Kauno technologijos universitetas

Informatikos fakultetas

Kolizijų belaidžiame DCF tinkle valdymo metodas

Baigiamasis magistro studijų projektas

Kajus Kevevari

Projekto autorius

dr. Dangis Rimkus

Vadovas

Kaunas, 2020

Kauno technologijos universitetas

Informatikos fakultetas

Kolizijų belaidžiame DCF tinkle valdymo metodas

Baigiamasis magistro krypties studijų projektas

Informacijos ir informacinių technologijų sauga (6211BX008)

Kajus Kevevari

Projekto autorius

dr. Dangis Rimkus

Vadovas

doc. Gediminas Činčikas

Recenzentas

Kaunas, 2020

Kauno technologijos universitetas

Informatikos fakultetas

Kajus Kevevari

Kolizijų belaidžiame DCF tinkle valdymo metodas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Kajaus Kevevari, baigiamasis projektas tema „Kolizijų belaidžiame DCF tinkle valdymo metodas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

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

Kevevari, Kajus. Kolizijų belaidžiame DCF tinkle valdymo metodas. Magistro baigiamasis projektas vadovas dr. Dangis Rimkus; Kauno technologijos universitetas, informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): informatikos mokslas.

Reikšminiai žodžiai: kolizijų valdymas belaidžiame tinkle.

Kaunas, 2020. 75 p.

Santrauka

Belaidžių tinklų informacijos perdavimo elektromagnetinėmis bangomis prigimtis, dėl tinkle susiformavusių paslėpto įrenginio sąlygų, sukelia eteriu perduodamų kadrų susidūrimus – kolizijas. Vietinio belaidžio tinklo protokolas apibrėžia kanalo prieigos valdymo mechanizmą, kuris sumažina kolizijų kiekį tinkle, tačiau sukelia papildomas eterio laiko sąnaudas. Dažnu atveju, kanalo pasiekimo mechanizmo numatoma, kanalo rezervacija yra nenaudinga ir padidina tinkle esančių valdymo kadrų kiekį neefektyviai išnaudojant eterio laiką. Baigiamajame projekte pasiūlytas dinaminis kanalo prieigos valdymo metodas atsižvelgia į belaidžio tinklo dinamiką ir parenka efektyviausią kanalo prieigos mechanizmo konfigūraciją, sumažindamas kolizijų tikimybę ir sutaupydamas sunaudotą eterio laiką perduodant tą patį duomenų kiekį. Atlikto tyrimo metu nustatomi statinės kolizijų mechanizmo konfigūracijos trūkumai. Pademonstruota, kad sukurtas metodas pritaikydamas tinklo įrangos konfigūraciją sumažina nereikalingas eterio laiko sąnaudas.

Kevevari, Kajus. Method for Control of Collision in DCF Wireless Network. Master's Final Degree Project supervisor dr. Dangis Rimkus; Faculty of Informatics, Kaunas University of Technology. Study field and area (study field group): computer science.

Keywords: dynamic collision management in wireless network.

Kaunas, 2020. 75.

Summary

The nature of wireless network information transmission by electromagnetic waves is known to cause collisions due to formed “hidden station” scenarios. The wireless network protocol defines a media access management mechanism that reduces the number of collisions in the network but incurs additional airtime costs. In many cases, channel reservation is useless and increases the number of control frames in the network causing inefficient use of airtime. The method proposed in the final project takes into account the dynamics of the wireless network and selects the most efficient configuration of the media access mechanism, reducing the probability of collisions and saving airtime. The study reveals the shortcomings of the static configuration of the current media access mechanism. It has been demonstrated that the developed method can adapt the configuration of network equipment by reducing the unnecessary cost of airtime.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

ACK – angl. „acknowledgement“, patvirtinimo kadras;

AP – angl. „access point“, prieigos taškas;

BEB – angl. „binary exponential back off“, binarinis eksponentinis atsitraukimas;

BSA – angl. „basic service area“ – centrinės stoties dengiama zona;

BSS – angl. „basic service set“, tarpusavyje komunikuojantis belaidžio tinklo įrenginių pogrupis;

CBS – angl. „central BSS station“, centrinė stotis;

CA – angl. „collision avoidance“, MPDU kolizijos vengimo skaitiklis;

CCA – angl. „clear channel estimation“, laisvo kanalo įvertinimo procedūra;

CSMA/CA – angl. „carrier-sense multiple access with collision avoidance“, nešlio pajautos daugialypės prieigos su kolizijos vengimu mechanizmas;

CSMA/CD – angl. „carrier-sense multiple access with collision detection“, nešlio pajautos daugialypės prieigos su kolizijos aptikimo mechanizmas;

CTS – angl. „clear to send“, kanalo rezervacijos atsakymas;

CW – angl. „carrier wave“, nešlio banga;

CW1 – angl. „congestion window“, susidūrimo langas;

DCF – angl. „distributed coordinated function“, paskirstytoji koordinuotoji funkcija;

DIFS – angl. „DCF interframe space”, DCF tarpkadrinis intervalas;

IFS – angl. „interframe space“, tarpkadrinis intervalas;

MAC – angl. „media access control“, kanalo lygmuo;

MPDU – angl. „MAC protocol data unit”, MAC protokolo duomenų vienetas;

NAV – angl. „network allocation vector“, tinklo alokacijos vektorius;

QOS – angl. „quality of service“, paslaugos kokybė;

PIFS – angl. „PCF interframe space“ PCF tarpkadrinis intervalas;

PPDU – angl. „physical protocol data unit”, fizinio protokolo duomenų vienetas;

PHY – angl. „physical layer“, fizinis lygmuo.

RTS – angl. „request to send“, kanalo rezervacijos užklausa;

RTT – angl. „round trip time“, laiku matuojamas atstumas nuo pradžios iki tikslo ir atgal;

SIFS – angl. „short interframe space”, trumpasis tarpkadrinis intervalas;

SNR – angl. „signal to noise ration“, signalo ir triukšmo santykis;

STA – angl. „station“, stotis;

WLAN – angl. „wireless local area network“, vietinis belaidis tinklas;

Terminai:

Pagrindinis paslaugų rinkinys (angl. „basic service set“) – asocijuotų belaidžio tinklo įrenginių pogrupis, kuris naudoja bendras fizinio lygmens charakteristikas (radijo dažnis, moduliacija, saugumo konfigūracija) tarpusavio informacijos mainams.

Įvadas

Tinklo administratoriai vis dažniau vartoja terminą – išmanieji tinklai (angl. „self-aware/self-managed networks“). Šis terminas atsirado ankstyvu kompiuterinių tinklų formavimosi metu, atsiradus pirmam patikimumą užtikrinančiam tinklo lygmens protokolui (TCP), kuris, reaguodamas į tinklo grūsties būklę vertinant duomenų kelio laiką pirmyn ir atgal (RTT (angl. „round trip time“)), pritaiko pakartotinio persiuntimo delsą ir duomenų lango dydį [1].

Nuo pirmųjų atsiradusių lokalaus belaidžio tinklo – IEEE 802.11 (angl. „Wi-fi“) protokolu veikiančių įrenginių, belaidžio ryšio technologija patyrė drastiškus pasikeitimus gerinant informacijos srauto perdavimo greitį, paslaugos kokybę (angl. „quality of service“, QoS) ir plečiant aprėpties zoną. Nors ir belaidžio tinko technologijų kokybė nepalyginamai pagerėjo, tačiau „Wi-fi“ tinko sparta išlaiko stipriai priklauso nuo elektromagnetinių (EM) bangų prigimties.

Belaidžio ryšio įranga dėl informacijos nešlio (angl. „carrier“) prigimties patiria paslaugos kokybės sutrikimus. Pernešančios informaciją elektromagnetinės bangos yra veikiamos radijo triukšmo, signalo atsispindėjimų, interferencijų ir energijos nuotekų iš gretimų kanalų. Prigimtinė elektromagnetinių bangų įtaka veikia belaidžio tinklo spartą, sukeldama duomenų kadrų susidūrimus – kolizijas.

Eterio pasiekimo kontrolės trūkumas yra dažna kolizijų priežastis. Bendra informacijos mainų erdvė verčia įrenginius konkuruoti dėl komunikacijos laiko. Skirtingų įrenginių perdavimo galios ir priėmimo jautrumo nesuderinamumai taip pat kaip ir atstumas tarp komunikuojančių įrenginių, sudaro sąlygas paslėpto įrenginio (angl. „hidden node“) scenarijui susiformuoti. Neteisingas eterio laiko paskirstymas, perteklinis aktyvus tinklo skenavimas, klientų migracijos delsos kyla dėl protokolo lygmens ir tiekėjų klaidingos programinės įrangos. Šios priežastys iš vartotojo perspektyvos pasireiškia kaip belaidžio tinklo spartos sumažėjimas.

Be perstojo sudėtingėjant radijo ryšio techninei ir programinei įrangai, produkcinės radijo aplinkos diagnostikos procesas tampa sudėtingas. Dėl unikalių aplinkos sąlygų, tokių kaip radijo taršos lygmuo, girdimų įrenginių kiekio, kliūčių išsidėstymo įrenginių dislokavimo vietoje, diagnostikos procesą sudėtinga automatizuoti. Rankinės diagnostikos metodai, vertinant tūkstančius besikeičiančių parametrų, tampa neefektyvūs bei susiduria su laiko, žmogiškųjų išteklių stokos ir kvalifikacijos trūkumo kliūtimis.

Tiriamojo darbo metu siekiama išanalizuoti belaidžio tinklo spartos suprastėjimą, ištikus duomenų kolizijoms; pademonstruoti esamų kolizijų valdymo mechanizmų trūkumus, analizuojant surinktus empirinius duomenis. Siekiama suprojektuoti ir realizuoti kolizijų valdymo metodą, atsižvelgiant į aplinkos dinamiką. Kuriamo metodo tikslas yra padidinti eterio laiko išnaudojimo efektyvumą ir tinklo spartą. Tiriamojo darbo tikslui įgyvendinti iškelti šie *uždaviniai*:

1. išanalizuoti belaidžių tinklų kolizijų atsiradimo priežastis;
2. išanalizuoti esamus kolizijų valdymo metodus;
3. suprojektuoti kolizijų valdymo metodą atsižvelgiantį į aplinkos dinamiką;
4. realizuoti kolizijų valdymo metodo prototipą;
5. sudaryti tyrimo metodiką ir ją taikant ištirti sukurtą metodą.

