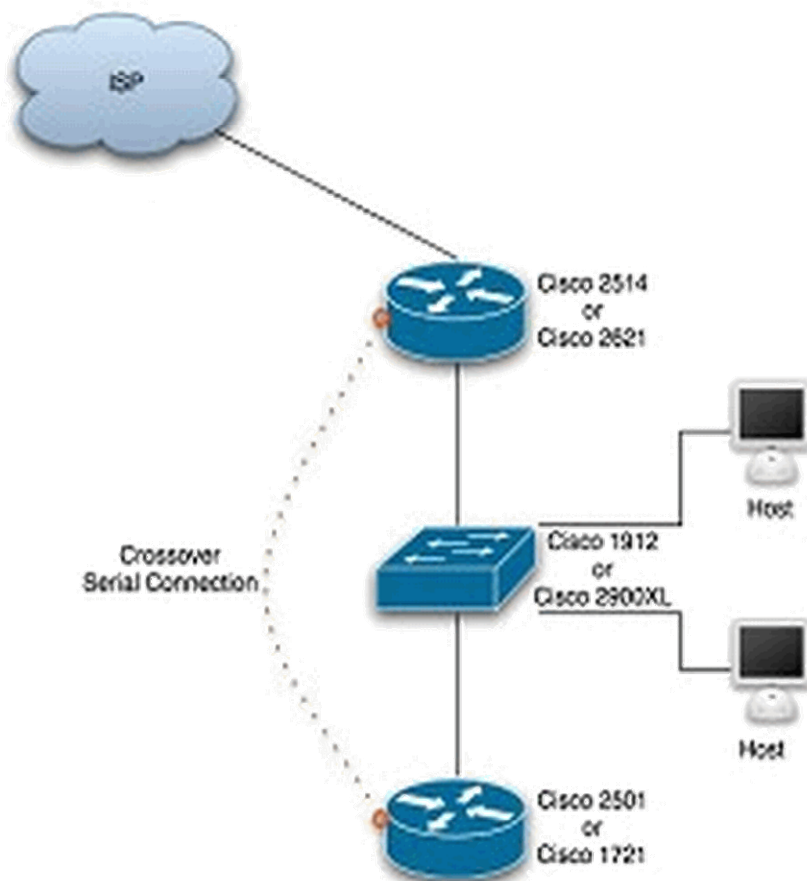


KOMUTAVIMAS IR MARŠRUTIZAVIMAS LOKALIUOSIUOSE TINKLUOSE



VILNIAUS UNIVERSITETAS
VILNIAUS PEDAGOGINIS UNIVERSITETAS

Feliksas Kuliešius, Kęstutis Svirskas, Artūras Rožė

KOMUTAVIMAS IR MARŠRUTIZAVIMAS LOKALIUOSIUOSE TINKLUOSE

Metodinė priemonė

Vilnius, 2008

Metodinė priemonė apsvartyta Vilniaus pedagoginio universiteto Fizikos ir technologijos fakulteto Teorinės fizikos ir informacinių technologijų katedros posėdyje 2008 m. kovo mėn. 20 d. (protokolo Nr. 6), Vilniaus pedagoginio universiteto Fizikos ir technologijos fakulteto tarybos posėdyje 2008 m. kovo 28 d. (protokolo Nr. 08-06-8) ir rekomenduota spausdinti.

Recenzentas doc. dr. Artūras Acus (VPU)

Mokslinis redaktorius doc. dr. Arūnas Maršalka (VU)

© Feliksas Kuliešius, Kęstutis Svirskas, Artūras Rožė, 2008

ISBN 9789955800378



Turinys

Pratarmė	4
Lokalusis kompiuterių tinklas	5
Vietinių tinklų realizacija	5
Transliacinis domenas	10
IP adresavimas.....	12
Tinklo dalijimas į potinklius.....	17
Maršrutizavimo protokolai	29
RIP protokolas	30
Standartinis maršrutizatoriaus konfigūravimas	34
IGRP protokolas	37
EIGRP protokolas.....	39
OSPF protokolas.....	46
Prieigos kontrolės sąrašai	57
Prieigos kontrolės sąrašo logika	57
Pakaitos simbolių šablonai	58
Prieigos kontrolės sąrašo konfigūravimas	59
Komutavimo pagrindai	62
Komutatorių funkcijos	62
Siejančio medžio protokolas (STP)	64
RSTP protokolas.....	73
Virtualieji tinklai ir VTP protokolas.....	76
Priedai	87
Skaičiavimo sistemos	87
OSI ir TCP/IP modeliai	92
Terminų žodynas	102

Pratarmė

Ši metodinė priemonė yra skirta universitetų ir kitų aukštųjų mokyklų studentams, studijuojantiems kompiuterinius tinklus, o taip pat ir visiems skaitytojams besidomintiems kompiuterinių tinklų darbo principais. Ji taip pat bus naudinga tiems, kurie ruošiasi laikyti CCNA ar CompTIA sertifikavimo egzaminus.

Šią discipliną autoriai jau eilę metų dėsto Cisco kompiuterinių tinklų akademijose, yra sertifikuoti ir sukaupę nemažą praktinės patirties. Tai leido atrinkti tik pagrindinės temas, sudarančias CCNA kurso branduolį, bei akcentuoti vietas, kurios studentams sunkiau suvokiamos. Šioje knygelėje nereikia ieškoti atsakymo į bet koki klausimą, ar mokytis vien iš jos, tai veikiau yra pagalbinė priemonė, nagrinėjanti sudėtingesnes kurso temas.

Nauji terminai tekste yra išskirti pastorintu šriftu. Už jų skliausteliuose visada pateikiama angliškas termino atitikmuo ir abreviatūra. Kur tik įmanoma, pateikiamos vartotinos specifinių terminų lietuviškos versijos, aprobuotos žodyne „Informatika. Keturkalbis terminų žodynas. Vilnius, 1999“.

Lokalusis kompiuterių tinklas

Norint suvokti lokaliajame tinkle vykstančius procesus kokybiniame lygmenyje, reikia gerai suprasti kolizijų domeno ir transliacinio domeno sąvokas, žinoti IP adresų klases ir mokėti jas skaidyti į atskirus potinklius, žinoti pagrindines 1-3 OSI lygmenims priklausančių įrenginių funkcijas (reikia pridėti – ir TCP/IP protokolus). Visi skaičiavimai vykdomi šešioliktainėje arba dvejetainėje skaičiavimo sistemose. Tai yra žinių minimumas, kuriomis disponuojant, galima pradėti studijuoti kompiuterinius tinklus.

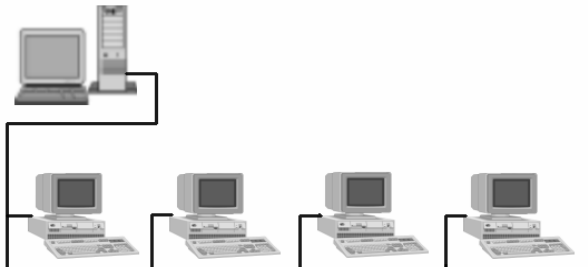
Vietinių tinklų realizacija

Vietinis tinklas yra kompiuterių (apibendrintai - mazgų, nes tinklo elementai gali būti įvairūs įrenginiai: spausdintuvai, maršrutizatoriai ir t.t.), kabelių ir kitų komunikacinių įrenginių visuma, valdoma tinklinės operacinės sistemos (TOS) ir taikomosios programinės įrangos.

Kompiuterių tinklų struktūra apibūdinama fizine (dažnai vadinama tiesiog topologija) ir logine (dažnai vadinama technologija) topologijomis bei tinklo programine ir aparatūrine sandara.

Tinklo topologija

Tinklo fizinė topologija yra jo kompiuterių jungimo į tinklą būdas – fizinis išsidėstymas vienas kito atžvilgiu, leidžiantis palyginti ir klasifikuoti įvairius tinklus. Yra trys pagrindiniai topologijų tipai: magistralė, žvaigždė ir žiedas. Dažnai, ypač didesni vietiniai tinklai, organizuojami taikant mišrią topologiją, t. y. dalis tinklo gali būti magistralės tipo, dalis – sujungta žvaigžde ar netgi naudojant perteklinę (mesh) schemą, kai kiekvienas mazgas yra sujungtas su visais likusiais.



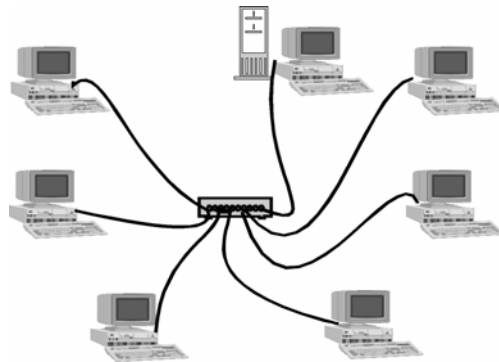
1 pav. Magistralės tipo vietinis tinklas.

Magistralės topologija (1 pav.) naudoja vieną bendrą visam tinklui perdavimo kanalą (paprastai koaksialųjį kabelį). Visi kompiuteriai sujungiami tiesiogiai prie šio kanalo. Pagrindinis šio jungimo

būdo trūkumas yra tai, kad pažeidimas vienoje vietoje išveda iš rikiuotės visą tinklą. Toks jungimo būdas šiandienos tinkluose beveik nebenaudojamas.

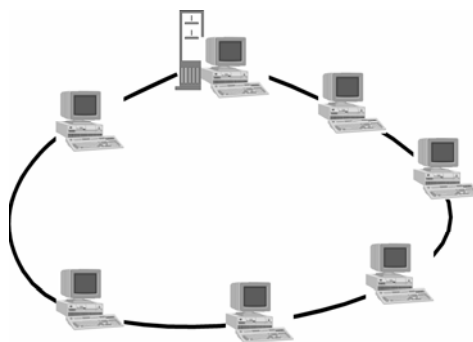
Naudojant **žvaigždės** topologiją (2 pav.), kompiuteriai jungiami tarpusavyje per koncentratorių (hub) arba komutatorių (switch). Tai yra specializuoti įrenginiai, skirti ryšio linijos išskaidymui, tačiau gali turėti ir papildomas logines ar tinklo apsaugos funkcijas. Ryšio linijoms naudojamas optinis arba vytos poros kabeliai.

Žvaigždės topologijos jungimui naudojama daugiau kabelio, tačiau tokios topologijos privalumas yra tai, kad pažeidus vieną jungtį tarp kompiuterio ir koncentratoriaus, likusi tinklo dalis veiks (blogiau, jei bus pažeistas pats koncentratorius). Be to naudojant komutatorius atsiranda papildomos galimybės padidinti tinklo įrenginių darbo našumą.



2 pav. Žvaigždės tipo vietinis tinklas.

Žiedo topologijai (3 pav.) būdingas uždaras informacijos perdavimo ciklas: Duomenys, išsiųsti



3 pav. Žiedo tipo vietinis tinklas.

viename žiedo taške, galų gale grįžta į pradžią. Skirtingai nuo žvaigždės, būtinas nenutrūkstamas kelias tarp visų kompiuterių, nes pažeidus tinklą vienoje vietoje, jis, kaip ir magistralės atveju, nustoja veikti. Žiedo topologijų tinkluose naudojami koaksialūs, optinis arba vytos poros kabelis. Dažnai naudojama žiedinio tinklo schema, kai žiedą sudaro centrinis valdymo įrenginys (VĮ), prie

kurio darbo stotys jungiamos panašiai kaip prie koncentratoriaus žvaigždės atveju. Tačiau, skirtingai nei koncentratorius, kuris tik išskaido liniją į atskiras atšakas, VĮ įjungia kompiuterius į tinklą vieną po kito žiedo principu. Žiedo topologija naudojama IBM Token Ring ir panašiuose tinkluose, sutinkamuose itin retai.

Tinklo technologijos

Kokią topologiją benaudotume, kai du ar daugiau kompiuterių pradeda perdavinėti duomenis vienu metu, tinkle kyla konfliktinė, neapibrėžta situacija, kadangi informaciją perduodantys mazgai turi pasidalinti bendru duomenų kanalu – vienas perduodantis įrenginys užima visą kanalą ir niekas kitas jo jau nebegali naudoti, nes tuo atveju informacijos nebebūtų galima išskirti. Procesas, skirtas šios problemos sprendimui yra vadinamas **magistralės arbitražu**. Jis nustato taisykles, kaip kompiuteris sužino, kad linija laisva ir galima perdavinėti duomenis. Yra du pagrindiniai arbitražo metodai: užimtumo aptikimas ir markerio (žymės, angl. – token) perdavimas. Pastaruoju atveju sakoma, kad tinklo valdymas centralizuotas, t. y., tinkle yra vienas arbitražo įrenginys, kuris sprendžia, kas turi teisę perduoti informaciją; pirmuoju gi – paskirstytas – kiekvienas tinklo kompiuteris pats sprendžia, ar jis gali pasiųsti pranešimą į tinklą.

Detaliau panagrinėsime tik paskirstytąjį arbitražą, kadangi jo pagrindu veikia dažniausiai sutinkami Ethernet technologijos, o labai panašiu – ir bevieliai Wi-Fi tinklai.

Užimtumo aptikimas dar vadinamas tuščio lango metodu arba kanalo paklausymo – užimtumo aptikimo (Carrier Sense Multiple Access with Collision Detection, CSMA/CD) metodu.

Kai tinklas dirba užimtumo aptikimo režime, kompiuteris pirmiausia analizuoja, ar kanalas laisvas, ir tik po to, jei nustatoma, kad kanale informacinių signalų nėra, pradeda perdavimą. Jei aptinkama, kad tinklas užimtas, kompiuteris palaukia perdavimo pabaigos ir po to kartuoja bandymą.

Jeigu du kompiuteriai pradeda savo perdavimą vienu metu (tokia situacija vadinama kolizija), abu kompiuteriai nutraukia perdavimą ir po trumpos atsitiktinės pertraukos vėl bando pradėti perdavimą. Pirmasis patikęs koliziją mazgas pasiunčia į liniją apie kolizijos buvimą informuojantį „jam“ signalą.

Ethernet

Ethernet 1975 m. sukūrė *Xerox* (*Xerox*, Palo Alto Research Center – PARC) mokslinė grupė, vadovaujama Bob Metcalf. Ethernet tinklas gali būti konstruojamas naudojant žvaigždės ir magistralės topologijas (jei naudojamas vytos poros kabelis – Ethernet'as konfigūruojamas tik kaip žvaigždė). Šiuolaikinė Ethernet'o versija priimta 1982 m. Standartinė Ethernet'o pralaida – 10 Mb/s, 100 Mb/s, 1 Gb/s arba 10 Gb/s. Ethernet technologijos pagrindas yra užimtumo aptikimo arbitražo metodas.

Standartinė (generic) Ethernet informacinio paketo struktūra pateikta 4 pav.

Preambulė	Paskirtis	Šaltinis	Tipas	Duomenys	CRC suma
8 baitai	6 baitai	6 baitai	2	46 – 1500	4

4 pav. Ethernet paketo struktūra.

Paketas susideda iš šių elementų:

- preambulė, skirta paketo sinchronizacijai yra sudaryta iš nulių ir vienetų sekų: pirmi 7 baitai yra vienodi – 10101010 ir tik 8-as baitas 10101011 – nesimetrinis;
- paskirtis ir šaltinis – tai unikalūs adresato ir siuntiklio Ethernet'iniai 48 bitų adresai, paprastai užrašomi šešioliktainiais skaičiais, pvz., 00 60 48 EC B2 C6. Šie adresai paskirstomi tinklo adapterių gamintojams ir fiksuojami adapterio pastoviojoje atmintyje, t. y. kiekvienas adapteris nuo pagaminimo momento jau turi savo identifikatorių – 48 bitų adresą;
- tipas (2 baitai) buvo įvestas Xerox ir naudotas siekiant nurodyti aukštesnio lygmens protokolą. Ethernete neinterpretuojamas. Aukštesnio lygmens protokolai pagal tipą gali atpažinti paketą, nenagrinėdami paketo turinio;
- duomenys; atkreipkime dėmesį, kad šis laukas negali būti trumpesnis, nei 46 baitai;
- CRC – perteklinės ciklinės sumos liekana (Cyclic Redundancy Checksum) – kontrolinė suma, skaičiuojama, naudojant CRC - 32 ar kitokio tipo polinomus. CRC naudojama perduodamos informacijos klaidų kontrolei.

Kolizijų domenas

Kolizijos reiškiny yra natūralus pasaulio LAN tinkluose vyraujančios Ethernet'o technologijos, išdava. Ethernet'o įrenginiai kadras privalo siųsti ir priimti, vadovaudamiesi nešlio paklausymo lygiagrečiosios kreipties su kolizijų aptikimu (CSMA/CD - Carrier Sense Multiple Access with Collision Detection) algoritmu. Bendrosios prieigos (shared access) Ethernet'o segmente, kur AK tarpusavyje bendrauja pusiau duplexiniu režimu, kiekvienas AK iš pradžių gali laisvai „kalbėti“, o po to klausytis, ar nėra kolizijų su kitais įrenginiais, kurie tuo metu tikriausiai taip pat „kalba“. Visas kolizijų aptikimo procesas yra pagrįstas faktu, kad Ethernet'o segmente kompiuteriai yra sujungti tam tikro maksimaliai leistino ilgio kabeliu ir todėl egzistuoja tam tikras maksimaliai leistinas laiko intervalas, per kurį kadras turi spėti nukeliauti nuo segmento vieno galo iki kito ir būti aptiktas tolimajame gale. Aišku, jeigu tarp mazgų yra šiek tiek didesnis atstumas, kartais įvyksta taip, kad du mazgai ima perduoti informaciją beveik vienu metu, nors abu prieš tai kabelyje nieko „negirdėjo“. Kolizijos metu informacija yra sugriauinama bitas po bito, nes įtampas susideda ir jų lygiai pakinta ir tinklo plokštės jų nebegali interpretuoti kaip informacijos. Tačiau mazgai moka susidoroti su kolizijomis. Susidūrus kadrui, pirmasis koliziją aptikęs mazgas ima generuoti specialų signalą,

informuojanti apie aptiktą koliziją. Generuojama pakankamai ilgai, kad visi segmento mazgai patikimai atpažintų koliziją. Po to visi mazgai staiga nutyla ir įjungia laikrodžius. Laikrodžių tikslėjimo trukmės yra išsibarstę. Pirmasis mazgas, kurio laikrodis jau baigė tikseti, „paklauso“, ar laide „tylu“ ir ima pardavinėti informaciją.

Kolizijų domenu yra vadinamas tinklo segmentas, kuriame atsiranda ir plečiasi kolizija. Šakotuvas (hub) išplečia kolizijų domeną, jis geografiškai tampa didesniu. Tai yra todėl, kad signalas, kuris patenka į vieną iš šakotuvo prievadų, automatiškai yra atkartojamas per likusius prievadus. Todėl prie šakotuvo prijungti mazgai gali atsidurti situacijoje, kai visi nori kalbėti, bet niekas negirdi. Galiausiai nusistovi režimas, kada kalbama iš eilės. Tai reiškia, kad visi mazgai pralaidumo juostos plotį dalijasi po lygiai. Iš čia seka pageidavimas kolizijų domenų tinkle turėti kuo mažesnius, nes tuomet tą patį dažnių juostos plotį tarpusavyje dalinsis mažesnis mazgų skaičius.

Kolizijų domeną į didesni skaičių mažesnių kolizijų domenų skaido 2 – lygmens įrenginys komutatorius. Su kiekvienu komutatoriaus prievadu yra susijęs atskiras kolizijų domenas. Kada prie atskirų komutatoriaus prievadų prijungti mazgai nori tarpusavyje „pakalbėti“, komutatorius, vadovaudamasis MAC adresų lentelės įrašais, juos elektroniskai sujungia, sudarydamas virtualią grandinę, egzistuojančią tik pokalbio metu. Pokalbiui nutrūkusi, grandinė nutraukiama. Komutatorius virtualias grandines gali sudaryti visiems prie prijungtiems mazgams. Svarbiausia yra tai, kad mazgai vienas kitam visiškai netrukdo „kalbėti“. Jie yra atskiruose kolizijų domenuose. Šiame kontekste yra sakoma, kad komutatorius dalija kolizijų domeną - kolizijų domenų atsiranda daugiau, bet mažesnių, o tai teigiamai atsiliepia tinklo funkcionavimui. Prie atskirų komutatoriaus prievadų prijungti PK nėra varginami nei kolizijų, nei dažnių juostos pločio dalybomis, nes perdavimas ir priėmimas vyksta atskiromis laidų poromis (1,2 ir 3, 6 laidai). Šioje situacijoje yra sakoma, kad AK tarpusavyje bendrauja duplexiniu režimu. Nepriklausomai nuo to, net ir pilno duplexinio režimo kanale kadrai privalo būti išsiųsti ir gauti per tam tikrą griežtai nustatytą laiko tarpą. Todėl maksimalus vytos poros (UTP) kabelio ilgis, jungiantis du PK, bendraujančius pilnu duplexiniu režimu, išlieka toks pat kaip ir pusiniui duplexiniam režimui.

Nors komutatorius praktiškai be delsos perduoda kadrus, bet jis yra skaidrus transliaciniam kadrui. Juos komutatorius privalo pakartoti per visus savo prievadus, išskyrus tą, per kurį transliacinis kadras buvo gautas. O jeigu mes turime ne vieną, o keletą komutatorių? Tada prie jų prijungtų mazgų skaičius tampa nemažu ir kiekvieną transliacinį kadrai, kurie tinklo veiklai yra gyvybiškai reikalingi, privalo apdoroti visi mazgai. Didindami komutatorių skaičių, mes greitai pasieksime ribą, kada visi mazgai bus užimti tik transliaciniais kadrais ir jokios vartotojui naudingos informacijos perduoti

negalės. Todėl yra reikalingas 3 - lygmens įrenginys, kuris gebėtų dalinti transliacinį domeną, kaip kad kolizijų domeną dalija komutatorius. Toks įrenginys yra maršrutizatorius.

Transliacinis domenas

Pagrindinė maršrutizatoriaus savybė, leidžianti logiškai struktūrizuoti tinklą, yra ta, kad maršrutizatorius neperduoda transliacinių paketų. Maršrutizatoriaus prievadai sulaiko transliacinius paketus, todėl kiekvienas prievadas yra atskirame transliaciniame domene.

Maršrutizatorius (Router) yra OSI 3 - jo lygmens įrenginys. Įrenginiai, susiję su maršrutizavimu 3 - jame lygmenyje, atlieka šias funkcijas:

- Perduoda paketus tarp tinklų, vadovaudamiesi 3-jo lygmens adresu.
- Kiekvieną paketą tinkle perduoda į tikslą optimaliausiu keliu.
- Paketo perdavimas susideda iš paskyrimo tinklų sąrašo peržiūros, sekančio šuolio maršrutizatoriaus adresu ir atitinkamo išėjimo (outbound interface) prievado, vedančio į tikslą, maršrutizatoriuje suradimo. Visa ši informacija yra maršrutizavimo lentelėje.
- Iš daugelio galimų kelių (maršrutų) išrenkamas optimalus kelias.
- Tarpusavyje maršrutizatoriai bendrauja, naudodami maršrutizavimo (routing), arba, kitais žodžiais tariant, **kelio parinkimo protokolus**

Pagal apibrėžimą maršrutizatorius neperduoda transliacinių paketų, todėl prie skirtingų maršrutizatoriaus prievadų prijungti tinklai (potinkliai) priklauso skirtingiems transliaciniams domenams. Tai leidžia kontroliuoti transliacijų sklidimą tinkle ir segmentuoti tinklą į dalis pagal IP adresus.

Faktiškai kadrai yra perduodami pagal 2- lygmens adresą. 3-jo lygmens įrenginiai tam tikru metodu susieja kanalinio lygmens adresą (MAC) su įrenginio tinklinio lygmens adresu (IP). Kiekvienam savo prievadui, prijungtam prie tinklo, maršrutizatorius taip pat privalo turėti priskirtus abiejų OSI lygmenų adresus. Visa tai maršrutizatoriui suteikia galimybę realizuoti tinklinio lygmens adresus, suteiktus fiziniams segmentams.

Prieš parenkant optimalų kelią kiekvienam paketui, maršrutizatorius privalo išanalizuoti paketo preambulėje įrašytą informaciją. Be to, kiekviename maršrutizatoriaus prievade, panaudojant šaltinio ir adresato adresus, protokolą arba kuri nors kitą 3-jo lygmens atributą, gali būti įdiegtos 3-jo lygmens saugumo ar kontrolės funkcijos, apsprendžiančios maršrutizatoriaus pasirinkimą praleisti paketą ar atmesti.

Optimalaus kelio parinkimas 3-jame lygmenyje atliekami procesorinėse sistemose. Kiekvieno paketo preambulėje esančios informacijos peržiūra reikalauja tam procesoriaus taktų skaičiaus. Kadangi maršrutizavimo lentelės gali būti labai didelės, jų peržiūra (lookup) trunka tam tikrą laiko tarpą. Taigi, kuo protingesnis tinklo įrenginys, kuo daugiau funkcijų jis atlieka, tuo didesnę delsą jis įneša į tinklą.

Taigi, maršrutizatorius geba perskaityti IP adresus, esančius paketo laukuose, ir jais operuoti, kiekvienam paketui parinkdamas geriausią kelią link tikslo ir perduodamas kadrą į atitinkamą prievadą. Geriausio kelio parinkimas kiekvienam paketui ir paketo komutavimas į atitinkamą maršrutizatoriaus prievadą, vedantį į tikslą, yra dvi pagrindinės maršrutizatoriaus funkcijos. Aišku, tam, kad maršrutizatorius galėtų perskaityti IP adresą, pirma jis privalo „nuvynioti“ kadro antraštę, t.y. vykdyti deinkapsuliavimą, o, parinkęs optimaliausią kelią ir nustatęs savo prievadą, vedantį link tikslo, privalo atlikti atvirkštinę - inkapsuliavimo operaciją. Jos metu kiekvieno segmento priekyje yra pridama antraštė, kurioje tarp kitų parametrų yra įrašoma siuntėjo ir gavėjo IP adresai, o gale - kontrolinė suma (CRC). Panašiai kanaliname lygmenyje paketas yra inkapsuliuojamas į 2-jo lygmens preambulę su siuntėjo ir gavėjo adresais ir kitais parametrais ir gale įrašomas CRC kodas.

Kiekvienas maršrutizatoriaus prievadas reiškia atskirą transliacinį domeną (aišku, ir atskirą kolizijų domeną). Maršrutizatorius dalija transliacinį domeną lygiai taip pat, kaip komutatorius dalija kolizijų domeną (tai nėra vienintelis maršrutizatoriaus privalumas). Taigi, kai transliacinis domenas tampa per didelis, tinkle instaliuojamas maršrutizatorius, suteikdamas galimybę prie kiekvieno prievado prijungti skirtingus tinklus, ar potinklius.

Perduodant informacijos paketus tinklu nuo vieno maršrutizatoriaus prie kito, kiekvienas maršrutizatorius vykdo duomenų deinkapsuliavimą iki 3-jo lygmens, o po to – inkapsuliavimą iki pat informacijos pavertimo bitais 1-me lygmenyje. Kelionės metu IP adresai nesikeičia, nes jie žymi siuntėjo adresą ir kelionės tikslą, tuo tarp MAC adresai kiekvieno šuolio metu kinta. Laiko momentu, kai kadras yra kažkuriame taške tarp dviejų eilinių maršrutizatorių, 2-jo lygmens preambulėje šaltinio MAC adresas rodo paskutinio maršrutizatoriaus prievadą, per kurį kadras iškeliaavo, o gavėjo MAC adresas rodo į sekančio maršrutizatoriaus prievadą, į kurį kadras pateks. Šių adresų kaita primena sustojimų kaitą troleibuso maršrute: išvykimo ir atvykimo vietų adresai nekinta, o esamo ir sekančio tarpinių sustojimų adresai (MAC adresai) nuolatos keičiasi.

IP adresavimas

Adresavimas Internete

Kompiuterių adresavimui Internete naudojama vieninga sistema: kompiuteriai adresuojami 32 bitų unikaliu adresu, nepasikartojančiu visame tinkle, vadinamu IP adresu. Tokius adresus galima užrašyti keliais būdais:

11000000 01100110 11111001 00000011	dvejetainė sistema,
3227973891	dešimtainė sistema,
C066F903	šešioliktainė sistema,
192.102.249.3	dešimtainė “su tašku” sistema.

Patogumo sumetimais pasirinktas paskutinyasis būdas, t. y. keturi IP adreso baitai užrašomi atskirtais tašku dešimtainiais skaičiais, kurie kinta nuo 0 iki 255.

32 bitų IP adresą sudaro 2 dalys: tinklo numeris ir kompiuterio (tiksliau – jo kortos) adresas tinkle. Internetas padalintas į tinklus pagal gana paprastą sistemą, kuri remiasi tinklų suskirstymu į klases A, B, C, D ir E. Kompiuterinių tinklų klasės skirstomos pagal minėtų tinklų dydį: A – dideli tinklai kompiuterių numeravimui tinkle skirti trys jaunesnieji baitai, t. y. 24 bitai – taip galima numeruoti iki 16777216 kompiuterių; B – vidutiniai tinklai – kompiuteriai tinkle adresuojami 2 baitais – iki 65536 kompiuterių; C – maži tinklai iki 255 kompiuterių – numeravimui tinkle skirtas tik jauniausias adreso baitas.

Tinklų klases IP adrese žymi pirmi vyriausieji adreso bitai (žr. 1 lent.):

1 lentelė. IP adresų skirstymas į klases

Klasė	Vyriausi bitai	Laisvi baitai tinklo kompiuteriams adresuoti
A	0	3
B	10	2
C	110	1
D	1110	klasė, skirta transliaciniam pranešimams
E	11110	rezervuoti adresai

Adresų erdvė, nepanaudota tinklo klasei bei kompiuterio numeriui tinkle, naudojama tinklo numeriui nurodyti: pvz., pilna C klasės IP adreso sandara bus tokia (žr. 2 lent.):

2 lentelė. C klasės IP adreso sandara

3 bitai	21 bitas	8 bitai
klasė (C)	tinklo Nr.	kompiuterio Nr.
110	00001 11011011 01010000	00001101

Vilniaus universiteto elektroninio pašto serveris, kurio IP adresas yra 193.219.80.13, priklauso C klasei, ir yra 13-tas kompiuteris 121640-tajame tinkle:

$$\begin{array}{rcl}
 & \text{Klasės požymis} & \text{Tinklo numeris} & \text{Kompiuterio numeris tinkle} \\
 & \underbrace{\hspace{1cm}} & \underbrace{\hspace{2cm}} & \underbrace{\hspace{1cm}} \\
 193.219.40.13 & \equiv & 11000001\ 11011011\ 01010000\ 00001101 \\
 & & 00001\ 11011011\ 01010000 & \equiv 121680 \\
 & & & 00001101 \equiv 13
 \end{array}$$

Reikia pastebėti, kad IP adreso struktūra yra apibrėžta tik dvejetainėje sistemoje. Konvertavus 32 bitų adresą į įprastai naudojamą 4 dešimtainių skaičių su taškais pavidalą, adreso skirstymas į klases, tinklo ir kompiuterių numerius tampa nebe toks akivaizdus: dešimtainiai skaičiai neturi jokios prasmės ir vartojami tik todėl, kad tokią užrašymo struktūrą lengviau atsiminti. Kadangi tinklo (kaip ir jo potinklio, žr. žemiau) numeris neturi prasmės (o ir naudoti jį nepatogu), tinklo (potinklio) numeriu dažniausiai vadinamas ne tinklo eilės numeris, o tinklo adresas, įskaitant ir klasės požymį, t.y visa IP adreso dalis be mazgo numerio. Dažnai ji vadinama **IP adreso prefiksu**.

Bet kokiame tinkle du adresai visada yra rezervuoti ir jų negalima skirti jokiame mazgui. Tai yra žemiausias ir aukščiausias tinklo adresai – pirmasis visada priskiriamas tinklui ir yra tinklo IP adresas (kartais vadinamas wire number), o paskutinis yra transliacinis tinklo adresas, kada adresuojami visi kompiuteriai tinkle.

Sudėjus visų klasių adresų diapazonus gautume, kad Internetas gali adresuoti daugiau nei 2 milijonus tinklų (ne kompiuterių, o tinklų!) ir 3,7 milijardus kompiuterių. Jei būtų naudojama 1 baitas tinklams numeruoti, o 3 – kompiuteriui tinkle, Internetė būtų galima adresuoti dar daugiau – virš 4 milijardų kompiuterių (255 tinklų po 16 milijonų kompiuterių), tačiau šią tokį adresavimo erdvės praradimą kompensuoja patogumas – smulkesnius tinklus lengviau administruoti.

Siekiant užtikrinti IP adresų vienareikšmiškumą, jie skirstomi centralizuotai: tai daro Interneto informacijos centras (Internet Network Information Center, InterNIC).

Grupiniai adresai (D klasė), kitaip vadinami transliaciniais adresais, kaip galima spėti iš pavadinimo, nusako tam tikrą kompiuterių grupę, kurios mazgai gali priklausyti skirtingiems tinklams. Grupė gali jungti neribotą mazgų skaičių, be to, kompiuterio priklausymas grupei yra dinamiškas, tai yra jis gali įsijungti ar išeiti iš grupės sudėties pagal savo nuožiūrą. Tam naudojamas IGMP (RFC 1112) protokolas. Tipinis grupinių adresų taikymas – video ar audio konferencijos, interaktyvūs žaidimai. IANA paskyrė eilę oficialių grupinių adresų.

Pasaulyje kompiuterių tinklai yra labai skirtingo dydžio. Transnacionalinėse korporacijose yra šimtai tūkstančių kompiuterių, universitetuose gal tik keliolika tūkstančių, o mažose įmonėse (SOHO-Small Office Home Office) – tik keli šimtai ar kelios dešimtys kompiuterių. Taigi, pasaulyje yra nedaug didžiulių tinklų, todėl jiems identifikuoti reikia nedaug bitų. Tam yra naudojami IP adreso vyriausieji bitai. Tuo tarpu mažų įmonių pasaulyje yra labai daug, todėl jų tinklams identifikuoti IP adrese reikia skirti žymiai daugiau bitų, o kiekvienos iš jų kompiuteriams suskaičiuoti užtenka tik keleto jauniausių IP adreso bitų.

Siekiant patenkinti tokius Interneto poreikius, skirtingų klasių tinklams numeruoti yra skirtas nevienodas bitų skaičius:

Tinklo klasė	IP adreso struktūra
A	N.H.H.H
B	N.N.H.H
C	N.N.N.H

Lentelėje raidėmis N pažymėti oktetai, skirti tinklams numeruoti, o raidėmis H – kompiuteriams.

Taigi, A klasei yra priskiriami IP numeriai, kuriuose pirmieji 8 bitai skirti tinklams numeruoti, o likusieji trys oktetai (24 bitai) – kompiuteriams. Iš viso pasaulyje yra ne daugiau, kaip 2^8 A klasės tinklų ir kiekviename iš jų yra po $2^{24} - 2$ kompiuterių. Minus 2 todėl, kad vienas IP adresas yra tinklo adresas, o kitas – transliacinis numeris. Kaip jau kalbėjome anksčiau, kai šis adresas būna įrašytas paketo antraštėje kaip tinklo adresas ir tada pranešimą turi perskaityti ir į jį reaguoti visi to tinklo kompiuteriai. Kada dėstytojas auditorijoje kreipiasi į visus studentus, tada galime skaityti, kad jis naudoja savotišką „transliacinį adresą“.

