklo valdymo stotimi (angl. *network management station* – NMS). Supaprastinta tokio modelio schema pateikta 6 paveiksle.

Paketų transportui SNMP naudoja UDP protokolą. SNMP rezervuoti 161 ir 162 UDP prievadai. 161 prievadas naudojamas užklausomas į įrenginius siųsti ir atsakymams gauti, o 162 – asinchroninėms pertrauktims (angl. *traps*) iš įrenginių priimti.




6 pav. SNMP modelio schema.

Įrenginių autentifikavimui SNMP naudoja slaptažodžius, kurie vadinami „bendruomenės eilutėmis“ (angl. *community strings* arba *communities*). Pirmoje (SNMPv1) ir antroje (SNMPv2c) versijoje šie slaptažodžiai paketuose yra siunčiami atviru tekstu (t.y. nešifruoti) ir dėl to jų perdavimas nėra saugus. Naujausia standartinė versija – SNMPv3 palaiko saugią įrenginių autentifikaciją.

SNMP (v1 ir v2) naudoja tris slaptažodžius [12]:

- Skaitymo (angl. *read-only*). Su šiuo slaptažodžiu NMS gali nuskaityti įrenginių duomenis, bet negali įrenginiuose keisti jokių nustatymų.
- Skaitymo ir rašymo (angl. *read-write*). NMS gali ir skaityti duomenis, ir keisti nustatymus.
- Pertraukčių (angl. *trap*). NMS gali priimti pertrauktis.

Valdomuose įrenginiuose visa informacija, kurią gali pasiekti SNMP, yra laikoma MIB duomenų bazėse, kurių struktūrą apibrėžia bendri standartai, atskirų protokolų (pvz. LLDP ar CDP) specifikacijos tų protokolų informacijai saugoti, ir gamintojų apibrėžtos MIB duomenų bazės.

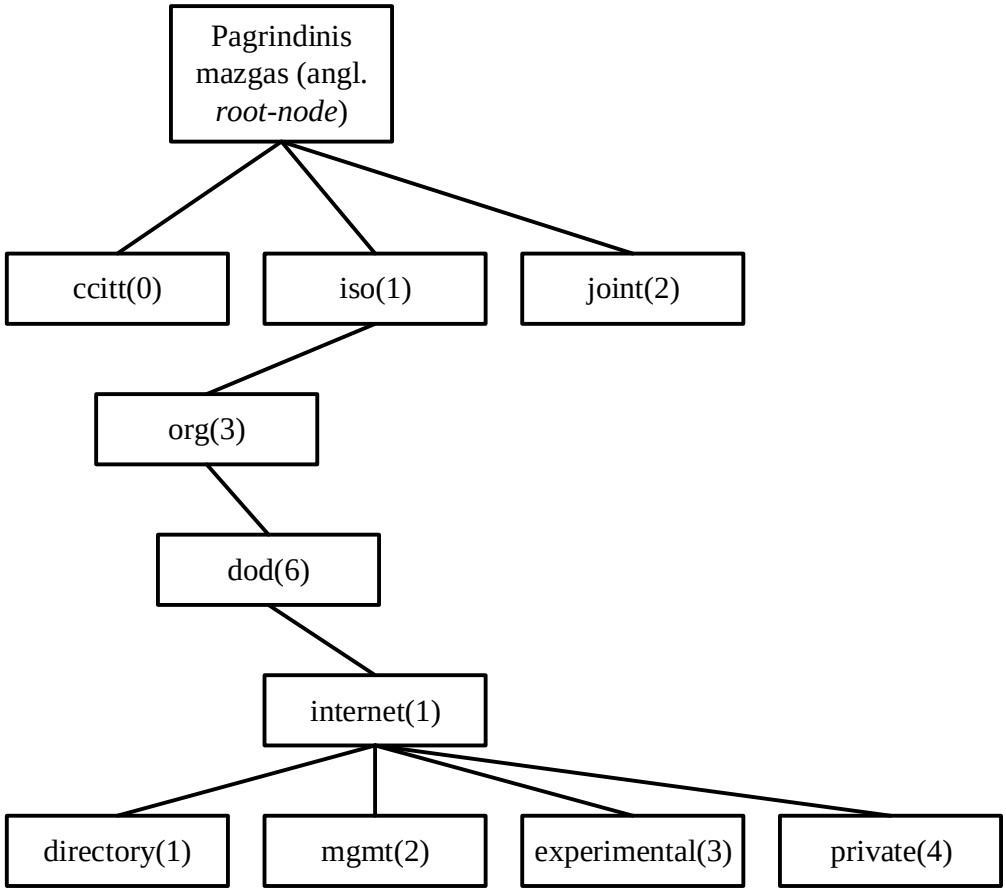
## MIB struktūra

MIB duomenų bazės yra sudarytos iš valdymo informacijos, kurios struktūrą nusako SMI (angl. *structure of management information*) standartai SMIv1 [13] ir SMIv2 [14]. Šiame poskyryje aptariama pirmoji versija.

SMI apibrėžia, kaip yra aprašomi valdomi objektai (angl. *managed objects*). Kiekvienas valdomas objektas apibrėžiamas trimis laukais [12]:

- Pavadinimo. Objekto identifikatorius (OID) yra unikalus kiekvienam objektui ir vienareikšmiškai apibrėžtas. OID būna skaitmeniniai arba tekstiniai[3] ir paprastai yra labai ilgi ir sudėtingi (pvz. 1.3.6.1.4.868.2.4.1.2.1.1.1.3.3562.3 arba iso.org.dod.internet.mgmt.mib-2.interfaces.2).
- Tipo ir sintaksės. Duomenų tipas pagal ASN.1 standartą [15] (angl. *abstract syntax notation one*). ASN.1 apibrėžia duomenų tipus ir jų atvaizdavimą, jis yra nepriklausomas nuo sistemos tipo, t.y. duomenys yra tokie patys UNIX, Windows sistemose ar tinklo įrenginių operacinėse sistemos, pvz. Cisco IOS.
- Kodavimo. Konkretus objektas (reikšmė) yra koduojamas pagal BER (angl. *basic encoding rules*) standartą [16].

Objektų identifikatorių struktūra yra medžio pavidalo. Konkretus OID yra sudarytas iš mazgų pavadinimų arba numerių, atskirtų taškais. Programiškai naudojami OID sudaryti iš dešimtainių skaitmenų, o atvaizduojami tekstiniu formatu. Pavyzdinis OID medis pateiktas 7 paveiksle.



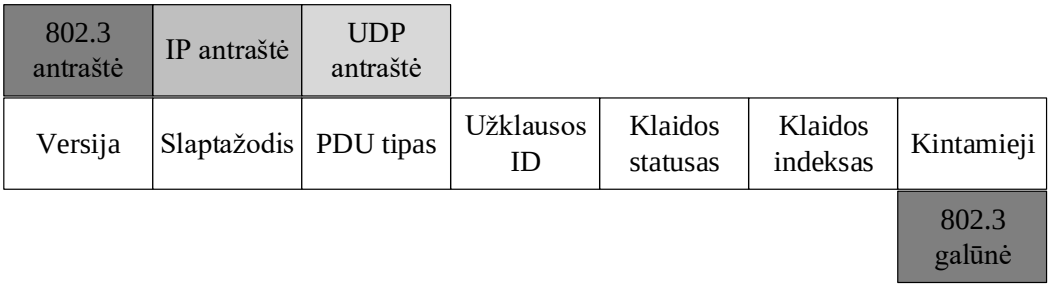

7 pav. OID medis.

[3] Skaitmeniniai ir tekstiniai OID yra ekvivalentūs, t.y. tekstinis OID yra tik atvaizdavimo būdas ir jis turi skaitmeninį atitikmenį, kuris ir yra naudojamas programinės įrangos.

## SNMP pranešimai

SNMP protokolo paketai, dar vadinami PDU (angl. *protocol data unit*), yra suformuojami pagal 8 paveiksle pateiktą struktūrą. SNMP palaiko 9 PDU tipus [12]:

- `GetRequest`. Šio tipo PDU serveris siunčia įrenginiui, o įrenginys atsako `GetResponse` paketu su serverio reikalaujamais duomenimis. OID, kurių serveris reikalauja apibrėžiami kintamųjų lauke.
- `GetNextRequest`. NMS siunčia įrenginiui, nurodydamas OID, o įrenginys atsako `GetResponse` paketu, kuriame yra sekantis objektas. Tokiu būdu, pradedant nuo pagrindinio taško, galima sužinoti visus konkrečios MIB duomenų bazės objektus. Serveris siunčia `GetNextRequest` užklausas tol, kol gauna klaidos pranešimą.
- `GetBulkRequest` (nuo SNMPv2). Optimizuota `GetNextRequest` užklausa. Šio tipo PDU naudojamas norint gauti visus norimus kintamuosius vienu kartu.
- `SetRequest`. Tokį paketą serveris siunčia įrenginiui, kad šis pakeistų tam tikrų objektų reikšmes. Siunčiami OID ir jų reikšmės. Įrenginys atsako `GetResponse` paketu su naujomis reikšmėmis.
- `GetResponse`. Atsako į `GetRequest`, `GetNextRequest`, `GetBulkRequest`, `SetRequest` ir `InformRequest` užklausas nurodant reikalaujamus kintamuosius bei klaidos statusą ir indeksą.
- `Trap` – pertrauktys. Asinchroniniai įrenginių pranešimai serveriui apie būsenos pakeitimus. Pertrauktys nereikalauja gavimo patvirtinimo. Pertrauktys turi kitokį paketo formatą.
- `Notification` (nuo SNMPv2). Pertraukties analogas pagal `GetRequest` užklausos formatą.
- `InformRequest` (nuo SNMPv2). Pertraukties analogas, kurio gavimas yra patvirtinamas. Paprastai naudojamas, kuomet yra keli NMS serveriai viename tinkle.
- `Report` (naudojamas nuo SNMPv3, aprašytas nuo SNMPv2). Šis tipas naudojamas SNMPv3 varikliams (angl. *engine*) bendrauti tarpusavyje.