Darbo struktūra sudaryta iš 4 pagrindinių skyrių: analitinės dalies (1 skyrius), kurioje analizuojama probleminė sritis, esami sprendimo metodai ir jų trūkumai; projektinės dalies (2 skyrius), kurioje projektuojamas kolizijų valdymo metodas; prototipo dalies (3 skyrius), kurioje aprašoma suprojektuoto metodo prototipo realizacija; tyrimo dalies (4 skyrius), kurioje sudaroma tyrimo metodika ir atliekamas sukurto kolizijų valdymo metodo tyrimas. Atlikus visus užsibrėžtus uždavinius pateikiamos darbo išvados ir atlikto tyrimo rezultatai (5 skyrius).

1. Kolizijų valdymo metodo analizė

Siekiant geriau suprasti probleminę sritį, šiame skyriuje atliekama detali probleminės srities analizė, kuri sudaryta iš 5 poskyrių. Pirmiausia detaliai aprašoma sprendžiama problema (1.1). Atliekama kolizijų belaidžiuose tinkluose atsiradimo priežasčių analizė (1.2). Apžvelgiami belaidžio ryšio 802.11 „Wi-fi“ protokolų rinkinio kolizijų valdymo mechanizmai (1.3). Apžvelgiami esamų kolizijų valdymo mechanizmų trūkumai (1.4). Analizuojami pasiūlyti probleminės srities sprendimo metodai (1.5). Pateikiamos analitinės dalies išvados ir suformuojamas darbo tikslas (1.6).

1.1. Probleminė sritis

Belaidžiuose tinkluose eteris yra dalijamas tarp visų vienoje BSA (angl. „base station area“) zonoje veikiančių įrenginių, vadinamų stotimis STA (angl. „stations“). Norint dalyvauti belaidžio tinko komunikacijoje stotys privalo palaikyti 802.11 MAC (angl. „media access control“) ir PHY (angl. „physical layer“) lygmens protokolus. Visos BSA stotys perduoda informaciją tuo pačiu fiksuoto pločio ir poslinkio kanalu per centrinę valdymo stotį [2] (angl. „central base station“, CBS), dar vadinamą prieigos tašku (angl. “access point”, AP). Vienu laiko momentu naudojant vieną kanalą, informaciją gali siųsti tik viena BSA zonos stotis. Perduodant duomenis tuo pačiu metu, nešlio signalų (angl. „carrier wave”) elektromagnetiniai laukai susiduria ir iškraipo moduliuojamą informaciją, kuri tampa neperskaitoma priimančiame įrenginyje. Esant daugiau nei vienai stočiai BSA zonoje, atsiranda būtinybė apibrėžti kanalo pasiekimo taisykles užtikrinant stočių informacijos perdavimą skirtingais laiko momentais.

Vietinio belaidžio paskirstytos koordinuotosios funkcijos tinklo (angl. „wireless local area network”, WLAN) protokolų rinkinys (802.11) stočių informacijos perdavimui eteriu valdyti naudoja mechanizmą CSMA/CA (angl. „carrier sense multiple access with collision avoidance”) [2][skyrius „10.2.2 DCF“] ir BEB (angl. „binary exponential back off“) [3]. Šių mechanizmų kombinacija yra priskiriama prie paskirstytosios koordinuotosios funkcijos (angl. „distributed coordinated function“, DCF) infrastruktūrinio tinklo.

Infrastruktūrinis DCF mechanizmas veikia pagrindiniu arba pasirenkamuoju režimu. Pagrindinis (angl. „basic“) režimas užtikrina stočių informacijos perdavimą radijo ryšiu skirtingais laiko momentais, atlikdamas laisvo kanalo įvertinimą CCA (angl. „clear channel assessment“), prieš siunčiant. Pasirenkamasis (angl. „optional“) režimas užtikrina kanalo rezervaciją RTS/CTS – „rankos paspaudimo“ metodu. Taip pat pasirenkamasis režimas numato galimybę apibrėžti RTS/CTS aktyvacijos ribą, kurios pagalba reguliuojama kanalo rezervacijos būtinybė. Abu kolizijų valdymo mechanizmo režimai valdo belaidžio tinklo eterio pasiekimo taisykles belaidžiame tinkle.

Pagrindinis DCF kanalo pasiekimo režimas yra efektyvus eterio laiko atžvilgiu tik tuo atveju, jei BSA zonoje visi įrenginiai gali girdėti vienas kitą, t. y., šis režimas nesprendžia paslėpto įrenginio problemos [4]. Tuo tarpu, pasirenkamasis režimas užtikrina visų BSS (angl. „basic service set“) esančių stočių kanalo pasiekimą, išvengiant kolizijų, tačiau jis drastiškai padidina informacijos srauto sulėtėjimą ir sunaudoja kritiškai didelę eterio laiko dalį, nešdamas mažą informacijos kiekį [3]. Belaidžio ryšio įrenginiai naudoja pagrindinį kanalo pasiekimo valdymo režimą kaip numatytąjį, kuris, susidarius paslėpto įrenginio sąlygai, nesuvaldo padidėjusio kolizijų skaičiaus ir suprastėjusio pralaidumo. Tai padidina kolizijų skaičių tinkle, mažinant tinklo spartą ir didinant komunikacijos delsą.

1.2. Kadrų kolizijos WLAN tinkle ir jų atsiradimo priežastys

Kolizijos belaidžiame tinkle atsiranda, kai dvi tokio paties ar panašaus dažnio elektromagnetinės bangos susiduria tame pačiame taške. Tuo metu elektromagnetiniai laukai, besijungdami iškraipo moduliuojamą informaciją ir dalis arba visa siunčiama informacija tampa neperskaitomi, įvyksta EM bangų susijungimas – kolizija.

Elektromagnetinių bangų EM kolizijas galima skirstyti pagal jų atsiradimo šaltinius į elektromagnetinius prietaisus ir aplinkos triukšmus. Prietaisai gali siųsti nešlio bangas CW (angl. "carrier wave") informacijos perdavimo tikslais, kaip BSS stotis ar mobiliojo ryšio įrenginys. Kitu atveju, prietaisai gali generuoti informacijos neperduodančias EM bangas, kaip mikrobangų krosnelės, elektriniai šilumos generatoriai.

Belaidžiuose tinkluose kolizijos fiksuojamos tuo metu, jei gavėjas, priėmęs paketą pripažįsta, kad jis yra sugadintas arba per nustatytą laiko tarpą siuntėjas negauna perdavimą patvirtinančio kontrolės kadro pažymėto ACK (angl. „acknowledgement") bitu. Sugadintą paketą gali parodyti negaliojanti (angl. „invalid") kontrolinė suma, klaidingai demoduliuotos aukštesniųjų protokolų antraštės ar nepavykęs kadro pranešimo (angl. „payload") iššifravimas. Tokios pasekmės atsiranda dėl paketų susidūrimo eteryje arba dėl aplinkos sukeltų triukšmų perdavimo metu [9][8].

1.2.1. Aplinkos triukšmų sukeliamos kolizijos

Aplinkos arba kanalo triukšmai padažnėja, kai priimto kadro signalo ir triukšmo santykis (angl. „signal to noise ratio", SNR) [9] sumažėja. Tai gali reikšti suprastėjusias radijo aplinkos sąlygas, kurias parodo aukštas triukšmo grindų (angl. „noise floor") matmuo arba pasikeitusi stoties komunikacijos kanalo kokybė. Radijo aplinkos sukeliamos žemo SNR priežastys gali būti signalo kelio praradimas, trajektorijos išnykimas (angl. „multipath fade") arba baltasis triukšmas. Dėl įvardytų priežasčių eteryje atsiranda neperskaitomi kadrai, kurie priimančiame įrenginyje yra šalinami. Išvardintos aplinkos triukšmo priežastys atsiradusios duomenų perdavimo metu tarp BSS stočių sukelia radijo bangų interferencijas ir sutrikdo komunikaciją. Įvardyti veiksniai sukelia belaidžio tinklo informacijos srauto sulėtėjimą ir sumažina spartą.

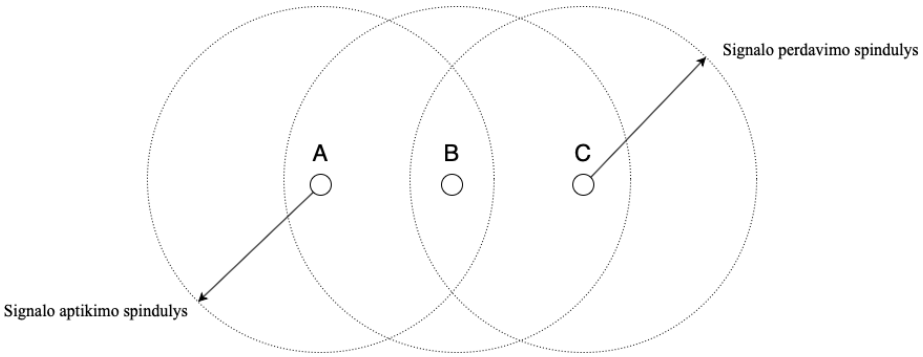
Deja aplinkos sąlygų higiena priklauso nuo visų radijo perimetre operuojančių įrenginių elgsenos, todėl užtikrinti, kad visi įrenginiai elgsis korektiškai negalima. Dėl šios priežasties aplinkos triukšmų sukeltos kolizijos toliau darbe nebus nagrinėjamos.

1.2.2. Susidūrusių kadrų kolizijos

Susidūrusių kadrų kolizija įvyksta, kai daugiau nei viena stotis perduoda duomenis tuo pačiu laiko momentu viename tinklo segmente. Kolizijos gali įvykti kai siunčiami kadrai persidengia laike, būdami per arti gavėjo [5] ir EM bangos susiduria. Toks perdavimas gali įvykti kai stotis neteisingai atlieka laisvo kanalo įvertinimą CCA arba įvykus atsitiktinio atsitraukimo sinchronizacijai. Dažnai neteisingai vertinamas laisvas kanalas gali būti tuo atveju, jei stotis negirdi siunčiamos informacijos, t. y., susiformuoja paslėpto įrenginio scenarijus. Daug rečiau kolizijos nutinka dėl sutapusios atsitiktinai pasirenkamos atsitraukimo intervalo vertės – atsitiktinio atsitraukimo sinchronizacijos.