B klasės IP adresuose tinklams numeruoti yra skirti du pirmieji oktetai, taigi, iš viso yra 2^{16} B klasės tinklų. Likę du oktetai numeruoja kompiuterius (kompiuterio sinonimas yra terminas „mazgas“. Mazgas (angl. Host) yra bet koks tinklinis įrenginys, ar jo prievadas, kuriam yra būtinas IP adresas. Kiekviename B klasės tinkle gali būti ne daugiau, kaip $2^{16} - 2$ mazgų.

C klasės tinklų yra net 2^{24} ir kiekviename iš jų yra ne daugiau, kaip $2^8 - 2 = 254$ mazgai. Vartotojams gali būti priskiriami tik A, B, C klasių IP adresai.

Su kiekvienos klasės IP adresu yra susijusi standartinė potinklio maskuotė (SM - Subnet Mask). Jos ilgis lygus IP adreso ilgiui, tik visi oktetai, skirti tinklams numeruoti, yra užpildyti vienetais, o likusioji dalis – nuliais. Žemiau pateiktos A, B, C adresų klasių standartinės potinklio maskuotės (nurodytos dvejetainės ir dešimtainės formos):

A klasė 11111111.00000000.00000000.00000000 255.0.0.0
B klasė 11111111.11111111.00000000.00000000 255.255.0.0
C klasė 11111111.11111111.11111111.00000000 255.255.255.0

Dabar gali kilti logiškas klausimas – o kaip gi maršrutizatoriai Internetu atskiria, kokiai klasei priklauso IP adresas. Tai yra aktualu, nes klasiniai (classfull) maršrutizavimo protokolai, tokie kaip RIP, IGRP savo atnaujinimuose neperneša potinklio maskuotės, todėl jie turi taikyti A, B, C pilnos klasės potinklio maskuotę. Bet tam jie turi atpažinti IP adreso klasę. Atsakymas būtų toks: maršrutizatoriai IP adreso klasę nustato pagal taip vadinamą pirmojo okteto taisyklę (pagal vyriausiuosius bitus). Kas tai per taisyklė ir kokia jos esmė? Trumpai sakant, A klasei priklauso IP adresai, kurių pirmajame oktete yra skaičius tarp 0 ir 126. B klasei – skaičius tarp 128 ir 191, o C – skaičius tarp 192 ir 239.

Šiuos skaičius atsiminti yra gana sunku ir tai ilgainiui atsitiks automatiškai, o pradžioje reikėtų žinoti principą. Jis yra akivaizdus dvejetainėje sistemoje. Nuosekliai išdėliokime dvejetainę formą A klasės pirmojo okteto bitus:

A klasė

00000000, dešimtainis skaičius 0,
00000001, dešimtainis skaičius 1,
.....,
01111111, dešimtainis skaičius 127,

A klasės adresuose pirmojo okteto vyriausiasis bitas yra 0 ir negali būti pakeistas, taip gauname mažiausią skaičių 00000000. Didžiausias dvejetainis skaičius yra 127, bet šis adresas yra rezervuotas

(jam priklauso IP adresas, kurio pagalba yra tikrinama kompiuterio tinklo plokštė – 127.0.0.1), todėl maksimalus yra 126.

B klasės IP adreso pirmojo okteto pirmieji du vyriausieji bitai yra 10 (fiksiuoti). Kada visi jaunesnieji pirmojo okteto bitai yra 0, tada gauname mažiausią dešimtainį skaičių 128. Skaičiai didėja, kol visi bitai (išskyrus pirmuosius du fiksuotus 01) tampa lygūs 1. Taip gauname didžiausią pirmojo okteto skaičių 10111111, reiškiantį 191.

B klasė

10000000, dešimtainis skaičius 128,
10000001, dešimtainis skaičius 129,
.....
10111111, dešimtainis skaičius 255-64=191.

C klasės IP adreso pirmajame oktete pirmieji trys bitai 110 yra fiksuoti. Kada likę penki bitai lygūs 0, gauname pirmąjį (mažiausią) skaičių 192. Kada jaunesni penki bitai lygūs 1, gaunamas didžiausias skaičius 11011111 (dešimtainis 223). Kaip gaunama 223? Jeigu visi bitai būtų 1, tada turėtume skaičių 255. Kadangi trečiasis iš kairės bitas yra 0 (atitinka 32), tai šį skaičių turime atimti iš 255.

C klasė

1100 0000, dešimtainis skaičius 192,
1100 0001, dešimtainis skaičius 193,
.....
1101 1111, dešimtainis skaičius 255-32=223.

D klasės IP adreso pirmajame oktete pirmieji keturi bitai yra 1101. Kada likę keturi bitai yra 0, gauname pirmojo okteto mažiausią vertę 1101000, t.y. 224. Kada likę keturi bitai lygūs 1, gauname 11011111, o tai atitinka dešimtainį skaičių 239. D klasės adresai naudojami specialioms reikmėms – grupiniams adresams (multicast addresses). Kaip matysime vėliau, iš dalies šios klasės adresus naudoja maršrutizavimo protokolai RIP v1, RIP v2, EIGRP, OSPF, PIM protokolai.

D klasė

1110 0000, dešimtainis skaičius 224,
1110 0001, dešimtainis skaičius 225,
.....
1110 1111, dešimtainis skaičius 255-16=239.

Dar yra E klasės IP adresai, pirmajame oktete turintys skaičius nuo 240 iki 255. Tai Interneto eksperimentinėms reikmėms skirti adresai. Vartotojams Internetu yra skiriami A, B, C klasės IP adresai.

Tinklo dalijimas į potinklius

Labai trumpai apžvelgsime tinklų dalijimo į potinklius metodiką ir ją iliustruosime, išspręsdami keletą charakteringų uždavinių. Norėdami suprasti priežastis, dėl kurių yra dalijamas tinklas, įsivaizduokime didelę auditoriją, pilną studentų (dėstytojo dar nėra!). Visi intensyviai naudoja transliavimo režimą (triukšmauja), todėl sunku susikalbėti su už keleto metrų esančiu draugu, nekalbant apie tolimesnius. Taigi, Jūsų pokalbį girdi visa auditorija, o Jūs girdite visų kitų pokalbius.

Dabar įsivaizduokime, kad auditorijoje apie kiekvieną triukšmingesnę kompaniją išaugo keletas garsui nepralaidžių sienelių. Todėl dabar atskiros kompanijos pokalbiai sklinda tik jos lokaloje patalpoje. Kompanijos viena kitos negirdi. Beliko tik viena neaptarta detalė – o kaipgi pakalbėti su draugu iš kitos kompanijos?

Labai panaši situacija atsiranda tinkle, kai transliacinis domenas yra didelis. Tada bet kurio mazgo generuojamą užklausą transliaciniu adresu privalo apdoroti visi tinklo mazgai. Tuo metu jie negali atlikti tiesioginio darbo, pernešdami vartotojo informaciją, o tik apdoroja transliacinius kadrus.

Dabar pereikime prie realesnės aplinkos. Pradėkime nuo to, kad kiekvienas nepadalintas tinklas turi tinklo numerį, eilę IP adresų, priskirtinų mazgams ir transliacinį numerį. Mažiausias IP adresas šioje grupėje yra skiriamas tinklui identifikuoti. Jeigu, tarkime, turime A klasės tinklą, tai tinklo adresas bus įrašytas į pirmąjį okteta. Paskutinis tinklui priklausantis IP adresas yra transliacinis adresas. Tarp šių dviejų numerių esančius IP adresus galima priskirti mazgams.

Tinklą galima padalinti į 2^n dalių – potinklių. Pagrindinė problema – kaip pažymėti t.y. sunumeruoti potinklius. Jeigu jie neturės numerių, maršrutizatorius negalės identifikuoti potinklių ir perduoti jiems informaciją. Tinklams numeruoti skirtų bitų liesti negalima, todėl reikiamas bitų kiekis yra skolinamas iš kompiuteriams numeruoti skirtų bitų, nes su jais administratorius gal elgtis laisviau. Taigi, skolinti bitai (borrowed bits) yra skirti potinklams numeruoti. Kaip matysime vėliau, identifikuoja tinklus ir prievadus ir keičiasi įrenginio perdislokavimo metu. Kiekvienas potinklis turi visus tinklui būdingus atributus: potinklio IP adresą, kompiuteriams priskirtinus IP adresus ir transliacinį adresą. Žiūrint į IP adresą, negalima pasakyti, kiek bitų yra skolintų. Šią informaciją sužinome iš nestandartinės potinklio maskuotės (Custom Subnet Mask). Joje visi bitai, skirti tinklams ir potinklams numeruoti, yra pažymėti vienetais, o likusieji – nuliais. Taip, pavyzdžiui, tarkime, kad duotas C klasės IP adresas 192.168.12.10 su 3 skolintais bitais. Tada bendras bitų, skirtų tinklams numeruoti, ir skolintų bitų skaičius yra nurodomas prefiksu, t.y. už pasviro brūkšnio parašytu

skaičiumi 192.168.12.0/27. Šis prefiksas yra vienodas visiems, šiuo atveju $23=8$ potinkliams. Bitai, skirti kompiuteriams numeruoti, šiuo atveju nėra svarbūs, todėl paskutiniajame oktete parašėme 0, o ne 10. Pagal prefiksą galime iš karto parašyti nestandartinę potinklio maskuotę. Tam tereikia parašyti 27 vienetus (jų skaičių matome prefikse), ir dešiniau jų dar 5 nulius, kad bendras bitų skaičius būtų 32. Gautąją seką suskaidome į keturis oktetus ir parašome dešimtainėje sistemoje 255.255.255.224.

Klasikinėje potinklų adresavimo sistemoje minimalus bitų, kuriuos galima pasiskolinti, skaičius buvo 2, nes potinklis, kurio numeris mažiausias, taip vadinamas nulinis potinklis, ir paskutinis potinklis būdavo išbraukiami. Nūdienėse sistemose komanda `ip subnet-zero`, kuri dažniausiai yra numatytoji standartiškai ir netgi nematoma konfigūraciniuose failuose, įgalina išskirti ir naudoti nuliniam potinklui priklausančius IP adresus. Paskutinis (visų vienetukų), potinklis irgi gali būti naudojamas.

Maksimalus skolinamų bitų skaičius yra toks, kad dar du bitai liktų mazgams numeruoti, o tai reiškia 22 IP adresus – vienas adresas naudojamas potinklio numeriui, vienas – transliaciniam adresui, o dar du adresai lieka mazgams numeruoti. Išimtinio atveju gali būti naudojamas vienetinis (mazgo) potinklis vienam kompiuteriui adresuoti.

Kaip pavyzdį, išnagrinėkime keletą A, B ir C klasės tinklų padalinimo į potinklius uždavinių. Pats paprasčiausias yra C klasės tinklas, nes skolinami bitai yra tik iš ketvirtojo okteto. Toks pat paprastas atvejis gaunamas, kai iš B klasės tinklo yra skolinama daugiau, nei 8 bitai, arba kai iš A klasės tinklo skolinama daugiau, nei 16 bitų.

1 uždavinys. Duotas C klasės tinklas 192.168.10.0. Padalinkime jį į keturis potinklius. Nustatykite kiekvieno potinklio numerį, potinklio IP adresą, mazgams priskirtinus IP adresus ir potinklų transliacinius IP adresus.

Sprendimas. Užrašykime uždavinio sąlygoje nurodytą IP adresą. Kadangi tinklą reikia dalinti į 4 dalis, o $2^2 = 4$, tai aišku, kad turime skolinti 2 bitus. Tokių potinklų prefiksas (prefiksas nurodo tinklams ir potinkliams numeruoti skirtų bitų, juos skaičiuojant iš kairės į dešinę skaičių) bus 26 bitai (/26), o potinklio maskuotė 255.255.255.192. Kadangi skolinti bitai priklauso ketvirtajam oktetai, tai dvejetainė forma užrašykime tik šį okteta, o pirmuosius tris palikime dešimtainėje formoje. Skolintus bitus išskirkime pajuodintu šriftu:

Nulinis potinklis	192.168.10. 00000000	192.168.10.0	nulinio potinklio IP adresas
	192.168.10. 00000001	192.168.10.1	pirmasis mazgui priskirtinas adresas
	192.168.10. 00000002	192.168.10.2	antrasis mazgui priskirtinas adresas
			
	192.168.10. 00111110	192.168.10.62	paskutinis mazgui priskirtinas adresas
Pirmasis potinklis	192.168.10. 00111111	192.168.10.63	transliacinis potinklio adresas
	192.168.10. 01000000	192.168.10.64	pirmojo potinklio IP adresas
	192.168.10. 01000001	192.168.10.65	pirmasis mazgui priskirtinas adresas
	192.168.10. 01000002	192.168.10.66	antrasis mazgui priskirtinas adresas
	.		
Antrasis potinklis	192.168.10. 01111110	192.168.10.126	paskutinis mazgui priskirtinas adresas
	192.168.10. 01111111	192.168.10.127	transliacinis potinklio adresas
	192.168.10.10000000	192.168.10.128	antrojo potinklio IP adresas
	192.168.10.10000001	192.168.10.129	pirmasis mazgui priskirtinas adresas
	192.168.10.10000010	192.168.10.130	antrasis mazgui priskirtinas adresas
Trečiasis potinklis			
	192.168.10.10111110	192.168.10.190	paskutinis mazgui priskirtinas adresas
	192.168.10.10111111	192.168.10.191	transliacinis potinklio adresas
	192.168.10.11000000	192.168.10.192	trečiojo potinklio IP adresas
	192.168.10.11000001	192.168.10.193	pirmasis mazgui priskirtinas adresas
Trečiasis potinklis			
	192.168.10.11111110	192.168.10.254	paskutinis mazgui priskirtinas adresas
	192.168.10.11111111	192.168.10.255	transliacinis potinklio adresas

Pastebėjime, koku „kvantu“ didėja potinklio adresai. Potinklius žymi pajuodinimu išskirti du bitai. Jauniausias bitas nurodo skirtumą tarp dviejų gretimų potinklių IP adresų, t.y. „kvanto“ dydį. Aukščiau pademonstruotas būdas greitam skaičiavimui nelabai tinka. Jeigu žinome skolintų bitų skaičių, tai iš karto žinome ir jauniausio skolinto bito dešimtainę vertę. Tada iš karto surašome potinklių adresus, pradėdami nuo nulinio, „kvantu“ vis didindami potinklio IP adresą:

192.168.10.0	nulinio potinklio IP adresas
192.168.10.64	pirmojo potinklio IP adresas
192.168.10.128	antrojo potinklio IP adresas
192.168.10.192	trečiojo potinklio IP adresas

Kadangi visi šie numeriai yra išsidėstę iš eilės, tai už nulinio potinklio transliacinio numerio seka pirmojo potinklio IP adresas. Vadinasi, iš pirmojo potinklio IP adreso 192.168.10.64 atėmę 1, gauname nulinio potinklio transliacinį numerį 192.168.10.63. Visi IP numeriai tarp 192.168.10.0 ir 192.168.10.63, t.y. 192.168.10.1-192.168.10.62 imtinai, yra priskirtini mazgams. Lygiai taip pat surandame trūkstantus kitų potinkių IP adresus.

2 uždavinys. Turime A klasės tinklą 10.10.0.0. Padalinkite jį į 8 potinklius, nurodydami kiekvieno potinklio numerį, potinklio IP adresą, transliacinį adresą ir mazgams priskirtinus IP adresus. Tą pačią užduotį pakartoti ir su kitų klasių tinklais.

Sprendimas. Pradedame nuo to, kad užrašome uždavinio sąlygoje nurodytą IP adresą. Kadangi tinklą reikia dalinti į 8 dalis, o $2^3=8$, tai aišku, kad turime skolinti 3 bitus. Kadangi skolinti bitai priklauso antrajam oktetui, tai dvejetainė forma rašykime tik šį oktetą. Skolintus bitus pajuodinkime (kaip ir anksčiau, dvejetainė forma rašykime tik tą oktetą, kuriam priklauso skolintieji bitai, šiuo atveju antrąjį):

Nulinis potinklis	10. 000 00000.0.0	10.0.0.0	potinklio IP adresas
	10. 000 00000.0.1	10.0.0.1	pirmajam mazgui priskirtinas adresas
	10. 000 00000.0.2	10.0.0.2	antrajam mazgui priskirtinas adresas
			
	10. 000 11111.255.254	10.31.255.254	paskutiniam mazgui priskirtinas adresas
Pirmasis potinklis	10. 000 11111.255.255	10.31.255.255	transliacinis potinklio adresas
	10. 001 00000.0.0	10.32.0.0.	potinklio IP adresas
	10. 001 00000.0.1	10.32.0.1.	pirmasis mazgui priskirtinas adresas
	10. 001 00000.0.2	10.32.0.2.	antrasis mazgui priskirtinas adresas
			
	10. 001 11111.255.254	10.63.255.254.	paskutinis mazgui priskirtinas adresas
	10. 001 11111.255.255	10.63.255.255.	transliacinis potinklio adresas

Antrasis potinklis	10. 010 00000.0.0	10.64.0.0	potinklio IP adresas
	10. 010 00000.0.1	10.64.0.1	
	10. 010 00000.0.2	10.64.0.2	
			
	10. 010 11111.255.254	10.95.255.254	
Trečiasis potinklis	10. 010 11111.255.255	10.95.255.255	transliacinis potinklio adresas
	10. 011 00000.0.0	10.96.0.0	potinklio IP adresas
	10. 011 00000.0.1	10.96.0.1	
			
	10. 011 11111.255.254	10.127.255.254	
	10. 011 11111.255.255	10.127.255.255	transliacinis potinklio adresas
	10. 100 00000.0.0	10.128.0.0	potinklio Nr.4 IP adresas
	10. 101 00000.0.0	10.160.0.0	potinklio Nr.5 IP adresas
	10. 110 00000.0.0	10.192.0.0	potinklio Nr.6 IP adresas
	10. 111 00000.0.0	10.224.0.0	potinklio Nr.7 IP adresas

Atkreipkite dėmesį, kad iš viso turime 8 potinklius, o ne 7, nes reikia nepamiršti pridėti ir nulinį potinklį. Potinklio adresas yra dvejetainis 3 bitų pajuodinimu išskirtas skaičius $\bar{u}$. Nepainiokite jo su potinklio IP adresu¹. Potinklio numeris yra tik trys bitai tarp brūkšnių. Kai rašome IP adresą dešimtainėje sistemoje, nekreipiame dėmesio į bitų priskyrimą potinkliams ar mazgams, o bitus interpretuojame pilnuose oktetuose.

Norėdami be klaidų padalinti tinklą, turime vadovautis paprastomis taisyklėmis:

- Skolinti bitai numeruoja potinklius. Šie bitai sudaro tolydžiai didėjančią bitų seką, kurio kiekvienai vertei mazgų bitai įgyja vertes nuo 00000 (visi nuliukai) iki 11111 (visi vienetukai).

¹ Kadangi potinklio numeris neturi prasmės, naudojamas tik siekiant įvertinti reikalingų potinklių skaičių (o ir naudoti jį nepatogu), potinklio numeriu, ypač turint daugiau potinklių ir/ar dirbant neklasiniame sistemoje, dažnai vadinamas ne potinklio eilės numeris, o tiesiog potinklio adresas.

■ Mazgų dalyje „visi nuliukai“ atitinka potinklio numerį, o „visi vienetukai“ – potinklio transliacinį numerį. Taigi, kai prievadams skirstome potinklio IP numerius, pats pirmasis ir pats paskutinis numeriai nenaudojami, nes jie rezervuoti aukščiau minėtoms reikmėms. Pavyzdžiui, jeigu potinkiems numeruoti iš C kasės skolinom 4 bitus, tai prievadams galėsime priskirti ne 2^4 , o tik 2^4-2 IP numerius.

Štai ir visos taisyklės, kurias taikome, spręsdami įvairiausius tinklų dalijimo uždavinius. Uždavinių į įvairovę nėra didelė.

Žemiau pateiksime keletą kitų tipinių uždavinių sprendimo pavyzdžių.

3 uždavinys. Koks yra potinklio, kuriam priklauso AK su IP adresu 192.16.54.236/28, adresas? Koks potinklio transliacinis adresas ir kokią IP adresų intervalą galime priskirti prievadams?

Sprendimas. Užrašas /28 (vadinamas prefiksu) reiškia, kad tinklams ir potinkiems numeruoti yra skirta 28 bitai, juos skaičiuojant iš kairės į dešinę. T.y., potinklio maskuotė SM yra 255.255.255.240. Kitaip tariant, mes iš paskutinio okteto paskolinome 4 bitus, kuriais numeruosime potinklius. Potinklio numerį galime nustatyti iš karto – tam tereikia SM ir IP numeriui atlikti loginę AND'ing operaciją (pirmuosius tris oktetus galėtume palikti dešimtainiame formate, kadangi skolinti bitai yra tik ketvirtajame oktete, bet šį kartą parašykime pilnai):

192.16.54.236	11000000.00010000.00110110.11101100	loginė daugybos (AND) operacija
255.255.255.240	11111111.11111111.11111111.11110000	
192.16.54.224	11000000.00010000.00110110.11100000	

Priminsime, kad loginės AND'ing operacijos rezultatas 1 tėra vieninteliu atveju – kai abu argumentai yra vienetai.

Rezultatą paverčiame dešimtainiais skaičiais ir gauname, kad potinklio IP numeris yra 192.16.54.224. Jeigu mus domina potinklio numeris, tai mes paprasčiausiai turime dešimtainiu skaičiumi paversti skolintus bitus (jie paryškinti juodai). Taigi mus dominančio potinklio numeris yra 1110, arba 14.

Vienintelis parametras, kuris nurodo, kiek bitų yra skolintų, yra potinklio maskuotė SM (kuri gali būti išreikšta prefiksu). Dvejetainėje sistemoje tinklams ir potinkiems skaičiuoti skirtose pozicijose SM turi 1, o mazgų pozicijose – 0. SM yra laipto pavidalo funkcija.

Atkreipkite dėmesį, kad jauniausio skolinio bito vertė yra lygi 16, t.y. dviejų gretimų potinklų IP adresai skiriasi per 16. Pirmąjį iš 16 numerių (kiekviename potinklyje) mes paskiriame potinklio adresui, paskutinįjį – transliaciniam numeriui, o tarp jų esančius IP numerius galime priskirti kompiuteriams. Tą procedūrą pailiustruosime žemiau pateikta schema. Šalia potinklio, kuriame yra mazgas 192.16.54.236, joje parodyti ir gretimi potinkliai. Mūsų nagrinėjamas yra vidurinysis.

192.16.54.11010000 Potinklio Nr. 13 IP adresas

192.16.54.11100000 Potinklio Nr.14 IP adresas (1-asis 14 pot. adresas)

.11100001 Priskiriame prievadui

.11100010 Priskiriame prievadui

192.16.54.11101111 Transliacinis adresas 192.16.54.239

192.16.54.11110000 Sekantis potinklis (Potinklis Nr. 15) 192.16.54.240

Pastorinimu išskirti skolinieji bitai. Matome, kad adresas, vienetu didesnis už 14 potinklio IP numerį yra 192.16.54.11100001, t.y. 192.16.54.225. Jis yra naudotinas adresas ir jį galime priskirti kuriam nors kompiuteriui, serveriui ar kitam mazgui. Pats paskutinis potinklio adresas 192.16.54.239 (visi vienetukai mazgų srityje) yra rezervuotas transliavimo reikmėms.

Uždavinio atsakymą formuluojame taip: mazgas 192.16.54.236 priklauso potinklui Nr.14. Potinklio IP adresas yra 192.16.54.224, transliacinis IP adresas - 192.16.54.239, o IP numeriai, kuriuos galima priskirti prievadams (naudotini adresai) yra 192.16.54.225-192.16.54.238.

4 uždavinys. Turime A klasės tinklą 10.0.0.0 255.255.128.0. Rasti 57 potinklio IP numerį, transliacinį numerį ir IP numerių, kuriuos galime priskirti prievadams, intervalą.

Sprendimas: Kadangi nepadalinto A klasės tinklo standartinė SM yra 255.0.0.0, tai iš užduotyje pateiktos SM matome, kad potinkiems numeruoti yra skolintas visas 2-asis oktetas ir vienas 3-jo okteto bitas (tai rodo skaičius 128). Vadinasi iš viso yra skolinta 9 bitai. Šiuos bitus išskiriame juos paryškindami (galėtume išskirti ir kitaip, pvz., vertikaliomis linijomis).

Išskirtojoje zonoje, naudojant visus 9 skolintus bitus, užrašome 57 dvejetainę formą – tai yra 000111001. Toliau telieka nuosekliai perskaityti visą adresą, atkreipiant dėmesį tik į grupavimą baitais (oktetais).

10.00011100.10000000.00000000	10.28.128.0	57-jo potinklio adresas
10.00011100.10000000.00000001	10.28.128.1	prievadų IP numeriai
10.00011100.10000000.00000010	10.28.128.2	

10.00011100.11111111.11111111	10.28.255.255	57-jo potinklio transliacinis adresas
10.00011101.00000000.00000000	10.29.0.0	58-jo potinklio adresas

Štai ir visa technologija! Atkreipkite dėmesį dar į vieną aplinkybę: nors pastorinimu išskirtais bitais užrašant potinklio numerį dvejetainėje formoje adreso oktetus skiriantis taškas atsidūrė išskirtojoje zonoje, bet, rašydami potinklio numerį, į tašką nekreipiame dėmesio. Tuo tarpu, rašant IP numerį dešimtaine forma, imame visą okteta, t.y. skaičius tarp dviejų taškų ir nekreipkime dėmesio į pastorinimus.

Dvejetainė sistema padeda geriau suvokti skaidymo į potinklius technologiją, bet ne itin patogi praktiniam naudojimui. Praktiškai sprendžiant su potinklių adresavimu susijusius uždavinius, kaip ir kitoje kasdienėje veikloje, vis tik yra patogiau naudoti dešimtainę sistemą.

Pakartokime 3-čią užduotį: Rasti potinklio, kuriam priklauso AK su IP adresu 192.16.54.236/28, parametrus. Koks potinklio transliacinis adresas ir kokį IP adresų intervalą galime priskirti prievadams?

Nustatome, kad /28 atitinkanti SM yra 255.255.255.240. Mums aktualus tik 4-tasis oktetas (240) – oktetai, kur visi elementai yra 1 ar 0 – neįdomūs. Iš 256 (visas okteto kombinacijų skaičius) atėmus 240 (SM vertė) gauname 16, t.y. potinklio elementų skaičių. Dabar belieka 236 (sąlygoje duotas tinklo elemento adresas) padalinti iš 16. Didžiausias dalinys (224), mažesnis už 236 ir bus ieškomo potinklio pradžia – potinklio adresas (verta pastebėti, kad kiti dalybos komponentai – dalmuo, liekana ir t.t. šiuo atveju visai nesvarbūs). Pridėjus potinklio elementų skaičių 16, gausime sekančio potinklio adresą (240). Vienetu mažesnis bus ieškomo potinklio transliacinis adresas (239). Analizuojant tik 4-tąjį okteta, gautume:

256	236 16	
-240	224	Potinklio adresas

= 16	225 – 238	Galimi mazgų adresai
	+16 239	Transliacinis adresas
	= 240	Potinklio adresas

Kaip ir reikėjo tikėtis, rezultatas sutampa su gautuoju skaičiuojant dvejetainėje sistemoje.

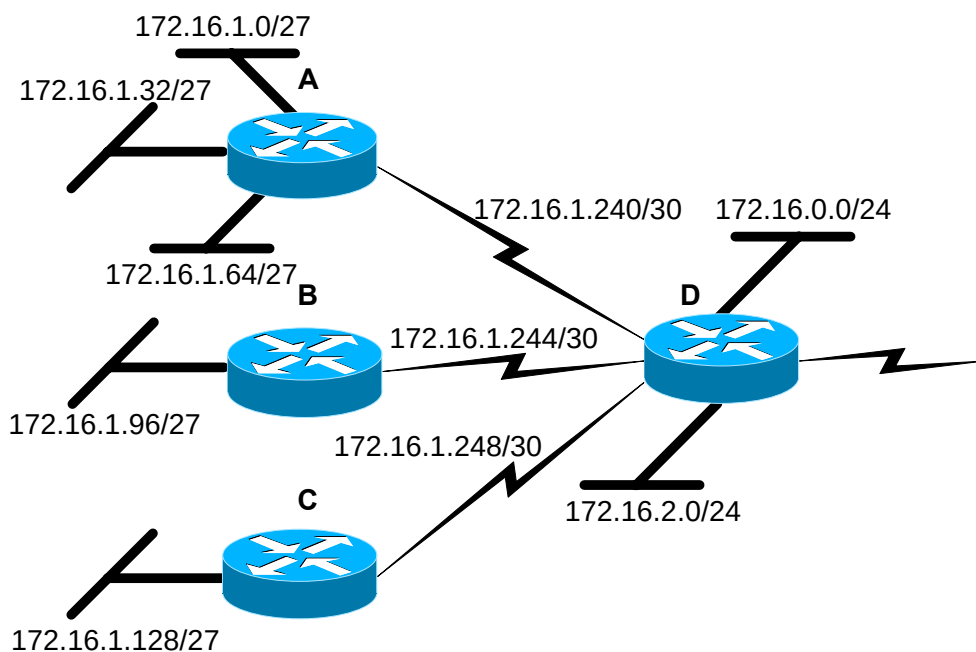
VLSM

Maksimaliai išnaudoti turimą adresų erdvę galima projektuojant tinklus su VLSM (Variable Length Subnet Masking) maskuote. Kintamo ilgio potinklio maskuotės įgalina naudoti skirtingas maskuotes tame pačiame tinkle, t. y. padalinti tinklą į skirtingas dalis. Kitaip sakant, VLSM yra problemų, susijusių su IP adresų trūkumu ir maršrutizavimo lentelių augimu, sprendimo mechanizmas.

Planuojant VLSM adresų erdvę ir apskaičiuojant potinkių adresus vadovaujamasi ta pačia logika ir taisyklėmis, kurios buvo naudojamos diegiant klasikinius potinklius ir aprašytos aukščiau. Svarbu visada atkreipti dėmesį, kad potinklio adresas visada (kaip ir dalinant tinklą į lygias dalis) būtų kartotinis to potinklio dydžiui.

Reikia nepamiršti, kad VLSM galima naudoti tik tada, kai yra galimybė naudoti neklasinius maršrutizavimo protokolus (RIP v2, OSPF, EIGRP ir pan.), kurie savo atnaujinimuose gali perduoti informaciją apie potinkių maskuotes (taip apibūrinant potinkių struktūrą). Klasiniai maršrutizavimo protokolai VLSM pagrindu suformuotame tinkle informaciją apie jo struktūrą platins neteisingai. Statinis maršrutizavimas, be abejo, leidžia tinklo administratoriui aprašyti tiek klasiniu, tiek neklasiniu principais suformuotus tinklus.

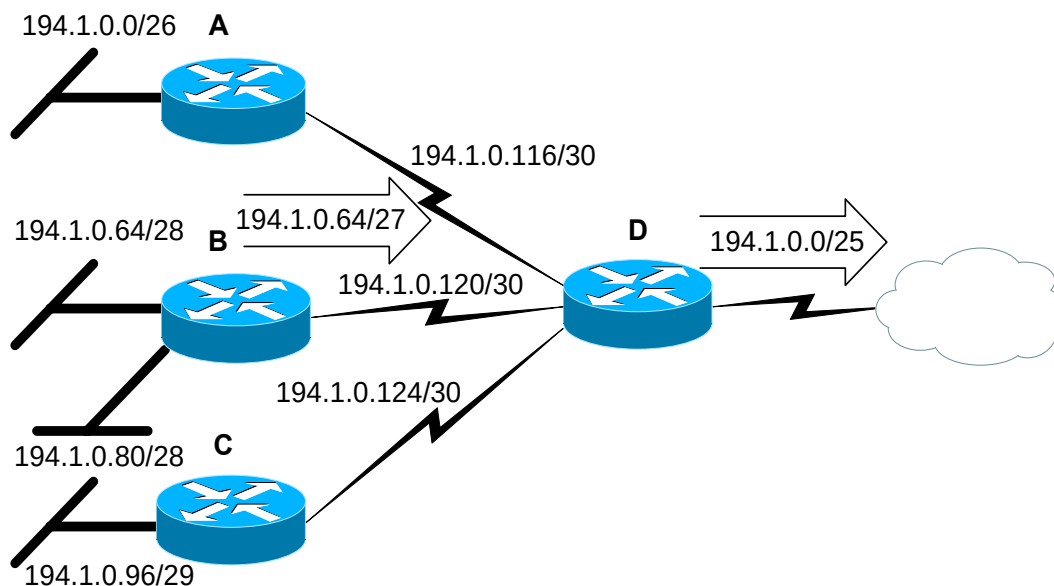
Pagrindinis VLSM privalumas – žymiai efektyvesnis adresų erdvės naudojimas. VLSM taikymas įgalina optimalizuoti tinklo struktūrą, kai planuojant tinklo adresavimą naudojamos hierarchinės schemos – tinklai dalinami į potinklius, pastarieji – į dar mažesnes grupes ir t. t.



4 pav. Tinklo dalinimas į potinklius naudojant VLSM.

Kaip pavyzdį nagrinėkime tinklą 172.16.0.0/16 (4 pav.). Pirmiausia, jis yra padalintas naudojant 24 bitų prefiksą (/24), o potinklis 172.16.1.0/24 padalintas į dar mažesnes dalis – prefiksai /27 ir /30.

Kaip ką tik minėjome, naudojant VLSM galima sukurti hierarchinę adresavimo struktūrą ir taip organizuoti efektyvesnį pasikeitimą maršrutizavimo atnaujinimų duomenimis. 5 pav. pateikta kelių lygmenų schema: maršrutizatorius B sumuoja du maršrutizavimo atnaujinimus apie potinklius



5 pav. Adresavimas ir sumavimas, naudojant VLSM maskuotę.

194.1.0.64/28 ir 194.1.0.80/28 kaip vieną įrašą 194.1.0.64/27. Maršrutizatorius D visus gautus iš A, B ir C atnaujinimus bei turimą informaciją apie tiesiogiai prijungtus tinklus susumuoja į vieną bendrą 194.1.0.0/25 potinklį. Toks sumavimas mažina maršrutizatorių resursų (atminties, procesoriaus laiko) ir tinklo apkrovos poreikį bei mažina delsą maršrutizatoriuose (tenka peržiūrėti mažesnes maršrutizavimo lenteles norint nukreipti paketą). Dar vienas svarbus sumavimo privalumas – taip galima izoliuoti vidiniuose potinkliuose vykstančius pokyčius nuo išorėje vykstančių procesų, t.y. šuoliukai (kokio nors kanalo atsijungimai ir po to sekantys pakartotiniai įsijungimai (flopping)) 194.1.0.0/25 viduje neįtakos agreguoto maršruto – joks maršrutizatorius, esantis domeno išorėje neprivalės keisti savo maršrutizavimo lentelių, nes „nepastebės“ ir nebus įtakojamas vidinių šuoliukų.