8 pav. SNMP PDU struktūra.

## SNMP versijų palyginimas

Šiuo metu Interneto standartu pagal IETF klasifikaciją yra laikoma trečioji SNMP versija (kitos laikomos pasenusiomis). Senesnės versijos nenaudotinos ir dėl nepakankamo (beveik jokio) saugumo SNMPv1 ir SNMPv2 versijose. Pagrindinis antrosios versijos patobulinimas pirmosios atžvilgiu buvo papildomų PDU tipų įvedimas veikimo optimizavimui.

Svarbiausias patobulinimai trečiojoje versijoje yra susiję su saugumu. Nuo šios versijos įrenginių autentifikavimas atliekamas vartotojais grįstu saugumo modeliu (angl *user-based security model* – USM). Jame slaptažodžiai jau nebėra siunčiami atviru tekstu ir yra šifruojami MD5 arba SHA-1 maišos (angl. *hash*) algoritmais. Taip pat yra palaikomi trys saugumo lygiai:

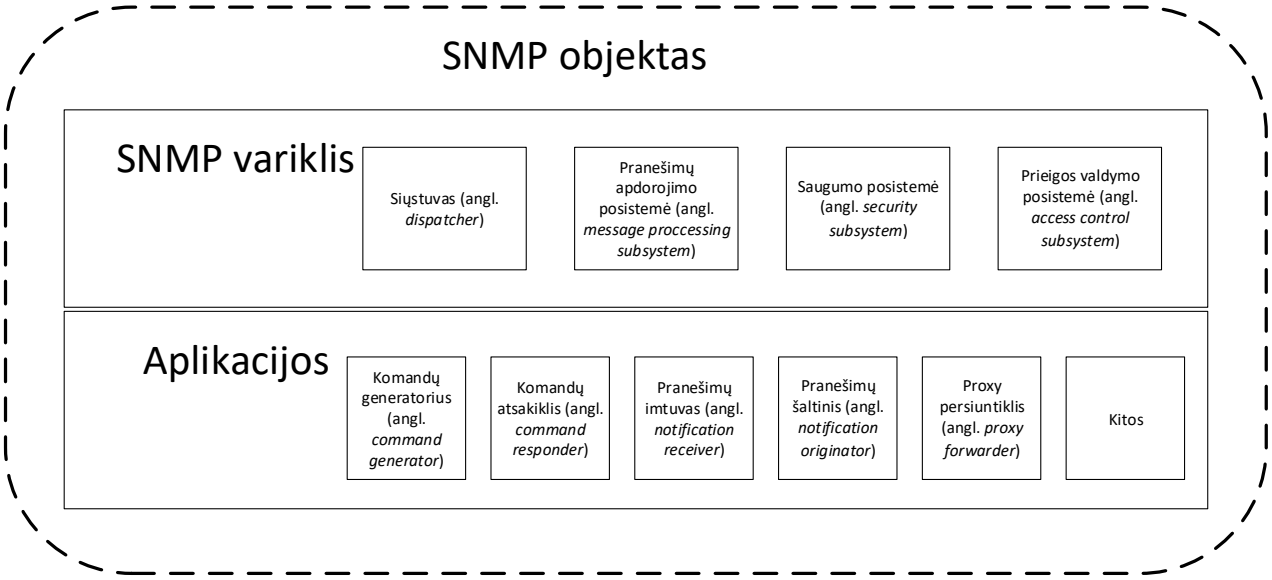
- NoAuthNoPriv. Nenaudojamas autentifikavimas ir nėra užtikrinamas privatumas (duomenų šifravimas).
- AuthNoPriv. Naudojamas autentifikavimo, bet nėra užtikrinamas privatumas.
- AuthPriv. Naudojamas ir autentifikavimas, ir duomenų šifravimas.

Taip pat svarbus pakeitimas yra sistemos architektūros apibrėžime. Nuo trečiosios versijos įrenginiai neskirstomi į valdomus įrenginius ir serverius – kiekvienas įrenginys yra SNMP objektas (angl. *SNMP entity*), kuris sudarytas iš SNMP variklio ir aplikacijų. SNMP objekto schema pateikta 9 paveiksle.

Šiame darbe yra naudojama SNMPv2c versija. Toks pasirinkimas grįstas tuo, jog topologijos nustatymui reikalingi PDU (konkrečiai – `GetBulkRequest`), kurių nėra SNMPv1 versijoje, o saugos funkcijų atsisakyta siekiant supaprastinti prototipo rengimą (jas galima nesunkiai įsidiegti vėliau).

12 lentelė. Pagrindiniai SNMP versijų skirtumai.

| | **SNMPv1** | **SNMPv2c** | **SNMPv3** |
|---|---|---|---|
| PDU kiekis | 5 | 8 | 9 |
| Autentifikavimas | Atviro teksto slaptažodis | Atviro teksto slaptažodis | Vartotojais pagrįstas su MD5 arba SHA-1 slaptažodžiai |
| Duomenų šifravimas | Nėra | Nėra | Jokio, DES, 3DES arba AES |
| Saugumo lygiai | Nėra | Nėra | NoAuthNoPriv, AuthNoPriv arba AuthPriv |
| Paketų integralumo užtikrinimas | Nėra | Nėra | Yra |
| Architektūra | Serveris - klientas | Serveris - klientas | SNMP varikliai ir aplikacijos |




9 pav. SNMP objekto struktūra.

# Topologijos informacijos surinkimas OpenFlow tinkluose

Topologijos nustatymas OpenFlow tinkluose nėra standartizuotas ir priklauso nuo naudojamo kontrolerio. Vis dėlto nuo pirmojo OpenFlow kontrolerio NOX [17] *de facto* standartu yra laikoma OpenFlow naudojama LLDP modifikacija vadinama OFDP (angl. *OpenFlow discovery protocol*) [18, 19].

OFDP naudojantys kontroleriai komutatorių įsijungimo metu išsiunčia komutatoriams savybių užklausas ir taip sužino, kiek ir kokių prievadų jie turi, bei įrašo po srauto įrašą, kuris LLDP paketus perduoda kontroleriui. Tuomet kontroleris kiekvienam komutatoriui suformuoja LLDP paketą kiekvienam fiziniam prievadui, kuriuose prievado ID TLV lauke yra įrašytas konkretus prievadas, kuriam skirtas paketas, ir juos perduoda komutatoriui, kad šis juos išsiųstų per atitinkamus prievadus. Jei prie atitinkamo prievado yra prijungtas kitas kontrolerio valdomas OF komutatorius, šis, gavęs LLDP paketą, perduoda jį kontroleriui. Kontroleris gauna paketą kartu su metaduomenimis, kuriuose yra įrašyta prievadas, į kurį atėjo paketas. Tokiu būdu, jei paketas yra gaunamas abiejose jungties pusėse, kontroleris tai traktuoja, kaip jungtį. Tokiu būdu yra surenkama topologija. Paprastai šio metodo įgyvendinimas skirtinguose kontroleriuose skiriasi tik laiko intervalais, kuriais topologija yra atnaujinama.