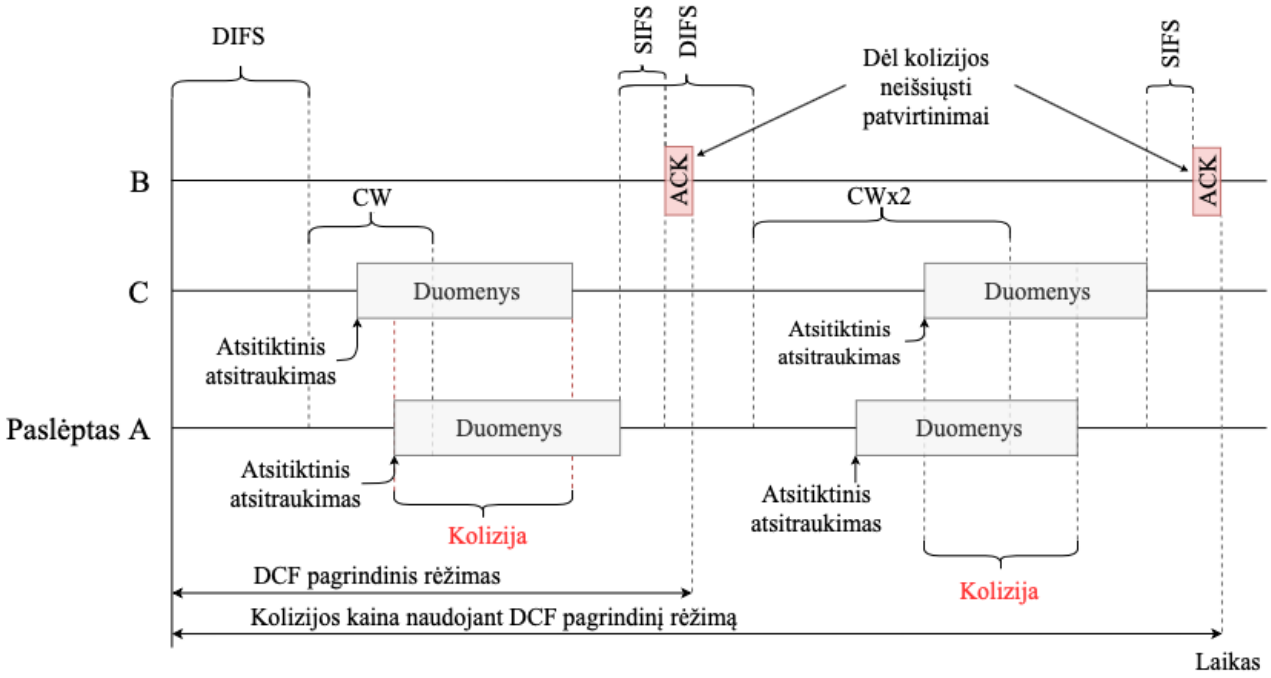
1.2.2.1. Paslėpto įrenginio sukeliamos kolizijos

Pradedant siuntimo operaciją, DCF infrastruktūriniame tinkle stotis atlieka laisvo kanalo įvertinimo CCA procedūrą, kurios metu pasiklausomas eteris ir nustatoma, ar juo naudojasi kita BSS stotis. Kanalas vertinamas kaip laisvas tuo atveju, kai per DIFS (angl. „DCF interframe spacing“) laiko intervalą (34 µs 802.11ac [5]) kanale nėra girdima jokia radijo transliacija.



1 pav. Paslėpto įrenginio scenarijus

Laisvo kanalo įvertinimas yra efektyvus tik tuo atveju, kai stotys gali girdėti visas kitas BSS stotis. Skirtingų įrenginių perdavimo galios ir priėmimo jautrumo nesuderinamumai, taip pat kaip ir nuotolis nuo BSS centrinės stoties dažnai yra paslėpto įrenginio scenarijaus priežastis. Esant tokiai situacijai (1 pav.) vienos iš BSA stočių (A) (paslėpta stotis) nepasiekia kitos stoties (C) siunčiamas signalas. Tokiu atveju stotis (A) negirdi kai stotis (C) perduoda kadrus centrinei stočiai (B). CCA klaidingai įvertina, kad kanalas yra laisvas ir inicijuoja perdavimą. Susidaro situacija, kai (A) ir (C) siunčia kadrus centrinei BSS stočiai sukeliant kadrų susidūrimus – kolizijas.

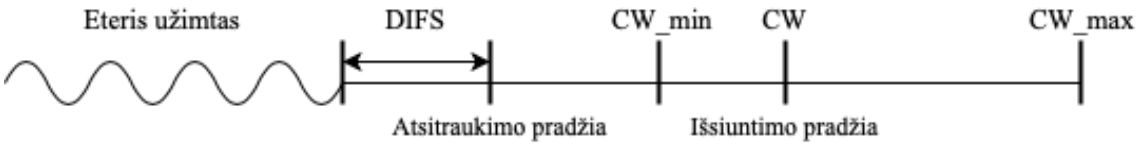


2 pav. Kolizijos kaina naudojant pagrindinį režimą

Pavaizduotame eterio laiko ir stočių perduodamų duomenų modelyje (2 pav.) matoma, kaip stotis (A) pradeda transliaciją, kai kanalas yra užimtas ir duomenų kadrai persidengia laike. Tokia situacija WLAN tinkluose yra vadinama paslėptos stoties problema. Susiformavus paslėptos stoties sąlygoms, kolizijų skaičius tinkle išauga drastiškai. Tokiu atveju CSMA/CA mechanizmas neveikia ir komunikacijos sparta tinkle mažėja.

1.2.2.2. Atsitiktinio atsitraukimo sinchronizacijos sukeliamos kolizijos

Kitas kadrų kolizijų šaltinis belaidžiuose tinkluose yra BEB atsitraukimo sinchronizacija. Didžiausia tikimybinė kolizijos rizikos zona laiko atžvilgiu yra pasibaigus komunikacijai tarp dviejų BSS įrenginių. Tokiu metu visos tinkle esančios stotys, norinčios perduoti duomenis, siekia užimti kanalą. Prieš perduodant bet kokio tipo duomenis, ar tai būtų vienas kadras, ar kadrų srautas, (angl. „back to back“) atliekama laisvo kanalo įvertinimo procedūra CCA pagal BEB algoritmą, siekiant sumažinti kolizijų tikimybes po pasibaigusio duomenų perdavimo. Kanalas DCF tinkle vertinamas kaip laisvas tada, kai atitinkamą laiko intervalą DIFS (angl. „DCF interframe spacing“ [2][skyrius „10.3.2.3 IFS“]) kanale nėra girdima jokia radijo transliacija. Priešingu atveju inicijuojama atsitraukimo (angl. „back off“) operacija, kurios metu atsitiktinai pasirenkama atsitraukimo intervalo vertė (skaičiuojamas laiko tarpais angl. „slots“) iš konkurencijos lango CW (angl. „contention window“) intervalo [11]. Atliekamas kanalo pasiklausymas, mažinant atsitraukimo skaitiklio vertę kiekvieną laiko vienetą, kai kanalas nustatomas laisvas. Skaitikliui išsekus, stotis interpretuoja eterį kaip laisvą ir pradeda kadrų siuntimo operaciją.



3 pav. Atsitiktinio atsitraukimo skaitiklio pasirinkimas

Esant fiksuoto dydžio atsitraukimo intervalui (3 pav.) didelės apkrovos tinkle egzistuoja tikimybė, kad dviejų stočių atsitraukimo skaitiklio vertė išseks tuo pat metu t. y. įvyks atsitiktinio atsitraukimo sinchronizacija. Jei stoties kadro perdavimas po atlaisvėjusio kanalo yra nesėkmingas, tada BEB algoritmas dvigubai padidina konkurencijos langą CW, taip sumažindamas atsitraukimo intervalo sinchronizacijos tikimybę. Eksponentinis konkurencijos lango didinimas taip pat turi neigiamą įtaką eterio laiko efektyvumui. Stočiai pasirinkus atsitraukimo intervalą, kuris yra arti maksimalios konkurencijos lango reikšmės, susidaro situacija, kad atlaisvėjus kanalui stotis kurį laiką negalės perduoti kadrų, nes pasirinktas skaitiklis nėra išsekęs [6]. Tokia situacija vėlina kadrų perdavimą ir atitolina kitų tinklo dalyvių komunikacijos pradžią, mažinant bendrą tinklo spartą.

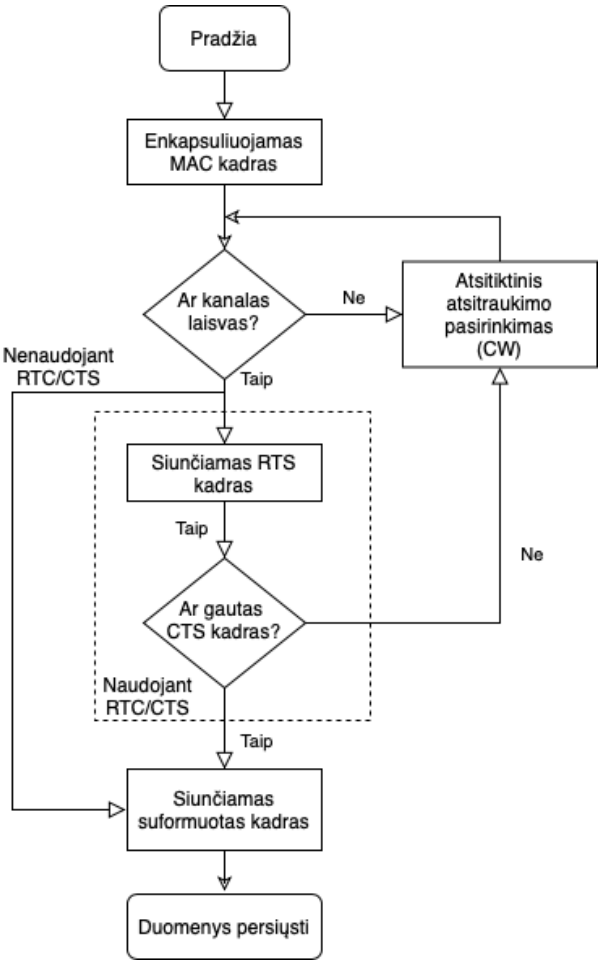
1.3. Kolizijų aptikimo ir valdymo mechanizmai DCF tinkle

DCF yra pagrindinis bendro eterio prieigos mechanizmas, palaikantis asinchroninį duomenų perdavimą geriausios pastangos (angl. „best effort“) principu. Kaip specifikuota WLAN protokole [1 - skirsnyje 9.3], DCF infrastruktūrinio tinklo atveju duomenų perdavimas yra paskirstytas t. y. perdavimo procesą inicijuoja BSS įrenginys [2][skyrius 10.2.2 „DCF“] (STA arba AP), kuris nori perduoti kadrą kitam tinklo įrenginiui. Kadrų susidūrimų įtakai valdyti 802.11 protokole yra numatyti CSMA/CA ir BEB kolizijų valdymo mechanizmai. CSMA/CA mechanizmas turi du darbo režimus t. y. gali naudoti tik nešlio pasiklausymą (pagrindinis režimas) arba nešlio pasiklausymą su kolizijų

valdymu (pasirenkamasis). Pasirenkamasis režimas numato eterio rezervaciją apsikeičiant trumpais kontrolės kadrais.