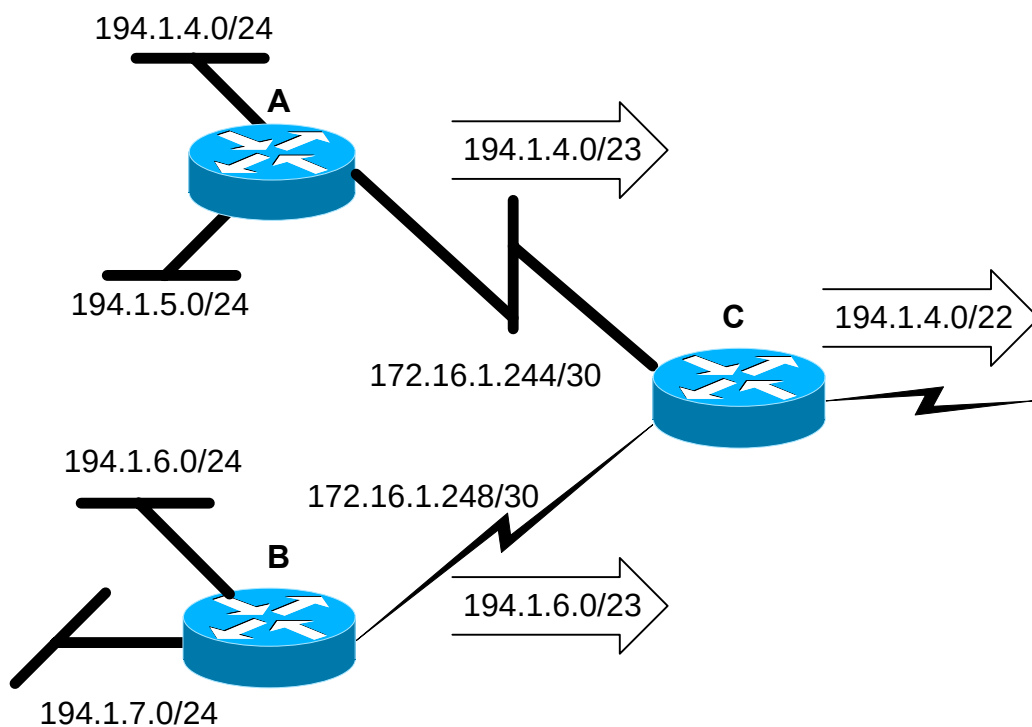
Maksimalų sumavimo efektą galima pasiekti tada, kai tinklo adresavimo schema yra parinkta teisingai: potinkliai išdėstyti nuosekliai, netrūkiais blokais. Jeigu adresai išdėstyti nenuosekliai

(noncontiguous), juos galima jungti į grupes ir sumavimą atlikti mažesnėse grupėse.

CIDR

Didelių tinklų sistemoje, kur gali būti šimtai ir tūkstančiai tinklų ir potinklų visai nebūtina juos aprašyti kiekviename maršrutizatoriuje. Maršrutų, kuriuos turi apdoroti maršrutizatorius skaičių, o tuo pačiu ir tinklinių resursų poreikį, galima sumažinti sumuojant šiuos potinklius ir tinklus (kartais šis procesas vadinamas agregavimu arba supertinklų formavimu).

Neklasinė tarpdomeninė maršrutizacija (Classless Interdomain Routing – CIDR) ir yra problemų, susijusių su maršrutizavimo lentelių augimu bei perpildymu, sprendimo mechanizmas. CIDR pagrindas yra galimybė grupuoti (agreguoti) klasiniu pagrindu suformuotus tinklus, taip sukuriant didelius neklasinių adresų rinkinius. Ši galimybė, kaip ir VLSM, atsirado įdiegus neklasinius maršrutizavimo protokolus, savo maršrutizavimo atnaujinimuose perduodančius ne tik žinomų tinklų adresus, bet ir potinklio maskuotes. Interneto paslaugų teikėjai (IPT) ar didelės įmonės sugrupuoja savo turimus adresus (tinklus) į neklasines grupes, taip sumažindami įrašų skaičių maršrutizavimo informacijos atnaujinimų srautus. CIDR yra detalai aprašyta RFC 1518 ir 1519. CIDR maršrutizavimo pavyzdys pateiktas 6 pav.



6 pav. CIDR maršrutizacija: IP adresų (tinklų) sumavimas

IPT maršrutizatorius visus 4 tinkles publikuoja tolesniems maršrutizatoriams kaip vieną neklasini adresą 194.1.4.0/22 (tuo pačiu ir vieną įrašą maršrutizavimo lentelėje bei maršrutizavimų atnaujinimuose), nurodydamas, kad turi prieigą prie visų įtrauktų į grupę tinklų.

Maršrutizavimo protokolai

Pagrindinė kiekvieno kompiuterinio tinklo paskirtis – perduoti vartotojo duomenis iš vieno tinklo taško į kitą. Ir nesvarbu, kaip toli jie būtų vienas nuo kito. Internetas yra sudarytas iš daugybės maršrutizatorių tinklo. Vartotojo duomenys paketais yra perduodami šiuo tinklu. Dažniausiai ne visi paketai keliauja tuo pačiu keliu, nes Interneto sąlygos nuolat kinta. Vienuose keliuose sąlygos gali pagerėti, kituose – pablogėti, o dar kiti – visai prapulti. Kiekvieno maršrutizatoriaus paskirtis – žinoti visą informaciją apie kelių būklę (ypatingai lokaliaje aplinkoje) ir perdavimo sąlygas, informaciją apie šias sąlygas saugoti tam tikrose – maršrutizavimo – lentelėse ir pastarųjų lentelių pagrindu, kiekvienam informacijos paketui parinkti patį optimaliausią kelią. Šis kelias yra parenkamas, nustatant prievadą, per kurį paketas turėtų būti išsiųstas sekančio maršrutizatoriaus link. Kiekvienam paketui maršrutizatorius išrenka optimaliausią kelią ir paketą nukreipia į atitinkamą išvesties prievadą – tai vadinama maršrutizatoriaus komutavimu. Komutavimą maršrutizatoriai vykdo, naudodami dviejų tipų protokolus. Vieni protokolai (tokie kaip IP, IPX ir kt.) nustato, kaip perduodama informacija tinkle – jie vadinami maršrutizuojamais (routed) protokolais. Kiti – (tokie kaip RIP, IGRP, EIGRP ir kt.) protokolai numato, kaip surenkama informacija apie tinklo struktūrą ir formuojamos maršrutizavimo lentelės. Jie vadinami maršrutizuojančiais (routing) protokolais. Pagal šios lentelės duomenis kiekvienam paketui maršrutizatorius privalo išrinkti optimaliausią kelią.

Yra skiriamas statinis ir dinaminis maršrutizavimas. Statinio maršrutizavimo atveju už tinklo struktūros aprašymą ir valdymą atsako tinklo administratorius. Jis, naudodamas statines `ip route` komandas nurodo, kaip pasiekiamas vienas ar kitas paskirties tinklas. Toks maršrutizavimo būdas yra varginantis ir neefektyvus, kadangi, ypač didesniuose tinkluose administratorius yra fiziškai nepajėgus laiku reaguoti į tinkle vykstančius pokyčius. Žymiai efektyvesnis yra automatizuotas – dinaminis maršrutizavimas, kai tinklo būsenos sekimo funkcija perduodama patiems tinklo įrenginiams, dažniausiai maršrutizatoriams. Negalima, vienok, išsiversti ir be statinio maršrutizavimo: naudojant statinius maršrutus aprašomos saugesnės linijos į uždaras zonas ir iš jų, formuojamos rezervinės linijos bei numatytieji (default) keliai. Pvz., komanda

```
Router(config)#ip route 0.0.0.0 0.0.0.0 Serial1
```

nurodoma į maršrutų lentelę įtraukti numatytąjį kelią – visus paketus, kurių paskirties tinklas

neaprašytas tiesiogiai, nukreipti per S1 prievadą.

Dinaminiai maršrutizavimo protokolai yra skirstomi į:

■ **vidinius maršrutizavimo protokolus** (Interior Gateway Protocols): RIP, IGRP, EIGRP, OSPF, IS-IS. Šie savo ruožtu gali būti skirstomi į atstumo vektorinius (RIP, IGRP), ryšio būsenos (OSPF, IS-IS) ir mišrius (EIGRP) protokolus

■ **išorinius maršrutizavimo protokolus** (Exterior Gateway Protocols: BGP).

BGP protokolus naudoja Interneto paslaugų teikėjai, nors, esant tam tikroms sąlygoms, šį protokolą turi naudoti ir didelės įmonės ar organizacijos. Toliau mes trumpai aptarsime RIP, EIGRP ir OSPF protokolus.

RIP protokolas

RIP yra paprasčiausias klasinis atstumo vektorinis protokolas, savo atnaujinimuose nesiunčiantis potinklio maskuotės SM. RIP yra paprastai konfigūruojamas ir administratoruojamas protokolas. Jis naudojamas nedideliuose tinkluose. Nuo 1988 metų RIP protokolas yra standartizuotas, jo pilną aprašymą galima rasti užklausoje komentarams (Request for Comments - RFC) 1058.

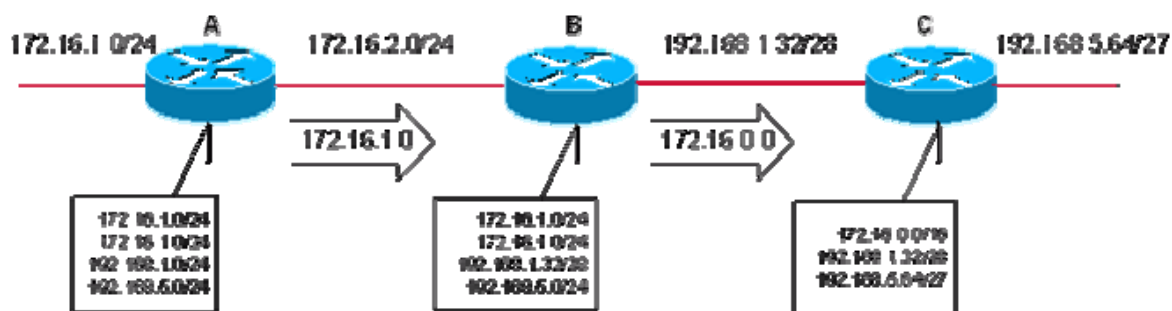
RIP protokole yra naudojama primityvi metrika – maršrutizatorių (arba šuolių) skaičius iki paskirties tinklo. Šis skaičius negali viršyti 15 šuolių: jeigu tinklo diametras viršija 15 maršrutizatorių, RIP protokolo naudoti negalima. RIP neatsižvelgia į dažnių juostos plotį, todėl, jeigu į paskirties tinklą veda du keliai, tarkime, pirmasis 56 kbps, vieno šuolio atstumu, o kitas - 1 Gbps, 2 šuolių atstumu, RIP protokolas visada parinks blogesnį kelią. Informaciją kaimynams RIP išplatina kas 30 s. siųsdamas atnaujinimus – visą maršrutizavimo lentelės turinį. Įvykus tinklo trikdžiui – išėjus iš rikiuotės arba atsiradus aktyviam prievadui, išsiunčiami vadinamieji trikžio generuoti atnaujinimai.

RIPv1, kaip ir kiti klasiniai protokolai (pvz., žemiau aprašytas IGRP), neperduoda informacijos apie potinklio maskuotę SM. Įdiegus tokius protokolus, potinklių struktūrą galima realizuoti tik naudojant vienodas potinklio maskuotes. Be to, tokiu atveju pasikeitimas informacija apie potinklius galimas tik pagrindinio tinklo ribose – peržengiant šią ribą, potinklių adresai sumuojami iki klasinio tinklo adreso. Taigi, gavęs atnaujinimo paketą, maršrutizatorius SM įvertina taip:

■ jeigu atnaujinimo įrašas deklaruoja pagrindinio tinklo, kuriam priklauso gaunantysis prievadas, potinklį, maršrutizatorius priskiria jam gaunančiajame prievade nustatytą SM;

- jeigu atnaujinimo įrašas susijęs su kitu tinklu, maršrutizatorius priskiria jam standartinę klasės SM.

Šiuos automatinio sumavimo principus iliustruoja 7 pav. pateiktas pavyzdys.



7 pav. Sumavimas klasiniuose (RIP, IGRP) maršrutizavimo protokoluose

Maršrutizatorius B turi prievadą 172.16.1.0/24 tinkle. Jeigu jis per šį prievadą gauna informaciją apie 172.16.0.0 tinklo potinklį, jam priskiriama prievado SM (/24). Siųsdamas atnaujinimą maršrutizatoriui C, informaciją apie potinklius iš 172.16.0.0 B sumuoja (/16), kadangi informacija išsiunčiama į kitą tinklą (192.168.1.0). Analogiški sumavimo procesai vyksta ir perduodant atnaujinimus iš C į A. A maršrutizavimo lentelėje matome informaciją apie 192.168.x.0 tinklus, bet ne apie potinklius. Jeigu 192.168.5.64/27 pakeistume į 172.16.5.0/24, maršrutizatorius A, negautų teisingos informacijos (nors jo dešinysis prievadas ir yra tinkle 172.16.2.0/24) apie 172.16.5.0/24, nes informacija jau būtų susumuota iki tinklo C->B atnaujinimuose. Tokia potinklų konfigūracija vadinama neasocijuota (incontiguous).

RIPv1 konfigūravimas yra gana paprastas. Tam tereikia pagrindiniame konfigūravimo režime aktyvuoti RIP protokolą (router rip) ir po to su network komanda deklaruoti visus pilnus tinklus, prie kurių prijungti konfigūruojamo maršrutizatoriaus prievadai:

```
R1#conf term
R1(config)#router rip
R1(config-router)#network 160.108.0.0
R1(config-router)#int s0
R1(config-if)#ip address 160.108.32.70 255.255.255.224
R1(config-if)#no shut
```

Deklaruojant tinklus RIP protokole (kaip ir kituose protokoluose) nurodoma, kad 1) atnaujinimai bus siunčiami ir 2) priimami per tinkle aprašytus prievadus ir, svarbiausia, įvardinti tinklai bus įtraukti į atnaujinimus (iš kitur gauta informacija, esanti maršrutų lentelėje į atnaujinimus įtraukiama visada).

RIPv2 protokolas yra patobulinta RIPv1 versija. Esminis patobulinimas yra tai, kad RIPv2 atnaujinimuose siunčia ir potinklio maskuotes (tam panaudoti RIPv1 pakete rezervuoti laukai). Be to, atnaujinimai siunčiami grupiniu adresu 224.0.0.9. Maršrutizatorius gali turėti prievadus, prijungtus prie tinklų, kuriuose naudojama skirtingo ilgio potinklio maskuotės. Tai leidžia naudoti beklasio tarpdomeninio maršrutizavimo kelių sumavimo (CIDR – Classless Interdomain Routing) ir/ar kintamo ilgio potinklio šablono technologijas (VLSM – Variable Length Subnet Mask). Nors RIPv2, kaip ir RIPv1, pagal nutylėjimą vykdo sumavimo klasiniu pagrindu funkciją, ji gali būti išjungiamą komanda **no auto-summary**. Papildomai, pageidaujamą sumavimą administratorius gali nurodyti komanda **ip summary**. Tai galima atlikti maršrutizavimo proceso arba prievado lygmenyje. Žemiau yra pateikta RIPv2 konfigūravimas aukščiau pateiktam tinklui.

```
R1#conf term
R1(config)#router rip
R1(config-router)#network 160.108.0.0
R1(config-router)#version 2
R1(config-router)#no auto-summary
R1(config)#int s0
R1(config-if)#ip address 160.108.32.70 255.255.255.224
R1(config-if)# ip summary-address rip 160.108.32.0 255.255.224.0
R1(config)#no shut
```

Išryškintos eilutė rodo RIPv2 konfigūravimą ir sumavimą prievado lygmenyje. Sumavimo komandoje kaip paskutinis argumentas yra naudojamas suminis potinklio šablonas (custom subnet mask).

Kovos su maršrutizavimo kilpomis metodai

Visą informaciją apie tinklų struktūrą RIP protokolas gauna iš kaimyninių maršrutizatorių, kurie kas 30 sekundžių savo kaimynams transliaciniu adresu siunčia atnaujinimus (updates) - pilnas maršrutizavimo lenteles, o šie, savo ruožtu, savo kaimynams ir t.t.. Todėl, netikėtai pasikeitus tinklo

topologijai, informacija apie pokyčius maršrutizatorių sistema sklinda gana lėta banga (trikčio generuotus atnaujinimus išsiunčia tik tiesiogiai trikdį užfiksavęs maršrutizatorius). Taip, net esant paprastam 5-6 maršrutizatorių tinklui, laikas, per kurį visas tinklas sužino apie pokytį (sakoma, tinklas konverguoja), gali siekti iki 500 sek. Taigi, tam tikrą laiko tarpsnį egzistuoja maršrutizatoriai, kurie jau žino apie pokyčius, ir tie, kurie dar nežino (Tai yra būdinga visiems vidiniams distanciniams vektoriniams protokolams). Abipus sklindančios atnaujinimų apie pakitusią topologiją bangos išsidėstę maršrutizatoriai disponuoja skirtinga informacija apie situaciją tinkle ir vieni iš jų gali stengtis perduoti paketus į jau nebepasiekiamus tinklus. Laiko tarpsnyje, kol tinklas dar nėra konvergavęs (kada dar ne visi maršrutizatoriai žino apie pokyčius), su perduodama informacija gali atsitikti eilė negerų dalykų. Kokie tie dalykai ir kokiomis priemonėmis su jais protokolai kovoja, aptarsime žemiau.

■ **Skaičiavimas iki begalybės** (Count to Infinity)– Maršrutizatorius (A) sužino apie naujus tinklus iš kaimyno (B) ir (C) ir įrašo šią informaciją į maršrutizavimo lentelę. Maršrutizavimo lentelė išsiunčiama visiems kaimynams (B) ir (C). Tačiau, jeigu kaimynui (B) buvo nusiųsta informacija apie tinklus, apie kuriuos jis pats pranešė maršrutizatoriui (A), gausime taip vadinamą **maršrutizavimo kilpą**. Kai vienas iš kelių prapuola, kiekvienas kilpoje esantis maršrutizatorius gali manyti, kad jis tebeturi atsarginį kelią, vedantį per kaimyną. Siekiant sumažinti galimybę atsirasti kilpoms, kiekvienas maršrutizatorius, prieš siųsdamas atnaujinimą kaimynui, padidina šuolių skaičių iki adresato vienetu. Kai šis skaičius pasiekia 16, tinklas yra paskelbiamas kaip nepasiekiamas ir yra pašalinamas, nes RIPv1 tinklo diametras negali viršyti 15. Tai ir yra **skaičiavimo iki begalybės** taisyklė, kur begalybė lygi 16.

■ **Atkirstas horizontas** (Split horizon) – Tai prieš maršrutizavimo kilpas nukreiptas mechanizmas. Jeigu atkirstas horizontas veikia, tai skaičiavimas iki begalybės nereikalingas. Atkirsto horizonto taisyklė teigia, kad maršrutizavimo procesas negali skelbti informacijos apie tinklus per prievadą, per kurį jis išgirdo apie šiuos tinklus.

■ **Atkirstas horizontas su atnaujinimų koregavimu** (Split horizon with poison reverse) – Viena, atskirai paimta, atkirsto horizonto taisyklė negali užkirsti kelio kilpų, nors ji ir draudžia išsiųsti informaciją apie tinklus atgal per sąsają, per kurią ši informacija buvo gauta. Tačiau atnaujinimų koregavimas prarasto tinklo atveju nepaiso atkirsto horizonto taisyklės. Atnaujinimų koregavimas išrenka visus iš kaimyno sužinotus tinklus ir juos paskelbia su metrika, lygia begalybei (16). RIP protokolas tokius tinklus supranta kaip nepasiekiamus. Maršrutizavimo procesas apie tai informuoja tinklus, bet kartu pasižymi, kad kelias į šiuos tinklus yra negaliojantis. Nors atkirsto horizonto su

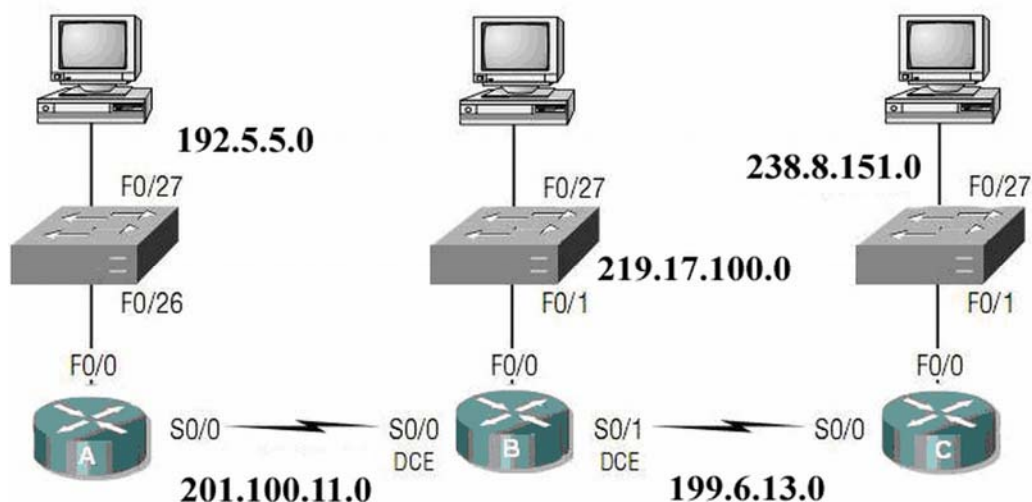
kelių koregavimu taisyklė ir padidina administracines atnaujinimų sąnaudas, bet ji geba eliminuoti kilpas.

■ **Atnaujinimų sulaikymas (Holddowns)** – Maršrutizavimo procesas, nusprendęs, kad kelias yra negaliojantis, standartiškai trijų atnaujinimų intervalą (90 sek RIP ir 270 sek – IGRP) laukia atnaujinimo su palankesne metrika. Kokia prasmė? Gali paprasčiausiai būti, kad koks nors prievadas periodiškai persijunginėja (is flapping), todėl geriau kurį laiką palaukti ir nekreipti į tai dėmesio.

Standartinis maršrutizatoriaus konfigūravimas

Šiame skirsnyje pateikiame standartinį maršrutizatoriaus konfigūravimą, įskaitant privilegijuotojo lygmens, konsolės, telneto linijų slaptažodžius, prievadus su jų aprašymais, RIP protokolą ir ip host lenteles. Šias komandas reikia įsiminti ir stengtis visada surinkti ta pačia tvarka. Taip greičiau pastebimos klaidos ir trūkstamos komandos.

Nagrinėsime laboratorinę maršrutizatorių sistemą, pateiktą 8 pav. Sistema yra sudaryta iš trijų



8 pav. Laboratorinė maršrutizatorių sistema.

maršrutizatorių A, B ir C. Prie kiekvieno maršrutizatoriaus yra prijungtas lokalus tinklas, kurio IP adresas yra nurodytas greta. Reikia sukonfigūruoti kairinį kraštinį maršrutizatorių. Secret slaptažodis – cisco. Konfigūracinis failas yra padalintas į keletą fragmentų. Kiekvienas iš jų aktyvuoja tam tikrą funkciją.

```
Router>enable  
Router#configure terminal  
Router(config)#hostname A  
Router(config)#enable password cisco  
Router(config)#enable secret class  
Router(config)#banner motd #  
Perspėjame, tai yra maršrutizatorius, priklausantis privačiam tinklui.  
#
```

Pirmoji aukščiau pateiktų komandų grupė nustato maršrutizatoriaus vardą A ir slaptažodžius. Komandos **enable password** ir **enable secret** nustato slaptažodžius, apribojančius prieigą į privilegijuotą operacinės sistemos režimą. Šie slaptažodžiai privalo būti skirtingi. Jeigu konfigūraciniame faile **enable secret** slaptažodis yra sukonfigūruotas, tai **enable password** nėra naudojamas. Konfigūraciniame faile enable password slaptažodis paprastai saugomas atviru tekstu, bet jį galima užšifruoti, panaudojus komandą **service password-encryption**.

Paskutinėje eilutėje esanti komanda **banner motd** sukuria įrašą, kuris prisijungiant prie įrenginio telneto linija pateikia informacinį apie įrenginį. Tekstas komandoje iš abiejų pusių turi būti atskirtas # simboliais.

```
A(config)#line console 0  
A(config-line)#password cisco  
A(config-line)#login  
  
A(config-line)#line vty 0 4  
A(config-line)#password cisco  
A(config-line)#login  
A(config-line)#exit
```

Šios dvi komandų grupės nustato konsolės ir penkių telnet'o linijų slaptažodžius atitinkamai. Telnet'u galima prisijungti tik sukonfigūravus telnet'o linijų slaptažodžius, priešingu atveju prisijungimas negalimas. Tiek konsolės, tiek vty slaptažodžiai leidžia prisijungti tik į vartotojo lygmenį. Norint patekti į privilegijuotąjį (enabled), reikia žinoto enable secret arba password slaptažodį.

```
A(config)#interface fa0/0
A(config-if)#ip address 192.5.5.1 255.255.255.0
A(config-if)#description A maršrutizatoriaus LAN prievadas
A(config-if)#no shut
```

Pastarosios komandos fa0/0 prievadui suteikia IP adresą ir įrašo aprašomojo pobūdžio informaciją apie prievado paskirtį. Tai ypač patogu, kai maršrutizatorius turi daug prievadų. Komanda **no shut** aktyvuoja prievadą.

Nuoseklieji prievadai konfigūruojami analogiškai, bet yra vienas niuansas. Realiomis sąlygomis organizacijos maršrutizatorius yra sujungtas su CSU/DSU įrenginiu. Šis įrenginys atlieka sinchronizavimo funkciją, t.y. jis yra DCE (Data Circuit Equipment) įrenginys. Kadangi laboratorijose tokių įrenginių nėra, tai maršrutizatorių, kurie standartiškai yra DTE (Data Terminal Equipment) įrenginiai, prievadai emuliuoja DCE (taip, kad viename kanalo gale visada būtų DCE prievadas). Jie yra sukonfigūruojami DCE prievade komanda **clock rate** nustatant linijos sinchronizaciją.

```
A(config-if)#interface serial 0/0
A(config-if)#ip address 201.100.11.1 255.255.255.0
A(config-if)#description WAN prievadas link B maršrutizatoriaus
A(config-if)#clock rate 64000
A(config-if)#no shut
A(config-if)#exit
```

RIP protokolas yra suaktyvinamas komanda **router rip** globaliajame lygmenyje. Po to, panaudojus komandą **network**, po vieną išvardijami visi prie maršrutizatoriaus prijungti tinklai, kurie bus įtraukiami į maršrutizavimo lentelės atnaujinimus ir taps žinomi kitiems maršrutizatoriams. Maršrutizavimo lentelėje (nepriklausomai, ar jie aprašyti RIP) šie tinklai skelbiami su indeksu C (connected).

```
A(config)#router rip
A(config-router)#network 192.5.5.0
A(config-router)#network 201.100.11.0
```

Jeigu norime sužinoti, koks maršrutizavimo protokolas yra įjungtas ir kokie tinklai yra skelbiami atnaujinimuose, naudojame komandą **show ip protocols**.

ip host komanda suformuoja lokaliai reikšmingą DNS lentelę, maršrutizatoriaus vardus, šiuo atveju B ir C, susiejančią su jų prievadų IP adresais. Tada į maršrutizatorių galima kreiptis ne IP adresu, o tiesiog vardu.

```
A(config)#ip host B 201.100.11.2 219.17.100.1 199.6.13.1
```

```
A(config)# ip host C 199.6.13.2 238.8.151.1
```

Kiekvieną kartą, kada yra pakeičiama konfigūracija, ją reikia įrašyti į konfigūracinį failą, esantį NVRAM, kad maršrutizatorių perkraunant, visi pokyčiai išliktų.

```
A#copy running-config startup-config
```

IGRP protokolas

IGRP yra išimamas iš apyvartos (nuo 2005 m. kuriamose IOS nebenaudojamas) vidinis maršrutizavimo protokolas (Interior Gateway Routing Protocol - IGRP), kurį Cisco Systems, Inc sukūrė 1980 metų viduryje. Patobulinta, neklasinė IGRP versija yra EIGRP. IGRP protokolas, parinkdamas paketui optimaliausią kelią, atsižvelgia į keletą parametrų: informacijos nešlio dažnių juostos plotį (bandwidth), delsą (delay), apkrovą (loading) ir patikimumą (reliability). Nors metrikos grupėje dažnai minimas maksimalus vienu kadru perduodamos informacijos kiekis (MTU – Maximum Transmission Unit), jis nėra metrikos parametras). Standartiškai protokolas atsižvelgia tik į juostos plotį ir delsą, kadangi kiti parametrai yra kintami ir įgyja prasmę tik tuomet, kai visame tinkle vykdomas tų parametrų monitoringas. Administratorius gali reguliuoti svorius, kuriais yra įskaitomi visi keturi parametrai, tačiau tai rekomenduojama daryti tik su profesionalų pagalba. Tinklo diametras praktiškai neribojamas (255 šuoliai).

Kadangi IGRP, kaip ir RIP, yra atstumo vektorinis protokolas ir turi daug bendrų bruožų, žemiau išvardinsime tik pagrindinius.

Periodiniai atnaujinimai. – Kas 90 s. kaimyniniams maršrutizatoriams yra siunčiami atnaujinimai – suminė maršrutizavimo lentelės informacija.

Transliuojami atnaujinimai - Atnaujinimai yra siunčiami transliaciniu numeriu visiems prie segmento prijungtiems kaimynams.

Visos maršrutizavimo lentelės atnaujinimas – Nors IGRP protokolas, kaip yra būdinga atstumo vektoriniams protokolams, esant pokyčiams, siunčia periodinius atnaujinimus. Jis standartiškai siunčiamas kas 90 sek. kaimynams siunčia visą savo maršrutizavimo lentelę.

Skaiciavimas iki begalybės – Jeigu maršrutizavimo lentelėje esančio kelio metrika padidėja, maršrutizatorius daro prielaidą, kad tai kitas maršrutizatorius pakoregavo kelią, todėl šis kelias yra pašalinamas iš maršrutizavimo lentelės.

Atkirstas horizontas – IGRP protokolas naudoja šią taisyklę taip –atnaujinimus jis siunčia visiems segmento kaimynams transliaciniu adresu, todėl tiki, kad visi tikrai gavo atnaujinimus ir nesitiki išsiųstą informaciją gauti iš kaimynų atgal (tai pažeistų atkirto horizonto taisyklę).

Proginiai atnaujinimai su kelių perskaičiavimu (triggered updates with route poisoning) – IGRP siunčia atnaujinimą, vos tik pastebėjęs pokyčius tinkle. Jeigu kelias pradingsta, jam nustatoma šuolių skaičius kaip nepasiekiamam. Taip kelias yra „užnuodijamas“ ir išvengiama kilpų atsiradimo.

Apkrovos balansavimą – Standartiškai balansuoja apkrovą keturiais skirtingais keliais(gali sugrupuoti skirtingos metrikos kelius) ir šį skaičių komanda *maximum-paths maximum-path* galima padidinti iki šešių.

Numatytieji keliai (default route) – Nors IGRP protokole galima sukonfigūruoti numatytąjį kelią, protokolas pažymi ir kandidatą į numatytąjį kelią. Geriausias iš kandidatų tampa numatytuoju keliu.

Bellman Ford maršrutizavimo algoritmas – Šis algoritmas sudaro atstumo vektorinių protokolų branduolį. IGRP jis yra modifikuotas, naudojant sudėtinę metriką, o komanda *variance* galima reguliuoti apkrovos balansavimą nevienodos metrikos keliais.

IGRP yra klasinis protokolas (classfull), nes atnaujinimuose jis operuoja priimančio maršrutizatoriaus prievado potinklio maskuote, sumuoja iki pilnų, nepadalintų A, B, C klasės tinklų bei nepraneša informacijos apie potinklio maskuotę SM.

Kadangi IGRP konfigūravimas sutampa su EIGRP protokolo konfigūravimu, todėl ties šiuo punktu neapsistosisime.

EIGRP protokolas

EIGRP (Enhanced Interior Gateway Protocol) - Patobulintas vidinis maršrutizavimo protokolas (Cisco privatus, t. y. nestandartizuotas), konverguojantis žymiai greičiau, nei IGRP. Jį galima vadinti tiek atstumo vektoriniu protokolu, tiek hibridiniu. EIGRP protokolui savitumą suteikia jame naudojamos technologijos:

- kaimyniniams santykiams užmegzti naudoja hello paketus;
- svarbiems paketams garantuotai pristatyti naudoja savo patikimą transporto protokolą RTP;
- DUAL baigtinių būsenų mašina išrenka optimaliausią, kilpų neturintį kelią į kiekvieną tinklą;
- protokolas turi modulius, nepriklausomai aptarnaujančius IP, AppleTalk ir Novell NetWare tinklus.

EIGRP lentelės ir terminologija

EIGRP protokolas pasižymi savita terminologija. EIGRP protokolas informaciją laiko trijose lentelėse: kaimynų, topologijos ir maršrutizavimo. Norėdami suprasti įrašų jose prasmę, turime žinoti terminologiją.

Pirmoji lentelė vadinama **kaimynų lentele** (neighbor table). Maršrutizatoriai privalo reguliariais intervalais su kaimynais keistis hello paketais. Visa informacija, kurią maršrutizatoriai surenka iš **hello** paketų, yra saugoma kaimynų lentelėje. Jeigu hello yra gaunamas, tai reiškia, kad kaimynas dar „gyvas“. Jeigu šie paketai pradingsta, tai po tam tikro laiko intervalo (hold time) DUAL yra informuojamas apie topologijos pasikeitimą. Taigi, kaimynų lentelėje yra informacija apie kaimyninius maršrutizatorius (su kuriais yra užmegzti kaimyniniai santykiai) ir sąsajas, per kurias pas tuos kaimynus galima nusiųsti informaciją.

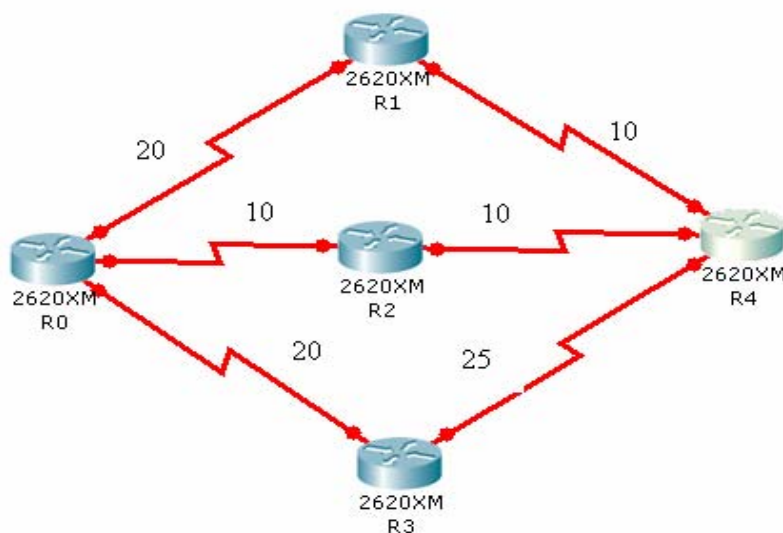
Kita lentelė yra EIGRP **topologijos lentelė**. Joje yra saugoma visų kaimyninių maršrutizatorių atsiųstuose atnaujinimuose išgirsta informacija apie visus jiems žinomus tinklus. Topologijos lentelėje saugomi ne tik patys geriausi keliai, kurie atrenkami į maršrutizavimo lentelę ir kuriais dabar siunčiami paketai, bet ir rezerviniai keliai, bei visi kiti keliai. Kiekvienam tiksliniam adresui

topologijos lentelėje yra nurodomi parametrai FD/AD (Feasible Distance/Advertised Distance). Kas tai yra?