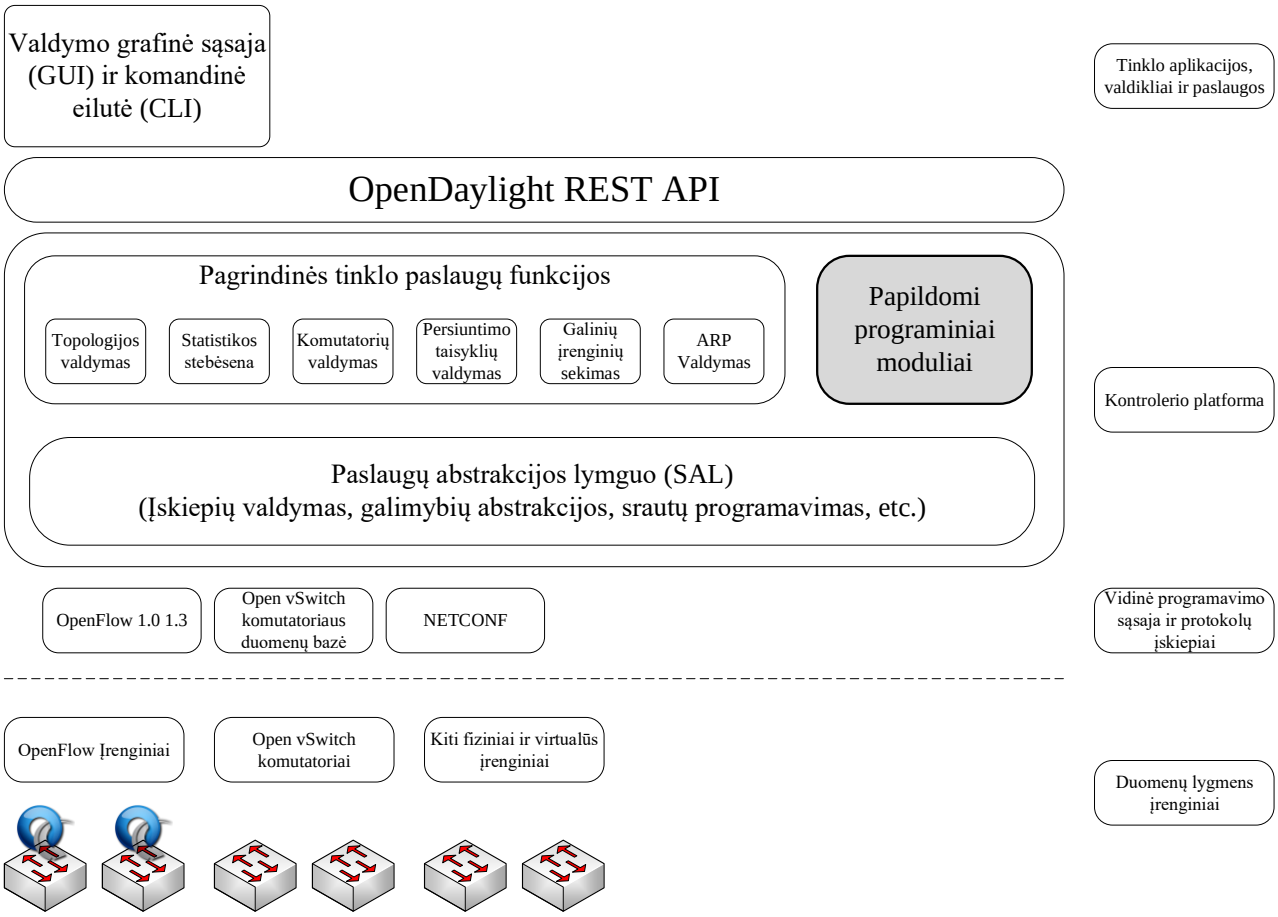
Šiuo būdu kontroleris gali aptikti OF palaikančius įrenginius, tačiau tinklo dalys, kuriose veikia tradiciniai įrenginiai, lieka neaptiktos. Taip pat, jei tarp dviejų OF komutatorių yra tradicinis tinklo įrenginys, tas sujungimas irgi nėra matomas. Šią problemą sprendžia BDDP (angl. *broadcast domain discovery protocol*) protokolas [18]. Tai yra nestandartinis sprendimas veikiantis kai kuriuose kontroleriuose (pvz. FloodLight [20], OpenDaylight [21]). BDDP yra analogiškas LLDP protokolas, tačiau jo Ethernet paketo tipas pakeistas iš 0x88CC į nestandartinį 0x8999, o taip pat naudojamas transliacinis (angl. *broadcast*) MAC adresas (FF:FF:FF:FF:FF:FF). Tokiu būdu tradiciniai tinklo komutatoriai neatmeta šių paketų (standartinis LLDP skirtas tik vieno šuoliuko L2 jungčiai), o perduoda juos į visus savo prievadus. Šiuo atveju, naudojant tą patį algoritmą, randami ir sujungimai, kur stovi tradiciniai tinklo komutatoriai. Vis dėlto patys komutatoriai nėra aptinkami ir rastoje topologijoje sujungimas atrodys kaip tiesioginė jungtis tarp dviejų OF komutatorių.

## OpenDaylight kontroleris

OpenFlow tinklo kontroleriu šiame darbe pasirinkta atviro kodo, Java programavimo kalba parašyta SDN platforma – OpenDaylight (ODL) [21, 22]. Šį pasirinkimą lėmė tai, jog šis kontroleris turi stiprų didžiausių IT bendrovių (pvz. Cisco, HP, RedHat...) palaikymą, o taip pat tai yra bene daugiausiai galimybių turinti tokio pobūdžio programinė įranga. Taipogi svarbu, jog kontrolerio dokumentacija bei kodas yra laisvai prieinami.

## Kontrolerio struktūra

OpenDaylight yra modulinė programa parašyta OSGi karkaso (angl. *framework*) pagrindu. Svarbiausias ODL komponentas yra paslaugų abstrakcijos lygmuo (angl. *service abstraction layer* – SAL), kuris įgalina tarpusavyje bendrauti programos moduliams (angl. *bundle*): įskiepiams (angl. *plugin*), paslaugoms (angl. *service*) ir t.t. Kiekvienas programos modulis laikomas nepriklausomu komponentu, kuris gali būti įjungtas, išjungtas bei panaudotas ir kitoje OSGi programinėje įrangoje. Apibendrinta kontrolerio struktūra pateikta 10 paveiksle.




10 pav. Apibendrinta OpenDaylight kontrolerio struktūra.

## Modeliu grįstas paslaugų abstrakcijos lygmuo

OpenDaylight kontroleryje yra naudojamas modeliu grįstas paslaugų abstrakcijos lygmuo (angl. *model-driven service abstraction layer* – MD-SAL) [23]. Jo paskirtis – informacijos apsikeitimas tarp programinių modulių per jų programavimo sąsajas (angl. *application programming interface* – API) bei konfigūracijos saugojimas. MD-SAL yra aprašytas YANG modeliavimo kalba. YANG modeliai aprašo duomenų struktūras bei programavimo sąsajas ir yra interpretuojami ir verčiami į Java kodą kontrolerio kompiliavimo metu. Tai atlieka specialus įrankis YANG Tools.

MD-SAL atžvilgiu, programiniai moduliai ar įskiepiai gali būti dviejų tipų:

- Vartotojai (angl. *consumers*).
- Teikėjai (angl. *providers*).

Vartotojai naudojasi teikėjų paslaugomis per MD-SAL, ir gali pasiekti duomenis duomenų saugykloje (angl. *datastore*), kuriuos įrašo teikėjai.

MD-SAL turi tris pagrindines funkcijas:

- Duomenų rašymas ir saugojimas. Visi duomenys yra saugomi medžio pavidalu. Kiekvienas medžio mazgas ar pomedis turi unikalų identifikatorių. Duomenų saugykloje yra 2 atskiri, bet susiję medžiai: operacinių duomenų medis (angl. *operational data tree*), kuriame saugomi sistemos būsenos duomenys, ir konfigūracijos duomenų medis (angl. *configurational data tree*), kuriame rašoma tinklo konfigūracija (pvz. kiekvienas OF komutatorius ir jo turimi srauto įrašai).
- Nuotolinių procedūrų iškvietimų (angl. *remote procedure call* - RPC) ir paslaugų maršrutizavimas tarp programos modulių. Vartotojų išsiunčiami pranešimai (dažniausiai į kitą programos modulį), kuriuos gauna teikėjai, ir įvykdo kažkokius veiksmus ar grąžina tam tikras vertes (pvz. srauto įrašo įrašymas į OF komutatorių).
- Pranešimų (angl. *notification*) imtuvų registravimas ir išsiuntimas. Įvykiai, kuriuos gali gauti imtuvų (angl. *listener*) registre esantys moduliai (pvz. naujo komutatoriaus įsijungimas, arba prievado būsenos pokytis).

## Programavimo sąsajos

SDN kontroleriuose paprastai yra skiriamos dviejų rūšių programavimo sąsajos: vidinės (angl. *southbound*) ir išorinės (angl. *northbound*). Jų atskyrimas paremtas objektų, su kuriais kontroleris bendrauja abstrakcijos lygiu. Vidinės sąsajos yra skirtos bendrauti su žemesnio abstrakcijos lygio objektais, o išorinės – su aukštesnio.

Vidinė programavimo sąsaja kontroleryje – tai jungtis tarp valdymo plokštumos (kontrolerio) ir duomenų plokštumos (tinklo įrenginių). OpenFlow, NETCONF ir SNMP yra tokių sąsajų pavyzdžiai. Programiškai tai įgyvendinama naudojant atitinkamo protokolo įskiepius, kurie yra kontrolerio dalis ir duomenų siuntimui ir priėmimui naudoja tam protokolui parašytą biblioteką.

Išorinė programavimo sąsaja yra jungtis tarp valdymo plokštumos (kontrolerio) ir aplikacijų plokštumos. Tai gali būti programos, automatizuojančios tinklo darbą, renkančios statistiką, orkestravimo įrankiai. OpenDaylight kontroleryje išorinė programavimo sąsaja yra įgyvendinta naudojant REST (angl. *representation state transfer*) architektūrą. Ji įgalina programas bendrauti tarpusavyje standartiniais HTTP (angl. *hypertext transfer protocol*) protokolo pranešimais (GET, POST, CONNECT, etc.). Tokia struktūra pasirinkta todėl, kad būtų galima rašyti išorines programas bet kokia programavimo kalba, o taip pat siekiant įgalinti nesudėtingą programų ir kontrolerio bendravimą tinklu. OpenDaylight kontroleryje šią sąsają naudoja komandinė eilutė bei grafinė vartotojo sąsaja. Būtent pastaroji sąsaja yra naudojama ir šiame darbe.