1.3.1. Kolizijų aptikimas nešlio pajauta pagrindinis režimas

Pagrindinis kolizijų valdymo mechanizmo režimas tinklo duomenims apsikeisti pirmiausia buvo specifikuotas laidiniuose lokaliuose tinkluose (802.3 protokolo CSMA/CD [7]), kaip kolizijų dedukcijos mechanizmas, valdantis perdavimo terpę nuo paketų susidūrimo. WLAN DCF kanalo prieigą 802.11 protokole [2][skyrius 10.2.2 „DCF“] nustato atsitiktinio atsitraukimo kanalo prieigos ir kolizijų valdymo mechanizmus CSMA/CA.



4 pav. DCF kolizijų valdymas naudojant ir nenaudojant RTS/CTS

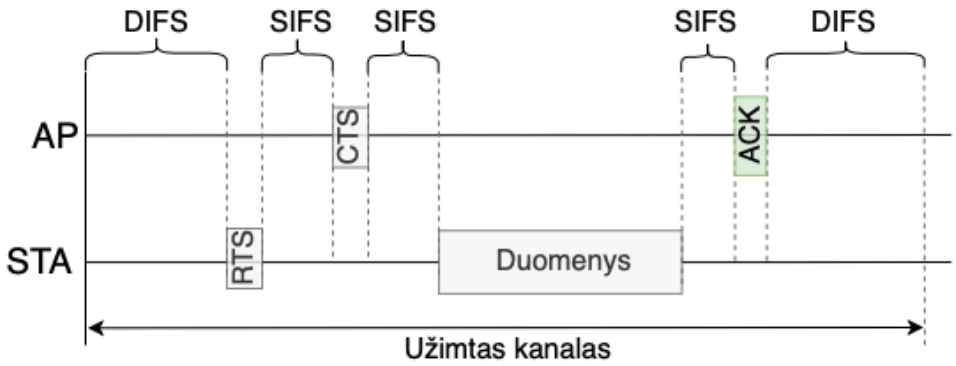
Pagrindinis DCF kolizijų valdymo mechanizmas naudoja CSMA ir BEB. Stotis, kuri nori perduoti duomenis eteriu pirmiausia privalo įvertinti ar eteris yra laisvas atliekant CCA, pasiklausant kanalo fiksuotą laiko tarpą – DCF tarpkadrinį laiko tarpo DIFS. Jei eteris yra laisvas, stotis siunčia kadrus, nenaudojant RTS/CTS (angl. „request to send/clear to send“)(1 pav.). Šis algoritmas yra veiksmingas ir užtikrina maksimalią duomenų spartą, kai įrenginiai geba teisingai įvertinti kanalą (t. y. vykdoma CCA procedūra teisingai nustato, kad kanalas yra laisvas).

1.3.2. Kolizijų aptikimas nešlio pajauta pasirenkamasis režimas

Pasirenkamasis kolizijų valdymo režimas buvo pasiūlytas paslėpto įrenginio problemai spręsti. Komunikacijos pradžioje, naudojant pasirenkamąjį režimą, stotis privalo atlikti kanalo rezervacija

prieš perduodant kadrus [1]. Rezervacija yra atliekama naudojant trumpus kontrolė kadrus. Šie kadrai gali būti RTS/CTS kontrolės tipo, kurie yra suprantami ir patikimi (angl. „robust“) įvairia moduliacija operuojantiems klientams arba trumpi duomenų/patvirtinimo (DATA/ACK) kadrai, kurios de-moduliuoti galės tik ta pačia moduliacija operuojančios stotys.

WLAN protokole (802.11) terpės pasiekimo prioritetas yra kontroliuojamas naudojantis tarpkadriniais laiko tarpais IFS (angl. „interframe space“), kurie numato laiko atskirtį tarp terpe keliaujančių kadrų. Protokole DCF tinklui specifikuoti du intervalai [2][skyrius „10.3.2.3 IFS“]: trumpas IFS – SIFS (angl. „short interframe spacing“) ir paskirstyto koordinavimo funkcijos IFS – DIFS. Šie intervalai skiriasi savo ilgiu, SIFS yra trumpas – 16 µs 802.11ac, o DIFS ilgas –34 µs 802.11ac.



5 pav. Pasirenkamasis RTS/CTS kanalo prieigos mechanizmas

RTS/CTS mechanizmo kadrai (5 pav.) naudojami siekiant sumažinti kolizijų kiekį kai tinkle egzistuoja paslėptos stotys. RTS (angl. „request to send“) siunčiamas stoties, kuri nori perduoti duomenis, reiškiantis prašymą radijo terpes laiko rezervacijai. Priimanti stotis (šiuo atveju AP) atsako su CTS kadru (angl. „clear to send“), duodant leidimą perduoti duomenis. Įvykus sėkmingai kanalo rezervacijai visos kitos BSS stotys priėmusios CTS kadrą, kuris yra skirtas perduodančiai stočiai, susilaiko nuo duomenų perdavimo, naudodamos tinklo alokacijos vektorių NAV (angl. „network allocation vector“), kaip virtualų kolizijos valdymo mechanizmą.

RTS/CTS kontrolės kadrai yra trumpesni nei tušti duomenų kadrai ir yra perduodami mažiausia tinklo stočių palaikoma sparta. Visos stotys gali demoduliuoti šiuos kadrus ir elgtis atitinkamai, todėl yra mažesnė tikimybė įvykti duomenų kolizijai nei siunčiant trumpus duomenų kadrus. Jei stotis girdi RTS kadrą, tačiau negirdi CTS, galima teigti, kad stotis yra atskirta.

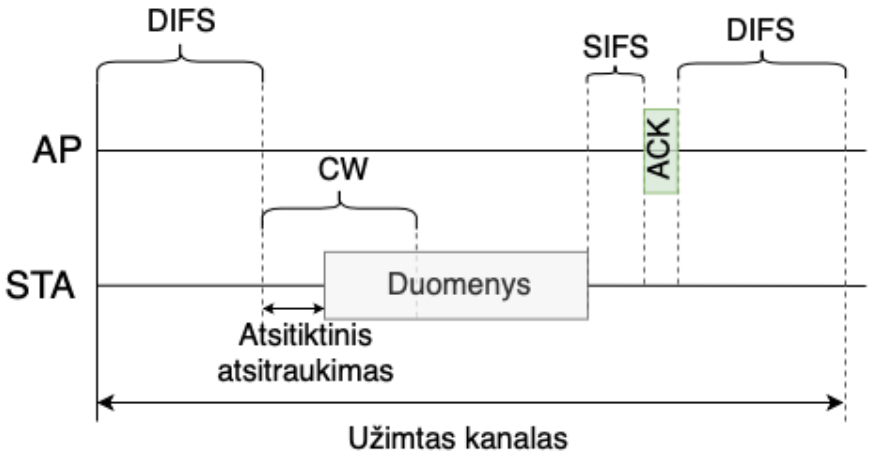
1.3.3. Tinklo alokacijos vektorius

Pasirenkamasis kolizijų valdymo mechanizmas papildomai naudoja ir virtualų eterio valdymo metodą. NAV tinklo alokacijos vektorius yra virtualus eterio jutimo mechanizmas, kuris gali būti suprantamas kaip skaitiklis, kuris sumažėja iki nulio. Kai skaitiklis yra lygus nuliui, laikoma, kad terpė yra laisva, priešingu atveju terpė yra užimta. Kiekvienas RTS/CTS kadras turi laiko argumentą CS, kurio reikšmė yra kanalo rezervacijos laikas nuo paskutinio PPDU nešamo MAC kadro, šiuo metu vykstančiai komunikacijai nustatyti. Kanalo rezervacijos laukas adresuojamas visų terpę pasiklausančių kaimyninių įrenginių, NAV skaitiklių atsinaujinimui. Naudojant šį virtualų kolizijų

valdymo metodą stotys išvengia kolizijų, išnaudodamos minimalų energijos kiekį. NAV leidžia nustatyti užimtą eterį nenaudojant fizinės kanalo pajautos (angl. „sensing“).

1.3.4. Binarinis eksponentinis atsitraukimo algoritmas

Esant užimtam kanalui, stotis atideda duomenų perdavimą su papildomu atsitiktiniu pridėtiniu laiku, pasirinktu iš konkurencijos lango intervalo [8] (angl. „contention window“ – CW)(6 pav.). Taip užtikrinama, kad atlaisvėjus kanalui, visos tinkle esančios stotys nepradėtų siuntimo tuo pačiu momentu, sukeliant kadrų koliziją. Kai stotis pradeda siuntimo operaciją kanalu, norint išlaikyti kanalą užimtą,, palieka trumus tarpus tarp kadrų vadinamus SIFS. Kita stotis negalės pertraukti siuntimo, nes CCA reikalauja užfiksuoti laisvą kanalą DIFS trukmę, kuri yra ilgesnė už SIFS.