Deklaruojamas atstumas AD (Advertised Distance) reiškia atstumą nuo sekančio šuolio maršrutizatoriaus iki tikslo. Parametras FD yra sekančio šuolio maršrutizatoriaus AD plius atstumas nuo lokalaus maršrutizatoriaus iki sekančio šuolio maršrutizatoriaus. Jeigu yra keletas skirtingos vertės kelių iki tikslo, tai mažiausios vertės kelias FD vaidina išskirtinį vaidmenį – jis nustato ne tik dabartinį perėmėją (pigiausias kelias), bet ir tikėtiną perėmėją.

Dabartinis perėmėjas (angl. Current Successor) yra sekančio šuolio maršrutizatorius, turintis pigiausią kelią be kilpų. Tikėtinas (atsarginis) perėmėjas (Feasible Successor) yra atsarginis maršrutizatorius, turintis kelią be kilpų ir kurio deklaruojamas atstumas AD yra mažesnis už dabartinio perėmėjo FD. Jeigu šis atstumas yra lygus, tai maršrutizatorius nėra tikėtinas perėmėjas. (Skaitytojo tegul nepainioja naudojami sinonimai vertė, atstumas ir metrika. Visi šie žodžiai EIGRP reiškia tą patį – metriką, kuri EIGRP yra skaičiuojama lygiai pagal tą pačią formulę, kaip IGRP protokole, tik su daugikliu 255.)

Pažiūrėkime, kaip šią teoriją pritaikyti realioje situacijoje. 8 pav. yra pavaizduota maršrutizatorių



8 pav. Tinklo su EIGRP schema: FD ir AD parametrų analizė.

sistema ir skaičiais nurodytos kelių vertės. Nuo R0 iki R4 turime tris skirtingos vertės kelius. Jų vertės yra atitinkamai lygios 30, 20 ir 45. Vidurinysis kelias yra geriausias, nes jo vertė mažiausia:

$FD=10+10=20$. Taigi, R2 maršrutizatorius yra dabartinis perėmėjas. Tuo tarpu deklaruojamus AD parametrus turime tris. Juos skelbia sekančio šuolio maršrutizatoriai: R1 – $AD=10$, R2 – $AD=10$ ir R3 – $AD=25$. Kadangi R3 skelbiamas AD yra didesnis už dabartinio perėmėjo $FD=20$, tai R3 nėra tikėtinas perėmėjas. Tuo tarpu R1 yra tikėtinas perėmėjas, nes jo deklaruojamas $AD=10$ yra mažesnis už dabartinio perėmėjo $FD=20$. Mažiausios vertės kelias bus įrašytas į maršrutizavimo lentelę, tuo tarpu topologijos lentelėje bus visi keliai.

IP tinkluose Cisco IOS standartiškai srautą balansuoja ne daugiau kaip keturiais vienodos vertės keliais ir ši skaičių komanda **maximum-paths** *maximum-path* galima padidinti iki šešių. Skaičių *maximum-path* prilyginus 1, balansavimas vienodos metrikos keliais yra išjungiamas. Jeigu keliai per R1 ir R2 būtų vienodos vertės, tai EIGRP balansuotų tik R0-R1-R4 ir R0-R2-R4 keliais. Kadangi jų vertės nevienodos, tai EIGRP visą srautą nukreips pigiausiu R0-R1-R4 keliu. Jeigu norime balansuoti visais skirtingos vertės keliais (angl. unequal load - balance), tai turime naudoti komandą.

```
Router(config-router)#variance daugiklis.
```

Tada yra balansuojama visais keliais, kurių FD yra mažesni už dabartinio perėmėjo FD, padauginą iš *daugiklio*. Šią sąlygą tenkina visi trys keliai, bet R0-R3-R4 kelias nedalyvaus vis tiek nedalyvaus, nes jis nėra tikėtinas perėmėjas. Kada naudojame **variance** komandą, tada į maršrutizavimo lentelę yra įrašomi visi balansavime naudojami keliai, ne vien dabartinis perėmėjas.

EIGRP topologijos lentelė

EIGRP protokolas topologijos lentelėje saugo informaciją apie visus jam žinomus kelius. Joje yra:

- kiekvieno adresato patys geriausi keliai (Successors), informacija apie kuriuos patenka į maršrutizavimo lentelę;
- kiekvieno adresato atsarginiai keliai (Feasible Successors), informacija apie kuriuos yra tik topologijos lentelėje;
- visi kiti keliai, neatitinkantys atsarginiams keliams keliamų reikalavimų (Feasibility Condition).

Žemiau yra pateikta EIGRP protokolo topologijos lentelė, kurią netrukus paaiškinsime.

R1#show ip eigrp topology all-links

IP EIGRP Topology table for AS(65001)/ID (172.20.10.1)

Codes: P - Passive, A - Active, U - Update, Q – Query,
r - reply Status, s - sia Status

P 172.20.3.128/25, 2 successors, FD is 30720, semo 9

via 172.20.10.2 (30720/28160), FastEthernet0/1

via 172.20.3.2 (30720/28160), FastEthernet0/3

P 10.69.0.0/24, 1 successors, FD is 156160, semo 16

via 172.20.3.2 (156160/128256), FastEthernet0/3

via 172.20.10.2 (153720/156160), FastEthernet0/1

P 172.20.10.0/24, 1 successors, FD is 28160, semo 1

via Connected, FastEthernet0/1

P 172.20.10.0/30, 1 successors, FD is 20514560, semo 15

via 172.20.1.1 (20514560/205122000), FastEthernet0/2

via 172.20.10.2 (20517120/20514560), FastEthernet0/1

P 172.20.1.0/24, 1 successors, FD is 28160, semo 2

via Connected, FastEthernet0/2

P 172.20.2.0/24, 1 successors, FD is 30720, semo 8

via 172.20.10.2 (30720/28160), FastEthernet0/1

via 172.20.3.2 (33280/30720), FastEthernet0/1

P 172.20.3.0/25, 1 successors, FD is 20160, semo 3

via Connected, FastEthernet0/3

R1#

Pirmojoje šios lentelės eilutėje esantis įrašas AS(65001)/ID (172.20.10.1) rodo, kad EIGRP procesas yra aktyvus autonominėje srityje AS65001 ir nurodo maršrutizatoriaus identifikatorių 172.20.10.1. Sekančioje eilutėje nurodytos abreviatūros, nusakančios kelio būseną. Įrašas

P 172.20.3.128/25, 2 successors, FD is 30720, semo 9

via 172.20.10.2 (30720/28160), FastEthernet0/1

via 172.20.3.2 (30720/28160), FastEthernet0/3

reiškia, kad į tinklą 172.20.3.128/25 yra du geriausi vienodos vertės $FD=30720$ keliai. Vienas kelias yra per FastEthernet0/1 prievadą ir toliau - į sekančio šuolio maršrutizatoriaus prievadą, kurio IP adresas yra 172.20.10.2. Skliausteliuose (30720/28160) esantys skaičiai yra (FD/AD). Matoma, kad sekančio šuolio maršrutizatorius yra arčiau tikslo. Sekančioje eilutėje matome, kad kitas kelias yra identiškos vertės. Į tikslą srautas iškeliauja per FastEthernet0/3 prievadą ir sekančio šuolio metu patenka į sekančio maršrutizatoriaus prievadą 172.20.3.2. Jeigu yra du pigiausi, identiškos vertės keliai, į maršrutizavimo lentelę bus įrašyti abu, ir, jei tik maršrutizatoriuje nėra sukonfigūruota komanda `maximum-paths 1`, srautas bus balansuojamas abiem šiais keliais.

Iš žemiau esančio įrašo

P 10.69.0.0/24, 1 successors, FD is 156160, semo16
via 172.20.3.2 (156160/128256), FastEthernet0/3
via 172.20.10.2 (153720/156160), FastEthernet0/1,

matome, kad į tinklą 10.69.0.0/24 turimi du nevienodos vertės keliai. Kelias per FastEthernet0/3 prievadą yra geresnis, nes, kaip rodo įrašas (156160/128256), jo $FD=156160$ yra mažiausias. Būtent ši metrika yra matoma eilute aukščiau “FD is 156160”. Antrojoje eilutėje “via 172.20.10.2 (153720/156160) FastEthernet0/1” yra informacija apie prastesnį kelią. Einant į 10.69.0.0/24 tinklą per FastEthernet0/1 prievadą, sekančio šuolio maršrutizatorius deklaruoja $AD=156160$ ir tai sutampa su dabartinio perėmėjo FD (pirmojoje eilutėje). Taigi, šis kelias nėra tikėtinas perėmėjas. Į maršrutizavimo lentelę bus įrašytas pirmasis kelias ir, jeigu tik jis išnyks, tai DUAL turės šį kelią pažymėti ne **P** raide, o **A** (Active), rodančia, kad yra ieškoma naujo geriausio kelio.

Įrašas

P 172.20.10.0/24, 1 successors, FD is 28160, semo 1
via Connected, FastEthernet0/1

skelbia, kad tinklas 172.20.10.0/24 yra tiesiogiai prijungtas prie FastEthernet0/1 prievado. Visi kiti įrašai topologijos lentelėje yra komentuojami analogiškai.

Maršrutizavimo lentelėje yra saugomi geriausi keliai. Jais yra perduodama visa informacija. Žemiau pateiktas maršrutizavimo lentelės pavyzdys. Pirmajame stulpelyje esančios raidės rodo ar tinklas, esantis antrajame stulpelyje: C - yra tiesiogiai prijungtas, D – yra EIGRP vidinis tinklas, D EX – yra „gautas“ iš kitų protokolų. Tuojau už galinio tinklo adreso yra metrika [90/2681856]. Pirmasis

skaičius yra EIGRP protokolo administracinis atstumas 90, o sekantis skaičius 2681856 yra dabartinio perėmėjo FD. Primename, kad topologijos lentelėje šis skaičius buvo pirmojoje pozicijoje (FD/AD). Jeigu kelias yra žinomas iš kitų protokolų, jo administracinis atstumas yra 170.

```
R1#show ip route
```

```
<praleista išvestis>
```

```
C      10.1.1.0 is directly connected, Serial0
```

```
D      172.16.1.0 [90/2681856] via 10.1.1.1, Serial0
```

```
D EX 192.168.1.0 [170/2681856] via 10.1.1.1, 00:00:00, Serial0
```

EIGRP metrika

EIGRP protokole metrika yra skaičiuojama pagal formulę

$$\text{Metrika} = [(K1 * BW) + [(K2 * BW) / (256 - \text{apkrova})] + (K3 * \text{delsa})] * [K5 / (\text{patikimumas} + K4)]$$

Standartinės koeficientų vertės yra $K1=1$, $K2=0$, $K3=1$, $K4=0$, $K5=0$ ir yra perdušamos hello pakete. Jeigu dviejų maršrutizatorių K vertės nesutampa, jie negali tapti kaimynais ir dėl to negali pasikeisti topologijos lentelės duomenimis, taigi, dėl to maršrutizavimo lentelės nėra atnaujinamos potencialių kaimynų žinomais keliais.

Formulėje delsa yra lygi delsų sumai kelyje, išreikštai dešimtimis mikrosekundžių (1 milisekundė = 1000 mikrosekundžių).

$$\text{Dažnių juostos plotis} = (10^7 / (\text{minimalus dažnių juostos plotis kelyje, kbps})) * 256.$$

Jeigu naudojamas nuoseklusis prievadas, tada, prievado lygmenyje panaudojus komandą

```
R1(config-if)#bandwidth kilobitai
```

galima nurodyti, kokį dažnių juostos plotį naudoti (pats dažnių juostos plotis linijoje nesikeičia!) EIGRP protokolui renkant kelius. Jeigu tokios komandos nėra, tai EIGRP protokolas mano, kad nuosekliojo prievado dažnių juostos plotis yra 1.544 Mbps.

`R1(config-if)#ip bandwidth-percent eigrp proceso_ID kanalo dalis_%`

nurodo EIGRP protokolui, kokią dažnių juostos pločio dalį jis gali naudoti atnaujinimams siųsti. Standartiškai EIGRP protokolas savo reikmėms naudoja ne daugiau kaip pusė dažnių juostos pločio, o kita pusė naudojama vartotojo informacijai siųsti.

EIGRP naudoja dar vieną parametą – MTU – maksimalų viename kadre perduodamos informacijos kiekį. Šis parametras nedalyvauja, skaičiuojant metriką, bet jis yra svarbus keičiantis informacija su kitais protokolais.

EIGRP pasižymi tokiomis savybėmis:

- Esant stabiliam tinklui EIGRP naudoja labai mažai resursų; maršrutizavimo duomenų bazėms palaikyti naudoja tik hello paketus.
- Atnaujinimai, informuojantys apie pokyčius tinkle yra siunčiami tik tada, kai jie atsiranda.
- Pirmojo atnaujinimo metu EIGRP siunčia visą maršrutizavimo lentelę, o vėliau - tik inkrementinius atnaujinimus, kuriuose yra tik informacija apie pokyčius. EIGRP kaimynų lentelę sudaro tik hello paketų pagalba, kurie pastoviai leidžia kontroliuoti, ar kaimynas „dar gyvas“ ir kanalas veikia. Jeigu per nustatytą laiko tarpą neišgirstama keleto hello paketų, maršrutizavimo procesas kitiems kaimynams siunčia atnaujinimą, skelbdamas apie pokytį tinkle.

EIGRP paketų tipai

EIGRP protokolas naudoja penkių tipų paketus, iš kurių tris siunčia, naudodamas savą patikimą transporto protokolą RTP (Reliable Transmission Protocol) protokolą:

- **hello** – skirtas užmegzti kaimyninius santykius ir yra mechanizmas, kuriuo nustatoma, ar kaimynas „pasiekiamas“. Šie paketai siunčiami grupiniu adresu ir nereikalauja patvirtinimo. Hello paketuose yra informacija apie maršrutizatoriaus IP adresą, K vertes ir kt. Jeigu K vertės nesutampa, tai kaimynais netampama ir maršrutizatoriai negali apsikeisti topologijos lentelės duomenimis ir maršrutizavimo lentelės neatnaujinamos kaimynų turimais duomenimis.
- **atnaujinimas** (update) – naudojamas atnaujinimams siųsti. Naudoja RTP. Atnaujinimai siunčiami tik suinteresuotiems kaimynams grupiniu, arba transliaciniu numeriu.
- **užklausa** (query) – skirtas kaimynų užklausiai maršrutizavimo informacijos. Naudoja RTP.

- **atsakymas** (reply) – atsakymui į užklausimą apie maršrutizavimo informaciją. Naudoja RTP.
- **patvirtinimas** (ACK - acknowledgement) – patikimo paketo patvirtinimui (paketo, naudojančio RTP)

EIGRP konfigūravimas

EIGRP protokolas konfigūruojamas komandomis:

```
R1(config)#router eigrp 123  
R1(config-router)#network 10.1.0.0  
R1(config-router)#network 172.16.2.0 0.0.0.255
```

Pirmoji komanda apibrėžia EIGRP kaip IP protokolą ir nustato jo proceso ID (autonominės srities numerį) 123. Visi maršrutizatoriai, kurie turi keistis maršrutizavimo duomenimis, privalo būti toje pačioje autonominėje srityje. Antroji komanda deklaruoja tiesiogiai prijungtus tinklus, kurie dalyvauja EIGRP procese. Tinklo adresai gali būti naudojami tiek su pakaitos simboliais, tiek be jų. Pakaitos simbolių bitai turi panašią prasmę, kaip ir prieigos sąrašuose (ACL), t.y. kaip interpretuoti adresą: vienetai reiškia atitikimą, o nuliai – atitinkami tinklo adreso bitai gali būti bet kokie. Jeigu norime, kad tik konkretus prievadas (ne tinklas) dalyvautų EIGRP procese, už IP adreso naudotini pakaitos simboliai **0.0.0.0**, o jeigu norime, kad procese dalyvautų maršrutizatoriaus visi prievada ir prijungti tinklai – reikia naudoti junginį **0.0.0.0 255.255.255.255**.

Komanda ip default-network

EIGRP protokole komanda `ip default-network tinklo-adresas` apibrėžia arba numatytąjį kelią (default route), arba paskutinės vilties kelią (gateway of last resort). Parametras *tinklo-adresas* turi būti pilnos klasės tinklas, pvz.: 10.0.0.0, 173.15.0.0, 193.100.12.0. Kadangi maršrutizatorius, kuriame ši komanda yra sukonfigūruota, paminėtą tinklą atnaujinimuose skelbs kaimynams kaip paskutinės vilties kelią, šis tinklas turi būti pasiekiamas iš duotojo maršrutizatoriaus.

OSPF protokolas

OSPF (Open Shortest Path First – Atviras pirmo trumpiausio kelio nustatymo protokolas) yra vidaus slenkstinis protokolas, t.y. jis užtikrina informacijos apie maršrutizavimo galimybes sklaidą tarp maršrutizatorių, priklausančių tai pačiai autonominei sistemai. OSPF, kaip atviros architektūros protokolą 1987-89 m. sukūrė IETF (Internet Engineering Task Force) grupė. IP v4 tinkluose dabar naudojama antroji OSPF versija, specifikuota RFC 2328.

OSPF protokolas buvo kuriamas norint išspręsti maršrutizavimo uždavinius dideliuose, plečiamuose tinkluose, kur paprastesni maršrutizavimo protokoliai tapo neefektyvūs, t.y. siekiant:

- paspartinti konvergavimą (didesniuose tinkluose RIP konvergencija gali siekti iki kelių minučių);
- rasti efektyvesnį kelio parinkties metodą (RIP įvertina tik šuolių (mazgų) skaičių iki tikslo tinklo, bet neatsižvelgia į užlaikymus bei kanalo parametrus): OSPF naudoja kanalo vertės (cost), atvirkščiai proporcingos juostos pločiui, metriką;
- išnaudoti kintamo ilgio potinklio maskuotės (VLSM) galimybes;
- padidinti maršrutizuojamų tinklų aprėptį (RIP ribojamas 15 šuolių);
- efektyviau išnaudoti tinklo kanalų pralaidą (RIP kas 30 s siunčia atnaujinimus kaimynams – pilnas maršrutų lenteles);
- užtikrinti tinklo plečiamumą.

Kadangi OSPF yra pilnai standartizuotas, tai jis dažniausiai naudojamas tiek dideliuose vietiniuose tinkluose, tiek Interneto paslaugų teikėjų. OSPF yra paremtas ryšio būsenos stebėseną (link state technology) ir sparčiai reaguoja į pokyčius tinkle.. Ryšio (link) sąvoka OSPF naudojama kaip sinonimas prievado sąvokai. Iš tiesų, jei dėl kokių nors priežasčių ryšys išeina iš rikiuotės, prievadas pereina į neaktyvią būseną ir atvirkščiai. Pokyčiams atsiradus, protokolas siunčia sužadintus atnaujinimus, o jeigu jų nebūna – tada kas 30 min. (palyginkime: RIP – kas 30 s) siunčia periodinius atnaujinimus (link-state refresh).

OSPF vertinamas kaip patikimas maršrutizavimo protokolas – jo administracinis atstumas (nusako maršruto šaltinio patikimumą) yra 110 ir yra geresnis negu RIP ar IS-IS.

OSPF protokolui būdinga hierarchinė struktūra. Pagrindinė sritis, šerdis, vadinama magistraline sritimi. Protokolo konfigūracijoje ji visada žymima kaip nulinė sritis (area 0). Prie magistralinės srities ABR (Area Border Router) maršrutizatoriumi yra jungiamos visos kitos sritys, vadinamos reguliariomis (yra ir kitokio tipo sričių, bet jos nepatenka į mūsų nagrinėjamų klausimų ratą). ABR maršrutizatorius turi atskiras LSDB ir magistralinei sričiai ir prijungiamai sričiai. Srautas, keliaujantis iš vienos zonos į kitą, būtinai turi praeiti per ABR maršrutizatorių, kuris magistralinėje srityje turi bent vieną prievadą. Hierarchinė struktūra garantuoja OSPF protokolo stabilumą, nes LSA paketų lavinos yra sulaikomos ties srities riba – dėl topologijos pokyčių vienoje srityje nereikia perskaičiuoti maršrutizavimo lentelių kitose srityse. Be to, atskirai paimtos srities maršrutizatoriams jau nebereikia turėti detalių žinių apie kitų sričių tinklus. Tarp sričių užtenka keistis tik tam tikra

sumine informacija ir standartiniais keliais. Taigi, hierarchinė sričių struktūra paspartina OSPF protokolo konvergavimą, lokalizuoja nestabilumus sričių ribose ir pagerina bendrą tinklo funkcionavimą, užtikrina efektyvias tinklo plečiamumo galimybes bei sumažina jo administracines išlaidas. Pažymėtina, kad netgi tada, kai neplanuojama hierarchinė tinklo struktūra ir tinklas neskaidomas į sritis, siekiant išlaikyti vieningą protokolo formą, būtina bent jau 0-linė sritis.

Pagrindiniai OSPF elementai

OSPF protokolo informacija perduodama specializuotais **ryšio būsenos paketais** (Link-State Packets – LSP). Įvairiuose maršrutizavimo etapuose OSPF naudojami 5 LSP tipai:

- **Hello** Hello paketai naudojami nustatant kaimyninius bei gretutinius (adjacent) maršrutizatorius, skelbiant OSPF maršrutizatorių parametrus, renkant išskirtąjį maršrutizatorių DR, bei atsarginį išskirtąjį BDR maršrutizatorių. Hello paketai grupiniu adresu 224.0.0.5 siunčiami visais OSPF procese dalyvaujančiais prievadais.
- **DBD...DBD** yra duomenų bazės aprašo (DBD - database Descriptor) paketas, kuriuo perduodamas sutrumpintas siunčiančiojo maršrutizatoriaus ryšio būsenos DB sąrašas, kurį gaunantysis maršrutizatorius sulygina su savąja ryšio būsenos DB.
- **LSR...** LSR yra užklauskos (Link-State Request) paketas, kuriuo reikalaujama papildomos informacijos apie DBD įrašą, kurio nėra gaunančiojo maršrutizatoriaus DB.
- **LSU** LSU yra ryšio būsenos papildymo (Link-State Update) paketas. Tai gali būti atsakas į LSR, perduodantis pilną užklausoje nurodyto ryšio būsenos informaciją, arba naujos ryšio būsenos informacijos skelbimas. LSU turi iki 10 skirtingų ryšio būsenos skelbimų (LSA Link-State Advertisement) tipų. LSU ir LSA terminų naudojimas (ir skirtumai tarp jų) yra gana komplikotas ir kai kada jie ne visai tiksliai naudojami kaip sinonimai.
- **LSAck** LSAck yra LSU gavimo patvirtinimas.

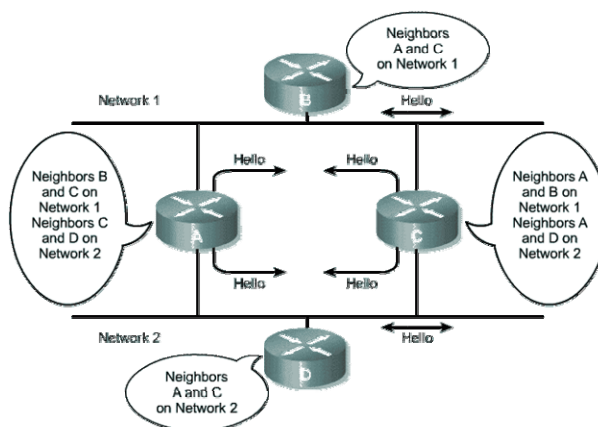
LSP tipai nurodomi LSP antraštės tipo lauke: 1-Hello, 2-DBD, 3-LSR, 4-LSU, 5-LSAck.

OSPF protokolo informacija – Ryšio būsenos paketai – perduodami tiesiogiai IP paketuose, nenaudojant TCP ar UDP inkapsuliacijos, IP paketo antraštėje įvardijant OSPF kaip 89 protokolą. OSPF pranešimai platinami grupiniais (multicast) adresais 224.0.0.5 (DROthers adresas) arba 224.0.0.6 (DR ir BDR adresas) bei su pastaraisiais susietais MAC adresais Ethernet tinkle 01-00-5E-00-00-05 ir 01-00-5E-00-00-06 (23 jaunesnieji bitai iš IP adreso tiesiog kopijuojami į grupinį MAC).

Bet kokio pasikeitimo informacija apie ryšio būsenas pradžioje visi OSPF procese dalyvaujantys maršrutizatoriai pirmiausia apibrėžia savo kaimynus ir nustato gretimybes - gretutinius (adjacent) maršrutizatorius. Kaimyniniai maršrutizatoriai yra visi tame pačiame potinklyje esantys maršrutizatoriai (gautas Hello paketas su maršrutizatoriaus identifikatoriumi (ID) nurodo, kad ID apibrėžtas įrenginys yra kaimynas). Gretutiniais vadinami maršrutizatoriai, kurie sinchronizuoja tarpusavyje savo duomenų bases.

Kaimyninio maršrutizatoriaus statusas į gretutinio keičiamas dviem etapais (žr. 9.pav):

- 2 žingsnių ryšio užmezgimas (Two-way communication), naudojant **Hello** paketus;
- Duomenų bazės sinchronizavimas, pasiekiamas pasikeičiant DB deskriptorių (Database Description – DD), Ryšio būsenos užklauso (Link State Request – LSR) ir Ryšio būsenos atnaujinimo (Link State Update – LSU) paketais.



9 pav. Kaimyninių santykių užmezgimas OSPF protokole.

Kaimyninių OSPF maršrutizatorių būsenos yra apibrėžiamos taip:

Down	Prievadas negauna Hello paketų
Attempt	Kaimynai turi būti aprašyti statišškai; naudojamas NBMA topologijoje.
Init	Prievadas gauna Hello paketus iš kaimynų.
2Way	Iš kaimynų gautuose Hello paketuose yra įtrauktas gaunančiojo maršrutizatoriaus ID.

ExStart Nustatyti pagrindinis/pagalbinis ryšiai, siekiant suformuoti gretimybes, apsieičiant DB deskriptorių (DD) paketais. (Maršrutizatorius su didesniu ID tampa pagrindiniu.)

Exchange Pasikeičiama maršrutizavimo informacija naudojant DD ir LSack.

Loading Siunčiami LSR reikalaujant atnaujinimo informacijos – LSA ir juos gaunant bei patvirtinant.

Full LSA informacija yra pilna ir sinchronizuota tarp gretutinių maršrutizatorių.

Beje, prieš pradedant šiuos procesus, turi būti nustatyti trys svarbūs parametrai: Atnaujinimų (Hello) periodas bei laukimo (Dead) intervalas ir tinklo tipas². Tinklo segmente šie parametrai turi griežtai sutapti.

Pagal nutylėjimą transliacinės bendrosios prieigos (pvz., Ethernet) ir dvitaškio sujungimo topologijų tinkluose Hello paketai siunčiami kas 10 s, o netransliacinės bendrosios prieigos tinkluose (NBMA, pvz., Frame Relay) - kas 30 s. Jei per laukimo intervalą negaunamas Hello paketas, yra deklaruojamas linijos trūkis. Laukimo intervalas paprastai būna lygus 4-iems atnaujinimo periodams ir yra 40 ir 120 s anksčiau įvardintiems tinklų tipams atitinkamai. Jei laukimo intervalo metu Hello paketas negaunamas, OSPF pašalina tokį kaimyną iš ryšio būsenos DB, ir maršrutizatorius per visus savo prievadus išplatina LSA apie ryšio nutrūkimą.

Siekiant sumažinti informacijos srautus bendrosios prieigos tinkluose, OSPF naudoja išskirtąjį DR bei atsarginį išskirtąjį BDR maršrutizatorius, kurie veikia kaip tarpiniai srauto paskirstytojai: gretimybės nustatomos ne tarp kiekvieno potinklio maršrutizatoriaus, bet tik tarp DR/BDR ir kitų (vadinamų DROther) įrenginių, taigi, informacija apie ryšio būseną (jos pokyčius) siunčiama ne visiems kaimynams, bet tik DR ir BDR; DR savo ruožtu išplatina informaciją visiems segmento maršrutizatoriams (DROthers) grupiniu adresu 224.0.0.5 (DR ir BDR grupinis adresas yra 224.0.0.6). BDR tik priima informaciją ir seka DR būseną. Pastarajam išėjus iš rikiuotės, BDR perima DR funkcijas.

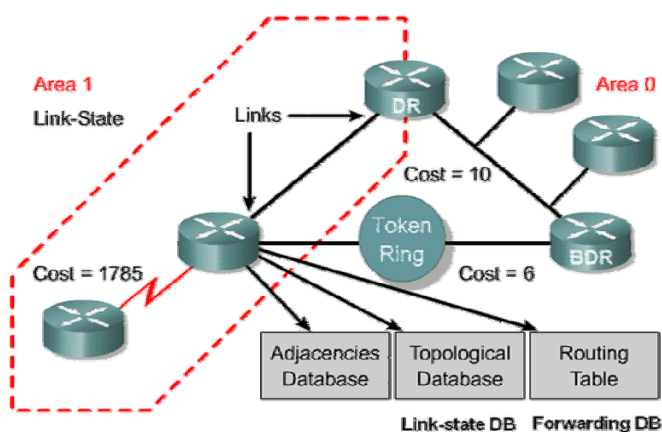
OSPF algoritmas

Siekiant užtikrinti efektyvų maršrutizavimo mechanizmą, OSPF naudoja tris duomenų bazes (žr. 10 pav.): gretimybų (adjacencies) DB, topologijos arba ryšio būsenos (topology or link-state)

² OSPF gali sėkmingai dirbti įvairiuose tinkluose, tačiau tinklo topologija įtakoja tam tikrus protokolo ypatumus, todėl derinant (konfigūruojant) jo darbą, skiriami keli režimai, sąlygoti transliacinės bendrosios prieigos (pvz., Ethernet), netransliacinės bendrosios prieigos (NBMA) (pvz., Frame Relay) ir dvitaškio sujungimo topologijų.

DB ir persiuntimo (forwarding) DB arba maršrutizavimo lentelę (routing table):

■ **Kaimynų lentelė** (Neighbor table, adjacency database), kurioje yra laikoma informacija apie atpažintus kaimynus. Jeigu maršrutizatorius neatsiunčia Hello paketo per laukimo intervalą, yra deklaruojamas linijos trūkis ir pradedamas griūtinio užtvindymo LSA su šiuo pranešimu procesas tinkle. Kaip ir EIGRP protokole, OSPF protokole kaimyninių santykių užmezgimas yra esminis – be jų maršrutizatorius nedalyvauja procese.



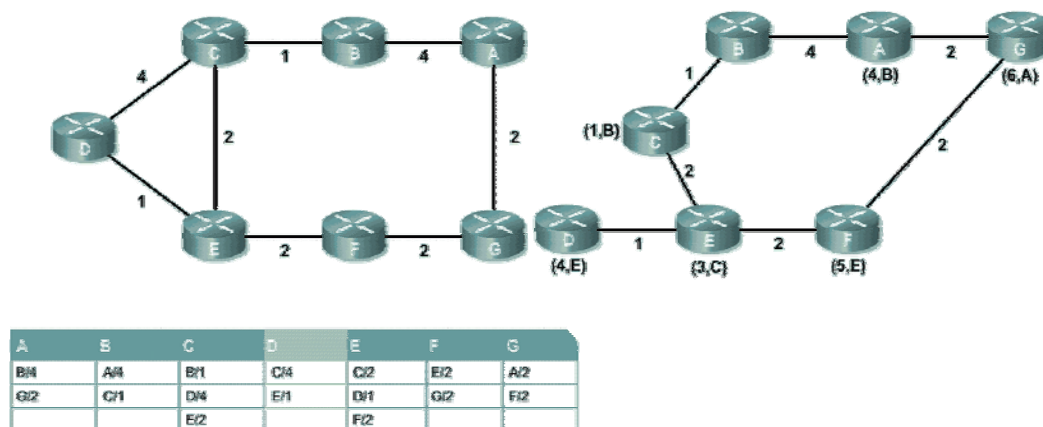
10 pav. Pagrindiniai OSPF protokolo objektai.

■ **Topologijos lentelėje**, kitaip vadinamoje ryšio būsenos DB LSDB, yra visų srities ar tinklo maršrutizatorių ir prie jų prijungtų ryšio kanalų sąrašas ir jų parametrai. Visi srities maršrutizatoriai turi identiškas LSDB, tačiau kiekvienas „išaugina“ skirtingą SPF medį, nes kiekvienas jį pradeda iš savo taško.

■ **Maršrutizavimo lentelėje**, dar vadinamoje persiuntimo lentele, yra pačių geriausių kelių į paskyrimo tinklus sąrašas. Kiekvienam maršrutizatoriui ji yra skirtinga. Maršrutizavimo lentelė yra Dijkstros algoritmo darbo su LSDB įrašais rezultatas. Kadangi srities maršrutizatoriai turi patikimus duomenis apie kelių į duotosios srities tinklus vertę, kiekvienas savarankiškai išrenka pigiausią kelią be kilpų į kiekvieną paskirties tinklą srityje.

Kiekvienas maršrutizatorius gauna ryšio būsenos informaciją iš gretutinių įrenginių (LSP) ir taip

sudaro savo ryšio būsenos DB. Tada, naudojant Dijkstra trumpiausio kelio algoritmą (SPF)³, sukuriamas SPF medis, kurio pagalba ir užpildomos maršrutizavimo lentelės (11 pav.).



11 pav. SPF medis.

11 pav. a dalyje pateikto tinklo maršrutizatoriams suformavus topologijos DB (visuose maršrutizatoriuose ji vienoda ir pateikta, įvardijant kaimynus ir kelio vertę iki jų, pvz., D turi jungtį su C (vertė = 4) ir su E (vertė = 1)) pagal SPF algoritmą apskaičiuojami trumpiausi keliai. Pavyzdyje pateikti skaičiavimai iki maršrutizatoriaus B. C pasiekia jį tiesiogiai, vertė 1. E pasiekia B per C, vertė 3. D yra pasiekia per E ir C, vertė 4. Kelias iki kitų maršrutizatorių randamas analogiškai. Taip gaunama stabili, be maršrutizavimo kilpų topologija. Apkrovos subalansavimui maršrutų lentelėje galimi 6 lygiaverčiai keliai į paskirties tinklą (numatytoji vertė yra 4).

Kada ryšio kanalo būseną pakinta, pirmasis pokytį aptikęs maršrutizatorius LSA (Link State Advertisement) paketą grupiniu adresu siunčia kaimynams (dvitaškėje topologijoje) arba DR/BDR (pastarieji replikuoja jį DROther's). Kiekvienas maršrutizatorius LSA turinį nusikopijuoja į savo ryšio kanalų duomenų bazę LSDB (Link-State Data Base) ir papildomai išplatina paketą per visus kitus prievadus srities maršrutizatoriams. Taip LSA paketai yra lavina perduodami visiems maršrutizatoriams ir garantuojama, kad visų maršrutizatorių LSDB yra atnaujintos, prieš jiems atspindint pakitusią tinklo topologiją maršrutizavimo lentelėse.