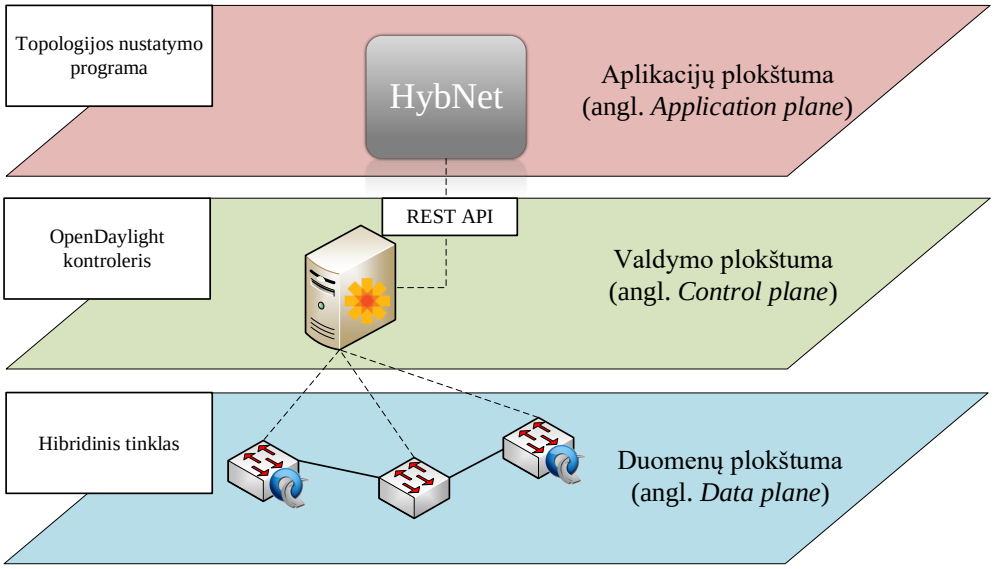
# Topologijos informacijos surinkimo hibridiniame tinkle realizacija

Svarbiausias šio darbo rezultatas – programa, kuri surenka tinklo topologijos informaciją iš OpenFlow ir tradicinių tinklo įrenginių centralizuotai. Šiame skyriuje aptariama topologijos nustatymo algoritmas, pati programa bei pateikiami ja gauti rezultatai

Darbo metu sukurtos programos HybNet kodas yra pateiktas internetinėje saugykloje (angl. *repository*) Bitbucket [24] ir yra laisvai prieinamas.

## Sistemos struktūra

Bendra sistema, kurioje veikia ir buvo testuojama programa yra pateikta 11 paveiksle. Parašyta programa veikia aplikacijų lygmenyje/plokštumoje ir su OpenDaylight kontroleriu sąveikauja per REST programavimo sąsają. OpenDaylight kontroleris taip pat atlieka ir SNMP serverio funkciją per savo SNMP įskiepį. Jo pagalba yra formuojamos SNMP užklausos. ODL yra sujungtas tiek su OpenFlow komutatoriais, kuriuos valdo, tiek su tradiciniais tinklo įrenginiais, siekiant kad būtų užtikrintas pasiekiamumas SNMP protokolo pranešimams.




11 pav. Sistemos struktūra.

## Naudojamos technologijos

Topologijos nustatymo programa yra parašyta kaip tinklo aplikacija (angl. *web app*) PHP programavimo kalba [25]. Tokio sprendimo pagrindinė priežastis yra ta, jog programa galima naudotis nepriklausomai nuo programos vartotojo (tinklo administratoriaus) sistemos, svarbu, kad joje būtų naršyklė. Aplikacija parašyta Symfony karkaso (angl. *framework*) [26] pagrindu. Tai suteikia tinklo aplikacijai standartinę MVC (angl. *model-view-controller*) architektūrą, taip pat

žymiai palengvina daug standartinių programavimo situacijų, kadangi Symfony turi platų bibliotekų, skirtų tinklo aplikacijoms, spektrą.

Topologijos nustatymo programa HybNet per REST API sąveikauja su OpenDaylight kontroleriu, kuris yra sujungtas su hibridiniu tinklu. Šiame tinkle naudojami kompanijos Cisco tradiciniai tinklo įrenginiai ir OpenFlow palaikantys kompanijos HP hibridiniai komutatoriai. Informaciją apie sujungimus įrenginiai kaupia naudodamiesi LLDP protokolu. Toks sprendimas priimtas todėl, kad naudojami OF įrenginiai nepalaiko CDP protokolo. LLDP informacija topologijos nustatymo programa surenkama SNMP protokolu, SNMP užklausas formuojant ODL kontroleriu.

Bandymams ir testavimams taip pat buvo suprogramuotos REST bei SNMP užklausų kūrimo konsolės, skirtos tiesioginiam bendravimui su kontroleriu. Jų pagalba galima greitai sugeneruoti norimo OID SNMP užklausą, arba išsiųsti bet kokią REST užklausą kontroleriui ir gauti JSON formato atsakymą.

Programos administravimas atliekamas per nustatymų paneles (skydelius) ir įrenginių panelę, kurioje importuojama pagrindinė tradicinių tinklo įrenginių informacija.

## Programos algoritmas

Supaprastintas programos algoritmo pseudokodas pateiktas 1 algoritme. Jį su daro pagrindinės dvi dalys:

- OpenFlow tinklo topologijos, kuri yra gaunama tiesiai iš OpenDaylight kontrolerio per REST užklausą ir transformuojama į programai tinkamą formatą, t.y. jungtis ir mazgus, radimas.
- Tradicinių įrenginių topologijos radimas. Kiekvienam tradiciniam tinklo įrenginiui siunčiama SNMP užklausa, reikalaujanti LLDP informacijos. Iš gautų *chassisId* išskaičiuojamos jungtys tarp mazgų.

Darbo metu buvo identifikuoti šie OID, būtini ir pakankami pilnos tinklo topologijos informacijos surinkimui:

- OID `1.0.8802.1.1.2.1.4.1.1.7`. LDDP objekto identifikatorius skirtas prie įrenginio prijungtų sistemų prievadų numeriams nustatyti.
- OID `1.0.8802.1.1.2.1.4.1.1.5`. Šis objekto identifikatorius skirtas prie įrenginio prijungtų LLDP protokolo sistemų identifikatoriams nustatyti (*chassisId*).
- OID `1.0.8802.1.1.2.1.3.7.1.3`. Šiame objekte saugomi įrenginio prievado numeriai.
- OID `1.0.8802.1.1.2.1.3.2`. Lokalių LLDP naudojančių prievadų MAC adresai.
- OID `1.0.8802.1.1.2.1.3.3`. Lokalios sistemos pavadinimas.

Visi naudojami OID yra iš LLDP MIB duomenų bazės [27].

Apskaičiuotą topologiją programa gali atvaizduoti grafiškai. Gauti rezultatai pateikti 2.4 skyriuje.

1 algoritmas. Hibridinio tinklo topologijos nustatymo algoritmas

```
1   topologija = nauja Topologija();
2
3   # Skaičiuojama OpenFlow topologija
4   openFlowInformacija[] = topologyUžklausaREST();
5   KIEKVIENAM openFlowInformacija[„mazgų informacija“] kaip mazgoInformacija
6     topologija->pridėtiMazgą(gautiOpenFlowMazgoInformaciją(mazgoInformacija));
7   PABAIGA
8
9   KIEKVIENAM openFlowInformacija[„jungčių informacija“] kaip jungtiesInformacija
10    # Pridedamas prievadas ir jungtis į kitą prievadą
11    JEI jungtisNeegzistuoja(jungtiesInformacija)
12      topologija->pridėtPrievadą(gautiOpenFlowJungtiesInformaciją(jungtiesInformacija));
13      topologija->pridėtiJungtį(gautiOpenFlowJungtiesInformaciją(jungtiesInformacija));
14    PABAIGA
15  PABAIGA
16
17  # Skaičiuojama tradicinių įrenginių topologija
18  tradiciniųĮrenginiųInformacija[] = gautiTradicininiųĮrenginiųSąrašą();
19  KIEKVIENAM tradiciniųĮrenginiųInformacija kaip įrenginioInformacija
20    mazgoInformacija = siųstiSNMPUžklausą(įrenginioInformacija[„IP adresas“]);
21    topologija->pridėtiMazgą(mazgoInformacija[„mazgo ID“]);
22    mazgoJungčiųInformacija = mazgoInformacija[„jungčių informacija“];
23
24    KIEKVIENAM mazgoJungčiųInformacija kaip jungtiesInformacija
25      topologija->pridėtPrievadą(jungtiesInformacija);
26      JEI nutolęsPrievadasEgzistuoja(jungtiesInformacija[„nutolęs prievadas“])
27        # Tikrinimas, ar yra OpenFlow jungčių, kurių viduryje turėtų stovėti tradicinis įrenginys
28        JEI nutolęsPrievadasTuriJungtį(jungtiesInformacija[„nutolęs prievadas“])
29          topologija-> pridėtiJungtįPerrašantEgzistuojančią(jungtiesInformacija);
30        KITU ATVEJU
31          topologija-> pridėtiJungtį(jungtiesInformacija);
32        PABAIGA
33      PABAIGA
34    PABAIGA
35  PABAIGA
36
37  GRĄŽINTI topologija;
```

## Pagrindinės savybės ir funkcionalumas

Nors pagrindinis programos tikslas yra hibridinio tinklo topologijos nustatymas, ji atlieka ir keletą papildomų su tuo susijusių užduočių. Pilnas galimybių sąrašas yra pateiktas 13 lentelėje.