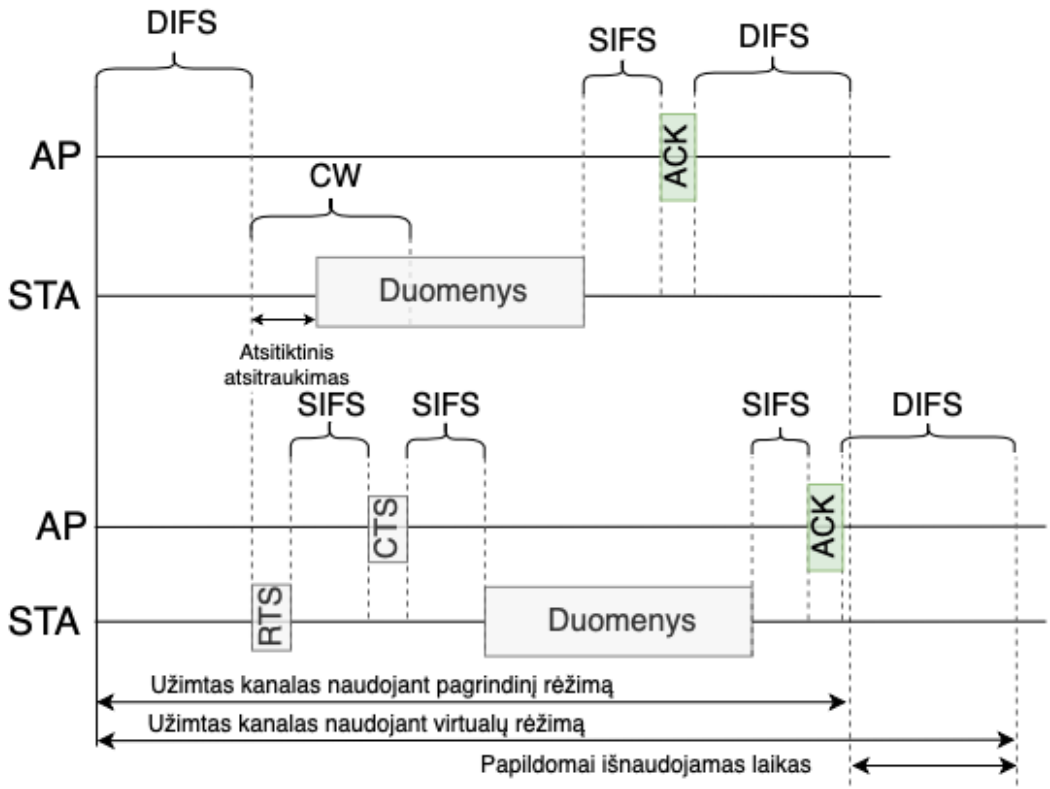


6 pav. pagrindinis DCF kanalo prieigos mechanizmas

Po pasibaigusios komunikacijos visos tinklo pasiklausančios STA yra pasiruošusios perduoti duomenis, todėl tokiais laiko tarpais kolizijos tikimybė yra didžiausia. Nepavykus pradėti komunikacijos atlaisvėjus kanalui, t. y. kadras pažymėtas ACK bitų negaunamas, manoma, kad įvyko kolizija ir kadras siunčiamas pakartotinai pasirinkus atsitiktinio atsitraukimo intervalą iš padvigubėjusio konkurencijos lango [6]. Toks eksponentinis CW augimas vadinamas binarinio eksponentinio atsitraukimo algoritmu. Šis algoritmas užtikrina kadrų perdavimo pradžią skirtingais laiko momentais po pasibaigusios komunikacijos.

1.4. Kolizijų valdymo mechanizmo trūkumai

Įvardinti DCF kolizijų valdymo mechanizmai yra efektyvūs, tačiau RTS/CTS „rankos paspaudimas“ gali taip pat neigiamai paveikti WLAN spartą sukeliant papildomą delsą (angl. „overhead“) ir padidinti vėlinimą. RTS/CTS mechanizmas duomenų perdavimui reikalauja kanalo rezervacijos. Rezervuojant kanalą stotys privalo į eterį perduoti papildomus kontrolės kadrus, kurių nereikalauja pagrindinis paskirstytosios koordinuotos funkcijos režimas.



7 pav. Kolizijų valdymo metodų palyginimas eterio laiko atžvilgiu

Parodyta [8], kad RTS/CTS galutinai neišsprendžia paslėptos stoties problemos, o tiesiog keičia vienas problemas į kitas. 802.11 DCF RTS/CTS mechanizmas (7 pav.) sunaudoja didesnę eterio laiko dalį valdymo kadrams, kurie paprastai perduoda minimalų informacijos kiekį. Pvz., RTS kadrui perduoti gali prireikti iki 60 μs [2], tai apima tiek laiko, kiek pakanka 3240 bitų perdavimui 54 Mb/s greičiu, per kurį jis perduoda vieną bitą svarbios informacijos.

Dėl įvardytos priežasties buvo pasiūlyta RTS konfigūruojama riba (*RT*), kuri naudojama RTS/CTS įjungimui/išjungimui [2]. *RT* numato eteriu keliaujančio kadro ilgį, kurį perduodant stotis privalo atlikti kanalo rezervaciją. Tačiau standartas neapibrėžia kokias *RT* reikšmes reikėtų naudoti. Įvairiose scenarijuose skirtingai parinktos aktyvacijos ribos gali rodyti skirtingus rezultatus. Dažnai RTS/CTS sukeliamos eterio laiko sąnaudos yra žalingos. Eteris pripildomas kontrolės kadrais neįvertinant ar tai yra naudinga esamoms tinklo sąlygoms.

1.5. Kolizijų aptikimo mechanizmo valdymo metodų analizė

Šiame poskyryje apžvelgiami pasiūlyti metodai darbe nagrinėjamai problemai spręsti. Aptariami esami kolizijų valdymo mechanizmai, jų privalumai ir trūkumai. Apžvelgiami šaltiniai ir publikacijos, analizuojančios bendrojo naudojimo eterio kolizijų valdymo mechanizmus ir pasiūlytus esamo mechanizmo pagerinimo metodus.

Nemažai analizuotų publikacijų nagrinėja belaidžio tinklo kolizijų aptikimo metodus. Pateiktas metodas [9] gebantis lokaliai (t. y. belaidžio tinklo prieigos taške) įvertinti prarasto kadro dėl įvykusios kolizijos tipą. Atskirtos aplinkos sukeliamos kolizijos bei pasiūlytas kolizijos tikimybės sekančiam perduotam/priimtam kadrui apskaičiavimas, diferencijuojantis tiesiogines (angl. „direct

collision“) ir neplanuotos (angl. „staggered collision“) kolizijas. Automobilių tarpusavio komunikacijos sferoje pasiūlytas [10] transliacijos duomenų sukeltų kolizijų atskirties nuo kanalo radijo terpės triukšmų metodas, nustatantis kolizijos tikimybę vertinant paketų pristatymo santykį (angl. „packet delivery rate“, PDR). Publikacijose [9] ir [10] sprendžiamos kolizijų belaidžiuose tinkluose aptikimo ir diferencijavimo problemos.

Dalis apžvelgtų publikacijų pateikia esamų kolizijos valdymo metodų vertinimą įvairiais aspektais. Įvertintas maksimalus WLAN šuolinio (angl. „multi-hop“) DCF tinklo pralaidumas [11], stebint tarp-tinklinį duomenų srautą keliaujantį per silpniausias tinklo grandis (angl. „bottle-neck nodes“). Tyrimas [4] parodo RTS/CTS mechanizmo sukeltą tinklo pralaidumo sumažėjimą nedidelės spartos tinkle. Taip pat sprendžiant pralaidumo sumažėjimo problemą DCF tinkle dėl kolizijų valdymo metodo pasiūlyta [6] varžybų lango valdymo schema, kuri pagerina esamą BEB. Atlikti tyrimai parodo mechanizmo sukeliamą duomenų spartos sumažėjimą.

Keletas rastų publikacijų parodo RTS/CST įtaką belaidžio ryšio tinklų saugumui. Parodyta [8], kad RTS/CTS mechanizmo naudojimas gali būti išnaudojamas tinklo trukdžiams sukelti sunaudojant minimalius energinius išteklius ir sukuriant tinklo grūsties iliuziją transliuojant RTS kadrus. Tyrimas [12] rodo, kad naudojant nekontroliuojamą RTS/CTS mechanizmo konfigūraciją piktavališki įrenginiai gali nesivadovaudami standarto (802.11) gauti eterio laiko pranašumą prieš kitus įrenginius. Įvykus tiesioginei duomenų kolizijai stotys pasirenka atsitiktinį atsitraukimo laiko tarpą iš CW (angl. „congestion window“) intervalo. Piktavališkos stotys susimažinusios šį intervalą gali įgauti terpės pasiekimo pirmenybę. Publikacijos parodo kolizijų valdymo mechanizmo papildomai atveriamą erdvę saugumo pažeidžiamumams.

Keletas publikacijų sprendžia RTS/CTS mechanizmo sukeltos delsos problemą. Pasiūlyta [13] 802.11 MAC lygmenyje įdiegti RTR (angl. „ready to receive“) kadrus, kurie būtų siunčiami komunikacijos pabaigoje, informuojant visas BSS stotis, kad kanalas laisvas. Kaip teigiama pasiūlytas, metodas padėtų sumažinti RTS/CTS delsą pagreitinat naujos komunikacijos pradžią.

Nemažai publikacijų ir mokslinių darbų nagrinėja su darbo problematika susijusias temas įvairiais aspektais, tačiau rastas tik vienas metodas sprendžiantis dinaminio kolizijų valdymo problemą DCF tinkle.

1.5.1. Belaidžių tinklų RTS/CTS valdymo metodas

Pasiūlytas dinaminis RTS/CTS valdymo mechanizmas [14] vertinant kanalo komunikacijos konkurenciją (angl. „contention“) ir signalinio protokolo spartą. Metodas lygina esamą WLAN tinklo pranašumą naudojant pagrindinį CSMA su galima tinklo delsa įjungus RTS/CTS kadrus. Apibrėžiama kolizijos kainos funkcija, kuri priklauso nuo stočių komunikacijos konkurencijos tikimybės ir duomenų perdavimo laiko. Komunikacijos konkurencijos tikimybei apskaičiuoti naudojamas kolizijos tikimybės matmuo yra vertinamas SENS (angl. „smart experts for network state estimation“) metodu. Apskaičiuojant nepavykusių priėmimų (angl. „transmission fail“) skaičių, vertinamas visų siuntėjo perduotų kadrų skaičiaus.