³ SPF algoritmą 1959 m. sukūrė olandų mokslininkas Edsger Dijkstra

OSPF maršrutizatoriaus ID

Kiekvienas OSPF procese dalyvaujantis maršrutizatorius įvardijamas identifikatoriumi ID. Jei yra bent vienas aktyvus (up) grįžtamo ryšio (loopback) prievadas, maršrutizatoriaus ID priskiriamas didžiausias iš šių prievadų adresas, nors fizinių prievadų adresai ir būtų didesni. Kitu atveju ID yra didžiausias aktyvaus fizinio prievado IP adresas, nesvarbu, kuriam prievadui jis priklausytų, netgi jei prievadas nedalyvauja OSPF procese. Grįžtamo ryšio prievadus patartina naudoti todėl, kad jie visada lieka gyvybingi (juos gali išjungti tik administratorius), tuo tarpu, dingus ryšiui, sutrikdoma fizinio prievado veikla, todėl naudojant jo adresus, ID gali nepageidautinai pasikeisti.

Esant reikalui, maršrutizatoriaus ID galima pakeisti komanda **router-id** (ši komanda galima tik naujesnėse operacinių sistemų versijose, Cisco IOS –12.0(T) ir vėlesnėse):

```
Router (config-router)#router-id ip-address
```

Paminėtina, kad čia svarbu ir eiliškumas: maršrutizatoriaus ID nustatomas, vos tik aprašius pirmąjį tinklą komanda **network**, todėl, jei komanda **router-id** ar grįžtamo ryšio prievadas nustatomi vėliau, ID jau bus parinktas pagal prievadų adresus ir siekiant, kad būtų priimtos paskesnių nustatymų vertės, teks išvalyti OSPF procesą (Router#**clear ip ospf proceso_ID**) arba net perkrauti maršrutizatorių. ID būseną galima pasitikrinti komandomis

```
Router#show ip protocols
```

```
Router#show ip ospf
```

```
Router#show ip ospf interface
```

OSPF DR ir BDR nustatymas

Kaip jau paminėta anksčiau, siekiant sumažinti informacijos srautus bendrosios prieigos tinkluose, OSPF naudoja išskirtąjį DR bei atsarginį išskirtąjį BDR maršrutizatorius, kurie veikia kaip tarpiniai srauto paskirstytojai: gretimybės nustatomos ne tarp kiekvieno potinklio maršrutizatoriaus, bet tik tarp DR/BDR ir kitų (vadinamų DROther) įrenginių, taigi, informacija apie ryšio būseną (jos pokyčius) siunčiama ne visiems kaimynams, bet tik DR ir BDR; DR savo ruožtu išplatina informaciją visiems segmento maršrutizatoriams.

DR ir BDR skiriami tik bendrosios prieigos tinkluose ir nenaudojami dvitaškėse topologijose. Jie išrenkami, apsieičiant Hello paketais pagal maršrutizatoriaus ID – maršrutizatorius su didžiausiu ID

tampa DR, sekantis pagal dydį – BDR. Maršrutizatorius, netapęs DR ar BDR yra įvardijamas kaip DROther. Administratorius gali valdyti šį procesą, keisdamas maršrutizatoriaus ar jo prievado OSPF prioritetą – pagal nutylėjimą jis yra lygus vienetui visuose maršrutizatoriuose, bet gali kisti nuo 0 iki 255. Didžiausias prioritetas lemia maršrutizatoriaus skyrimą DR. Nulinis prioritetas nurodo, kad maršrutizatorius jokių būdu negali tapti DR ar BDR. DR ir BDR renkami atskiruose potinkliuose, todėl tas pats maršrutizatorius gali būti DR viename potinklyje ir DROther kitame. Tinkle atsiradus naujam įrenginiui su didesniu prioritetu (ar didesniu ID, jei prioritetai vienodi), nei DR, nei BDR neperrenkami. DR išėjus iš rikiuotės, jo funkcijas perima BDR ir pradamas naujo BDR nustatymo procesas, kaip ir išėjus iš rikiuotės pačiam BDR.

OSPF metrika

OSPF metrika yra įvardinta kaip **vertė** (cost), galinti kisti nuo 1 iki 65535, ir yra atvirkščiai proporcinga kanalo (išeinančio prievado) perdavimo terpės juostos plotiui:

$$\text{vertė} = (\text{santykinis juostos plotis} = 100 \text{ Mbps}) / (\text{juostos plotis})$$

Bendra maršruto vertė gaunama sumuojant kanalų vertes iki paskirties tinklo. Esant būtinybei, pvz., esant spartesnėms nei 100 Mbps linijoms, vertę galima pakeisti, modifikuojant juostos plotį arba santykinį juostos plotį:

```
Router(config-if)#bandwidth bandwidth-kbps
```

```
Router(config-router)#auto-cost reference bandwidth reference-bandwidth
```

Pastaruoju atveju reikėtų komandą naudoti visuose maršrutizatoriuose, kad metrika visur būtų adekvati. Pagaliau, kanalo vertę galima tiesiog nurodyti:

```
Router(config-if)#ip ospf cost 10
```

Pastarasis būdas ypač rekomenduotinas nehomogeniškoje tinklo aplinkoje, kai naudojama įvairių gamintojų įranga.

OSPF diegimas

OSPF aktyvuojamas komanda

```
Router(config)#router ospf process-id
```

Proceso identifikatorius *process-id* yra administratoriaus parinktas skaičius nuo 1 iki 65535, nusakantis bendrą OSPF domeną. Komanda **network** aprašytą tinklo adresą atitinkantis prievadas, panašiai kaip ir atstumo vektoriaus protokoluose, aktyvuojamas gauti ir siųsti OSPF paketus, o pats tinklas ar potinklis bus įtrauktas į maršrutizavimo atnaujinimus.

```
Router(config-router)#network network-address wildcard-mask area area-id
```

Skirtingai nei RIP, OSPF reikalaujama nurodyti potinklį. Tam naudojami pakaitos simboliai (kai kur vadinami inversine maskuote) (*wildcard-mask*), pvz., potinklui 10.10.10.1/26, inversinė maskuotė bus 0.0.0.63. Kai kurios IOS leidžia aprašyti potinklį potinklio maskuote (mūsų pavyzdyje – 255.255.255.192), kaip ir nurodant prievado adresą. Sritis (*area*) yra OSPF įrenginių, kurie dalinasi ryšio būsenų informacija ir turi vienodas topologijos bazes, grupė. Nors ir nadrinėjame tik vienos srities atvejį ir hierarchinės struktūros nėra, OSPF procese privalo būti įvardinta bent jau nulinė (magistralinė) sritis.

Numatytojo maršruto deklaravimas

Kaip ir EIGRP ar RIP, maršrutizatorius, sujungtas su Interneto paslaugų teikėju ISP, gali būti panaudotas, siekiant deklaruoti maršrutą į neištrauktus į OSPF procesą tinklus. Toks įrenginys vadinamas ribiniu, įėjimo ar šliuziniu. OSPF terminologijoje maršrutizatorius, jungiantis OSPF domeną ir tinklus, nepriklausančius OSPF, vadinamas ribiniu autonominės sistemos maršrutizatoriumi (Autonomous System Boundary Router, ABSR). Tada, jei ryšys su ISP užfiksuotas per S0 prievadą, srautas į ISP nukreipiamas standartine komanda

```
Router (config)#ip route 0.0.0.0 0.0.0.0 S0 ,
```

o pastarasis kelias skelbiamas OSPF domene

```
Router (config-router)#default-information originate
```

■ **Maršrutizavimo lentelėje**, dar vadinamoje perdavimo lentele, yra pačių geriausių kelių į paskyrimo tinklus sąrašas. Kiekvienam maršrutizatoriui ji yra skirtinga. Maršrutizavimo lentelė yra Dijkstros algoritmo darbo su LSBDD įrašais rezultatas. Kiekvienam paketui, patenkančiam į duotąjį maršrutizatorių, geriausias kelias parenkamas pagal įrašus maršrutizavimo lentelėje.

Naudodami Edsger Dijkstra algoritmą, dar žinomą SPF algoritmo (Shortest Path First) vardu, kiekvienas maršrutizatorius LSDB esančių įrašų pagalba suformuoja savo SPF medį, kurio pagrinde vaizduoja patį save. Šio medžio mažiausios vertės kelias į dominantį tinklą yra kopijuojamas į maršrutizavimo lentelę. OSPF protokolo atveju maršrutizatoriai viską sužino „iš pirmų lūpų“, o ne remiasi tuo, ką pasakė kaimynas, kaip kad yra RIP ir IGRP protokoluose.

OSPF protokolui būtina hierarchinė struktūra. Pagrindinė sritis, šerdis, vadinama magistraline sritimi. Protokolo konfigūracijoje ji visada žymima kaip nulinė sritis (area 0). Prie magistralinės srities ABR (Area Border Router) maršrutizatoriumi yra jungiamos visos kitos sritys, vadinamos reguliariosiomis (yra ir kitokio tipo sričių, bet jos nepatenka į mūsų nagrinėjamų klausimų ratą). ABR maršrutizatorius turi atskiras LSDB ir magistralinei sričiai ir prijungiamai sričiai. Srautas, keliaujantis iš vienos zonos į kitą, būtinai turi praeiti per ABR maršrutizatorių, kuris bent vieną prievadą turi magistralinėje srityje. Hierarchinė struktūra garantuoja OSPF protokolo stabilumą, nes LSA paketų lavinos yra sulaikomos ties srities riba – dėl topologijos pokyčių vienoje srityje nereikia perskaičiuoti maršrutizavimo lentelių kitose srityse. Be to, atskirai paimtos srities maršrutizatoriams jau nebereikia turėti detalias žinias apie kitų sričių tinklus. Tarp sričių užtenka keistis tik tam tikra sumine informacija ir standartiniais keliais. Taigi, hierarchinė sričių struktūra sumažina OSPF protokolo administracines išlaidas, pagreitina jo konvergavimą, lokalizuoja nestabilumus sričių ribose ir pagerina bendrą tinklo funkcionavimą.

Prieigos kontrolės sąrašai

Prieigos kontrolės sąrašai (toliau, ACL – Access Control List) sudaro kompiuterių tinklo prieigos saugumo bazę. Kontrolės sąrašų pagalba maršrutizatoriai gali filtruoti paketus, tarkime, blokuoti Interneto prieigą, arba uždrausti pasirinkto potinklio vartotojams naudotis tam tikru protokolu. IP ACL'ai dar gali būti naudingi maršrutizatorių atnaujinimams filtruoti, patikrinti sąlygas paketams skiriant prioritetus ar įdiegiant aptarnavimo kokybės elementus (QoS).

ACL'ą sudaro nuosekli seka **permit** ir **deny** teiginių, taikomų šaltinio ar paskirties punktų adresams, protokolams bei jų portams. ACL'ą galima parašyti bet kuriam maršrutizuojamam protokolui - IP, IPX Apple Talk ir t.t. Prieigos kontrolės sąrašų skaičių riboja prievadų skaičius, kryptys ir sukonfigūruoti protokolai. Tai reiškia, kad pasirinktam maršrutizatoriaus prievadui galime parašyti vieną IP protokolo ACL'ą, kuris filtruotų išeinantį srautą ir vieną, kuris filtruotų įeinantį srautą. Įeinantis srautas maršrutizatoriaus prievade yra filtruojamas prieš maršrutizavimo lentelės peržiūrą, tuo tarpu išeinantis srautas filtruojamas po maršrutizavimo lentelės peržiūros, prieš paketui paliekant prievadą.

Prieigos kontrolės sąrašo logika

Cisco ACL'o pagrindą sudaro nuosekli seka teiginių, kurie yra išdėstomi griežta tvarka, pradedant detaliesiais (viršuje) ir baigiant pačiais bendriausiais teiginiais, rašomais ACL'o gale. ACL'o logiką sudaro du pagrindiniai veiksmai: atitikimo (matching) ir veiksmo (action). Palyginimo logika analizuoja kiekvieną paketą ir nustato, ar paketas atitinka ACL'o teiginio sąlygą. Jeigu sąlygą atitinka, tai paketas arba praleidžiamas (permit), arba filtruojamas (deny). Tolimesni teiginiai nėra tikrinami. Jeigu paketas teiginio sąlygos neatitinka, tai tikrinamas sekantis teiginys. Jeigu paketas netenkina nė vieno ACL'o teiginio sąlygos (pasiekiamas ACL'o galas), tai paketas yra atmetamas, t.y. ACL'o pabaigoje visada yra neišreikštinis (nerašomas) deny įrašas. Taigi, ACL'o logiką galime trumpai išreikšti keliais etapais:

1 etapas Access-list teiginio atitikimo parametrai palyginami su paketo parametrais.

2 etapas Jeigu parametrai atitinka, tai yra vykdoma šiame ACL'o teiginyje nustatyta operacija (**permit** arba **deny**).

3 etapas Jeigu 2 etape parametrai neatitinka, 1 ir 2 etapai yra kartojami, naudojant sekantį ACL'o teiginį.

4 etapas Jeigu atitikimas neaptiktas, patikrinus visus ACL'o teiginius, tai paketas yra atmetamas.

Ši logika vienoda ir standartiniams, ir išplėstiniais ACL'ams. Palyginimo parametrai ACL'o teiginiuose yra pagrįsti IP, TCP ir UDP protokolų preambulėse esančiais laukais. Išplėstiniai ACL'ai greta kitų galimų laukų gali tikrinti ne tik siuntėjo ir gavėjo IP adresus, bet ir siuntėjo ir gavėjo protokolus, portus. Tuo tarpu standartiniai ACL'ai gali tikrinti tik siuntėjo IP adresą.

Pakaitos simbolių šablonai

Pakaitos simbolių šablonas leidžia nurodyti, kurią IP adreso dalį turi tikrinti ACL'o teiginys. Būtina atkreipti dėmesį, kad, skirtingai nuo potinklio maskuotės ar inversinę potinklio maskuotę atitinkančių pakaitos simbolių, naudojamų maršrutizavimo protokoluose, ACL naudojami pakaitos simboliai gali būti ne tik vieno laipsto funkcija, bet įgyti bet kokias reikšmes. Tarkime, kad maršrutizatoriaus Fa0 prievadas gavo paketą, kuriame yra nurodytas IP adresas 172.164.10.33, o maršrutizatoriuje yra sukonfigūruotas ACL'as, per Fa0 prievadą praleidžiantis tik paketus, skirtus 172.164.10.0/24 potinkliui. Šią situaciją išreikškime dvejetainėje sistemoje:

10101100.10100100.00001010. 00100001	172.164.10.33
00000000.00000000.00000000. 11111111	0.0.0.255

Apatinėje eilutėje yra užrašytas pakaitos simbolių šablonas (wildcard mask). Jame nuliais pažymėti tie IP adreso bitai, kurie yra tikrinami ir privalo sutapti su nurodytais ACL'e , o vienetais - netikrinami bitai.

Skirtingai nuo pakaitos simbolių, naudotų aprašant tinklus OSPF, kurie sudaromi tvarkingais blokais, atitinkančiais potinklų ribas pradžioje yra tik nuliai, o po to – tik vienetai (dėl šios priežasties pakaitos simboliai dažnai vadinami inversine potinklio maskuote), ACL'o pakaitos simbolių šablonas, pagal poreikį, gali būti sudarytas iš bet kokios nulių ir vienetų sekos. Pastaruoju atveju ACL'ą analizuoti bus labai sunku, ypač, jeigu jame bus dar keliolika ar kelios dešimtys papildomų teiginių. Dėl šios priežasties pakaitos simbolių šablonai sudaromi tvarkingais blokais, atitinkančiais potinklų ar tinklų ribas - pradžioje yra tik nuliai, o gale – tik vienetai. Parašykime keletą pakaitos simbolių šablonų didėjančia tvarka:

Pakaitos simbolių šablonų pavyzdžiai

Pakaitos simb. šablonas	Dvejetainė šablono versija	Aprašymas
0.0.0.0	00000000.00000000.00000000.00000000	Tikrinami visi IP adreso bitai
0.0.0.255	00000000.00000000.00000000.11111111	Tikrinami tik pirmieji 24 bitai
0.0.255.255	00000000.00000000.11111111.11111111	Tikrinami tik pirmieji 16 bitai
0.255.255.255	00000000.11111111.11111111.11111111	Tikrinami tik pirmieji 8 bitai
255.255.255.255	11111111.11111111.11111111.11111111	Visi bitai gali būti bet kokie
0.0.15.255	00000000.00000000.00001111.11111111	Tikrinami tik pirmieji 20 bitai
0.0.3.255	00000000.00000000.00000011.11111111	Tikrinami tik pirmieji 22 bitai

Parašykime pakaitos simbolių šabloną, atitinkantį duotąjį potinklio šabloną (Subnet Mask). Primename, kad potinklio šablono dvejetainėje formoje vienetai žymi bitus, skirtus tinklams ir potinklams numeruoti, o nuliai – mazgams. Vienoje eilutėje parašykime duotąjį potinklio šabloną, tarkime 255.255.252.0, dvejetainė forma, o apačioje - atvirkščią šabloną, t.y. gautą, potinklio šablone sukeitus nulius ir vienetus vietomis:

11111111.11111111.11111100.00000000	255.255.252.0/22
00000000.00000000.00000011.11111111	0.0.3.255

Apatinėje eilutėje gavome pakaitos simbolių šabloną 0.0.3.255. Ryšys tarp šių dviejų objektų yra akivaizdus. Viršutinėje eilutėje nuliai žymi bitus, skirtus mazgams numeruoti, o apatinėje eilutėje – netikrinamus bitus.

Prieigos kontrolės sąrašo konfigūravimas

Prieigos kontrolės sąrašas konfigūruojamas dviem etapais. Pirma parašomas kontrolės sąrašas, po to jis susiejamas su reikiamu prievadu. Kontrolės sąrašas veikia tik tada, kai yra įvykdyti abu punktai. Pirmiausia reikia žinoti tikslą: ar reikia filtruoti (ar praleisti) visus tam tikrų šaltinių paketus, ar reikia blokuoti prieigą tam tikrais protokolais prie tam tikrų šaltinių. Pirmuoju atveju reikia naudoti standartinį IP ACL'ą, kuris tikrina tik siuntėjo IP adresą. Antruoju atveju yra reikalingas išplėstinis IP ACL'as, turintis didesnę tikrinamų parametrų skaičių.

Standartiniai prieigos kontrolės sąrašai

Tarkime, kad norime parašyti ACL'ą, kuris per maršrutizatoriaus Fa0 prievadą išorės kryptimi praleistų tik paketus, skirtus potinklui 172.167.10.0/24. Kadangi protokolai neminimi, reikia naudoti standartinį

ACL'ą ir jam suteikti numerį, priklausantį intervalams 1 – 99, 1300 – 1999:

R1(config)# access-list 1 permit 172.164.10.0 0.0.0.255	kontrolės sąrašas
R1(config)# interface Fa0	
R1(config-if)# ip access-group 1 out	sąrašo susiejimas su prievadu

Kontrolės sąrašas šiuo atveju yra sudarytas iš vienos viršutinės eilutės. Po jos yra neišreikštinė neigimo eilutė **deny**, kuri niekada nerašoma. Trečiojoje eilutėje esančia komanda ACL'as yra susiejamas su FA0 prievadu. Komandos gale esantis žodelis **out** reiškia, kad bus tikrinamas srautas, einantis per Fa0 prievadą išorės kryptimi. Priešingos krypties (in) srautas tikrinamas nebus.

Standartinį ACL'ą turime susieti ne su šaltinio, o su vieno iš paskyrimo adresų, kurį norime užblokuoti. Kadangi standartinis ACL tikrina tik šaltinio IP adresą, tai jį susieję su šaltinio prievadu, potencialiai užblokuotume srautą visiems potencialiems gavėjams.

Prieigos kontrolės sąraše teiginiai turi būti išdėstyti griežta tvarka, pradedant pačiais detaliausiais ir baigiant pačiais bendriausiais. Norėdami įprasminti šį reikalavimą, išnagrinėkime kiek sudėtingesnę standartinio ACL'o atvejį:

```
Sw(config)#access-list 99 deny 172.164.10.15 0.0.0.0
Sw(config)#access-list 99 permit 172.164.10 0.0.0.255
Sw(config)#access-list 99 deny 172.164.0 0.0.255.255
Sw(config)#access-list 99 deny 172.0.0 0.255.255.255
```

Pirmasis teiginys yra detaliausias, nes jame kalbama apie konkretų mazgą. Jeigu paketas neatitiks pirmojo teiginio sąlygos, bus tikrinama antrojo teiginio sąlyga. Jeigu jiniai bus išpildyta, bus vykdoma instrukcija permit, o sekantys teiginiai nebebus tikrinami. Dabar tarkime, kad žemiausią teiginį access-list 99 deny 172.0.0.0 0.255.255.255 įrašėme pirmuoju. Jame esančią tikrinimo sąlygą atitiks visi paketai, gaunami iš tinklų 172.0.0.0, o žemiau esantys detalesni teiginiai niekada nebus tikrinami ir srautas iš potinklio 172.164.10 visada bus filtruojamas.

Pirmojoje eilutėje esantį pakaitos simbolių šabloną 0.0.0.0 prieigos kontrolės sąraše galima pakeisti žodeliu **host**, rašomu prieš IP adresą, t.y. vietoje teiginio **access-list 99 deny 172.164.10.15 0.0.0.0** galima rašyti **access-list 99 deny host 172.164.10.15**. Panašiai, teiginį **access-list 99 deny 172.164.10.15 255.255.255.255** galima pakeisti trumpesniu **access-list 99 deny any** (visi IP adresų bitai gali būti bet kokie, vadinasi tai yra bet koks IP adresas).

Išplėstiniai prieigos kontrolės sąrašai

Pagrindinis išplėstinių prieigos kontrolės sąrašų (Extended ACL) skirtumas nuo standartinių yra didesnis tikrinamų parametrų skaičius.

Išplėstiniame IP prieigos kontrolės sąraše yra tikrinami šie parametrai:

- Šaltinio IP adresas
- Šaltinio IP adreso dalis, panaudojant pakaitos simbolių šabloną
- Paskirties IP adresas
- Paskirties IP adreso dalis, panaudojant pakaitos simbolių šabloną
- Protokolo tipas (TCP, UDP, ICMP, IGRP, IGMP ir kt.)
- Šaltinio TCP/UDP portas.
- Paskirties TCP/UDP portas.
- Established – atitinka visi TCP srautai, išskyrus pirmąjį – sinchronizacijos.
- IP Paslaugos tipas (Type of Service)
- IP pirmumas (IP precedence)

Išplėstiniai IP ACL'ai numeruojami skaičiais iš intervalų 100 – 199, 2000 – 2699. Tam, kad būtų įvykdytas veiksmas **permit** arba **deny**, teiginyje turi atitikti visi tikrinami parametrai. Jeigu bent viena sąlyga nėra išpildyta, teiginys yra praleidžiamas ir pereinama prie sekančio teiginio. Parametrų išdėstymo tvarka juose yra labai svarbi. Visi parametrai, kurie liečia šaltinį, rašomi iš karto už šaltinio IP adreso, o kurie liečia paskyrimo punktą – už paskyrimo IP adreso. Išnagrinėkime vieną išplėstinį prieigos sąrašą.

```
R1(config)#access-list 101 permit ip 192.168.14.0 0.0.0.15 172.16.2.0 0.0.0.255  
R1(config)#access-list 101 deny tcp 192.168.14.0 0.0.0.255 any eq 80  
R1(config)#access-list 101 ip any any  
R1 (config-if)#ip access-group 101 in
```

Pirmoji eilutė leidžia potinklio 192.168.14.0/28 vartotojams visais protokolais susisiekti su 172.16.2.0 potinkliu. Antroji 101 prieigos sąrašas eilutė draudžia visam 192.168.14.0 tinklui prisijungti prie Interneto, naudojantis www protokolu. Trečioji eilutė praleidžia visą TCP/IP protokolų steką. Šis prieigos sąrašas yra susietas su vienu iš prievadų. Kaip ir standartinio prieigos sąrašo atveju, pirmasis teiginys yra detaliausias.

Išplėstinis prieigos sąrašas yra susiejamas su prievadu, esančiu kuo arčiau šaltinio.

Komutavimo pagrindai

Primename, kad bendrosios prieigos tinkluose viename kolizijų ir transliaciniame domene yra daugybė mazgų ir bet kurio mazgo siunčiamą transliacinį kadrą apdoroja visi transliacinio domeno mazgai. Kolizijų domene bet kurie du vienu metu transliuojantys mazgai sukelia koliziją ir ją visi pagal žinomą algoritmą privalo apdoroti, taigi, visi mazgai dažnių juostos plotį dalijasi tarpusavyje. Mazgai yra verčiami dirbti pusiau duplexiniu režimu – arba tik kalbėti, arba tik klausyti.

Situaciją bendrosios prieigos tinkluose daugeliu aspektų pagerina komutatoriai. Jau vien komutavimo funkcija gali sumažinti tinklo perkrovas (congestion) ir padidinti dažnių juostos plotį, jeigu komutatoriui vieta tinkle yra tinkamai parinkta ir jis yra tinkamai sukonfigūruotas.

Komutatoriaus sugebėjimų visuma priklauso ir nuo jo tipo ir nuo jame instaliuotos operacinės sistemos versijos. Šiame skyriuje susipažinsime su pačiais pagrindinėmis taisyklėmis, kuriomis vadovaujasi komutatorius, perduodamas kadrus, siejančio medžio protokolo (STP) funkcionavimo pagrindais ir virtualiųjų tinklų (VLAN) konfigūravimu su ryšiu tarp atskirų VLAN'ų.

Komutatorių funkcijos

Bet kuris antrojo OSI lygmens įrenginys (komutatorius, tiltas), perduodantis kadrus, atlieka tokias pagrindines funkcijas.

- MAC adresus sužino, gautų kadrų preambulėje perskaitęs esančius šaltinių MAC adresus (source address).
- Sudaro ir prižiūri MAC adresų ir su jais susijusių komutatoriaus ar tilto prievadų lentelę.
- Transliaciniai ir grupiniai (multicast) kadrai yra perduodami per visus prievadus (išskyrus prievadą, per kurį kadras buvo gautas).
- Nežinomiems paskyrimams skirti kadrai yra perduodami per visus prievadus (išskyrus prievadą, per kurį kadras buvo gautas).

■ komutatoriai ir tiltai eliminuoja komutavimo kilpas (bridging loops), panaudodami siejančio medžio protokolą (STP – Spanning Tree Protocol).

Šios funkcijos apibūdina kaip skaidrų tiltą (transparent bridge), taip 2 – jo lygmens komutatorių. Tarp tilto ir 2 – jo lygmens komutatoriaus yra tokie maži skirtumai, kad komutatorių galima įsivaizduoti kaip daug tiltų vienoje vietoje. Be to, tiltas sprendimą perduoti ar neperduoti kadrą priima programiškai, o komutatorius gali turėti eilę prievadų, kuriuose komutavimo funkcija atliekama aparatiškai optimizuota įranga – specialiosios paskirties mikroschemomis (ASIC – Application – Specific Integrated Circuit). ASIC kadrus komutuoja praktiškai be delsos.

Kadras yra perduodamas, perskaičius jame esantį paskirties mazgo MAC adresą. Šį adresą komutatoriui arba reikia nurodyti statiškai komandine eilute, prievado lygmenyje mazgo MAC adresą susiejant su atitinkamu prievadu ir VLAN'u, arba leisti komutatoriui jį sužinoti pačiam dinamiškai. Pastaruoju atveju komutatorius nagrinėja per prievadus ateinančių kadrų antraštes ir, perskaitęs juose šaltinio MAC adresus, juos įsirašo į MAC adresų lentelę, kartu įsirašydamas ne tik prievadą, per kurį nustatė duotąjį MAC adresą, bet ir VLAN'ą, kuriam priklauso tas prievadas. Taigi, MAC adresų lentelė informuoja, kokie MAC adresai priskirti tam tikriems prievadams ir kuriems VLAN'ams priklauso. Jeigu komutatorius randa MAC adresą, kurio jis dar neturi savo MAC adresų lentelėje, ši yra papildoma nauju MAC adresu, kartu nurodant ir atitinkamo prievado, ir VLAN'o numerį.

Įeinančių kadrų preambulėse yra ir paskyrimo punktų MAC adresai, pagal kuriuos komutatorius MAC adresų lentelėje suranda juos atitinkančius prievadų ir VLAN'ų numerius. Jeigu toks MAC adresas randamas (known address), kadras yra perduodamas per atitinkamą prievadą, jeigu ne (unknown address) – perduodamas per visus prievadus (flooding), priklausančius šaltinio VLAN'ui, išskyrus prievadą, per kurį kadras buvo gautas. Transliacinius kadrus ir kadrus, skirtus grupei žinomų adresatų (multicast destination address) komutatorius pagal apibrėžimą privalo perduoti per visus prievadus.

Kai kadras patenka į komutatorių, jis kopijuojamas į prievado įėjimo eilę, kurioje kartu gali būti ir skirtingą prioritetą (QoS – Quality of Service) turintys kadrai. Svarbesni kadrai aptarnaujami greičiau. Aptarnavęs kadrą šioje eilėje, komutatorius turi žinoti ne tik kur perduoti kadrą, bet ir ar jį iš viso reikia perduoti ir kaip jį perduoti. Visus šiuos tris dalykus komutatorius sužino vienu metu, šioms operacijoms naudodamas optimizuotą komutavimo įrangą – CAM (Content Addressable Memory) ir TCAM (Ternary Content Addressable Memory). CAM atmintyje yra MAC adresų lentelė, dar vadinama komutavimo lentele. Paskyrimo mazgo MAC adresas yra naudojamas kaip raktas į CAM slypintį atsakymą – išėjimo prievadą. TCAM atmintyje (Ternary CAM, ternary reiškia trigubą) kompiliuota forma saugoma prieities

sąrašai ACL ir perdavimo kokybės sąrašai (QoS ACL). Apie ACL'us kalbėsime sekančiuose skirsniuose, dabar tik pasakysime, kad jie realizuoja saugumo politiką ir filtruoja kadrus, atsižvelgiant į eilę parametrų, tokių kaip MAC adresai, IP adresai, protokolai, 4 lygmens portai. TCAM tuo tarpu sprendžia QoS klausimus. Abiejų tipų atmintys peržiūrimos vienu metu ir atsakymas gaunamas vienos „žvilgsnio“ metu.

Nepaisant visų komutatoriaus privalumų, jis tinkle atlieka tam tikras funkcijas ir jos nustato komutatoriaus panaudojimo ribas. Komutatorių tinklas negali būti plečiamas be galo, kadangi komutatoriai transliacinius kadrus perduoda per visus savo prievadus, taip suformuodami labai didelius transliacinius domenų. Todėl tampa aišku, kad, naudodami vien tik komutatorius, negalime turėti didelio, plečiamo tinklo.

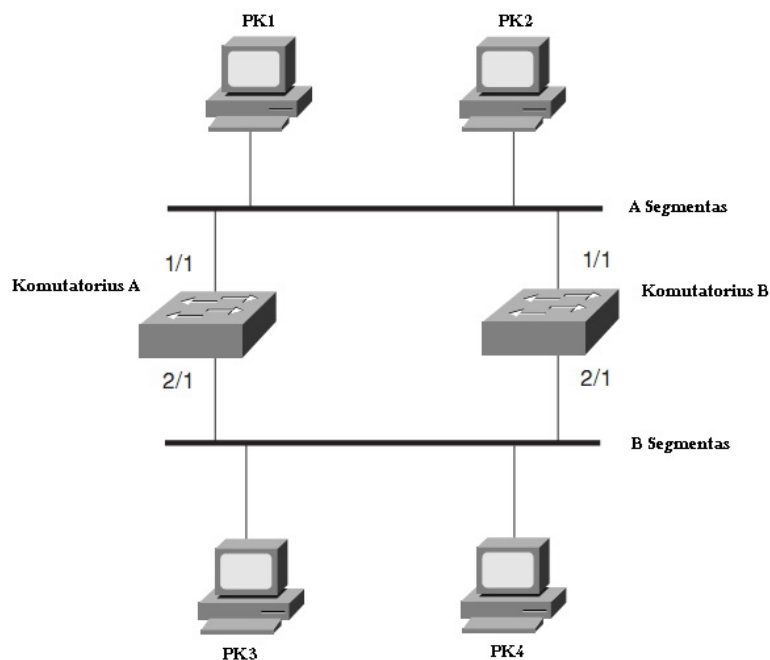
Yra reikalingas aukštesnio lygio įrenginys, kuris galėtų apriboti transliacinių kadrų perdavimą. Toks prietaisas OSI trečiojo lygmens prietaisas maršrutizatorius. Jo funkcijas aptarsime sekančiame skyriuje.

Siejančio medžio protokolas (STP)

Kritiškuose tinklo segmentuose yra būtini atsarginiai keliai. Išėjusius iš rikiuotės kelius pakeičia kiti. Trečiojo lygmens požiūriu tai yra savaime suprantamas dalykas, juk visas Internetas sudarytas tokiu principu, kad atskiri vieno failo paketai į paskyrimo punktą gali nukeliauti labai skirtingais keliais. Tarp maršrutizatorių esančios kilpos problemų nesukelia, nes kelio paieškos protokolai su kilpomis moka susidoroti. Tam jie taiko atitinkamas kovos su kilpomis technologijas. Jeigu koks paketas ir pagaunamas į maršrutizavimo kilpą, tai joje neužsibūna per ilgai, nes paketo preambulėje esantis TTL (Time To Live) skaitiklis fiksuoja praeitų maršrutizatorių skaičių ir kai šis pasiekia tam tikrą nustatytą ribinę vertę (kiekvienam protokolui skirtingą), paketas yra atmetamas.

Antrajame lygmenyje kadras savo preambulėje panašios paskirties lauko neturi, todėl kadras gali be galo ilgai cirkuliuoti kilpoje ir komutatorius niekada nesupras, kad šis paketas jam, gal būt, jau šimtą kartų matytas. Taigi, situacija yra prieštaringa: tinklo stabilumas reikalauja turėti atsarginius kelius, bet šie neišvengiamai sudaro 2 – jo lygmens kilpas (bridging loops).

Norėdami geriau įsivaizduoti, kaip susidaro kilpos, nagrinėkime kadrų kelią tinkle, kurio schema pateikta 12 pav.



12 pav. Kilpa komutatorių sistemoje.

Tarkime, kad abu komutatoriai nieko nežino apie PK1 ir PK4. Patys kompiuteriai žino vienas kito adresus, todėl jie gali normaliai inkapsuliuoti informaciją ir suformuoti kadrus. Mazgas PK1 siunčia kadrą PK4 mazgui per tinklo segmentą A. Toliau besirutuliojančius įvykius sekime etapais.

1 etapas. Abu komutatoriai kadrą gauna į 1/1 prievadą. Kadangi komutatoriai apie PK1 išgirdo pirmąjį kartą, abu jie PK1 MAC adresą susieja su 1/1 prievadu, padarydami įrašus MAC adresų lentelėse. Nuo dabar jie jau žino, kad PK „gyvena“ segmente A, todėl, jeigu ateityje gaus kadrą, adresuotą mazgui A, tikrai galės jį pristatyti adresatui.

2 etapas. Kadangi abu komutatoriai nežino, kur prijungtas PK4, jie kadrą privalo išsiųsti per visus savo prievadus, tikėdamiesi, kad PK4 atsilieps ir komutatoriai taip sužinos, kuriame segmente yra PK4.