13 lentelė. Programos galimybių sąrašas.

| Funkcionalumas | Paaiškinimas |
|---|---|
| Topologijos informacijos nustatymas | Tradicinio, OpenFlow ir hibridinio tinklo topologijos informacijos nustatymas pagal 1 algoritmą. |
| Dinaminis topologijos atvaizdavimas | Dinaminis topologijos (tinklo įrenginių, prievadų ir jungčių) atvaizdavimas neperkraunant puslapio. |
| Statinis topologijos atvaizdavimas | Testavimui skirtas topologijos atvaizdavimas perkraunant puslapį. |
| Topologijos nustatymo greičio matavimas | Topologijos informacijos nustatymo pakartojimas pasirinktą skaičių kartų ir vidurkio bei standartinio vidurkio nuokrypio apskaičiavimas. |
| Nustatymų panelė | Nustatymų susijusių su kontroleriu ir įrenginių SNMP nustatymais redagavimas ir saugojimas duomenų bazėje. |
| Tradicinių tinklo įrenginių sąrašas | Tradicinių tinklo įrenginių sąrašo redagavimas ir saugojimas duomenų bazėje. |
| REST užklausų formavimo panelė | HTTP užklausų OpenDaylight kontroleriui formavimas ir siuntimas. |
| SNMP užklausų formavimo panelė | SNMP užklausų įrenginiams formavimas ir siuntimas. |

## Gautos topologijos

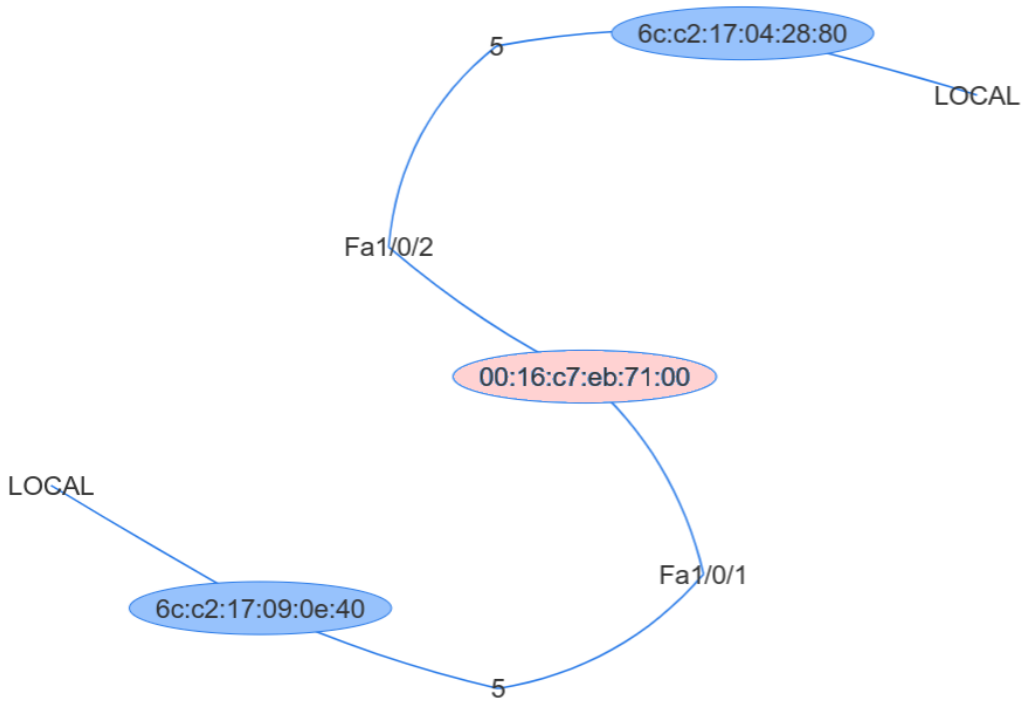
Parašyta programa buvo panaudota nustatant 5 fizines hibridinio tinklo topologijas ir vieną fizinės ir virtualios topologijos kombinaciją. Kaip OpenFlow įrenginiai buvo naudoti HP-2920 komutatoriai (konfigūracijos failai pateikti 1 priede), o kaip tradicinio tinklo įrenginiai – Cisco Catalyst 3750 komutatoriai (konfigūracijos failai pateikti 2 priede). Apskaičiuota šių topologijų nustatymo trukmė esant skirtingoms topologijoms. Rezultatai pateikti 14 lentelėje. Priklausomai nuo topologijos, jos nustatymo trukmė svyruoja nuo 0,9 iki 9,8 s. Įvertinant tinklo elementų padidėjimą iki vidutinės organizacijos dydžio tinklo, trukmė būtų maždaug dešimt kartų didesnė. Tai yra priimtina trukmė realiam darbiniam tinklui, įvertinant, kad pilnas topologijos nustatymas reikalingas tik tinklo inicijavimo metu, o vėliau sekami tik topologijos pokyčiai.

Topologijų atvaizdai, gauti parašyta programa, pateikti 12 – 17 paveiksluose (mėlynai pavaizduoti OF įrenginiai, o raudonai – tradiciniai). Aktyvus įrenginio prievadas paveiksluose vizualizuojamas iš įrenginio išeinančia jungtimi ir prievado įvardijimu (pvz. „6“ 13 paveiksle). Toliau jungtis gali tęstis į kitą prievadą (pvz. Gi2/0/2), o tuomet į kitą įrenginį. Jei prievadas

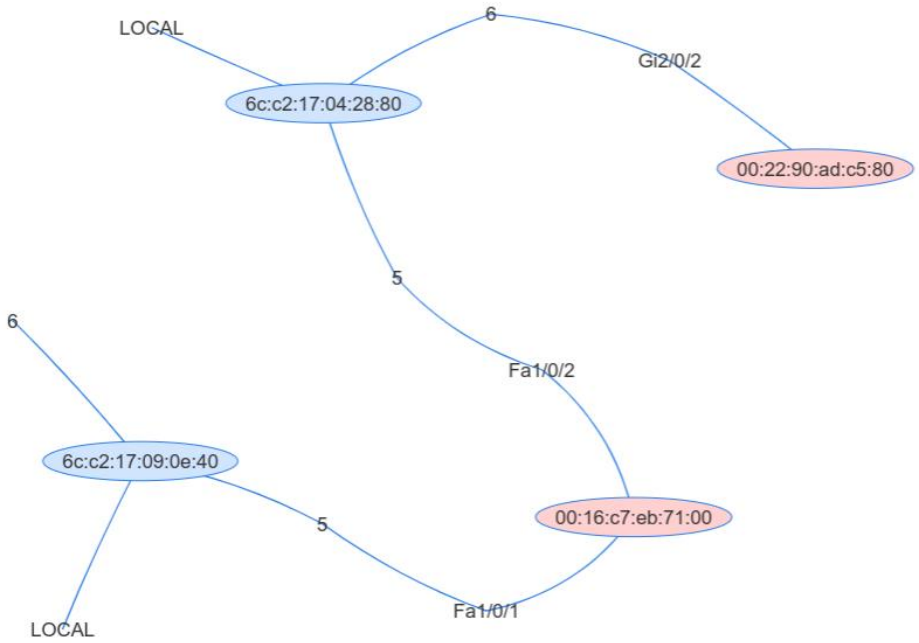
aktyvus, bet prie jo neprijungtas kitas įrenginys, jis bus rodomas kaip nutraukta jungtis (pvz LOCAL prievadas OF komutatoriuose). Kaip įrenginio indentifikatorius rodomas LLDP *chassisId*.

14 lentelė. Topologijos nustatymo trukmė. Rezultatai gauti atlikus 1000 bandymų su kiekviena topologija.

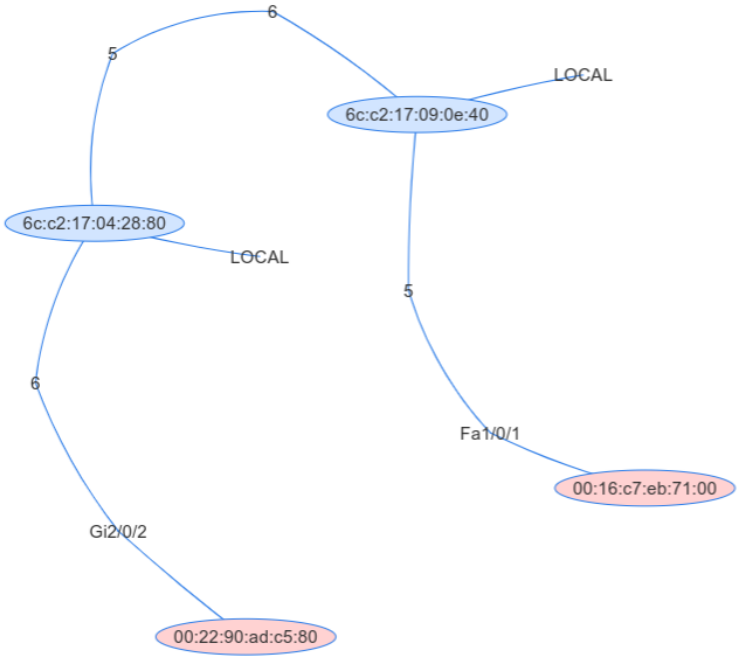
| **Topologija** | **Vidutinė nustatymo trukmė $\bar{t}$, ms** | **Vidurkio standartinis nuokrypis $s$, ms** | **Paaiškinimas** |
|---|---|---|---|
| 1 topologija | 919,39 | 5,15 | Du OpenFlow komutatoriai su tradiciniu komutatoriumi tarp jų (12 paveikslas). |
| 2 topologija | 9820,25 | 33,44 | Iš eilės einantys OF, tradicinis, OF ir vėl tradiciniai komutatoriai (13 paveikslas). |
| 3 topologija | 9811,39 | 31,7 | Du sujungti OF komutatoriai ir prie jų po vieną prijungti du tradiciniai komutatoriui (14 paveikslas). |
| 4 topologija | 6840,09 | 34,99 | Du sujungti tradiciniai komutatoriai, prie vieno kurių prijungti du OF komutatoriai (15 paveikslas). |
| 5 topologija | 3278,56 | 23,99 | Du sujungti tradiciniai komutatoriai ir prie jų po vieną prijungti du OF komutatoriui (16 paveikslas). |
| Virtualios ir 1 topologijos kombinacija | 1000,55 | 4,26 | Fizinė topologija 1 ir kartu prie kontrolerio prijungti virtualūs OF komutatoriai (17 paveikslas). Virtualūs komutatoriai gauti Mininet įrankiu [28]. |