SENS yra 2019 metais pasiūlytas mašininio apsimokimo metodas naudojantis (angl. „fixed-share“) algoritmą. Metode laikas padalinamas į fiksuoto dydžio langus. Kiekvieno lango pradžioje skiriamas laikas, kurio metu keletą kartų apskaičiuojama kolizijos tikimybė, kuri yra naudojama likusią periodo

lango dalį. Sprendžiant iš atlikto tyrimo išvadų, sukurtas metodas pagerina tinklo pralaidumą, lyginant tinklo spartą nenaudojant RTS/CTS ir naudojant statinę RTS/STA ribos RT reikšmę.

1.6. Analizės išvados

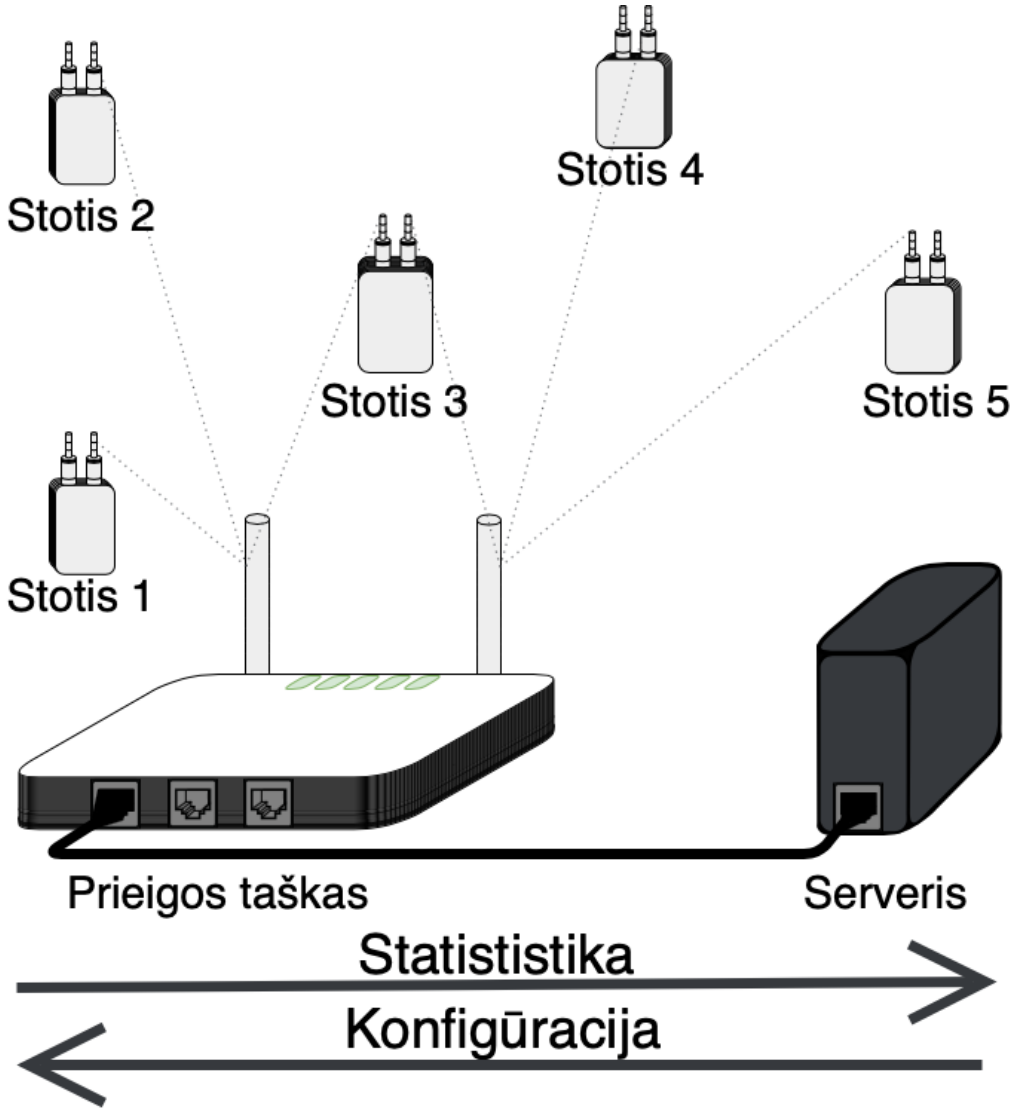
Atliktos analizės metu nustatyta, kad esamas paskirstytosios koordinuotos funkcijos tinklų WLAN protokolo kolizijų valdymo mechanizmas gali sumažinti kolizijų kiekį dėl eteryje susidūrusių kadrų, tačiau dažnu atvejų, kai visi radijo įrenginiai girdi vienas kitą, RCT/CTS mechanizmo užimamas eterio laikas sukelia papildomą vėlinimą ir tinklo pralaidumas sumažėja. Esant unikalioms tinklo sąlygoms kiekviename BSS tinkle statiška RTC/CTS valdymo riba nėra efektyvi. Besikeičiant tinklo sąlygoms naudojamas pasirenkamasis kolizijų valdymo mechanizmo režimas užpildo eterį pertekliniais kontrolės kadrais.

Analizuoti literatūriniai šaltiniai [13] [11] [4] parodo drastišką tinklo pralaidumo sumažėjimą ir delsos padidėjimą naudojant RTS/CTA mechanizmą. Kai kurie šaltiniai [8] [12] išryškina saugumo spragas naudojant RTS/CTS kadrus kolizijoms vadyti. Analizuotas sukurtas dinaminis WLAN protokolo kolizijų valdymo metodas [14] parodo teigiamus rezultatus tinklo spartos pagerinimui esant paslėptos stoties sąlygoms, tačiau metodas varijuoja kolizijų valdymo mechanizmo režimu naudojat statinę ($RT = 1$) aktyvacijos ribą.

Iškeltas baigiamojo darbo tikslas yra sukurti metodą valdantį paskirstytosios koordinuotosios funkcijos kanalo prieigos mechanizmą nustatant optimalią ribą (RT), užtikrinančią maksimalią tinklo spartą ir efektyvų eterio laiko išnaudojimą.

2. Dinaminio kolizijų valdymo metodo projektas

Dinaminis kolizijų valdymo metodas DKVM – šiame darbe projektuojamas kolizijų valdymui belaidžiame DCF tinkle metodas. Projektinėje dalyje numatyta suprojektuoti kolizijų valdymo metodą, atsižvelgiant į analitinėje dalyje analizuotas protokolo spragas. Dinaminio kolizijų valdymo metodo tikslas, sumažinti kolizijų kiekį belaidžiame tinkle pritaikant tinklo įrangos (prieigos taško) konfigūraciją prie tinklo sąlygų. Paveiksle (8 pav.). pavaizduota kuriamo metodo koncepcinė diagrama. Projektuojamas metodas yra sudarytas iš dviejų komponentų: belaidžio tinklo prieigos taško (AP), kuris periodiškai surenka stebimas statistikas ir perduoda į serverį; dedikuoto serverio, kuris priima statistikas ir parenka tinkamą konfigūraciją.

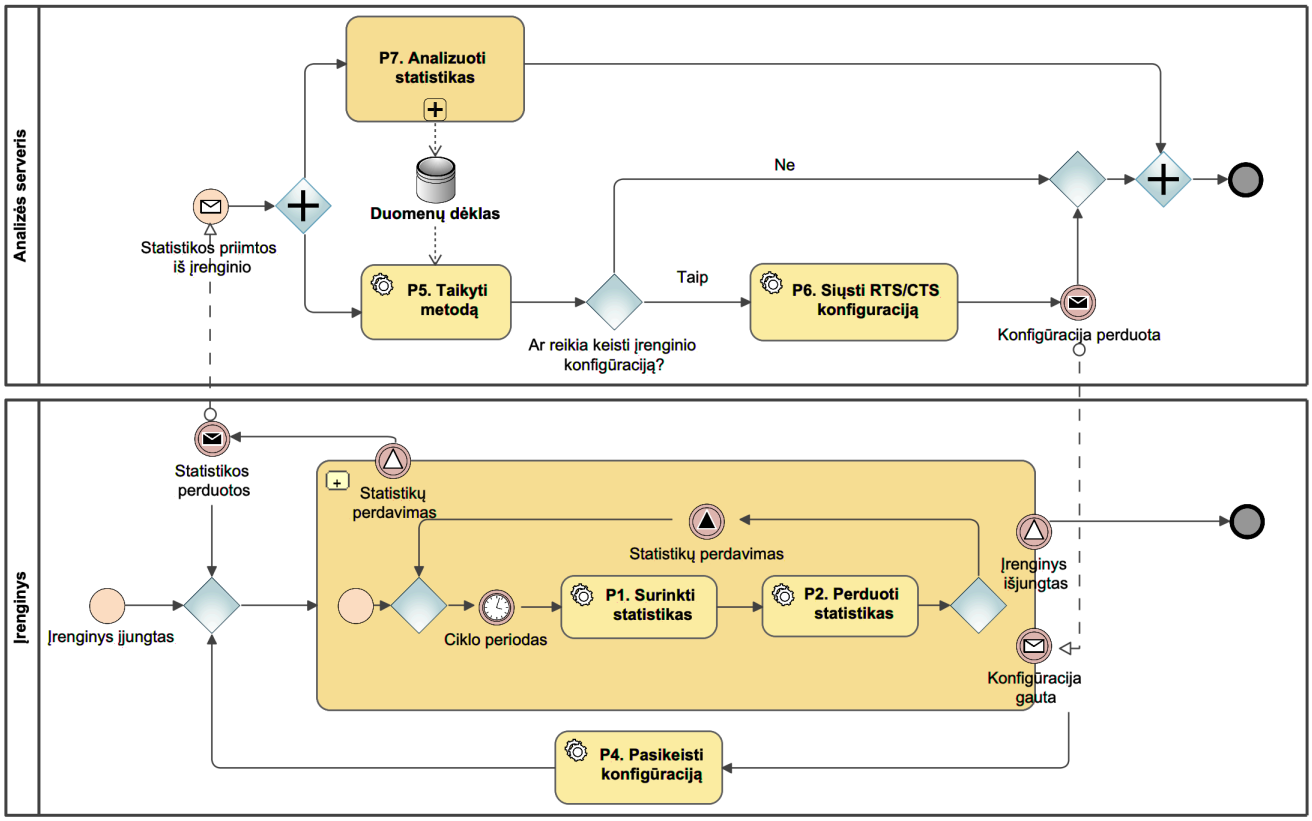


8 pav. Dinaminio kolizijų valdymo metodo vizija

Dinaminio kolizijų valdymo metodo projektinė dalis sudaryta iš 9 poskyrių: sumodeliuojamas kuriamo metodo veiklos procesų modelis (2.1); sudaroma panaudos atvejų diagrama (2.2); pateikiamas DKVM koncepcinis modelis su detaliu kiekvieno komponento aprašu (2.3); aprašomi projektuojamo metodo manipuliuojamų duomenų tipai (2.4); pateikiamos projekto išvados (2.5).