3 etapas. Kiekvienas komutatorius per prievadą 2/1 perduoda kadrą segmentui B ir PK4 gauna dvi to paties kadro kopijas. Jeigu šių kopijų daugiau niekas negautų, viskas tuo ir pasibaigtų, nes, jeigu PK4 mazgas atsilieptų, komutatoriai PK4 atsaką kaip mat išgirstų ir jo MAC adresą įsirašytų į savo MAC adresų lenteles. Bet kadro kopijas gauna ir abu komutatoriai į 2/1 prievadus.

4 etapas. Abu komutatoriai mato, kad PK1 mazgas, prijungtas prie 1/1 prievadų, dabar girdisi iš kito segmento. Jie nusprendžia, kad PK1 buvo perkeltas į segmentą B. Jie panaikina senąjį įrašą ir PK1 susieja

su 2/1 prievadais.

5 etapas. Šiame etape nė viena komutatorius dar nežino, kuriame segmente yra PK4, kadangi nė vienas negavo PK4 siųsto kadro, kuriame jo MAC adresas būtų įrašytas siuntėjo adreso lauke. Todėl, tikintis rasti PK4 dislokacijos vietą, naujas kadras yra vėl siunčiamas per visus prievadus. Taip per komutatorių 1/1 prievadus kadras vėl grįžta į A segmentą.

6 etapas. Abu komutatoriai iš naujo sužino, kad PK1 yra prijungtas prie 1-jo segmento ir viskas kartojasi iš naujo.

Kaip matome, vienas konkrečiam adresatui skirtas kadras buvo pagautas kilpoje. Netrukus pradinis kadras ir jo kopija ėmė cirkuliuoti kilpa priešingomis kryptimis. Kiekvieno apsisukimo metu PK4 mazgas gauna po du naujus kadrus. Turint galvoje komutatoriaus pajėgumus, tokioje situacijoje PK4 mazgas turi daug darbo.

Komutatorių kilpos sukelia keletą nepageidautinų reiškinių:

■ **Transliacinius kamščius** (broadcast storm). Jeigu, tarkime, PK1 (žr. pav.3) kompiuteris pasiuntė transliacinį kadrą į 1 –ąjį segmentą, tai jis patenka į komutatorių 1/1 prievadus (aišku, jį priima bei apdoroja ir PK2 mazgas). Pagal skaidraus tilto taisyklės jo kopijas komutatoriai privalo perduoti per visus savo prievadus, išskyrus tą, per kurį kadrą gavo. Naujos kartos transliacinius kadrus, patekusius į 2 –jį segmentą, privalo apdoroti visi segmentui priklausantys prievadai: abiejų komutatorių 2/1 prievadai, bei PK3 su PK4 mazgai. Abu komutatoriai, 2/1 prievaduose gavę transliacinį kadrą, vėl visais prievadais išsiunčia kopijas. Taip 1-jį segmentą pasiekia padidėjęs transliacinių kadrų skaičius. Toliau viskas cikliška kartojasi. Transliacinių kadrų skaičius greitai auga, kol galiausiai užkemša segmentą. Ši situacija vadinama transliaciniu kamščiu. Vienintelė išeitis iš susidariusios padėties – perkrauti komutatorius, arba išjungti prievadą.

■ **Dubliuotus kadrus.** Tinkle su atsarginiais 2 – jo lygmens keliais mazgai dėl susidariusių kilpų gali gauti keletą to paties kadro kopijų. Pasekime pav.3, kas atsitinka, kai PK1 mazgas siunčia kadrą PK4 mazgui. Tarkime, kad abu komutatoriai savo MAC adresų lentelėse turi visų kompiuterių (mazgų) adresus. PK1 siunčia kadrą į tinklo segmentą A. Abu komutatoriai gauna kadrą 1/1 prievaduose. Sutinkamai su įrašais MAC adresų lentelėse, abu jie šį kadrą perduoda per atitinkamą prievadą 2/1 į tinklo segmentą B. Tokiu būdu PK4 mazgas iš komutatorių gauna dvi to paties kadro kopijas. Aišku, tai tragedijos nesukelia.

■ **MAC adresų lentelės nestabilumą.** Kilpos komutatorių klaidina, kaip žmogų klaidina aidas miške. Dėl kilpų komutatorius mazgo MAC adresą susieja ne su tuo prievadu, prie kurio mazgas yra iš tikrųjų prijungtas.

BPDU kadrai

STP protokolas funkcionuoja, komutatoriams tarpusavyje keičiantis specialiais **BPDU** (Bridge Protocol Data Unit) kadrais. Išsiųsdamas BPDU kadra, komutatorius siuntėjo adreso lauke įrašo savo MAC adresą. Kadangi pradiniu laiko momentu komutatoriai vienas kito nežino, tai visi gavėjo lauke nurodo specialų STP grupinį MAC adresą.

BPDU kadrai yra dviejų tipų. STP skaičiavimams naudojamas **konfigūravimo BPDU**, o pranešimams apie topologijos pokyčius – **TCN BPDU** (Topology Change Notification).

Žemiau 3 lentelėje yra išvardinti konfigūravimo BPDU viduje esantys laukai.

3 lentelė . Konfigūravimo BPDU kadro laukai

Lauko aprašymas	Baitų skaičius lauke
Protocol ID (visada 0)	2
Version (visada 0)	1
Message Type (Configuration or TCN BPDU)	1
Flags	1
Root Bridge ID	8
Root Path Cost	4
Sender Bridge ID	8
Port ID	2
Message Age (sek. 256-osiomis dalimis)	2
Maximum Age (sek. 256-osiomis dalimis)	2
Hello Time (sek. 256-osiomis dalimis)	2
Forward Delay (sek. 256-osiomis dalimis)	2

Pradiniame etape pagrindinis keitimosi BPDU kadrais tikslas – iš visų komutatorių išrinkti vieną pagrindinį komutatorių, vadinamą **pagrindiniu komutatoriumi** (root bridge). Jis turi būti atskaitos taškas, nuo kurio bus išaugintas topologinis medis be kilpų. BPDU kadrais komutatoriai keičiasi kas 2 sekundės, tad topologijos pokytis pastebimas greitai. STP protokolo konvergavimo metu yra išrenkamas pagrindinis komutatorius ir po tam tikrų skaičiavimų užblokuojami tie prievada, taip gaunant komutatorių sistemą be kilpų.

Pagrindinio komutatoriaus rinkimai

Bendrą nuomonę apie tinklo topologiją be kilpų komutatoriai gali turėti tik turėdami bendrą atskaitos tašką – pagrindinį komutatorių.

Kiekvienas komutatorius turi unikalų 8 bitų **tilto identifikatorių** (Bridge ID). Jį sudaro du laukai:

- Komutatoriaus prioritetas (2 baitai) – Skaičius nuo 0 iki 65 535. Standartinė prioriteto vertė lygi 32 768. Kuo mažesnis tilto prioritetas, tuo didesnė tikimybė komutatoriui tapti išrinktam pagrindiniu.
- MAC adresas (6 baitai) – Unikalus komutatoriaus MAC adresas (Juo dažnai išrenkamas supervizoriaus modulio MAC adresas).

Pradiniu laiko momentu po įjungimo kiekvienas komutatorių save laiko pagrindiniu komutatoriumi ir ima siuntinėti BPDU kadrus, kurių šaknies tilto ID ir siuntėjo tilto ID laukuose nurodo savo komutatoriaus ID.

Gavęs BPDU, komutatorius lygina jame esantį pagrindinio komutatoriaus ID su savuoju ID. Geresniu laikomas mažesnis ID. Iš pradžių palyginami pirmajame lauke esantys komutatoriaus prioritetai. Jeigu dviejų komutatoriaus prioritetų vertės yra vienodos, tada geresniu ID tampa tas, kurio antrajame lauke esantis MAC adresas yra mažesnis. Kai komutatorius išgirsta BPDU su mažesniu pagrindinio komutatoriaus ID, jis liaujasi reklamavęs save kaip pagrindinį komutatorių, o ima reklamuoti geresnįjį, savo siunčiamų BPDU kadrų pagrindinio komutatoriaus ID lauke nurodydamas geresniojo komutatoriaus ID, nors ir toliau savo ID nurodo siuntėjo komutatoriaus ID lauke. Rinkimai laikomi baigtais, kai visi komutatoriai savo siunčiamų BPDU pagrindinio komutatoriaus lauke vieningai nurodo vieną ir tą patį komutatoriaus ID.

Kaip pavyzdį, nagrinėkime tris komutatorius su tokiais tilto ID:

Sw1	32768.00-00-00-00-00-0a
Sw2	32768.00-00-00-00-00-0b
Sw3	32768.00-00-00-00-00-0c

Kadangi visi tilto prioritetai yra lygūs standartinei 32768 vertei, tad laimi tas, kurio MAC adresas yra mažiausias, t.y. SwA.

Jeigu į tinklą yra įjungiamas nauja komutatorius, turintis mažesnę komutatoriaus ID, negu pagrindinis, jis pagal protokolą ima siuntinėti BPDU, save skelbdamas pagrindiniu komutatoriumi. Aišku, tokiu jis ir

tampa. Todėl reikia saugotis, kad pagrindiniu komutatoriumi netaptų silpnas komutatorius, arba esantis toli nuo tinklo šerdies (core).

Pagrindinių prievadų rinkimai

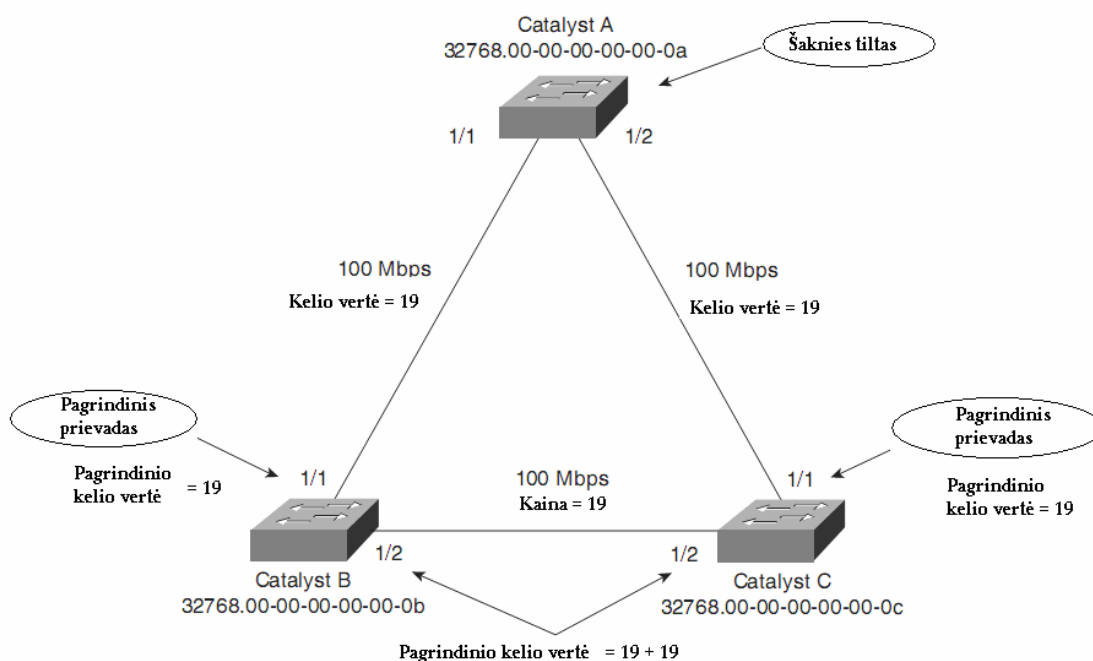
Išrinkus pagrindinį komutatorių, kiekvienas komutatorius turi gebėti nurodyti, kurioje vietoje yra pagrindinis komutatorius. Tai yra pasiekama kiekviename nepagrindiniame komutatoriuje išrenkant **pagrindinį prievadą**. Tai yra prievadas, iš kurio kelias iki pagrindinio komutatoriaus yra mažiausios vertės (trumpiausias). Tik šiuo prievadu yra gaunama informacija iš pagrindinio komutatoriaus. Kiekvienas ne pagrindinis komutatorius turi tik vieną pagrindinį prievadą. Visa struktūra primena tikrą medį.

Pagrindinio prievado rinkimas yra procesas, į kurį įtraukti ir kiti komutatoriai. Rinkimuose naudojami du parametrai: **Pagrindinio kelio vertė** (Root Path Cost) ir STP kelio vertė (STP Path Cost). Pirmasis parametras, kaip jau pastebėjote 3 lentelėje, yra pernešamas BPDU ir yra adityvus dydis. Pasibaigus pagrindinio komutatoriaus rinkimams, BPDU siuntinėja tik pagrindinis komutatorius. Antrasis parametras - STP kelio vertė yra susijusi su kiekvieno prievado pralaidumu. Tiksliau, tai yra skaičius, netiesiškai priklausantis nuo prievado pralaidumo (Originalioje IEEE 802.1D versijoje šis skaičius yra lygus 1000 Mbps, padalintam iš prievado pralaidumo, išreikšto MBps). 4 lentelėje yra pateiktos STP kelio vertės, atitinkančios skirtingus prievadų pralaidumus kaip originaliojoje STP versijoje, taip ir naujojoje.

4 lentelė. STP kelio vertė.

Prievado pralaidumas	Senoji STP kelio vertė	Naujoji STP kelio vertė
4 Mbps	250	250
10 Mbps	100	100
16 Mbps	63	62
45 Mbps	22	39
100 Mbps	10	19
155 Mbps	6	14
622 Mbps	2	6
1 Gbps	1	4
10 Gbps	0	2

13 pav. prie kiekvieno nepagrindinio komutatoriaus prievadų, nukreiptų pagrindinio tilto link, yra nurodyta pagrindinio kelio vertė. Komutatorius, gavęs BPDU iš pagrindinio tilto, pagrindinio tilto vertės lauke esantį skaičių padidina savo prievado pralaidumą atitinkančia verte, gautąjį skaičių įsirašo į atmintį ir pakoreguotą BPDU siunčia sekančiam komutatoriui. Pavyzdžiui, pagrindinis komutatorius Catalyst A siunčia BPDU, kuriame nurodyta pagrindinio kelio vertė, lygi 0. Catalyst B šį BPDU priima 1/1 prievadu, kurio pralaidumas yra 100 Mbps ir vertė 19. Todėl Catalyst C gautame BPDU pagrindinio kelio vertę padidina nuo 0 iki 19, gautąjį skaičių įrašo į atmintį (jį susiedamas su prievadu 1/1) ir pakoreguotą BPDU siunčia toliau. Taip atmintyje yra sukaupiami įrašai apie kelio vertę iki pagrindinio tilto nuo kiekvieno prievado, „žiūrinčio“ į šaknies tiltą. Prievadas, kurio kelio iki pagrindinio tilto vertė yra mažiausia, išrenkamas pagrindiniu prievadu.



13 pav. Pagrindiniai STP elementai

- Keletą pavyzdžių patartina paskaičiuoti patiems. Skaičiuodami, laikykitės tokių taisyklių:
- Pagrindinis komutatorius siunčia BPDU, nuroydamas pagrindinio kelio vertę, lygią 0, kadangi jo prievadas yra tiesiogiai prijungtas prie pagrindinio tilto.
- Kai BPDU gauna sekantis artimiausias kaimyninis komutatorius, jis prie pagrindinio kelio vertės prideda savo prievado, kuris priėmė BPDU, vertę.

- Kaimyninis komutatorius išsiunčia BPDU su nauja, padidinta, pagrindinio kelio verte.
- Sklindant BPDU nuo pagrindinio komutatoriaus į tolimesnius, pagrindinio kelio vertė kiekvieną kartą yra padidinama įėjimo prievado verte.
- Pagrindinio kelio vertė yra padidinama priimant BPDU. Taigi, nauja vertė yra gaunama, kai kadras įeina, o ne išeina iš komutatoriaus.
- Suskaičiavę pagrindinio kelio vertę nuo kiekvieno įėjimo prievado iki pagrindinio komutatoriaus, išrenkate prievadą, kuriam ši vertė mažiausia. Tai ir bus pagrindinis prievadas.

Išskirtųjų prievadų rinkimai

Kiekviename komutatoriuje išrinkus pagrindinius prievadus visi prievadais yra veikiantys, tad tebeturime fizinę topologiją su kilpomis. 11 pav. komutatorių B ir C 1/2 prievadais yra prijungti prie to paties segmento, taip sudarydami kilpą. Šiame etape STP turi nutraukti kilpą, vieną iš prievadų užblokuodamas, o kitą prievadą paskelbdamas **išskirtuoju prievadu** (Designated Port, toliau DP) duotajam segmentui.

Išskirtasis prievadas yra renkamas vadovaujantis kriterijais, kurie yra išvardinti pagal jų prioritetus:

1. Mažiausias pagrindinio komutatoriaus ID
2. Mažiausia pagrindinio kelio vertė iki pagrindinio komutatoriaus
3. Mažiausias siuntėjo komutatoriaus ID
4. Mažiausias siuntėjo prievado ID

Visi trys komutatoriai savo DP išrinko, vadovaudamiesi tokiais samprotavimais:

- **Catalyst A** – Pagal apibrėžimą, visi aktyvūs pagrindinio komutatoriaus prievadais yra DP. Kiekvienam pagrindinio komutatoriaus prievadui pagrindinio kelio vertė lygi 0.
- **Catalyst B** – Catalyst A 1/1 prievadas yra DP segmentui A-B, kadangi jo pagrindinio kelio vertė yra mažiausia (0). Catalyst B 1/2 prievadas yra DP segmentui B-C, nes DP rinkimo kriterijų sąrašė 1-2 punktuose nurodyti parametrai sutampa su Catalyst C, o pagal 3 punktą rinkimus laimi Catalyst B. Jo siuntėjo komutatoriaus ID yra mažesnis (32768.00-00-00-00-00-0b): komutatoriaus prioritetai vienodi, tačiau MAC adresas mažesnis.

■ **Catalyst C** - Catalyst A 1/2 prievadas yra DP segmentui A-C. Catalyst B 1/2 prievadas yra DP segmentui B-C. Todėl Catalyst C 1/2 prievadas negali būti nei DP, nei pagrindiniu prievadu. Todėl jis užblokuojamas.

STP protokolo prievadų būsenos

Paprastai laikoma, kad komutatorių sistemos diametras yra 7. STP protokolo konvergavimas užima tam tikrą laiko intervalą. Aišku, kad prievadų funkcinės savybės konvergavimo metu turi būti ribojamos. Į tinklą įjungtas naujas komutatorius per tam tikrą laiko tarpą turi susipažinti su tinklo topologija, nustatyti, kas yra prijungta prie jo prievadų, jeigu reikia, kai kuriuos savo prievadus užblokuoti ir tik tada jis gali sėkmingai perduoti kadrus. Taigi, STP prievadai nuoseklia tvarka pereina per keletą būsenų:

■ **Išjungtas** (Disabled) – Prievadą uždarė administratorius (administratively down), arba sistema dėl klaidos. Tai nėra STP būsena

■ **Blokavimo** (Blocking) – Suaktyvinus prievadą, jis stengiasi nesudaryti kilpų, todėl pereina į blokavimo būseną. Šioje būsenoje prievadas negali nei priimti, nei perduoti duomenų ir negali MAC adresų rašyti į savo MAC adresų lentelę. Tačiau jis gali priimti BPDU.

■ **Klausymosi** (Listening) – Kai komutatorius supranta, kad prievadas gali būti išrinktas pagrindiniu prievadu arba DP, jis pereina į klausymosi režimą. Būdamas šiame režime, prievadas dar negali operuoti duomenimis, bet gali priimti ir siųsti BPDU.

■ **Mokymosi** (Learning) – Tam tikrą laiko tarpą, vadinamą **perdavimo delsa** (Forward Delay), pabuvęs klausymosi būsenoje, prievadas pereina į mokymosi būseną. Šiame režime prievadui leidžiama gauti ir perduoti BPDU. Kaip sako pats būsenos pavadinimas, prievadas jau gali MAC adresus įrašyti į savo MAC adresų lentelę.

■ **Perdavimo** (Forwarding) – MAC adresų lentelei užpildyti apmokymo režime yra skiriamas kitas perdavimo delsos laiko intervalas. Prievadas pilnai pasiruošia perduoti duomenis. Prievadas pilnai pasiruošia perduoti duomenis, todėl pereina į **perdavimo būseną**. Prievadas gali perduoti ir priimti duomenis ir BPDU. Perdavimo režime leidžiama būti pagrindiniams prievadams ir išskirtiesiems prievadams.

Iš komutatorių sudarytas tinklas laikomas konvergavusiu, kai visi tiltų ar komutatorių prievadai yra arba perdavimo, arba blokavimo būsenoje. Laikas, per kurį konverguoja nauja siejančio medžio topologija, naudojant IEEE 802.1D standartą, lygus 50 sek. = 20 sek. (max amžius) + 15 sek (klausymosi perdavimo delsa) + 15 sek. (mokymosi perdavimo delsa).

Keletą žodžių reikia tarti apie STP naudojamus laiko skaitliukus (Timers). Jų yra trys: **hello** intervalas, **perdavimo delsos** ir **max** amžiaus (Max Age) intervalai. Hello intervalas nurodo, kas kiek sekundžių komutatoriai tarpusavyje keičiasi BPDU kadrais. Kuo jis mažesnis, tuo greičiau STP konverguoja, bet tada BPDU kadrai daugiau apkrauna tinklą. Perdavimo delsa yra laikas, kurį prievadas privalo išbūti klausymosi būsenoje. Max amžiaus intervalas nustato laiką, kurį BPDU yra saugomas atmintyje. Praėjus šiam laiko intervalui, BPDU yra atmetamas. Standartinė max amžiaus trukmė – 20 sek. Visų intervalų trukmės yra parinktos tariant, kad tinklo diametras yra 7 komutatoriai, atstumą matuojant nuo šaknies tilto (jį įskaitant).

TCN BPDU

TCN BPDU kadrus generuoja komutatorius, kurio vieno iš prievadų būseną pakito iš „up“ į „down“ arba atvirkščia kryptimi, t.y. kai prievadas pereina į perdavimo būseną, arba iš perdavimo ar apmokymo būsenos – į blokavimo būseną. TCN BPDU kadrai perduodami per pagrindinį prievadą, siekiant informuoti pagrindinį komutatorių apie įvykusius topologijos pokyčius (tik konstatuojant patį faktą ir jokių detalių). TCN BPDU pernešami nuosekliai nuo vieno komutatoriaus prie kito, kol galiausiai patenka pagrindiniam komutatoriui. Gavę TCN BPDU kadrą visi komutatoriai patvirtina jo gavimą ir tik pagrindinis komutatorius generuoja konfigūravimo BPDU kadrą, skirtą visiems komutatoriams. Konfigūravimo BPDU kadre pamatę nustatytą topologijos pokyčio požymį, visi komutatoriai sumažina savo MAC adresų lentelės amžiaus trukmę nuo standartinės vertės (300 sek.) iki perdavimo delsos (15 sek.). Todėl visi MAC adresų lentelės įrašai labai greitai pašalinami kaip pasenę, jų tarpe ir pokyčių kaltininkai. Lieka tik MAC adresai mazgų, su kuriais lokalusis komutatorius intensyviai bendrauja. Taip tinklas ima funkcionuoti naujos topologijos sąlygomis. Perėjimo trukmė lygi perdavimo delsos (15 sek.) ir Max amžiaus (20 sek.) sumai.

RSTP protokolas

Dabartiniuose tinkluose standartinės STP konvergavimo trukmės (30 – 50 sek.) po topologijos pokyčių yra per ilgos, todėl IEEE 802.1D protokolas buvo patobulintas ir jo pagrindu buvo sukurtas naujas standartas – IEEE 802.1w. Jis dar yra žinomas RSTP vardu (Rapid STP). Šis protokolas apibrėžia kai

kuriuos naujus ir labai efektyvius komutatorių sąveikos principus, leidžiančius tinklui konverguoti žymiai greičiau. Trumpai supažindinsime su RSTP idėja.

Pagal 802.1D protokolą, komutatoriaus prievadui gali būti priskirtas vienas iš trijų galimų vaidmenų - pagrindinio, išskirtojo, arba blokavimo prievado. Prievado vaidmuo tam tikru laipsniu yra susietas su galimomis penkiomis būsenomis – atjungtas, blokavimo, klausymosi, mokymosi ir perdavimo. Pavyzdžiui, pagrindinis ir išskirtasis prievadai niekada negali būti blokavimo būsenoje.

RSTP protokole konvergavimo greitis iš esmės yra pasiekiamas, leidžiant komutatoriams bendrauti su kaimynais visais prievadais. Ši sąveika greičiau yra pagrįsta prievado vaidmeniu, o ne vien pagrindinio tilto siuntinėjamais BPDU kadrais. RSTP protokole, išaiškėjus prievado vaidmeniui, jam yra priskiriama būseną, nusakanti, ką jis privalo daryti su gautais duomenimis.

RSTP protokole pagrindinis komutatorius yra renkamas pagal tuos pačius kriterijus, kaip ir STP – mažiausią komutatoriaus ID. Sekančiame etape prievadams priskiriami vaidmenys:

RSTP protokolo prievadų vaidmenys

- **Pagrindinis prievadas** (Root Port) – Kiekviename komutatoriuje yra išrenkamas pagrindinis prievadas, turintis mažiausią pagrindinio kelio vertę iki pagrindinio komutatoriaus.
- **Išskirtasis prievadas** (Dedicated Port) – Komutatoriaus prievadas, tinklo segmente turintis mažiausią pagrindinio kelio vertę iki pagrindinio komutatoriaus (kaip ir STP protokole).
- **Alternatyvus prievadas** (Alternate Port) – Tai yra prievadas, kuriuo informacija pagrindinį komutatorių pasiekia kitu keliu, nei per pagrindinį prievadą. Šio kelio vertė yra truputį didesnė, nei pagrindinio prievado.
- **Atsarginis prievadas** (Backup Port) – Tai prievadas, kuris turi atsarginį, didesnės vertės kelią iš segmento į pagrindinį komutatorių, negu išskirtasis prievadas. Iš segmento pagrindinis komutatorius yra pasiekiamas dviem keliais per du skirtingus prievadus – išskirtuoju ir atsarginiu, be to, atsarginio prievado pagrindinio kelio vertė truputį didesnė (mažiau geidžiama), negu išskirtojo prievado.

Bet kuris iš aukščiau išvardintų prievadų gali būti vienoje iš trijų būsenų, nusakančių, ką prievadas gali daryti su gaunamais kadrais.

RSTP protokolo prievadų būsenos

■ **Atmetimo** (Discarding) – Šiame režime visi gaunami kadrai yra atmetami ir MAC adresai neįsimenami. Šis režimas savyje apjungia pirmąsias tris STP protokolo būsenas – išjungtą, blokuotą ir klausymosi, kuriose kadrai nėra perduodami. Klausymosi stadijos atsisakyta, kadangi RSTP protokolas gali labai greitai pakeisti net neklausydamas BPDU.

■ **Mokymosi** (Learning) – Gaunami kadrai atmetami, tačiau MAC adresai įsimenami.

■ **Perdavimo** (Forwarding) – Gaunami kadrai perduodami sutinkamai su įsimintais MAC adresais.

STP ir RSTP protokolai yra tarpusavyje suderinami. BPDU kadruose nauja tai, kad RSTP protokolo atveju juose yra nurodoma prievadų vaidmenys ir būsenos, bei RSTP protokolo versija 2. Prievadų būsenos keičiamos kaimyninių komutatorių derybų metu. Kadangi kiekvienas komutatorius kas 2 sekundes siunčia BPDU kaimyniniams komutatoriams, todėl topologijos pokyčius greitai (standartiškai per tris Hello intervalus – 6 sek.) gali aptikti bet kuris komutatorius. Jeigu iš kaimyninio komutatoriaus negaunama trijų **hello** paketų iš eilės, yra daroma prielaida, kad kaimynas išsijungė ir tuojau pat yra pašalinama visa informacija, susijusi su šiuo kaimynu. Tokiu būdu laikas, per kurį yra aptinkama kaimyninio komutatoriaus neveika, lygus 6 sek. (tuo tarp 802.1D protokole šis laikas siekia 20 sek.).

RSTP protokole prievadas pats prisiderina prie gaunamo protokolo versijos – STP (versija 0) ar RSTP (versija 2). Siekiant prievadą apsaugoti nuo nuolatinio protokolų perjunginėjimo, kai iš eilės yra gaunami skirtingos versijos BPDU, yra naudojamas specialus laiko skaitliukas, nustatantis minimalų migravimo laiką iš vienos versijos į kitą.

Kada komutatorius yra įjungiamas arba yra aptinkamas topologijos pokytis, visi komutatoriai kartu sudaro naują siejantį medį. Savo sprendimus perduoti ar atmesti kadrus komutatoriaus prievadai pagrindžia prievado tipu.

RSTP protokolo prievadų tipai

■ **Krašto** prievadas (Edge port) – Tai yra tinklo pakraštyje esantis prievadas, prie kurio yra prijungtas pavienis mazgas. Suaktyvinus STP protokolo **PortFast** savybę, nurodančią, kad prie prievado yra prijungtas pavienis vartotojas, kad kilpos nėra galimos ir kad prievadas, esant reikalui,

gali iš karto būti pervestas į perdavimo režimą, prievadas tampa krašto prievadu. Tačiau, jei tik krašto prievado kada nors gauna BPDU, šis statusas yra iš karto prarandamas.

■ **Pagrindinis** prievadas (Root port) – Kaip ir STP protokole, komutatorius turi tik vieną aktyvų pagrindinį prievadą, kurio šaknies kelio vertė yra mažiausia ir, kaip jau žinome, keletą alternatyvių prievadų. Jų pagrindinio kelio vertės truputį didesnės už pagrindinio prievado. Išėjus iš rikiuotės pagrindiniam prievadui, vienas iš alternatyvių prievadų iš karto pervedamas į perdavimo režimą.

■ **Tiesioginis** prievadas (Point-to-point) – Taip yra vadinamas bet kuris prievadas, per kurį patenkama į kitą komutatorių ir kuris tampa išskirtuoju prievadu. Kaimyniniai komutatoriai greitos rankų paspaudimo procedūros pagalba nustato prievadų būseną, apsikeisdami BPDU kadrais. Vienas komutatorius praneša, kad jo prievadas bus išskirtuoju; jeigu kitas sutinka, tai jis siunčia sutikimo pranešimą. Yra daroma prielaida, kad komutatoriai tarpusavyje bendrauja duplexiniu režimu, todėl šiuo režimu veikiantys prievadais automatiškai tampa tiesioginio tipo prievadais. Segmentuose su tiesioginiais prievadais RSTP protokolas konverguoja greitai, naudodamas RSTP rankų paspaudimo procedūrą.

Tuo tarp RSTP daro prielaidą, kad pusiau duplexiniu režimu veikiantys prievadais yra prijungti prie bendrosios prieigos terpės, kurioje yra daugiau, nei du komutatoriai. Šio tipo terpėje naudojamas standartinis 802.1D protokolas, taigi, prievadais neišvengiamai privalo pabuvoti ir klausymosi, ir mokymosi būsenose, tad konvergavimas yra lėtesnis.

RSTP tinkle (tiesioginių prievadų srityje) konvergavimas prasideda nuo pagrindinio komutatoriaus. Jis RSTP rankų paspaudimo procedūros pagalba nustato savo prievadų būseną ir užsitikrina, kad tarp jo ir kaimyninių komutatorių jokios kilpos nėra. Sekančiame etape kaimyniniai komutatoriai rankų paspaudimu derina savo prievadų būsenas su savo kaimynais ir įsitikina, kad nėra kilpų t.t. Tokiu būdu konvergavimo procesas primena bangas, sklindančias nuo pagrindinio tilto.

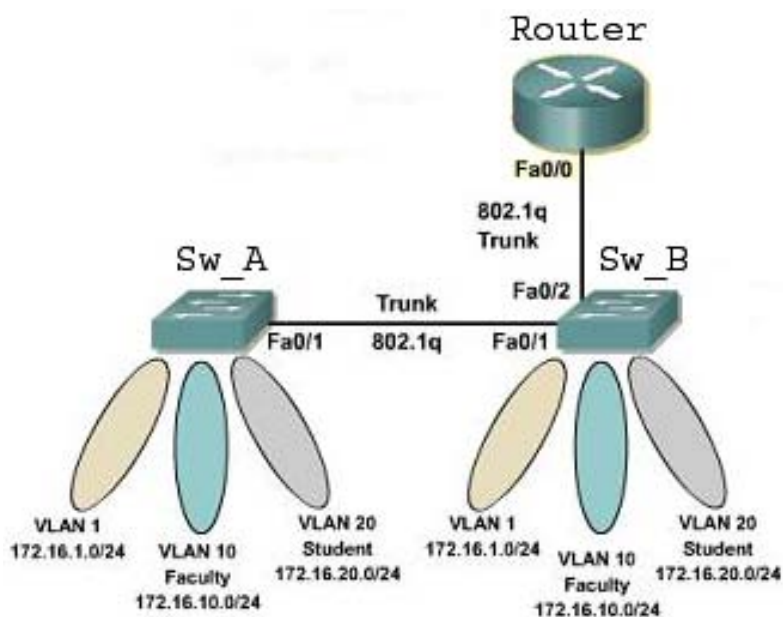
Virtualieji tinklai ir VTP protokolas

Jeigu tinklas yra sudarytas tik iš 2-jo lygmens komutatorių, tai turime vieną didelį transliacinį domeną. Mazgų skaičiui didėjant, didėja ir transliacinių kadrų skaičius. Tokiame tinkle gali būti rezerviniai keliai, o kilpas eliminuoja STP protokolas. Tačiau srautą nukreipti vienu metu keletu skirtingų kelių neįmanoma, nes tam yra reikalingas 3-jo lygmens maršrutizavimo funkcionalumas.

Virtualieji tinklai (toliau, VLAN - Virtual Logical Area Networks) logiškai segmentuota tinklą, vieną didelį transliacinį domeną padalindami į didesnę skaičių mažesnių transliacinių domenų. Kiekvienas VLAN'as yra atskiras transliacinis domenas. Tam tikrame VLAN'e esantys mazgai girdi tiksliai duotajame VLAN'e atsiradusius transliacinius paketus. Į vieną VLAN'ą galima apjungti skirtinguose tinklo segmentuose esančius mazgus. Bendrai tariant, mazgai gali būti net skirtinguose tinkluose, tik nuo tokios praktikos reikia atsiriboti ir visada stengtis, kad VLAN'ų ribos sutaptų su potinklų ribomis.

Mazgus VLAN'ams galima priskirti arba statiškai, arba dinamiškai. Pagal tai ir VLAN'ai yra vadinami atitinkamai statiniais arba dinaminiais. CCNA lygmenyje yra kalbama tik apie statinius VLAN'us, kada konkrečius komutatoriaus prievadus virtualiesiems tinklams priskiria administratorius. Kiekvienam prievadui priskyrimo metu suteikiamas VLAN identifikatorius (Port VLAN ID, trumpiau PVID), susiejantis prievadą su VLAN'u. Tam, kad atsirastų ryšys tarp skirtingiems VLAN'ams priklausančių mazgų, būtinas 3-jo lygmens įrenginys. Jis užtikrina ryšį tarp VLAN'ų.