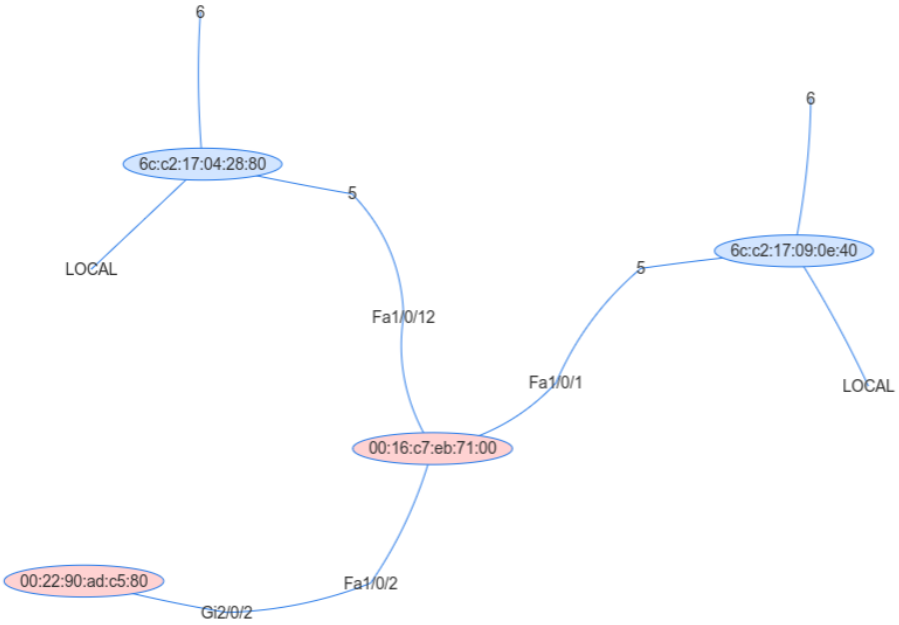

12 pav. 1 topologijos atvaizdas.



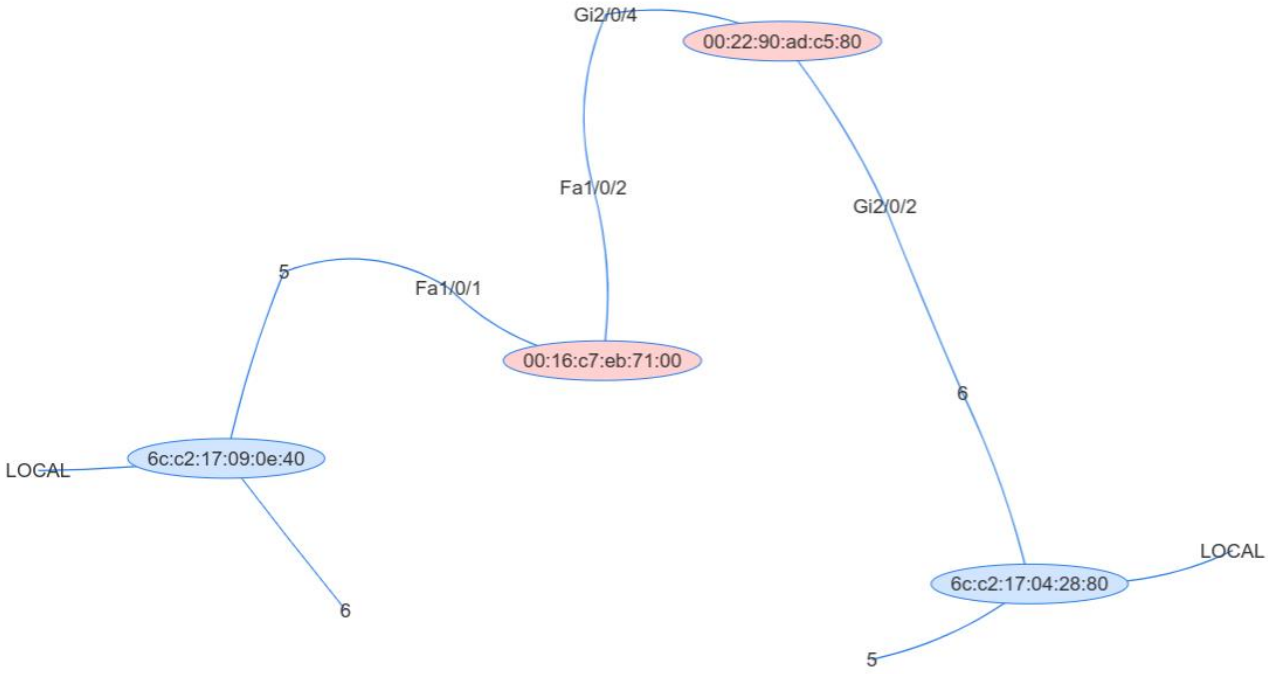

13 pav. 2 topologijos atvaizdas.



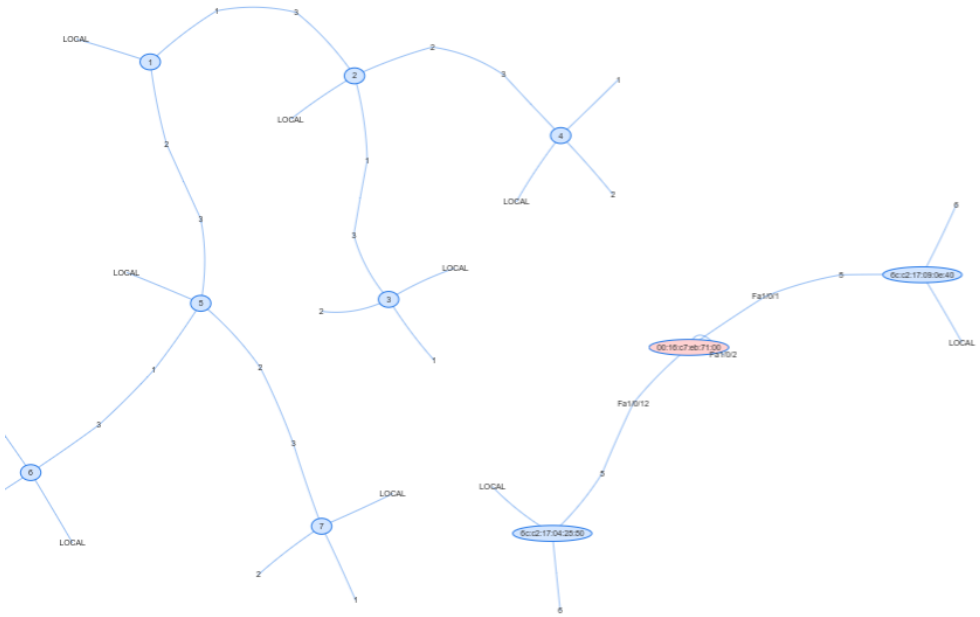

14 pav. 3 topologijos atvaizdas.




15 pav. 4 topologijos atvaizdas.




16 pav. 5 topologijos atvaizdas.



17 pav. Virtualios ir 1 topologijos atvaizdas.

# Pagrindiniai rezultatai ir išvados

## Rezultatai

1. Darbo metu buvo išnagrinėti tinklo topologijos nustatymo būdai ir protokolai tradiciniuose ir OpenFlow tinkluose, jų galimybės, bei apribojimai.
2. Išanalizuota SNMP protokolo struktūra ir sintaksė.
3. Nustatyti topologijos informacijai gauti būtini ir pakankami objektų identifikatoriai (OID).
4. Parašyta programa, nustatanti hibridinio tinklo topologiją, atvaizduojanti ją grafiškai bei apskaičiuojanti nustatymo trukmę. Programos kodas yra viešai prieinamas internetinėje saugykloje [24].

## Išvados

1. Programos pagalba tiksliai nustatoma hibridinio tinklo topologija.
2. Nustatyta, kad priklausomai nuo topologijos, jos nustatymo trukmė svyruoja nuo 0,9 iki 9,8 s. Tai yra priimtina trukmė tinklo veiklai.

# Literatūra


[1] H. Lu, N. Arora, H. Zhang, C. Lumezanu, J. Rhee, and G. Jiang, Hybnet: Network manager for a hybrid network infrastructure, in Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference, ser. Middleware Industry ’13. New York, NY, USA: ACM, pp. 1 – 6., (2013)

[2] D. K. Hong, Y. Ma, S. Banerjee, and Z. M. Mao, Incremental Deployment of SDN in Hybrid Enterprise and ISP Networks, in Proceedings of the Symposium on SDN Research – SOSR 16, Santa Clara, CA, USA, 7 p., (2016).

[3] S. Livingstone, L. A. Lievrouw, and L. Lievrouw, Handbook of New Media: Student Edition. Sage Publications, 253 p., (2006)

[4] S. Pandey, M.-J. Choi, Y. J. Won, and J. W.-K. Hong, SNMP-based enterprise IP network topology discovery, in International Journal of Network Management, vol. 21, no. 3, pp. 169 – 184, (2010).