2.1. Dinaminio kolizijų valdymo metodo veiklos proceso modelis

Veiklos procesų modelyje (9 pav.) pavaizduotas kuriamo metodo veiklos procesas. Prieigos taške veikiantis statistikų surinkimo agentas periodiškai (kas t_p) atlieka statistikų surinkimo užduotį (P1), kurios metu perimamos belaidžio tinklo plokštės tvarkyklės kaupiamos veiklos statistikos ir perduodamos (P2) į serverį, kuriame veikia statistikų kolektorius. Kolektoriaus tikslas priimti statistikas ir pateikti jas tinkamu formatu tolimesnei analizei. Serveryje statistikos yra analizuojamos (P7) ir rezultatai kaupiami duomenų dėkle (angl. „stack“).



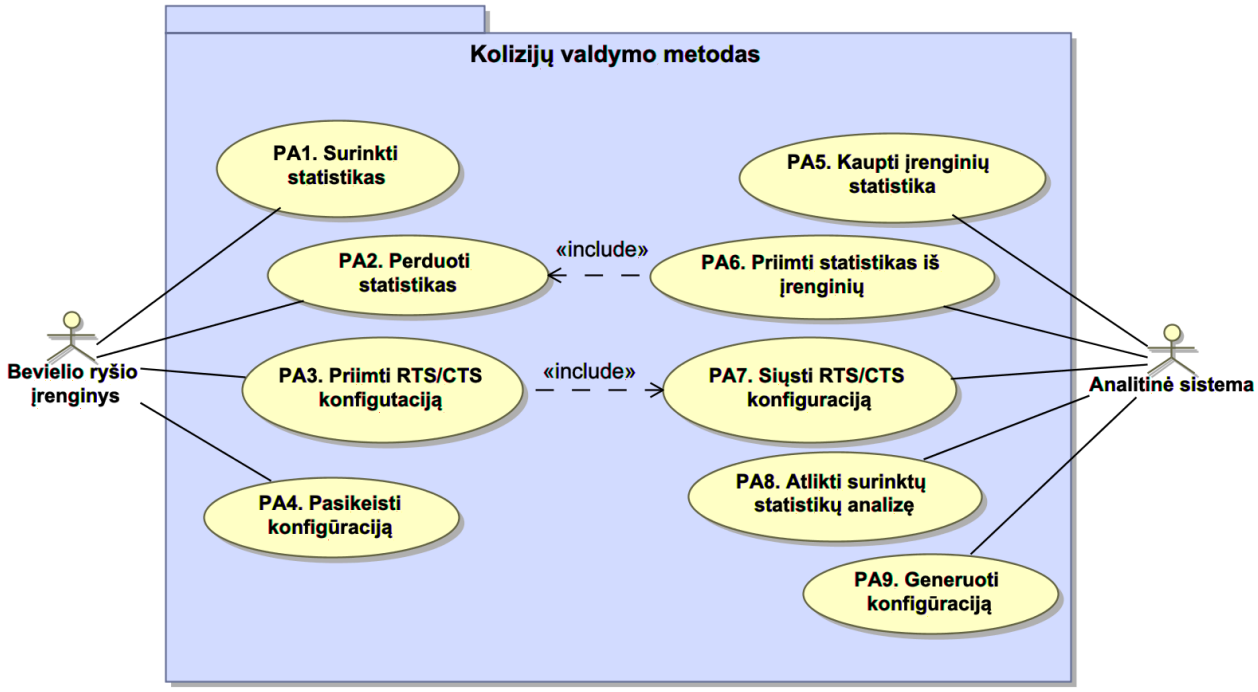
9 pav. Dinaminio kolizijų valdymo metodo veiklos procesų modelis

Analizuojant (P7) atliekami skaičiavimai, kurių metu išgryninami belaidžio tinklo parametrai, žymintys kolizijų belaidžiame tinkle tikimybę. Apskaičiuotos reikšmės talpinamos į istorinių duomenų dėklą pažymint laiko žyme (angl. „timestamp“). Naudojant sukauptus istorinius duomenis taikomas kolizijų valdymo sprendimo metodas (P5), kuris grąžina naują mechanizmo konfigūraciją ir jo aktyvacijos ribą. Jei metodo išeities konfigūracija skiriasi nuo esamos įrenginio konfigūracijos, tokiu atveju nauja konfigūracija perduodama įrenginiui (P6). Įrenginys atnaujina konfigūracija ir tęsia veiklą (P4). Pavaizduotas proceso modelis leidžia įrangai reaguoti į tinklo pasikeitimus ir atitinkamai keisti konfigūraciją.

2.2. Dinaminio kolizijų valdymo metodo panaudos atvejų diagrama

Atsižvelgiant į sudarytą veiklos procesų modelį, buvo iškelti reikalavimai projektuojamam metodui. Sudaryta kolizijų valdymo metodo panaudos atvejų diagrama (10 pav.) apibrėžiant funkcinius, kuriamo metodo reikalavimus. Diagramoje pažymėti du aktoriai: stebimas belaidžio ryšio įrenginys,

kuris paskirstytosios koordinuotosios funkcijos tinkle yra prieigos taškas ir analitinė sistema, kuri reprezentuoja visą kuriamo metodo funkcionalumą serverio pusėje.



10 pav. Kolizijų valdymo metodo panaudos atvejų diagrama

Panaudos atvejų diagrama numato projektuojamam metodui keliamus reikalavimus:

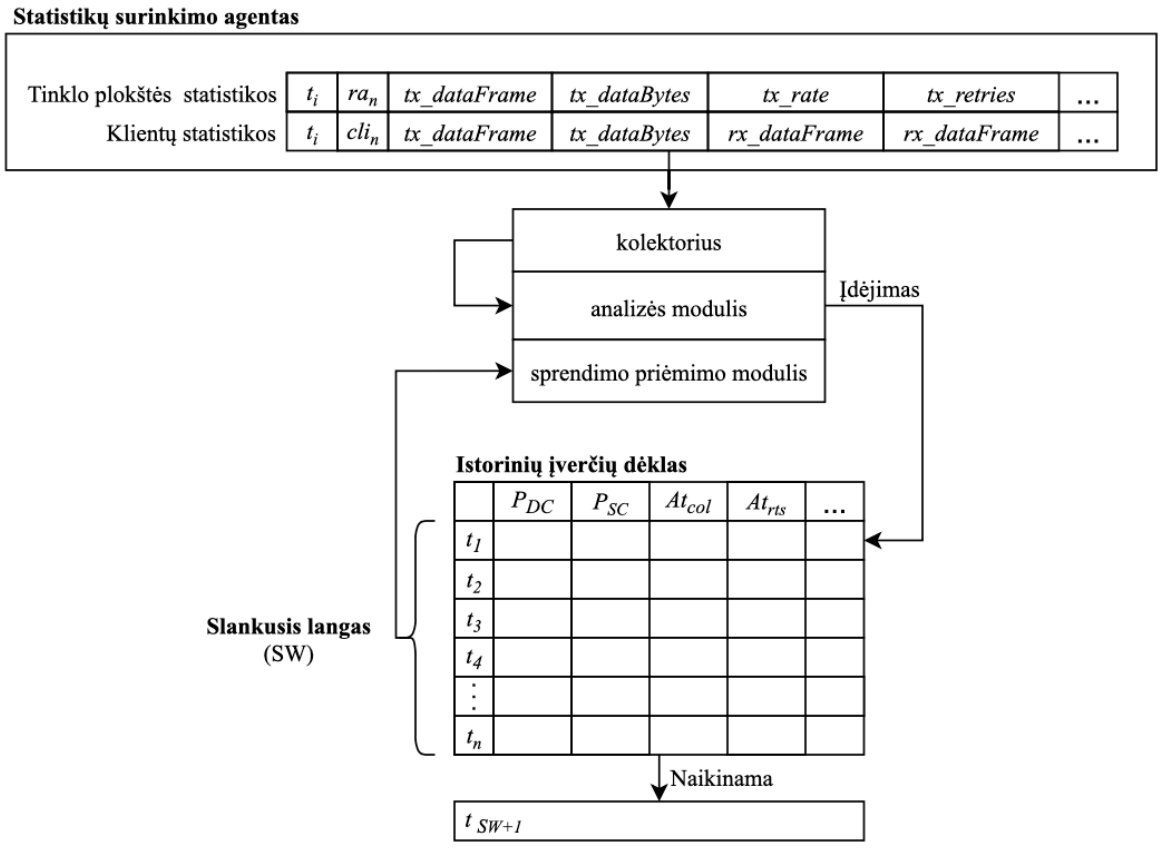
- PA1. Surinkti statistikas – WLAN prieigos taškas periodiškai surenka veiklos statistikas ir suformuoja pranešimą;
- PA2. Perduoti statistikas – WLAN prieigos taškas perduodą suformuotą pranešimą su užfiksuotomis tinklo veiklos statistikomis į dedikuotą serverį;
- PA3. Priimti RTS/CTS konfigūraciją – WLAN prieigos taškas priima serverio siunčiamą apskaičiuotą naują RTS/CTS konfigūraciją iš dedikuoto serverio;
- PA4. Pakeisti konfigūraciją – WLAN įrenginys atnaujina konfigūraciją priimtą iš dedikuoto serverio;
- PA5. Kaupti įrenginių statistikas – analitinė sistema kaupia surinktas įrenginių veiklos statistikas atmintyje;
- PA6. Priimti statistikas iš įrenginio – dedikuotas serveris priima statistikas iš WLAN įrenginio;
- PA7. Siųsti RTS/CTS konfigūraciją – analitinė sistema perduoda sugeneruotą RTS/CTS konfigūraciją WLAN įrenginiui;
- PA8. Atlikti surinktų statistikų analizę – analitinė sistema atlieka iš WLAN įrenginio surinktų statistikų analizę;
- PA9. Generuoti konfigūraciją – analitinė sistema sugeneruoja tinklo įrenginio konfigūraciją pagal apskaičiuotas reikšmes.