Šiame paragrafe supažindinsime su svarbiausiomis komutatoriaus konfigūravimo komandomis, pradėdami nuo konsolės, privilegijuoto lygmens, telnet'o linijų slaptažodžių konfigūravimo ir baigdami VTP protokolu, VLAN'ais ir ryšio tarp VLAN'ų konfigūravimu. Kaip darbinį pavyzdį imsime 14 pav. pateiktą tinklo schemą, sudarytą iš dviejų komutatorių Catalyst 2950 ir



14 pav. Virtualieji tinklai ir ryšys tarp jų.

maršrutizatoriaus. Kiekviename komutatoriuje eilė prievadų bus priskirti VLAN'ams 10 ir 20. Sw_B komutatoriaus prievadas fa0/1 verstu kabeliu yra sujungtas su Sw_A komutatoriaus fa0/1 prievadu, o Sw_A komutatoriaus fa0/2 prievadas tiesiu kabeliu sujungtas su maršrutizatoriaus fa0/0 prievadu. Per šiuos prievadus vienu metu kelias skirtingiems VLAN'ams priklausantys kadrai. Kad jie tarpusavyje nesimaišytų, minėti prievadai privalės dirbti magistraliniame (*trunk*) režime, bet apie tai kiek vėliau.

Jeigu painiojate komutatoriaus ir maršrutizatoriaus konfigūravimo komandas, tai pirmajame artinyje galite įsivaizduoti, kad dirbame prie maršrutizatoriaus. Pradinės slaptažodžių konfigūravimo, vardų suteikimo komandos praktiškai sutampa. Esminis skirtumas tarp maršrutizatoriaus ir komutatoriaus yra tai, kad komutatoriaus prievadams negalima suteikti IP adresų, kai tuo tarpu kiekvienam maršrutizatoriaus prievadui privalu priskirti IP adresą. Įjungus komutatorių, visi jo prievadai priklauso virtualiajam valdymo tinklui (**management VLAN**) VLAN 1. Tai tarsi bazė, iš kurios prievadai imami ir priskiriami konkretiems virtualiesiems tinklams. Virtualieji tinklai sukuriama dviem etapais - pirma sukuriamas VLAN'as ir tik vėliau jam priskiriami prievadai. Priskyrimo metu prievadas iš VLAN 1 perkliamas į naująjį VLAN'ą. IP adresą komutatoriuje galima priskirti tik virtualiajam valdymo tinklui VLAN1, kuris tokiu būdu atstovauja visą komutatorių ir jis tinkle tampa „matomas“, t.y. atsiranda galimybė komutatorių konfigūruoti, prisijungus telnet'u. Taigi, pradėkime nuo vtp domeno, VLAN'ų ir VTP serverio sukūrimo. Iš karto pasakysime, kad šią užduotį galima atlikti dviem būdais. Vienas iš būdų - naudojantis **VLAN-Database** režimą - yra nueinantis nuo arenos ir kai kuriose naujose operacinėse sistemose jau nebus naudojamas, kitas – pagrįstas **VLAN-Configuration** režimu – yra rekomenduojamas VLAN'ams kurti.

VLAN'ų ir VTP (Virtual Trunk Protocol) domeno sukūrimas VLAN-Database režime:

```
Sw_A>enable
Sw_A#vlan database
Sw_A(vlan)#vtp server
Sw_A(vlan)#vtp domain ccna
Sw_A(vlan)#vlan 10 name faculty
Sw_A(vlan)#vlan 20 name student
Sw_A(vlan)#vlan 30
Sw_A(vlan)#exit
```

VLAN'ų ir VTP domeno sukūrimas VLAN-Configuration režime:

```
Sw_A>enable
Sw_A#config terminal
Sw_A(config)#vtp mode server
Sw_A(config)#vtp domain ccna
Sw_A(config)#vlan 10

Sw_A(config)#name faculty
Sw_A(config)#vlan 20
Sw_A(config)#name student
Sw_A(vlan)#vlan 30
Sw_A(vlan)#exit
```

Kadangi tarp šių dviejų komandų grupių skirtumai yra nedideli, todėl pakomentuosime tik pirmąją komandų grupę, nes į CCNA egzaminą įeina būtent pirmosios grupės komandos. Pirmąją komandą mes patenkame į **vlan** režimą, kuriame galime sukurti VTP domeną, komutatorių Sw_A paskirti vtp serveriu ir sukurti VLAN'us. Kam reikalingas VTP (**VLAN Trunking Protocol**) domenas ir VTP serveris? VTP yra Cisco privatus protokolas, skirtas suderintam VLAN'ų sukūrimui, nutrynimui ir vardų pakeitimui bendrame administraciniame domene. Tarkime, kad mums reikia sukurti VLAN'us ne viename komutatoriuje, o didesniame skaičiuje, tarkime, septyniuose. Paprastai turėtume įvesti aukščiau pateiktas komandas iš eilės visuose komutatoriuose. O ateityje gali prireikti ištrinti vieną iš VLAN'ų, sukurti naujus, ar pakeisti jų pavadinimus – ir kiekvieną kartą reikėtų aplankyti visus komutatorius ir kiekviename įvesti visas komandas. Darbas ne iš maloniųjų, be to, atsiranda klaidos tikimybė. VTP protokolas administratorių išvaduoja nuo šių rutininių darbų. Jam tereikia vieną iš komutatorių, šiuo atveju Sw_A, komanda **vtp server** paskirti vtp serveriu ir visas komandas sukonfigūruoti tik serveryje. Serveris kas 300 sekundžių (jeigu yra pokyčiai, tada greičiau) šią informaciją praneša grupei komutatorių, priklausančių tam pačiam vtp domenui (šiuo atveju domenui, vardu ccna). Šie pagal serverio nurodymus sukuria tokius pačius VLAN'us. Tokiu būdu VTP serveris pasirūpina, kad visi pasikeitimai nutrynus, pervardinus ar sukūrus naują VLAN'ą būtų perduoti visiems vtp domeno komutatoriams.

Informaciją apie VLAN'us komutatoriai saugo **vlan.dat** faile, esančiame flash atmintyje, atskirai nuo konfigūracinio failo NVRAM'e. Dėl šios priežasties VLAN'ų negalima panaikinti, ištrinus konfigūracinį failą. **Vlan.dat** failą galite pamatyti, surinkę komandą **show flash**. Sukonfigūruotus

VLAN'us galite pamatyti, surinkę komandą **show vlan**. Jeigu VLAN'ų nėra, matysite, kad visi prievadai priklauso VLAN 1. Jeigu norite ištrinti visus VLAN'us (išskyrus VLAN 1, kurio neįmanoma ištrinti), naudokite komandą **delete flash:vlan.dat**. Atkreipkite dėmesį, kad tarp dvitaškio ir **vlan.dat** jokio tarpo nėra.

Jeigu norite ištrinti vieną VLAN'ą, tarkime, VLAN 20, tai tame pačiame režime, kuriame VLAN'ai buvo sukurti, reikia įvesti komandą su žodeliu **no** priekyje:

```
Sw_A(vlan)#no vlan 20, arba
Sw_A(config)#no vlan 20.
```

Prieš ištrindami VLAN'ą, pirma iš jo pašalinkite visus prievadus, priešingu atveju prievadai bus neveikiantys, nes priklausys neegzistuojančiam VLAN'ui.

VTP domene gali būti vienas ar daugiau vtp serverių. Kiti komutatoriai turi būti arba kliento, arba skaidriame režime (**vtp client**, **vtp transparent**). Klientai negali sukurti VLAN'ų, jie tik klausosi serverio pranešimų ir išsaugoja informaciją savo VLAN duomenų bazėje ir serverio pranešimus perduoda kitiems komutatoriams. Tuo tarpu skaidriame režime esantys komutatoriai gali kurti VLAN'us, bet negali jų reklamuoti pranešimuose kitiems, tad jų VLAN'ai tėra tik lokaliai reikšmingi. Aišku, serverio pranešimus jie turi perduoti kitiems.

Kiekvienas serverio pranešimas yra numeruojamas revizijos numeriu augančia tvarka. Kiti komutatoriai, matydami didesnę revizijos numerį, atnaujina savo VLAN duomenų bazes naujausia informacija. Kadangi standartiškai kiekvienas Cisco komutatorius yra serverio režime, tai yra didžiulis pavojus, kad naujas į tinklą įjungtas komutatorius, jeigu jo vtp revizijos numeris yra didesnis, nei serverio, gali ištrinti visuose domeno komutatoriuose egzistuojančius VLAN'us. Todėl reikia pasirūpinti, kad, prieš įjungiant naują komutatorių į tinklą, jo vtp revizijos numeris būtų 0. VTP revizijos numerį galite patikrinti komanda **show vtp status**.

VTP revizijos numerį pakeisti nuliniu galima dviem būdais:

- komutatorių nustatyti į **transparent** režimą ir vėl grąžinti į serverio režimą,
- komutatorių priskirti kitam, neegzistuojančiam vtp domenui ir vėl grąžinti į pradinį domeną.

Sukūrę VLAN'us, pakeitimus privalome išsaugoti VLAN duomenų bazėje, padidinti revizijos numerį ir išeiti iš **VLAN Database** režimo (arba **VLAN – Configuration**). Visas šias funkcijas atlieka komanda **exit** arba **apply**. Niekad nenaudokite CTRL+Z klavišų derinio, nes ši seka anuliuoja

padarytus pakeitimus. Sekančiame etape galima sukonfigūruoti visus slaptažodžius ir komutatoriui suteikti vardą.

Slaptažodžių ir vardų konfigūravimas:

```
Switch>enable
Switch#configure terminal
Switch(config)#hostname Sw_A
Sw_A(config)#enable secret cisco
Sw_A(config)#service password-encryption
Sw_A(config)#line console 0
Sw_A(config-line)#password cisco
Sw_A(config-line)#login
Sw_A(config-line)#exit
Sw_A(config)#line vty 0 15
Sw_A(config-line)#password cisco
Sw_A(config-line)#login
Sw_A(config-line)#exit
```

Šia komandų grupę komutatoriui suteikėme Sw_A vardą. Kita komanda **enable secret** mums jau žinoma – kaip ir maršrutizatoriuje ji nustato privilegijuoto lygmens slaptažodį. Visada patartina naudoti šią komandą, nes ji naudoja stipresnį šifravimo algoritmą, nei normali **enable password** komanda, nustatanti vartotojo lygmens slaptažodį. Konfigūraciniame faile šie slaptažodžiai yra matomi atviru tekstu ir juos gali perskaityti net ir nepageidautini asmenys. Norėdami slaptažodžius konfigūraciniame faile užšifruoti, naudojame komandą **service password-encryption**. Tokiu būdu jų niekas negalės perskaityti, net matydamas konfigūracinį failą.

Toliau sukonfigūruosime **standartinio šliuzo** (default-gateway) IP adresus ir komutatorių IP adresus (priskiriamus loginiams VLAN1prievadams). Visi šie adresai turi priklausyti vienam potinklui, tarkime 172.16.1.0. Jeigu tarsime, kad tinkle turime du komutatorius ir vieną maršrutizatorių, reikalingą užtikrinti ryšiui tarp VLAN'ų, tai 172.16.1.1 adresą priskirsime maršrutizatoriaus prievadui, prie kurio prijungsime komutatorius. Šis adresas komutatoriuose bus sukonfigūruotas kaip standartinio šliuzo adresas. Komutatorių valdymo VLAN'ams priskirsime to paties potinklio IP adresus 172.16.1.2 ir 172.16.1.3.

Standartinio šliuzo konfigūravimas ir IP adresų priskyrimas valdymo VLAN'ui

```
Sw_A(config)#ip default-gateway 172.16.1.1
Sw_A(config)#interface vlan 1
Sw_A(config-if)#ip address 172.16.1.2 255.255.255.0
Sw_A(config-if)#exit
Sw_A(config)#
Sw_B(config)#ip default-gateway 172.16.1.1
Sw_B(config)#interface vlan 1
Sw_B(config-if)#ip address 172.16.1.3 255.255.255.0
Sw_B(config-if)#exit
Sw_B(config)#
```

Standartinio šliuzo paskirtis yra analogiška, kaip kompiuteryje. Šis adresas yra „vartai į pasaulį“, kai adresatas yra kitame VLAN'e, ar tinkle. Matome, kad komutatoriuje IP adresas yra priskiriamas loginiam prievadui, susijusiam su VLAN 1. Dabar komutatoriai turi atsakyti į vienas kito **ping**'us, kadangi abu komutatoriai yra viename potinklyje. Maršrutizatorius šiame etape į komutatorių **ping**'us atsakyti negali, kadangi jis dar nėra sukonfigūruotas. Tolimesnė užduotis - VLAN'ams priskirti prievadus ir pasirūpinti magistraliniais kanalais, kuriais vienu srautu galėtų būti perduodami visų VLAN'ų kadrai.

Prievadų priskyrimas VLAN'ams.

```
Sw_A(config)#interface fa0/4
Sw_A(config-if)#switchport mode access
Sw_A(config-if)#switchport access vlan 10

Sw_A(config)#interface fa0/5
Sw_A(config-if)#switchport mode access
Sw_A(config-if)#switchport access vlan 10

Sw_A(config)#interface fa0/6
Sw_A(config-if)#switchport mode access
Sw_A(config-if)#switchport access vlan 10
```

Visiškai analogiškai prievadus fa0/7 – fa0/9 priskiriame VLAN 20. Jeigu VLAN'ui reikia priskirti ne vieną, du, o didelę grupę prievadų, galime pasielgti gudriau ir tai atlikti viena komanda. Pavyzdžiui, aukščiau pateiktas fa0/4-fa0/6 prievadų priskyrimo VLAN 10 komandas pakeičiame tokia trumpa komandų seka:

```
Sw_A(config)#interface range fa0/4-6
Sw_A(config-if)#switchport mode access
Sw_A(config-if)#switchport access vlan 10
```

Analogiškai fa0/7-9 prievadus priskiriame VLAN'ui 20:

```
Sw_A(config)#interface range fa0/7-9
Sw_A(config-if)#switchport mode access
Sw_A(config-if)#switchport access vlan 20
```

Tokias pat komandas įvedame ir Sw_B komutatoriuje. Taip, abiejuose komutatoriuose turime fa0/4-6 prievadus, priskirtus VLAN'ui 10, fa0/7-9 – VLAN'ui 20. Visi likusieji prievadai tebeprisiklauso VLAN 1. Panaudoję komandą **show vlan**, galime patikrinti, ar prievadus teisingai priskyre VLAN'ams. Pastaroji komanda neparodo magistralinių (*trunk*) prievadų, o tik prieigos (*access*).

VLAN'ų magistralės ir magistralinių prievadų konfigūravimas.

Dabar abiejų komutatorių fa0/1 prievadus sukonfigūruokime darbui magistraliniame režime:

```
Sw_A(config)#interface fa0/1
Sw_A(config-if)#switchport mode trunk

Sw_B(config)#interface fa0/1
Sw_B(config-if)#switchport mode trunk
Sw_B(config-if)#interface fa0/2
Sw_B(config-if)#switchport mode trunk
```

Kaip jau minėjome anksčiau, komutatoriaus prievadas gali dirbti viename iš dviejų režimų – **prieigos** (*access*) arba **magistraliniame** (*trunk*). Komanda **switchport mode access** nurodo, kad prievadas yra priskiriamas tik vienam VLAN'ui ir prie jo gali būti prijungtas mazgas, bet ne kitas komutatorius. Tuo tarpu komanda **switchport mode trunk** prievadą paverčia magistraliniu. Jeigu naudojamas

magistralinis režimas, būtina nurodyti inkapsuliaciją, išskyrus atvejus, kad komutatoriuje galima nustatyti tik vieną inkapsuliacijos tipą, pvz.: Catalyst 2950 komutatoriuje egzistuoja tik 802.1q inkapsuliacija, todėl jos paminėjimas tampa beprasmiu.

Kadangi konkrečiam VLAN'ui priklausantys mazgai yra abiejuose komutatoriuose, tai magistralinėmis linijomis yra perduodami skirtingiems VLAN'ams priklausantys kadrai. Kad jie tarpusavyje nesusimaišytų, kiekvieno VLAN'o kadrai magistralėje yra „nuspalvinami“ skirtinga „spalva“. Realiai tai reiškia, kad kadruose vienokiu ar kitokiu būdu yra įrašoma VLAN'o identifikatorius. Nagrinėjamoje sistemoje turime dvi magistralines linijas. Viena iš jų jungia tarpusavyje komutatorių fa0/1 prievadus, o kita veda iš Sw_A komutatoriaus fa0/2 prievado į maršrutizatoriaus fa0/0 prievadą.

Egzistuoja du kadrų „spalvinimo“ būdai magistralėje. Pirmasis yra Cisco privati ISL inkapsuliacija, kai 2-jo lygio kadras yra tikrai inkapsuliuojamas, t.y. jam priekyje pridėdama 26 baitų preambulė su VLAN'o ID, o pabaigoje – 4 baitų CRC. Kadro dydis dėl to padidėja ir 30 baitų viršija maksimaliai leistiną Ethernet'o tinkluose 1518 Kb ribą. Jeigu abiejuose magistralės galuose inkapsuliacijos nesutampa, tada ISL kadrus komutatoriai fiksuoja kaip *Giant baby*. Kai įranga yra skirtingų gamintojų, naudojama standartinė IEEE 802.1Q inkapsuliacija, dar vadinama *dot1q* inkapsuliacija. Šiuo atveju į kadrą, iš karto už šaltinio adreso lauko, yra įrašoma 4 baitų žymė, kurios paskutiniuose 12 bitų yra įrašomas VLAN'o identifikatorius.

Kadangi Catalyst 2950 komutatoriuose galima tik *dot1q* inkapsuliacija, tai komutatoriuje ji iš viso neminima. Tuo tarpu Catalyst 2900 komutatoriuje yra galimos abi inkapsuliacijos - *dot1q* ir *isl*, - todėl jo magistralinio prievado konfigūravime būtina išreikštai nurodyti inkapsuliaciją:

```
Switch(config)#interface fa0/1
Switch(config-if)#switchport mode trunk
Switch(config-if)#switchport trunk encapsulation isl
```

Kada kadras patenka į komutatoriaus prievadą, komutatoriaus viduje jam yra priskiriama VLAN'o, kuriam priklauso prievadas, žymė. Pasiėkus išėjimo prievadą, ši žymė yra panaikinama. Jeigu tai yra magistralinis prievadas, tada VLAN'as yra identifikuojamas arba ISL inkapsuliacija, arba 802.1q žyme. Pasiėkus magistralės galą, visos žymės ar inkapsuliacijos yra panaikinamos, patikrinama VLAN'ų žymės ir įrašomos vidinės žymės. Tuo atveju, kai išėjimo prievadas priklauso galiniam

virtotojui, tada siunčiamas originalus, nepakeistas kadras, tad pats VLAN'ų panaudojimo faktas virtotojams tampa skaidriu.

Prievadai gali tapti magistraliniais arba juos konfigūruojant statiškai, arba dinaminių derybų metu, panaudojant **dinaminį magistralės protokolą** (DTP – Dynamical Trunking Protocol). Prievadas gali būti penkiuose DTP režimuose:

- **Prieigos** (Access) – Prievadas priklauso kažkurio tai VLAN'o virtotojui.
- **Magistralinis** (Trunk) – Prievadas derina magistralės parametrus su kitame gale esančiu prievadu.
- **Nesiderėjimo** (Non – negotiate) – Prievadas yra magistralinis ir nenaudoja DTP deryboms su kitame gale esančiu prievadu.
- **Dinaminis pageidaujamas** (Dynamic Desirable) – Aktyviai deramasi su kitu prievadu. Jis tampa magistraliniu, jeigu kitame gale prievadas dirba **magistraliniame**, **dinaminiam pageidaujamame**, arba **dinaminiam auto** režimuose.
- **Dinaminis auto** (Dynamic Auto) – Pasyviai laukia, kol su juo pradės derybas kitas komutatorius. Jis tampa magistraliniu tik tuo atveju, kai kitame gale prievadas dirba **magistraliniame** arba **dinaminiam pageidaujamame** režimuose.

Su 802.1q inkapsuliacija yra susijusi **gimtojo** VLAN'o (native VLAN) sąvoka. Magistralėje gimtasis VLAN'as nustatomas komanda:

```
Switch(config-if)#switchport trunk native vlan vlan nr
```

Čia *vlan nr* yra gimtojo VLAN'o numeris. Gimtojo VLAN'o kadrai magistralėje yra perduodami išskirtiniu būdu – be žymės, tarsi prievadas nepriklausytų jokiam VLAN'ui. Abiejuose magistralės galuose gimtieji VLAN'ai privalo sutapti. Standartiškai VLAN 1 visiems prievadams yra gimtasis.

Standartiškai magistrale perduodami visų VLAN'ų kadrai. Jeigu magistrale norite praleisti tik tam tikrus VLAN'us, naudokite komandą

```
Switch(config-if)#switchport trunk allowed vlan vlangs
```

Ryšio tarp VLAN'ų konfigūravimas maršrutizatoriuje

Paskutiniame etape beliko organizuoti ryšį tarp VLAN'ų. Kaip tai padaroma? Turime tris VLAN'us – 10, 20 ir VLAN 1. Kadangi kiekvienas VLAN'as yra atskiras transliacinis domenas, tai kiekvienam VLAN'ui priklausančiame kompiuteryje turi būti sukonfigūruotas savas standartinis šliuzas (default gateway). Kadangi komutatoriuje VLAN 1 yra priskirti IP adresai atitinkamai 172.16.1.2 ir 17.16.1.3, tai IP adresą 172.16.1.1 galima sukonfigūruoti VLAN 1 priklausančiuose kompiuteriuose kaip standartinio šliuzo adresą. Tada VLAN 10 ir VLAN 20 priklausančiuose kompiuteriuose šliuzo IP adresais galime pasirinkti atitinkamai 172.16.10.1 ir 172.16.20.1. Tačiau yra trys skirtingi šliuzo adresai, maršrutizatorius turi tik vieną fizinį prievadą fa0/0.

Problemos sprendimas paprastas – fizinis maršrutizatoriaus prievadas fa0/0 yra skaidomas į tris loginius prievadus - **subprievadus**, ir jiems priskiriami standartinių šliuzų adresai:

```
Router(config)#interface fa0/0
Router(config)#no shutdown
Router(config)#interface fa0/0.1
Router(config-subif)#encapsulation dot1q 1
Router(config-subif)#ip address 172.16.1.1 255.255.255.0

Router(config)#interface fa0/0.10
Router(config-subif)#encapsulation dot1q 10
Router(config-subif)#ip address 172.16.10.1 255.255.255.0
Router(config)#interface fa0/0.20
Router(config-subif)#encapsulation dot1q 20
Router(config-subif)#ip address 172.16.20.1 255.255.255.0
```

Kiekvienam VLAN'ui priklausančioms mazgai turi sukonfigūruotus skirtingus standartinius šliuzus – maršrutizatoriaus subprievadų IP adresus. Šiose komandose atkreipkite dėmesį į tai, kad:

- fizinis prievadas gali neturėti IP adreso
- loginiai prievadai konfigūruojami, panaudojant tašką, pavyzdžiui: *fa0/0.10*
- komandoje **encapsulation dot1q 10**, skaičius 10 reiškia VLAN'o numerį.

Priedai

Skaiciavimo sistemos

Kompiuterinius tinklus aptarnaujantys darbuotojai kiekvieną dieną susiduria su skaičiavimais dvejetainėje sistemoje. Tai yra tinklų abėcėlė ir ją reikia puikiai žinoti. Šį klausimą stengsimės pateikti kiekvienam suprantama forma. Visos skaičiavimo sistemos yra panašios ta prasme, kad visos jos yra pagrįstos tuo pačiu principu. Jeigu mokate skaičiuoti vienoje sistemoje (o dešimtainėje sistemoje skaičiuojame puikiai visi), tada, išsiaiškinę jos dėsningumus, Jūs iš karto įvaldote bet kurią skaičiavimo sistemą. Kompiuteriniuose tinkluose reikia žinoti dvejetainę ir šešioliktinę skaičiavimo sistemas.

Dvejetainė skaičiavimo sistema

Kompiuteryje duomenys ir visos komandos sklinda elektrinių impulsų pavidalu. Kompiuterio darbas pagrįstas trivialiu faktu – tam tikru laiko momentu įtampa laide yra arba jos nėra. Šios dvi būsenos dvejetainėje skaičiavimo sistemoje atitinka loginius vienetus ir nulius (arba atvirkščiai), todėl ši skaičiavimo sistema yra kompiuterio „gimtoji“ kalba. Būtent šioje sistemoje atsiskleidžia giluminiai bet kokio kompiuteryje vykstančio proceso dėsningumai, kurie dešimtainėje formoje nėra matomi ir akivaizdūs. Reikia mokėti dešimtainius skaičius paversti dvejetainiais, šešioliktiniais ir atvirkščiai, bei pereiti nuo dvejetainių prie šešioliktinių. Tinklo administratorius šiose sistemose skaičiavimus turi mokėti atlikti greitai ir be klaidų. Šešioliktinė sistema yra naudojama daug kur – tai ir kompiuterio fizinis adresas (MAC - Media Access Control), išdegingtas tinklo plokštės ROM atmintyje, RAM atminties resursai, IPv6 protokolo IP adresai ir pan. Skaičiavimo dvejetainėje sistemoje įgūdžiai pravers skaidant tinklus, vykdant tokią rutininę operaciją, kaip IP adreso priskyrimas tinklo mazgo prievadui, maršrutų sumavimas, kintamo ilgio potinklio maskuotė (VLSM – Variable Length Subnet Mask), filtrai (ACL – Access List) ir t.t. Tinklo administratorius su panašaus tipo užduotimis susiduria beveik kasdien.

Visos skaičiavimo sistemos pagrįstos vienu algoritmu, kurį paprastais žodžiais galima iliustruoti tokiu gyvenimišku klausimu: kiek samčių smėlio yra kieme esančioje smėlio krūvoje? Aišku, Jūs paklausite – o kokio dydžio samtis? Didelių samčių bus daugiau, o mažesnių – mažiau. Taigi, tarp skirtingų skaičiavimo sistemų tik toks ir yra skirtumas. Tą patį dešimtainį skaičių šešioliktinėje

sistemoje užrašysime, tarkime, tik dviem simboliais, o dvejetainėje sistemoje jo užrašas užims net aštuonis simbolius. Toliau pateiksiu bendrą dešimtainio skaičiaus pervedimo į kitą skaičiavimo sistemą algoritmą, kuris studentui turėtų būti lengvai suvokiamas.

Tarkime, kad mums reikia dešimtainį skaičių x paversti dvejetainiu. Matematiškai tai reiškia, kad reikia rasti skleidinio

$$x = a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_1 2^1 + a_0 2^0 = \sum a_i 2^i \quad (1)$$

koeficientus $a_1, a_2, \dots, a_n$. Čia pagal pasikartojančius indeksus vyksta sumavimas. Kadangi, dirbdami su kompiuteriu, dažniausiai susiduriame su skaičiais, neviršijančiais 255, tai galime tarti, kad (1) formulėje $n = 7$. Jau iš (1) išraiškos galime matyti vieną patogų ir paprastą dešimtainio skaičiaus pavertimo dvejetainiu algoritmą. Skaičių x nuosekliai dalinkime iš 2, kol galų gale gausime 1. Kiekvieną kartą, kai skaičius dalijasi be liekanos, rašome 0, o kai su liekana, rašome 1. Galiausiai užrašome gautąjį nulių ir vienetų seką atvirkščia tvarka ir mūsų darbas baigtas! Pavyzdžiui, tarkime, kad yra duotas skaičius $x = 125$. Nuosekli dalijimų grandinėlė bus tokia: $125:2=62$ ir liekana **1**, $62:2=31$ be liekanos, tad rašome **0**, $31:2=15$ ir liekana **1**, $15:2=7$ ir liekana **1**, $7:2=3$ ir liekana **1**, $3:2=1$, tad rašome **1** ir dar turime prirašyti liekaną **1**. Dabar, užrašę gautus vienetų ir nulių atvirkščia tvarka, gauname ieškomą dvejetainį skaičių **1111101**.

Studentams, ypač ne fizikams ir ne matematikams, kurie yra tik girdėję apie dvejetainę sistemą, yra geriau suprantamas kitas metodas, pagrįstas kitokiu (1) formulės užrašu:

$$x = 128a_7 + 64a_6 + \dots + 2a_1 + 1a_0. \quad (2)$$

Šiuo atveju skaičių x atitinka dvejetainis skaičius $a_7a_6a_5a_4a_3a_2a_1a_0$. Šį būdą galima labai paprastai prisiminti ir suprasti, nagrinėjant pirkimo operacijos pavyzdį, todėl ir pačiam metodui suteikime kodinį pavadinimą „pirk televizorių“. Kadangi visi mes klaidų nedarome, kai reikia skaičiuoti savo pinigus, tai tarkime, kad pas mus įvyko nelogiška pinigų reforma ir dabar visi daiktai kainuoja ne daugiau, kaip 255 Lt (maksimalus 8 bitų skaičius). Apyvartoje yra 128, 64, 32, 16, 8, 4, 2 ir 1 Lt banknotai. Kaip pastebime, šie skaičiai yra ne kas kita, o (2) sklaidinio koeficientai. Reikia laikytis vienos taisyklės - mokant pinigų negalima duoti dviejų vienodos vertės banknotų. Taigi, tarkime, kad parduotuvėje perkame televizorių (Jūs galite pirkti kompiuterį), kuris įvertintas aukščiau minėta 125 Lt suma. Pinigus mokėkime, pradėdami nuo stambiausios kupiūros ir baigdami smulkiausia.

Apatinėje 5 lentelės eilutėje, ties atitinkamo nominalo 2^n banknotu, rašome skaičių 1, jeigu tokį banknotą davėme ir skaičių 0 – jeigu nedavėme.

5 lentelė Dešimtainio skaičiaus pavertimas dvejetainiu

2^n	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Skaičiaus 2^n laipsniai	128	64	32	16	8	4	2	1
Taip – 1, ne – 0.	0	1	1	1	1	1	0	1

Apatinėje eilutėje yra ne kas kita, kaip dešimtainio skaičiaus 125 išraiška dvejetainėje formoje, t.y. 01111101. Gautąjį dvejetainį skaičių norėdami parašyti dešimtainėje sistemoje, mums tereikia sudėti nominalus banknotų, kuriuos davėme pardavėjai, t.y. sudėti viršutinėje eilutėje esančius skaičius, ties kuriais yra vienetai: $64+32+16+8+4+1=125$.

Šešioliktinė skaičiavimo sistema.

Pažiūrėkime dar kartą į (1) formulę. Mums ji labai primena vektoriaus išdėstymą baziniais vektoriais. Panašiai kaip bazę sudarančius vektorius mes galime pasirinkti daugybe būdų, lygiai taip (1) formulėje esančius dvejetukus galime pakeisti bet kuriais kitais skaičiais, kad ir skaičiumi 16. Taip gausime dešimtainio skaičiaus skleidinį skaičiaus 16 (vadinamo skaičiavimo sistemos pagrindu) laipsniais. Dabar vietoje (1) išraiškos gausime

$$x = a_n 16^n + a_{n-1} 16^{n-1} + \dots + a_1 16^1 + a_0 16^0 = a_i 16^i. \quad (3)$$

Taigi, norėdami tą patį skaičių 125 parašyti šešioliktinėje formoje, mes turime rasti koeficientus a_i . Pastebėkime, kad bet kurioje skaičiavimo sistemoje atitinkamo skleidinio koeficientai yra mažesni už pagrindą. Taip dvejetainėje sistemoje (pagrindas lygus 2) koeficientai gali būti tik 0 arba 1. Būtent iš čia atsirado draudimas duoti tą patį banknotą du kartus. Šešioliktinėje sistemoje tai reiškia, kad visi koeficientai bus skaičiai tarp 0 ir 15. Dabar grįžkime prie skaičiaus 125 ir (3) formulės. Iš karto matome, kad nelygūs nuliui tebus paskutiniai du jauniausi koeficientai: a_1 ir a_0 . Dalindami 125 iš 16 gauname koeficientą $a_1 = 7$ (tai yra ne kas kita kaip dalmuo), o liekana 13 reiškia koeficientą $a_0 = 13$. Taigi, gavome lygybę $125 = 7 \cdot 16^1 + 13 \cdot 16^0$. Vadinasi, dešimtainis skaičius 125 šešioliktinėje sistemoje bus užrašomas dviem simboliais: 7(13), skliausteliuose esantį skaičių suprantant kaip vieną nedalomą koeficientą. Patogumo dėlei jį galime pažymėti viena raide, tarkime D. Tada skaičiaus 125 šešioliktainis atitikmuo bus skaičius 7D. Šešioliktinėje sistemoje vietoje

koeficientų, sudarytų iš dviejų skaitmenų (10, 11, ..., 15) yra naudojamos pirmosios abėcėlės raidės, t.y. priimti tokie žymėjimai

$$10 = A, 11 = B, 12 = C, 13 = D, 14 = E, 15 = F. \quad (4)$$

Norint pažymėti, kad skaičių 7D reikia suprasti kaip šešioliktąjį (angl. Hexadecimal), Cisco IOS (Internet Operatig System) prieš skaičių yra rašomi simboliai 0x, arba pažymima raide h. Taigi, maršrutizatoriuje šešioliktąjį skaičių Jūs matysite, kaip 0x7D. Žemiau esančioje 6 lentelėje yra pateiktas ryšys tarp įvairių skaičių įvairiose skaičiavimo sistemose. Šia lentele patogiu naudotis, kai norime atlikti konversiją iš dvejetainės (šešioliktinės) į šešioliktinę (dvejetainę).

6 lentelė. Ryšys tarp dvejetainių, dešimtainių ir šešioliktinių skaičių

Dešimtainis	Dvejetainis	Šešioliktinis
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Jeigu norime šešioliktąjį skaičių paversti dvejetainiu arba atvirkščiai, yra naudojama elementari procedūra, kurios pagrindą sudaro faktas, kad skaičius 16 yra gaunamas, skaičių 2 pakėlus laipsniu 4. Tai

reiškia, kad bet kurį šešioliktainį skaičių, esantį paskutiniajame 6 lentelės stulpelyje, galime išreikšti keturių bitų dvejetainiu skaičiumi, kuris nurodytas viduriniame stulpelyje.

Turėdami tokią lentelę, galime lengvai paversti šešioliktainiu bet kokį dvejetainį skaičių, kad ir koks didelis jis bebūtų, tarkime 0101 1100 0001 1101 1111 0110 0101 1100 0011 1110 1001 0110. Tam tereikia duotąjį dvejetainį skaičių suskirstyti į grupes po keturis (kvartetais), einant iš galo į pradžią, ir pagal lentelę kiekvieną nuliukų ir vienetukų kvartetą pakeisti atitinkamu šešioliktainiu skaičiumi iš trečiojo stulpelio. Taigi, nurodytas ilgas skaičius šešioliktainėje formoje bus žymiai kompaktiškesnis: 0x5C1-DF6-5C3-E96 (0x priekyje reiškia šešioliktainį skaičių). Gal būt, kažkur pasaulyje yra kompiuteris, kurio tinklo plokštės ROM atmintyje yra išdegintas toks MAC adresas. Kiekviena dvejetainio skaičiaus pozicija reiškia vieną bitą, vienas šešioliktainis skaičius – keturi bitai.

OSI ir TCP/IP modeliai

OSI modelis

Siekdama standartizuoti bei struktūrizuoti tinklinę aparatinę ir programinę įrangą, Tarptautinė standartų organizacija ISO (International Standard Organization)⁴ paskelbė kompiuterinių tinklų architektūros modelį – Rekomenduojamą atvirų sistemų sąveikos modelį⁵ – (The Reference Model of Open Systems Interconnection – OSI), kuris buvo sukurtas 1977 – 1984 m. OSI modelis aprašytas ISO7498 standartu (yra papildomų specifikacijų, praplečiančių šį – pagrindinį – standartą). OSI modelis nėra tinklo architektūra, nes jis nurodo tik kiekvieno lygmens funkcijas, bet ne tai, kokios tikslios paslaugos ir protokolai turi būti naudojami kiekviename lygmenyje.