[5] Open Networking Foundation, OpenFlow Switch Specification Version 1.5.1, 285 p., (2015)

[6] W. Stallings, Software-Defined Networks and OpenFlow, The Internet Protocol Journal 16 (1), pp. 2 – 14, (2013)

[7] S. Azodolmolky, Software Defined Networking with OpenFlow, PACKT publishing, Birmingham, 138 p., (2013)

[8] X. Ma and G. Xia, Autonomous System Network Topology Discovery Algorithm Based On OSPF Protocol, in Proceedings of the 3rd International Conference on Material, Mechanical and Manufacturing Engineering8 p., (2015)

[9] S. Avallone, S. D’Antonio, M. Esposito, A. Pescap`e, S. P. Romano, A Topology Discovery Module based on a Hybrid Methodology, Inter-domain Performance and Simulation Workshop, Budapest (Hungary), 8 p., (2004)

[10] S. Pandey, M. Choi, S. Lee, James W. Hong, IP Network Topology Discovery Using SNMP, Proceedings of the 23rd International Conference on Information Networking, ICOIN’09, IEEE Press, pp. 33 – 37, (2009)

[11] IEEE, Station and Media Access Control Connectivity Discovery, 190 p., (2009)

[12] D. R. Mauro, K. J. Schmidt, Essential SNMP 2nd Edition, O‘Reilly Media, 444 p., (2005)

[13] IETF, RFC 1155, Structure and Identification of Management Information for TCP/IP-based Internets, 22 p., (1990)

[14] IETF, RFC 2578, Structure of Management Information Version 2 (SMIv2), 43 p., (1999)

[15] ITU-T, X.680, Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation, 146p., (2002)

[16] ITU-T, X.690, Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), 36 p., (2002)

[17] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. Mckeown, and S. Shenker, Nox, ACM SIGCOMM Computer Communication Review, vol. 38, no. 3, pp. 105 – 110 (2008)

[18] L. Ochoa-Aday, C. Cervello-Pastor, A. Fernandez-Fernandez, Current Trends of Topology Discovery in OpenFlow-based Software Defined Networks, IEEE, 6 p., (2015)

[19] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, Efficient topology discovery in software defined networks, 2014 8th International Conference on Signal Processing and Communication Systems (ICSPCS), 8 p., 2014.

[20] Project Floodlight, http://www.projectfloodlight.org/, (žiūrėta 2017-05-25)

[21] The OpenDaylight Platform, http://opendaylight.org/, (žiūrėta 2017-05-25)

[22] The Linux Foundation, OpenDaylight documentation, https://wiki.opendaylight.org/view/OpenDaylight_Controller, (žiūrėta 2017-05-25)

[23] J. Medved, A. Tkacik, R. Varga, K. Grey, OpenDaylight: Towards a Model-Driven SDN Controller Architecture, 2014 IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Sydney, 6 p., (2014)

[24] https://bitbucket.org/mindaugasgiedraitis/hybnet/

[25] PHP: Hypertext Preprocessor, http://php.net/, (žiūrėta 2017-05-25)

[26] Symfony, High Performance PHP Framework for Web Development, https://symfony.com/, (žiūrėta 2017-05-23)

[27] IEEE, Management Information Base module for LLDP configuration, statistics, local system data and remote systems data components, 32 p., (2005)

[28] Mininet: An Instant Virtual Network, http://mininet.org/, (žiūrėta 2017-05-25)

# Santrauka

## INFORMACIJOS APIE TOPOLOGIJĄ HIBRIDINIUOSE TINKLUOSE SURINKIMAS PRITAIKANT SNMP

Mindaugas Giedraitis

**Šio darbo tikslas** – pritaikyti SNMP protokolą ir SDN kontrolerio galimybes centralizuotam hibridinio tinklo topologijos informacijos surinkimui, siekiant integruoti tradicinio tinklo įrenginius į SDN struktūrą ir užtikrinti vieningą centralizuotą tinklo valdymą (o taip pat suteikiant tinklo administratoriams aiškų tinklo vaizdą). Tikslui pasiekti buvo iškelti šie uždaviniai:

- išnagrinėti tinklo topologijos nustatymo būdus ir protokolus tradiciniuose ir OpenFlow tinkluose;
- išanalizuoti SNMP protokolo struktūrą ir sintaksę;
- nustatyti topologijos informacijai gauti reikalingus objektų identifikatorius (OID);
- parašyti ir išbandyti programą, galinčią centralizuotai surinkti hibridinio tinklo topologijos informaciją ir atvaizduoti topologiją grafiškai.

Darbe pateikiama SDN ir OpenFlow technologijų, topologijos informacijos surinkimo būdų tradiciniuose ir OF tinkluose apžvalga, bei išnagrinėti SNMP ir LLDP protokolai.

**Darbo rezultatas** – sukurtas ir programiškai pritaikytas algoritmas hibridinio tinklo topologijai nustatyti. Parašyta programa nustato ir grafiškai atvaizduoja hibridinio tinklo topologiją.

Nustatyta, kad priklausomai nuo topologijos, jos nustatymo trukmė svyruoja nuo 1099 iki 9923 ms. Tai yra priimtina trukmė tinklo veiklai.

# Summary


## TOPOLOGY INFORMATION COLLECTION USING SNMP IN HYBRID NETWORKS

Mindaugas Giedraitis

**The objective** of this paper is to adapt SNMP protocol and SDN controller for centralized hybrid network topology information gathering this way aiming to integrate traditional networking equipment into SDN network structure and ensure unified and centralized network management (as well as to give network administrators a clear network view). To reach this objective these tasks were set:

- analyze network topology discovery methods and protocols in both traditional and OpenFlow networks;
- analyze structure and syntax of SNMP protocol;
- determine object ids (OID) that are needed to gather network topology information;
- write and test a program, that can centrally gather hybrid network topology information and display that topology graphically.

This paper includes reviews of SDN and OpenFlow technologies, information gathering methods in OF and traditional networks, and analysis of SNMP and LLDP protocols.

**The main result** of this paper is a new algorithm for hybrid network topology information gathering and a program that implements the algorithm and graphically displays network topology.

Topology discovery times, that where measured for different topologies, varies from 1099 to 9923 ms. This is an acceptable time for network operation.

# Priedas nr. 1

# HP-2920 konfigūracijos failai

## HP-2920-3

```
hostname "HP-2920-3"
module 1 type j9726a
ip route 172.16.83.0 255.255.255.128 192.168.226.150
interface 1
   disable
   exit
interface 2
   disable
   exit
interface 6
   speed-duplex 100-full
   exit
interface A1
   disable
   exit
interface A2
   disable
   exit
snmp-server community "public" unrestricted
openflow
   controller-id 1 ip 10.226.120.1 controller-interface vlan 120
   controller-id 2 ip 172.16.83.58 controller-interface vlan 226
   instance "my1"
      member vlan 100
      controller-id 1
      enable
      exit
   instance "hybnet1"
      member vlan 2260
      controller-id 2
      enable
      exit
   enable
   exit
oobm
   ip address dhcp-bootp
   exit
vlan 1
   name "DEFAULT_VLAN"
   no untagged 5-13,21-24
   untagged 1-4,14-20,A1-A2,B1-B2
   no ip address
```

```
   exit
vlan 5
   name "VLAN5"
   no ip address
   exit
vlan 6
   name "VLAN6"
   no ip address
   exit
vlan 7
   name "VLAN7"
   untagged 8
   no ip address
   exit
vlan 8
   name "VLAN8"
   no ip address
   exit
vlan 9
   name "VLAN9"
   untagged 9-10
   no ip address
   exit
vlan 10
   name "VLAN10"
   no ip address
   exit
vlan 11
   name "VLAN11"
   untagged 11-12
   no ip address
   exit
vlan 12
   name "VLAN12"
   no ip address
   exit
vlan 100
   name "VLAN100"
   untagged 21-24
   no ip address
   exit
vlan 120
   name "VLAN120"
   untagged 13
   ip address 10.226.120.103 255.255.255.0
   exit
vlan 226
   name "VLAN226"
```

```
   untagged 7
   ip address 192.168.226.152 255.255.255.0
   exit
vlan 999
   name "VLAN999"
   no ip address
   exit
vlan 2260
   name "VLAN2260"
   untagged 5-6
   no ip address
   exit
```

## HP-2920-4

```
hostname "HP-2920-4"
module 1 type j9726a
ip default-gateway 192.168.226.254
ip route 172.16.83.0 255.255.255.128 192.168.226.150
ip routing
interface 2
   disable
   exit
interface 6
   speed-duplex 100-full
   exit
interface A1
   disable
   exit
interface A2
   disable
   exit
snmp-server community "public" unrestricted
openflow
   controller-id 2 ip 172.16.83.58 controller-interface vlan 226
   instance "hybnet1"
      member vlan 2260
      controller-id 2
      enable
      exit
   enable
   exit
oobm
   ip address dhcp-bootp
   exit
vlan 1
   name "DEFAULT_VLAN"
   no untagged 5-7,13,23-24
   untagged 1-4,8-12,14-22,A1-A2,B1-B2
```

```
   no ip address
   exit
vlan 14
   name "VLAN14"
   untagged 23
   ip address 192.168.14.254 255.255.255.0
   exit
vlan 24
   name "VLAN24"
   untagged 24
   ip address 192.168.24.254 255.255.255.0
   exit
vlan 120
   name "VLAN120"
   untagged 13
   ip address 10.226.120.104 255.255.255.0
   exit
vlan 226
   name "VLAN226"
   untagged 7
   ip address 192.168.226.153 255.255.255.0
   exit
vlan 998
   name "VLAN998"
   no ip address
   exit
vlan 2260
   name "VLAN2260"
   untagged 5-6
   no ip address
   exit
```