Reikia pripažinti, kad iki šiol nėra gaminių, kurie pilnai tenkintų OSI, nors stengiamasi, bent dalinai, jo laikytis. Beje, OSI modelis nėra taikytinas vien kompiuteriniams tinklams: kadangi jis sukurtas kaip telekomunikacijų standartas, jo modeliu veikia ir stacionarūs bei mobilūs telefonijos bei daugelis kitų telekomunikacinių tinklų.

Siekiant supaprastinti tinklų struktūrą, ji suskaidyta į tam tikrus lygmenis su apibrėžtomis funkcijomis (15 pav.). Minėtieji lygmenys sudaro taip vadinamą OSI rinkinį, susidedantį iš 7 lygmenų (sluoksnių). Tinkluose gali būti panaudoti ir kiti pvz., TCP/IP skaidymo į lygmenis modeliai.

7.	Taikomasis lygmuo
6.	Vaizdavimo lygmuo
5.	Seansinis lygmuo
4.	Transporto lygmuo
3.	Tinklinis lygmuo
2.	Kanalinis lygmuo
1.	Fizinis lygmuo

15 pav. OSI modelio struktūra.

⁴ Tarptautinė standartų organizacija (International Standard Organization, ISO) įkurta 1946 m. Ją sudaro >160 techninių komitetų, tame tarpe ir tokios žinomos organizacijos, kaip ANSI - American National Standard Institute), jungia >75 valstybių.

⁵ Terminas “atvira sistema” reiškia, kad tos sistemos architektūra ir savybės nėra kieno nors nuosavybė, t. y. vartotojas turi pilnus sistemos aprašymus ir galimybę bei leidimą panaudoti sistemą savo tikslams, ją keisti ir tobulinti (taip tikimasi išplatinti sistemą didesniai vartotojų skaičiui ir paversti sistemą standartu *de facto*).

Skaidant tinklines struktūras į lygmenis, vadovaujamasi šiomis pagrindinėmis taisyklėmis:

1. Kiekvienas lygmuo turi vykdyti griežtai apibrėžtas funkcijas;
2. Lygmenų funkcijos turi būti pasirinktos orientuojantis į esamus ar prognozuojamus tarptautinius protokolų standartus;
3. Lygmenų ribos turi būti pasirinktos taip, kad būtų minimizuotas duomenų srautas per sąsajas tarp lygmenų;
4. Lygmenų skaičius turi būti pakankamai didelis, kad skirtingos funkcijos, esant galimybei, būtų atskirtos į atskirus lygmenis, ir pakankamai mažas, kad struktūra netaptų griozdiška.

Fizinis lygmuo nusako bitų srauto perdavimą ryšio kanalu. Pagrindinė šio lygmens funkcija persiųsti informaciją iš mazgo į mazgą ir jos neiškraipyti, t. y. kai vienas tinklo mazgas siunčia 1, kitas turi tai suprasti kaip vienetą. Šis lygmuo susijęs su fizine signalų perdavimo aplinka bei mechaninėmis, elektrinėmis ir procedūrinėmis sąsajomis. Fiziniame lygmenyje numatoma:

- loginių signalų forma, įtampa, trukmė, kodavimo algoritmas;
- ryšio tipas (duplexiškumas);
- pradinis susijungimas ir seanso pabaiga;
- kabeliai, tinklinės jungtys ir jų išvadų panaudojimas.

Kanalinis (jungties) lygmuo perduoda duomenis į ryšio liniją bei priima neapdorotą duomenų srautą iš linijos ir atlikus klaidų kontrolę, perduoda į tinklinį lygmenį. Tai pasiekama suskaidant duomenis į duomenų kadrus, paverčiant juos nuosekliomis bitų sekomis, perduodant šias sekas kitam mazgui ir, panaudojus atsiųstus iš imtuvo patvirtinimo kadrus, taisant klaidas. Kanalinio lygmens programinė įranga turi pakartoti kadro siuntimą, jei įvyko klaida ar nebuvo atsiųstas patvirtinimo kadras, t. y. kanaliname lygmenyje turi būti išspręstos problemos, susijusios su kadro pažeidimu, praradimu ir dubliavimu. Šalia klaidų kontrolės kanaliname lygmenyje gali būti integruota srauto valdymo (flow control) – perdavimo reguliavimo, kai siųstuvas greitesnis negu imtuvas, galimybė. Kanalinis lygmuo turi suformuoti kadro pradžios ir pabaigos žymes, kadangi fizinis lygmuo perduoda ir priima bitų srautą be jokios prasmės. Paprastai tai padaroma kadro pradžioje ir gale pridedant specialų bitų rinkinį. Transliaciniuose tinkluose kanalinis lygmuo dažnai skaidomas į 2 polygmenius. Taip įvestas papildomas MAC (Media Access Control – kreipties į aplinką valdymas) polygmenis, skirtas priėjimo prie bendro kanalo kontrolei, susijęs su kreiptimi į tinklą – markerio perdavimu Token Ring ar kolizijų aptikimu Ethernet'e – bei jų valdymu. LLC (Logical Link Control), susijusi su vartotojo pranešimų siuntimu ir priėmimu, struktūriškai yra virš MAC polygmenio bei naudoja pastarojo teikiamas paslaugas kaip paprogramę.

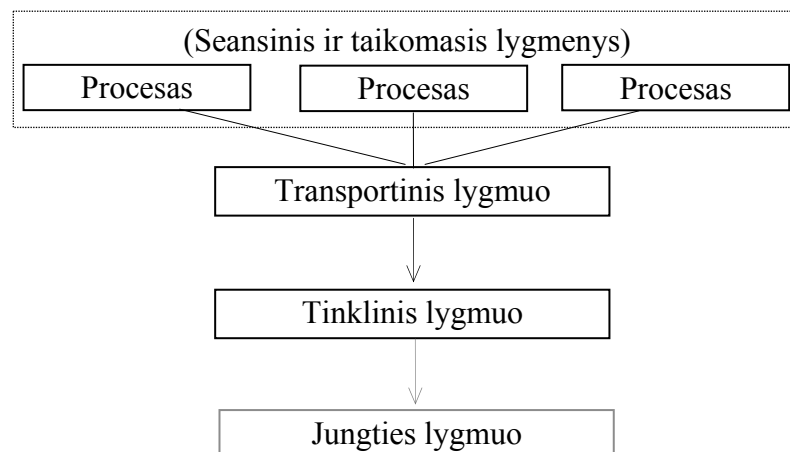
Kanaliniame lygmenyje naudojami Ethernet, Token Ring, FDDI protokolai.

Tinklinis lygmuo yra susijęs su potinklio (subnet) darbo valdymu. Potinkliu vadinama tinklo mazgų, kurių pagrindinė paskirtis yra duomenų retransliacija, visuma. Svarbiausia šio lygmens funkcija – informacinių paketų maršrutizavimas (nukreipimas) nuo šaltinio iki tikslo. Maršrutai gali būti paremti statinėmis, retai keičiamomis lentelėmis arba nustatomi kiekvienam paketui, pagal realų tinklo apkrovimą esamu momentu (dinaminis maršrutizavimas).

Tinkliniame lygmenyje sprendžiamos perkrovimų – tinklo kamščių problemos. Šiame lygmenyje sprendžiamos ir tokios perėjimo iš vieno tinklo į kitą problemos, kaip skirtingas adresavimas, protokolų neatitikimai, paketų ilgių skirtumai ir pan.

Tinklinio lygmens protokolų šeimai priklauso IP, ICMP, IPX, NetBEUI.

Transportinis lygmuo (16 pav.) duomenų failą, gautą iš sesijinio lygmens, suskaido į paketus ir perduoda juos į tinklinį lygmenį bei užtikrina, kad paketai neiškraipyti pasiektų adresatą ir būtų reikiama tvarka suformuoti į failą, identišką išsiųstajam. Visa tai turi būti padaryta efektyviai, greitai ir patikimai, nepriklausomai nuo įvairiose grandyse panaudotos aparatūros.



16 pav. Transporto lygmens funkcinės sąsajos. Atkreiptinas dėmesys, kad paveiksle pateiktoje diagramoje seansinis, vaizdavimo ir taikomasis lygmenys sujungti į vieną.

Šiuolaikiniuose tinkluose vyrauja paketų pristatymas netikrinant pristatymo eilės (failas sukomponuojamas pagal paketų sekos numerius) bei pranešimų tiražavimas didelėms adresatų grupėms. Vienas iš populiariausių transportinio lygmens sujungimų, ypač ankstyvuosiuose tinkluose, yra porinio jungimo kanalas, kuris pristato pranešimus tokia tvarka, kokia jie buvo išsiųsti ir tikrina klaidas.

Transportiniame lygmenyje atitinkamos šio lygmens sujungime dalyvaujančių kompiuterių programos pilnai sąveikauja tarpusavyje, nors fiziškai tai gali būti toli vienas nuo kito esantys mazgai, atskirti daugybe maršrutizatorių ir kitų tinklo elementų. Tuo tarpu žemesniuose lygmenyse tiesioginė sąveika įmanoma tik tarp artimiausių kaimynų, t. y. tik grandies ar vietinio tinklo ribose.

Paprastai transportinis lygmuo sukuria atskirą tinklinį sujungimą kiekvienam transportiniam sujungimui, kurio reikalauja sesijinis ir taikomasis lygmenys, tačiau tai nėra taisyklė: jei reikia, transportinis lygmuo gali sukurti keletą tinklinių sujungimų vienam transportiniam sujungimui ir atvirkščiai. Kadangi dauguma šiuolaikinių kompiuterių vienu metu gali vykdyti daugelį uždavinių, vienu metu gali egzistuoti daug sujungimų. Transporto lygmens protokolų antraštėse numatytas sujungimus atitinkančių pranešimų žymėjimas. Nurodant pranešimo prievadą (portą), apibrėžiami procesai, dalyvaujantys tarpkompiuteriniame informacijos pasikeitime. Srautų valdymo mechanizmo pagalba transportiniame lygmenyje suderinamas pasikeitimas duomenimis tarp greitų ir lėtų tinklo mazgų.

Transportiniame lygmenyje naudojami TCP, UDP, SPX protokolai.

Seansinis (sesijinis) lygmuo leidžia vartotojams organizuoti ryšio seansus tarp skirtingų kompiuterių. Tai yra lygmuo, kuriame atliekamas ryšyje dalyvaujančių kompiuterių dialogo valdymas, pvz., sesija gali būti panaudota vartotojui įsijungti į nutolusią kompiuterinę sistemą, leisti ar drausti perdavinėti duomenis abiemis kryptimis tuo pačiu metu, valdyti operacijų sekas naudojant žymės metodą ir pan. Seanso lygmenyje užtikrinama vardų, slaptažodžių identifikacija bei vartotojo teisių kontrolė.

Viena iš šio lygmens paslaugų – sinchronizacija. Pvz., jei perduodant failą nepatikimu kanalu į duomenų srautą įterpiami sinchronizacijos signalai, tai nutrūkus ryšiui ir pastarąjį atnaujinus galima pratęsti siuntimą nuo trūkio vietos.

Vaizdavimo lygmuo sietinas ne tiek su patikimu bitų perdavimu iš vieno tinklo mazgo į kitą, kiek su perduodamos informacijos sintakse ir semantika. Tam, kad esantys tinkle įvairių sistemų kompiuteriai galėtų sėkmingai keisti duomenimis, pastarųjų kodavimas, masyvų formatai, turi būti pateikiami vieningu standartu. Įvairūs kompiuterio elementai, pasiekiami tinklo vartotojams, pvz., terminalas, taip pat turi būti emuliuojami visiems suprantamais. Tai ir yra lygmens, kuriame atliekamos informacijos pateikimo standartų suderinimo operacijos, funkcija.

Taikomajame lygmenyje sukoncentruotos funkcijos, susijusios su vartotojų taikomosiomis programomis. Taikomojo lygmens funkcijų pavyzdžiais gali būti taikomosios elektroninio pašto, įsijungimo į nutolusį kompiuterį ar paskirstytųjų duomenų bazių programos, t. y. bet kokios

programos, skirtos galutiniam vartotojui. Tikslus tokių funkcijų apibrėžimas yra ypač svarbus tokių programų kūrėjams. Taikomajame lygmenyje turi būti išspręstos visos nesuderinamumo, pvz., terminalo emuliatoriaus problemos.

Taikomojo lygmens protokolai yra FTP, TELNET, SMTP...

Modelis “klientas – serveris”

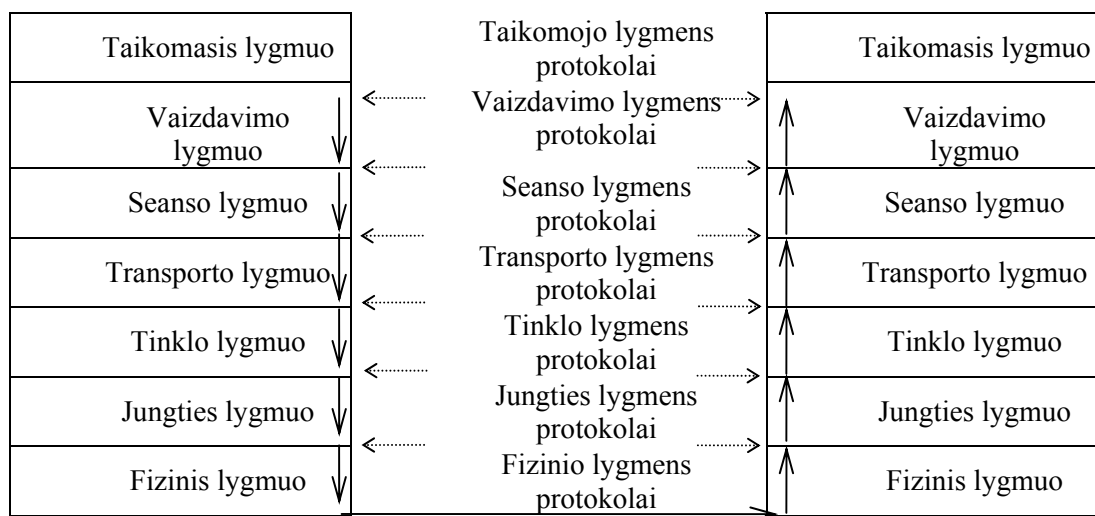
Nagrinėjant taikomąjį lygmenį ir siekiant supaprastinti taikomojo lygmens programinę įrangą įvedamas **“kliento – serverio” modelis**. Modelis “klientas – serveris” supaprastina tinklinės programos kūrimą, padalindama užduotį į dvi dalis: klientinę ir serverinę. Šiame modelyje laikoma, kad tinklinis sujungimas, o tuo pačiu ir taikomieji procesai yra dvipusiai. Tas pats taikomasis procesas skirtingose sujungimo pusėse vykdomas skirtingai – jis susideda iš klientinės ir serverinės dalių. Laikoma, kad klientas pateikia užklausą informacijai ar paslaugoms, o serveris atsako į užklausą, t. y. pateikia informaciją ar įvykdo paslaugą. Daugumoje atvejų tinklinės taikomosios programos sudarytos iš dviejų nepriklausomų dalių – kliento ir serverio, nors kartais šios abi funkcijos būna sujungtos į vieną programą. Gali būti, kad serveris nepajėgus įvykdyti užklausą, tada jis kreipiasi į kitą serverį, kuriam perduoda užklausą, pats tapdamas klientu.

Serverių programinė įranga skirstoma į du tipus pagal tai, kaip apdorojamos užklaustos: skiriami nuoseklaus bei lygiagretaus užklausų tvarkymo serveriai. Nuoseklaus užklaustos apdorojimo serveriai, kaip galima spėti iš pavadinimo, klientų užklaustas tvarko paeiliui, pagal jų pateikimo eiliškumą. Serveris iš karto imasi vykdyti gautą užklausą ir nereaguoja į naujus kreipinius, kol nebaigia darbo. Kada negalima numatyti užklaustos vykdymo laiko, serveris kiekvienai užklausiai sukuria atskirą procesą. Sukuriama tiek procesų, kiek yra kreipimųsi į serverį. Įvykdęs užklausą, serveris neveikos režime laukia sekančio kliento. Lygiagretaus užklausų vykdymo serverio konstrukcija reikalauja, kad kompiuterio operacinė sistema užtikrintų lygiagretumą, t. y. galimybę vienu metu vykdyti kelis procesus. Lygiagretaus užklausų vykdymo serveriai naudojami, pvz., failų perdavimui tinkle, nes neįmanoma numatyti failo perdavimo laiko, kadangi failo ilgis bei perdavimo sparta proceso pradžioje nežinomi.

Sąveika tarp tinklo kompiuterių

Programų požiūriu lygmuo N viename kompiuteryje sąveikauja su tuo pačiu lygmeniu kitame kompiuteryje (17 pav.). Programuotojo požiūriu visiškai nesvarbu, kaip informacija pasiekia atitinkamą lygmenį. Pavyzdžiui, nagrinėjant transporto lygmenį kuriame informacija suskaidoma į

paketus ir, persiuntus į vartotojo mazgą, vėl suformuojama į pradinį failą, nesvarbu per kokius maršrutizatorius, linijas vyksta informacijos perdavimas tarp kompiuterių, o taip pat ir tai, kokio turinio informacija perduota. Visa tai paliekama kitų, žemesnių arba aukštesnių lygmenų kontrolei. Taisyklės ir susitarimai, taikomi šiam ryšiui, vadinami N lygmens protokolu. Bendraujama horizontaliai, naudojant atitinkamo lygmens protokolą, tai yra nagrinėjamas virtualus N -tojo lygmens ryšys ir susidaro įspūdis, kad vienoduose lygmenyse esantys procesai komutuoja tarpusavyje.



17 pav. OSI modelis ir sąveika tarp tinklo kompiuterių. Punktyrais pažymėti virtualūs ryšiai tarp analogiškų lygmenų; ištiesine linija – realus informacijos perdavimas.

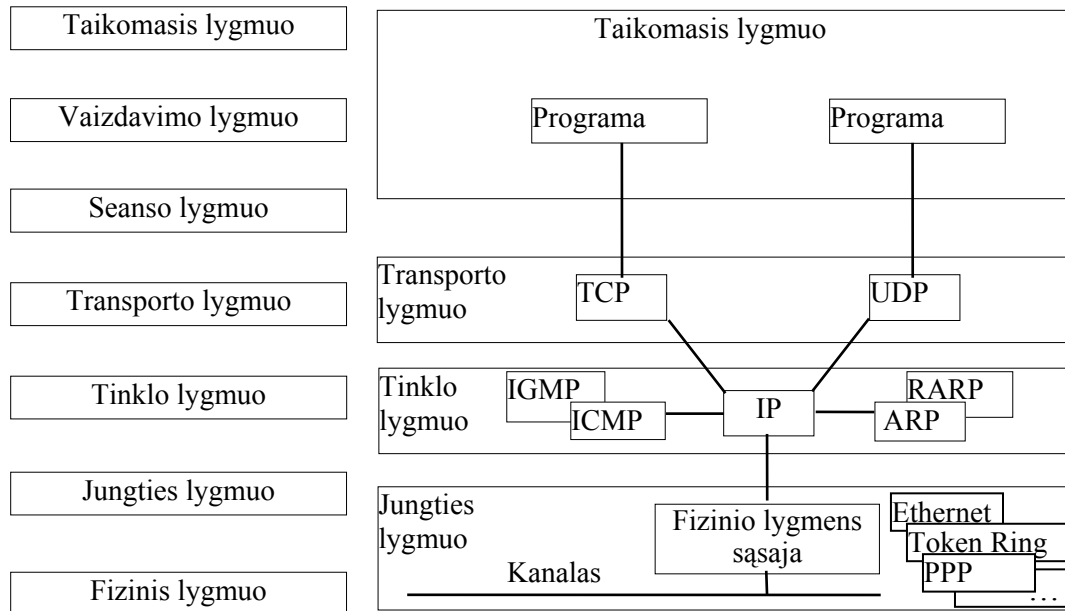
Fiziškai informacija perduodama iš aukštesnio lygmens į žemesnį, pasiekia fizinį lygmenį ir elektrinių ar optinių signalų pagalba perduodama į kitą kompiuterį. Adresato kompiuteryje praeinamas atvirkščias kelias iki atitinkamo lygmens.

Siekiant užtikrinti tokio proceso suderinamumą bei galimybę pakeisti vieną kurio nors lygmens realizaciją kita, būtina aiškiai apibrėžti sąsajas tarp lygmenų, nusakančias, koku būdu tarp jų perduodama informacija.

TCP/IP modelis

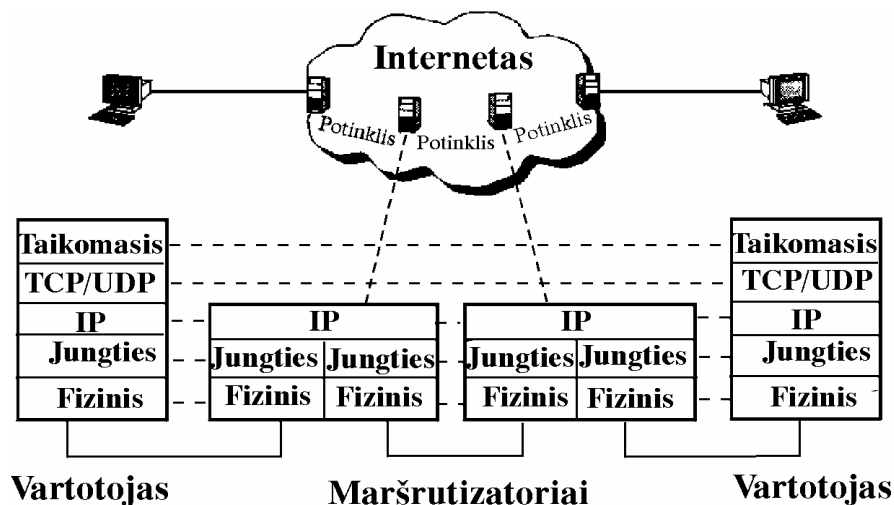
OSI modelis taikomas aprašant standartus, o praktinėms reikmėms dažniau naudojamas **TCP/IP modelis**. Dešimtmečiu anksčiau nei OSI priimtas TCP/IP modelis sudarytas iš keturių, OSI modelis –

iš septynių lygmenų. Nors skirstymo į lygmenis TCP/IP ir OSI modeliuose principai tie patys, patys lygmenys šiuose modeliuose nesutampa (18 pav.).



18 pav. OSI ir TCP/IP modelių palyginimas. Kairėje pateiktas OSI modelio skirstymas į lygmenis, dešinėje – TCP/IP lygmenys ir jų protokolai.

Kiekviename TCP/IP modelio lygmenyje vykdomos specifinės atitinkamo lygmens funkcijos. Visa kita paliekama kitų, žemesnių arba aukštesnių lygmenų kontrolei. TCP/IP modelis sukurtas veikiančių tinklo protokolų pagrindu, todėl jame nėra ypatingai ryškiai atskirtos sąsajos tarp gretimų lygmenų, kaip tai padaryta OSI modelyje, kuris kurtas kaip rekomendacinis modelis satandartams vystyti. OSI modelyje ryškiai atskirti protokolai, sąsajos ir paslaugos. Sąsajos nurodo, kaip įvairūs procesai gali pasiekti atitinkamą lygmenį, kaip informacija perduodama tarp lygmenų bei apibrėžia perdavimui reikalingus parametrus ir rezultatų laukus. Nusakant paslaugą nurodoma, kas atliekama aprašomame lygmenyje, nesigilinant, kaip tas lygmuo veikia. Taisyklės ir susitarimai, taikomi konkrečiam lygmeniui, vadinami N lygmens protokolu. Kadangi žemesniųjų lygmenų protokolai yra skaidrūs aukštesniųjų lygmenų atžvilgiu, t. y. pakeitus, pvz., tinklinio lygmens protokolą kita versija, aukštesnieji lygmenys (transportinis ar taikomasis) to nepajus ir dirbs kaip iki pakeitimo, susidaro išpūdis, kad vienoduose lygmenyse esantys procesai komutuoja tarpusavyje (19 pav.). Taigi, naudojant atitinkamo lygmens protokolą bendraujama horizontaliai, tai yra nagrinėjamas virtualus N-tojo lygmens ryšys.



19 pav. Interneto protokolų ir lygmenų sąveika.

Fiziškai informacija perduodama iš aukštesnio lygmens į žemesnį, pasiekia fizinį lygmenį ir elektrinių ar optinių signalų pagalba perduodama į kitą kompiuterį. Adresato kompiuteryje praeinamas atvirkščias kelias iki atitinkamo lygmens.

Siekiant užtikrinti tokio proceso suderinamumą bei galimybę pakeisti vieną kurio nors lygio realizaciją kita, būtina aiškiai apibrėžti sąsajas tarp lygmenų, nusakančias, koku būdu tarp jų perduodama informacija.

Duomenys Internetu perduodami keliais etapais (19 pav.):

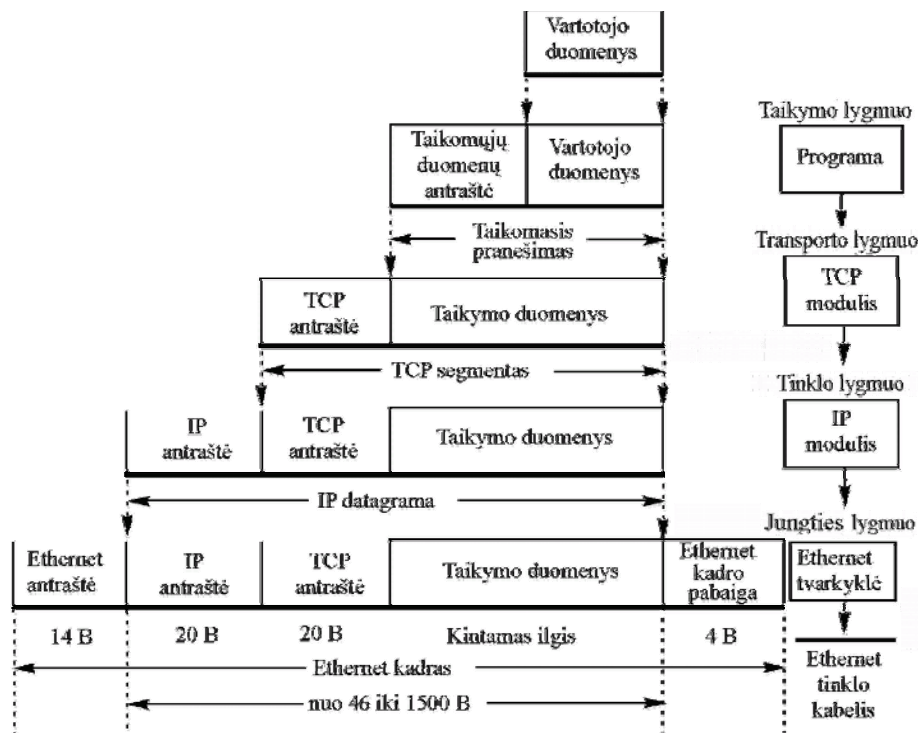
1. Informacija perduodama tarp taikomosios programos ir tinklo, tai yra steku žemyn iki fizinio lygmens.
2. Nustatomas artimiausio tarpinio mazgo IP adresas, pastarasis susiejamas su fiziniu adresu (pvz., MAC) ir, žingsnis po žingsnio, naudojant maršrutizacijos mechanizmus, duomenys perduodami į adresato mazgą (19 pav.).
3. Kildami steku aukštyn, duomenys pasiekia taikomąjį lygmenį.

Duomenų blokai įvairiuose lygmenyse paprastai vadinami skirtingai. Terminologija dažnai tiesiogiai neatspindi procesų, todėl labai svarbu gerai suvokti duomenų judėjimą ir pasidalijimą TCP/IP šeimoje. Duomenų tvarkymo proceso metu kiekvienas protokolas duomenis įvelka (inkapsuliuoja) į

pavidalą, tinkamą kaimyniniam steko protokolui. Taip leidžiantis steku žemyn, duomenys inkapsuliuojami būdingai kiekvienam lygmeniui, paruošiant juos tolesniam perdavimui (20, 21 pav.).

Jungties lygmuo ir jo protokolai

Jungties lygmenį TCP/IP steke galima suskirstyti į du polygmenius: jungties ir fizinį. Pastarąjį, kaip ir OSI modelyje, sudaro fizinis nešlys – ryšio kanalas. Tai gali būti vytos poros, optiniai ar

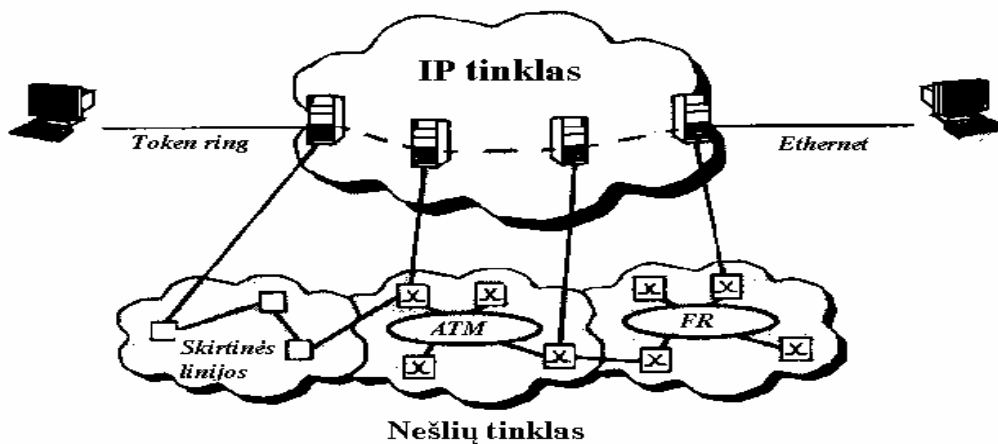


21 pav. TCP/IP duomenų inkapsuliacija, kai jungties lygmenyje informaciją perneša Ethernet tinklas.

koaksialieji kabeliai, radio ar palydovinio ryšio linijos, naudojami įvairūs kodavimo mechanizmai. Taikomųjų programų autoriui užtenka žinoti, kad prijungtas prie kompiuterio kabelis yra tinklo fizinis lygmuo. Jungties polygmenį, dažniausiai vadinamą tiesiog jungties lygmeniu sudaro aparatūros (fizinio lygmens) sąsaja.

Jungties lygmenyje sutvarkomas informacijos srautas tarp tinklinio lygmens ir fizinio duomenų perdavimo kanalo. Šalia to, jungties lygmuo, kaip ir visi kiti TCP/IP (ar kitokio tinklinio modelio) lygmenys paslepia konkrečias lygmenyje veikiančių protokolų realizacijos detales. Konkrečiai, jungties lygmuo paslepia nuo tinklinio lygmens fizinio kanalo funkcionavimo smulkmenas. Jeigu jungties lygmuo suprojektuotas gerai, tinkliniame lygmenyje nebus jaučiama kokiais tinklo

adapteriais ir kabeliais, kokiomis tinklo linijomis naudojamosi – tai gali būti Ethernet, Token Ring, FDDI, skirtinės linijos, kadru retransliacijos (Frame Relay, FR), ISDN, asinchroninio perdavimo režimo (Asynchronous Transfer Mode, ATM), X.25 ir kt. linijos (22 pav.).



22 pav. Įvairių linijų panaudojimas jungties lygmenyje ir aukštesnių lygmenų atskyrimas

Terminų žodynas

Access list	prieigos sąrašas
Access Port	prieigos prievadas
Adjacent router	gretutinis maršrutizatorius
Adjacency	gretimybė
Advertised Distance	deklaruojamas atstumas
Advertise	skelbti
Area border Router ABR	srities ribinis maršrutizatorius
Bandwidth	dažnių juostos plotis
Bidirectional communication	dvikryptis perdavimas
Carrier	nešlys
Carrier signal	nešlio signalas
Count to infinity	skaičiavimas iki begalybės
Congestion	perkrova
Classfull protocol	klasinis protokolas
Current Successor	esamas perėmėjas (žr. EIGRP)
Custom subnet mask	nestandartinė potinklio maskuotė
Data presentation protocol	duomenų pateikties protokolas
Dead interval	laukimo intervalas
Default Route	numatytasis kelias
Default Gateway	standartinis šliuzas
Delay	delsa
Designated Port	išskirtinis prievadas
Designated router	išskirtasis maršrutizatorius
Egress Port (interface)	išvesties prievadas
Enabled	aktyvus
Enquiry	užklausa
Feasible Successor	tikėtinas perėmėjas
Forward Delay	perdavimo delsa
Full-duplex	dupleksinis
Half-duplex	pusiau duplexinis
Header	antraštė
Holddown	atnaujinimų sulaikymas
Hello interval	atnaujinimų periodas
Inbound interface	įvesties prievadas
Ingress Port (interface)	įvesties prievadas
Interior Gateway Protocol	vidaus šliuzo protokolas
Internet protocol	tarptinklinis protokolas
Inquiry, Enquiry, Query	užklausa
Line protocol	linijos protokolas
Link state	sąsajos būseną
Max (maximum) Age	max intervalas
Multiple access	lygiagrečioji kreiptis
Multipoint connection	daugiataškis sujungimas
Neighbour router	kaimyninis maršrutizatorius

Non – negotiate	nesiderinantis tarpusavyje
Outbound interface	išvesties prievadas
Output	išvestis
Parallel port	lygiagretusis prievadas
Point-to-point connection	dvitaškis sujungimas
Query	užklausa
Request	užklausa
Root Bridge	pagrindinis komutatorius
Root Path Cost	pagrindinio kelio vertė
Root Port	pagrindinis prievadas
Routed Protocol	maršrutizuojamas protokolas
Routing Protocol	maršrutizuojantis protokolas
Serial interface	nuosekioji sąsaja
Serial port	nuoseklusis prievadas
Shared access	bendroji prieiga
Shutdown	išjungimas, sutrikdymas
Spanning Tree Protocol	siejančio medžio protokolas
Split Horizon	atkirstas horizontas
Standard Subnet mask	standartinė potinklio maskuotė
Subinterface	subprievadas
Triggered Update	proginis atnaujinimas
Transfer medium,	perdavimo terpė
Transfer medium,	perdavimo terpė
Traffic	(informacijos) srautas
Transport protocol	perdavimo protokolas
Trunk Port	magistralinis prievadas
Unicast	kryptingas perdavimas
Wildcard symbols	pakaitos simboliai

Feliksas Kuliešius, Kęstutis Svirkas, Artūras Rožė, Komutavimas ir maršrutizavimas lokaliuosiuose tinkluose, Metodinė priemonė (CD), VPU, 2008, ISBN 978-9955-800-37-8
Vilnius: Vilniaus pedagoginio universiteto leidykla, 2008.

ISBN 978-9955-800-37-8

Metodinėje priemonėje yra glaustai pateikiamos pagrindinės žinios apie lokaliuosius kompiuterių tinklus, apsiribojant svarbesnėmis ir sunkiau suprantamomis temomis. Priemonė skirta universitetų ir kolegijų studentams. Gali būti naudinga ruošiantis CCNA sertifikavimo egzaminui.