# Priedas nr. 2

# Cisco Catalyst 3750 konfigūracijos failai

## Legacy1

```
version 12.2
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Legacy1
!
boot-start-marker
boot-end-marker
!
!
!
!
no aaa new-model
switch 1 provision ws-c3750-48ts
system mtu routing 1500
!
!
!
!
crypto pki trustpoint TP-self-signed-3354095872
 enrollment selfsigned
 subject-name cn=IOS-Self-Signed-Certificate-3354095872
 revocation-check none
 rsakeypair TP-self-signed-3354095872
!
!
crypto pki certificate chain TP-self-signed-3354095872
 certificate self-signed 01
  30820240 308201A9 A0030201 02020101 300D0609 2A864886 F70D0101 04050030
  31312F30 2D060355 04031326 494F532D 53656C66 2D536967 6E65642D 43657274
  69666963 6174652D 33333534 30393538 3732301E 170D3933 30333031 30303031
  33385A17 0D323030 31303130 30303030 305A3031 312F302D 06035504 03132649
  4F532D53 656C662D 5369676E 65642D43 65727469 66696361 74652D33 33353430
  39353837 3230819F 300D0609 2A864886 F70D0101 01050003 818D0030 81890281
  8100BDAE 3007F471 E0786E7D 1EC3A8C3 15A8513F 1E03061D F6033AE2 E64154AC
  26789B03 9153ED61 C3E9037C A124A523 1CE8234C E4D7E92C BFF2BEF8 1934DCF8
  F298ADF8 29CCE2BE BE3E02FE E42D3E7D C49BE2A4 8148C48D 4D9A56A3 3A4A77EF
```

```
      2F9EACE2 4FE43BFA 30CE5FC3 7AC4E4A8 C44E440F B0BC40D6 C09B7830 4219BA9E
      1D8D0203 010001A3 68306630 0F060355 1D130101 FF040530 030101FF 30130603
      551D1104 0C300A82 084C6567 61637931 2E301F06 03551D23 04183016 80143F69
      1BDD0941 3632A663 63747E13 28050581 44AB301D 0603551D 0E041604 143F691B
      DD094136 32A66363 747E1328 05058144 AB300D06 092A8648 86F70D01 01040500
      03818100 7DA8A109 E9DAF1B8 0C24CE04 D64C9ABE 6F4B41E0 A1B213EB CF7AA47D
      BAADDAFA 9E570EDC B4C339E5 4B4E85ED C2E4BF30 3F67D8C8 299DC100 E7DB1365
      E82DD1F3 9FE59DDA 07757F4A 90B0D3B5 719F6E7F 721028B6 607D5F7F 2769827C
      22B4E9E2 65CC2890 A59D6C6B 663CE968 6DBB05AC 153EF7A8 C5EDFE44 FA3E50CD
6BCFE2A8
      quit
    !
    !
    !
    spanning-tree mode pvst
    spanning-tree extend system-id
    !
    vlan internal allocation policy ascending
    lldp run
    !
    !
    !
    !
    interface FastEthernet1/0/1
    !
    interface FastEthernet1/0/2
     duplex full
    !
    interface FastEthernet1/0/3
    !
    interface FastEthernet1/0/4
    !
    interface FastEthernet1/0/5
     switchport access vlan 226
     switchport mode access
     load-interval 30
    !
    interface FastEthernet1/0/6
    !
    interface FastEthernet1/0/7
    !
    interface FastEthernet1/0/8
    !
    interface FastEthernet1/0/9
    !
    interface FastEthernet1/0/10
    !
    interface FastEthernet1/0/11
```

```
!
interface FastEthernet1/0/12
!
interface FastEthernet1/0/13
!
interface FastEthernet1/0/14
!
interface FastEthernet1/0/15
!
interface FastEthernet1/0/16
!
interface FastEthernet1/0/17
!
interface FastEthernet1/0/18
!
interface FastEthernet1/0/19
!
interface FastEthernet1/0/20
!
interface FastEthernet1/0/21
!
interface FastEthernet1/0/22
!
interface FastEthernet1/0/23
!
interface FastEthernet1/0/24
!
interface FastEthernet1/0/25
!
interface FastEthernet1/0/26
!
interface FastEthernet1/0/27
!
interface FastEthernet1/0/28
!
interface FastEthernet1/0/29
!
interface FastEthernet1/0/30
!
interface FastEthernet1/0/31
!
interface FastEthernet1/0/32
!
interface FastEthernet1/0/33
!
interface FastEthernet1/0/34
!
interface FastEthernet1/0/35
```

```
!
interface FastEthernet1/0/36
!
interface FastEthernet1/0/37
!
interface FastEthernet1/0/38
!
interface FastEthernet1/0/39
!
interface FastEthernet1/0/40
!
interface FastEthernet1/0/41
!
interface FastEthernet1/0/42
!
interface FastEthernet1/0/43
!
interface FastEthernet1/0/44
!
interface FastEthernet1/0/45
!
interface FastEthernet1/0/46
!
interface FastEthernet1/0/47
!
interface FastEthernet1/0/48
!
interface GigabitEthernet1/0/1
!
interface GigabitEthernet1/0/2
!
interface GigabitEthernet1/0/3
!
interface GigabitEthernet1/0/4
!
interface Vlan1
 no ip address
 no ip route-cache
 no ip mroute-cache
!
interface Vlan226
 ip address 192.168.226.151 255.255.255.0
 no ip route-cache
 no ip mroute-cache
!
ip default-gateway 192.168.226.150
ip classless
ip http server
```

```
ip http secure-server
!
!
ip sla enable reaction-alerts
!
tftp-server flash:c3750-ipservicesk9-mz.122-55.SE3.bin
snmp-server community cisco RO
snmp-server enable traps config
snmp-server host 172.16.83.58 cisco
!
!
line con 0
line vty 0 4
 exec-timeout 0 0
 privilege level 15
 password 7 110A1016141D
 logging synchronous
 login
 transport preferred telnet
 transport input telnet
line vty 5 15
 no login
!
end
```

## Legacy3

```
version 12.2
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Legacy3
!
boot-start-marker
boot-end-marker
!
enable password cisco
!
!
!
no aaa new-model
switch 2 provision ws-c3750g-12s
system mtu routing 1500
!
!
!
!
```

```
crypto pki trustpoint TP-self-signed-2427307392
 enrollment selfsigned
 subject-name cn=IOS-Self-Signed-Certificate-2427307392
 revocation-check none
 rsakeypair TP-self-signed-2427307392
!
!
crypto pki certificate chain TP-self-signed-2427307392
 certificate self-signed 01
  30820240 308201A9 A0030201 02020101 300D0609 2A864886 F70D0101 04050030
  31312F30 2D060355 04031326 494F532D 53656C66 2D536967 6E65642D 43657274
  69666963 6174652D 32343237 33303733 3932301E 170D3933 30383232 32333130
  31375A17 0D323030 31303130 30303030 305A3031 312F302D 06035504 03132649
  4F532D53 656C662D 5369676E 65642D43 65727469 66696361 74652D32 34323733
  30373339 3230819F 300D0609 2A864886 F70D0101 01050003 818D0030 81890281
  8100E1FD F3C6F940 42CAE2A9 6FBD04E7 AB5EB8B5 3BA91E80 5CDA5069 7BF51379
  0AA7182D 74A82E88 0ACD2C93 C8D2C7CA B038D3F3 154197CD 67CAE6BC 57325680
  B63071F2 8089128C DDE10E3A 16FF5194 D0551B94 120C8B85 91A4AD2F CE234571
  9647DF8D 30001D04 B08A0ADA 9420190A D005118D 117C69C4 60B2DE60 9BEFFC81
  E1B10203 010001A3 68306630 0F060355 1D130101 FF040530 030101FF 30130603
  551D1104 0C300A82 084C6567 61637933 2E301F06 03551D23 04183016 8014B937
  D38BB4B6 8E7D329A 48048705 CA558F78 224D301D 0603551D 0E041604 14B937D3
  8BB4B68E 7D329A48 048705CA 558F7822 4D300D06 092A8648 86F70D01 01040500
  03818100 42B61BFC 9700DE9B C9B1FEAA 415BA5EF 2D31F9C6 5CD8637F 0725035F
  4C8AE93E ADD4A0CB B0083548 DAA62C71 D365DEC2 AB81D127 65B81204 85043A1E
  B8551A9B 8EEE1602 6E2AA8A5 0FB94B4D 59CD2DC8 CD428DA7 4FB9EF60 78EF8035
  08732F42 EB7E7484 B08D82B4 1C566D86 5CB3BCAA CEEB627A C242FBD6 98BF91CE
0D0789B1
  quit
!
!
!
spanning-tree mode pvst
spanning-tree extend system-id
!
vlan internal allocation policy ascending
lldp run
!
!
!
!
interface GigabitEthernet2/0/1
!
interface GigabitEthernet2/0/2
!
interface GigabitEthernet2/0/3
 switchport access vlan 226
 switchport mode access
```

```
 load-interval 30
!
interface GigabitEthernet2/0/4
!
interface GigabitEthernet2/0/5
!
interface GigabitEthernet2/0/6
!
interface GigabitEthernet2/0/7
!
interface GigabitEthernet2/0/8
!
interface GigabitEthernet2/0/9
!
interface GigabitEthernet2/0/10
!
interface GigabitEthernet2/0/11
!
interface GigabitEthernet2/0/12
!
interface Vlan1
 no ip address
!
interface Vlan226
 ip address 192.168.226.200 255.255.255.0
!
ip default-gateway 192.168.226.150
ip classless
ip http server
ip http secure-server
!
!
ip sla enable reaction-alerts
!
snmp-server community cisco RO
snmp-server enable traps config
snmp-server host 172.16.83.58 cisco
!
!
line con 0
line vty 0 4
 login
line vty 5 15
 login
!
end
```