2.3. Dinaminio kolizijų valdymo metodo koncepcinis modelis

Dinaminis kolizijų valdymo metodas (DKVM) yra tiriamojo darbo metu pasiūlytas sprendimas DCF WLAN tinkle atsirandančių kolizijų įtakai, duomenų perdavimui valdyti. DKVM periodiškai renka ir analizuoja WLAN tinklo veiklos statistikas dedikuotame serveryje, kuriomis remiantis nustatoma BSS eterio valdymo mechanizmo konfigūracija.

Metodą sudaro keturios atskiros dalys – moduliai, kurie yra atsakingi už tam tikras sistemos veiklos dalis (11 pav.):

- Statistikų surinkimo agentas – periodiškai surenka WLAN tinklo įrangos tvarkyklės registruojamas kintamųjų reikšmes prieigos taške ir perduoda į serverį;
- Statistikų kolektorius – priima statistikas iš prieigos taško dedikuotame serveryje ir inicijuoja analinės modulį;
- Analizės modulis – atlieka skaičiavimus vieno periodo ribose ir registruoja istorinius rezultatus;
- Sprendimo priėmimo modulis – atlieka slankaus lango ilgio analizės modulio duomenų vertinimą ir padaro kolizijos mechanizmo valdymo konfigūracijos sprendimą.

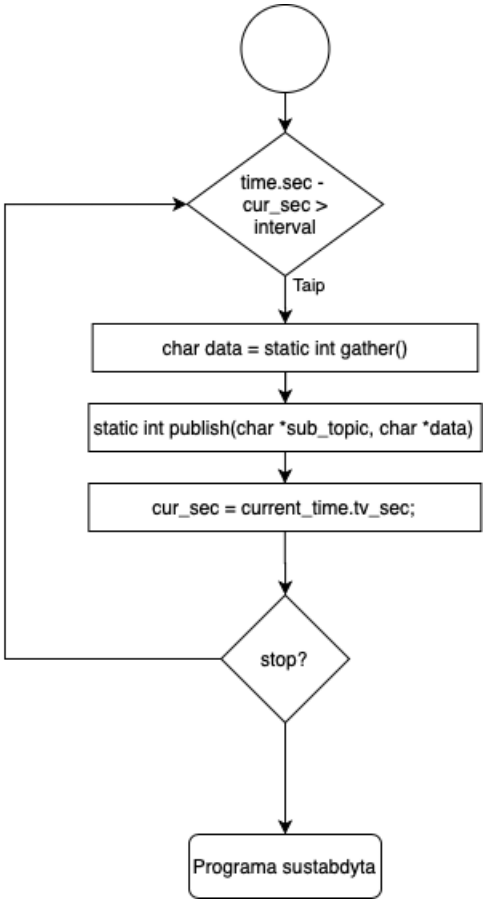


11 pav. Kolizijų aptikimo metodo koncepcinė diagrama

Pagrindinė DKVM dalis yra sprendimo priėmimo modulis, kuris atlieka konfigūracijos parinkimo funkciją. Įvertinus istorinius dėklo įrašus modulis parenka efektyvią RTC/CTS mechanizmo konfigūraciją ir perduoda ją prieigos taškui.

2.3.1. Statistikų surinkimo agentas

Prieigos taškas yra įterptinė sistema, kuri turi ribotus skaičiavimo ir atminties talpos resursus, todėl tinklo parametrų istorijos saugojimas ir papildomi skaičiavimai įrenginyje gali sutrikdyti įrenginio paskirties veiklą. Dėl šios priežasties veiklos statistikos yra perduodamos ir kaupiamos dedikuotame serveryje, kuriame vykdoma stebinių analizė ir padaromas RTS/CTS įjungimo/išjungimo sprendimas.



12 pav. Statistikų surikimo agento veiklos diagrama

Prieigos taške įdiegtas statistikų surinkimo agentas, kuris kas nustatytą laiko intervalą (t_p) kreipiasi į WLAN tinklo plokštės sąsają, naudojant operacinės sisteminiai kreipinius (angl. „ioctl“). Tinklo plokštės sąsaja gražina struktūrą su kintamųjų reikšmėmis. Panaudojant gražintą struktūrą pasiekiami WLAN techninės įrangos tvarkyklės (UMAC (angl. „upper media access control“) lygmens) registruojami veiklos parametrai. Surinktos kintamųjų reikšmės laikinai išsaugomos įrenginio operatyvioje atmintyje tolesniam apdorojimui.

Įrenginyje periodiškai stebimi didėjantys skaitikliai, kurių vertės nesuteikia jokios statistinės naudos. Stebint WLAN techninės įrangos tvarkyklės registruojamus didėjančius skaitiklius, svarbus stebinių stacionarumas. Laiko eilutės duomenys yra stacionarūs tuo metu, kai statistinės stebinio savybės nekinta laike. Tai yra, kai vidurkis, dispersija ir kovariacija yra nepriklausomos nuo laiko. Šių skaitiklių nauda pasireiškia tik tuo metu, kai stebime jų pokytį (Δ) per vieną periodą. Kintamojo Z reikšmės stebimos bėgant laikui. Laikoma, kad yra žinomos reikšmės $Z(t_i)$ laiko momentais, kai $t_1 < t_2 < \ldots < t_3$. Visi stebėjimai atliekami vienodais laiko intervalais, t. y. $t_{i+1} - t_i = t_p$, kai t_p – yra

periodo trukmė. Skaitiklių pokyčio skaičiavimai atliekama prieigos taške atimant ankstesnio periodo skaitiklių vertes $Z(t_{i-1})$, iš einamojo periodo $Z(t_i)$ skaitiklių verčių.

Atlikus skaitiklių pasikeitimų skaičiavimus, suformuojamas tekstinis pranešimas naudojant „JSON“ notaciją. Apskaičiuotos kintamųjų stacionarios reikšmės perduodamos „MQTT“ kliento programai, kuri atlieka publikaciją į dedikuotame serveryje veikiančio MQTT brokerio temą.

2.3.1.1. Stebimos statistikos

Prieigos taško agentas stebi tinklo plokštės tvarkyklės registruojamas dviejų tipų statistikas: stoties (angl. „nodestats“), kurios registruojamos kiekvienai UMAC lygmens sukurtai mazgo struktūrai ir tinklo plokštės (angl. „device“), kurios registruojamos kiekvienai HAL (angl. „hardware abstraction layer“) lygmens įrenginio struktūrai (t. y. fizinei WLAN tinklo plokštei).

Lentelėje (1 lentelė.) pateiktos metodo sprendimo priėmimo moduliui aktualios stoties perspektyvos statistikos registruojamos prieigos taško pusėje, kurios periodiškai perduodamos į serverį.

1 lentelė. Stoties statistikos

Pavadinimas	**Aprašymas**
$tx\_dataFrames$	Stoties perduotas duomenų kadrų skaičius;
$tx\_dataBytes$	Stoties perduotas duomenų kadrų dydis, baitais;
tx_rate	Stoties paskutinio sėkmingai perduoto kadro perdavimo sparta, Mb/s;
$tx\_retries$	Stoties perdavimo pakartojimų po nesėkmingo pristatymo skaičius;
$rx\_dataFrame$	Stoties priimtų duomenų kadrų skaičius;
$rx\_dataBytes$	Stoties priimtų duomenų kadrų dydis, baitais;
rx_rate	Stoties priėmimo sparta, Mb/s;
rx_decap	Stoties priimtų nepavykusių dekapsuliuoti kadrų skaičius;
$rx\_decryptcrc$	Stoties priimtų nepavykusių iššifruoti dėl nesutapusios CRC (angl. „check“) patikros;
$rx\_tkipivc$	Stoties priimtų kadrų su nepavykusia ICV patikra (TKIP);
$rx\_wepfail$	Stoties priimtų neteisingai iššifruotų WEP (angl. „check wireless enhancesd protection“) apsaugos mechanizmo kadrų skaičius;
$signals$[]	Stoties signalų stipriai iš AP perspektyvos;
$band$	Fizinio lygmens naudojamas radijo bangų spektras [5g arba 2g];
$phyMode$	Stoties Lygiagrečių antenų konfigūracija [1x1, 2x2, 3x3];
$chWidth$	Naudojamas kanalo plotis [20, 40, 80];

Lentelėje (2 lentelė.) pateiktos metodo sprendimo priėmimo moduliui aktualios tinklo plokštė perspektyvos statistikos, kurios yra periodiškai perduodamos į serverį. Stoties ir tinklo plokštės statistikos yra įvesties duomenys, kurie yra perduodami analizės moduliui.

2 lentelė. Tinklo plokštės statistikos

Pavadinimas	**Aprašymas**
$tx\_dataFrames$	AP vieno tinklo įrenginio perduotas duomenų kadrų skaičius

$tx\_dataBytes$	AP vieno tinklo įrenginio perduotas duomenų kadrų dydis, baitais
$rx\_dataFrame$	AP vieno tinklo įrenginio priimtų duomenų kadrų skaičius
$rx\_dataBytes$	AP vieno tinklo įrenginio priimtų duomenų kadrų dydis, baitais

2.3.2. Statistikų kolektorius

Statistikų kolektorius yra dinaminio kolizijų valdymo metodo sudedamoji dalis, kuri yra atsakinga už statistikų priėmimą iš prieigos taško serverio pusėje ir jų apdorojimą. Statistikų priėmimas vykdomas naudojant MQTT protokolą. Kolektorius prenumeruoja MQTT temą (angl. „subscription topic“), kurioje prieigos taškas periodiškai skelbia statistiką. Kolektorius atlieka triukšmo filtravimą, tai yra šalinamos visos reikšmės neatitinkančios maksimalių ir minimalių rėžių, kurie gali iškraipyti skaičiavimo rezultatus. Pašalina nereikalingus parametrus, kurie nebus naudojami duomenų analizės proceso metu. Apdoroti duomenys paverčiami į žodyno duomenų tipą ({„raktas“: „reikšmė“}) ir perduodami analizės moduliui.

2.3.3. Analizės modulis

Analizės modulis atlieka vieno periodo reikšmės skaičiavimus, kurie yra registruojami į istorinių duomenų dėklą. Apskaičiuojamos reikšmės naudojamos atpažinti tinklo sąlygų pasikeitimus ir įvertinti kolizijos įvykio tikimybę laiko periode.

Siekiamas kuriamo metodo rezultatas – valdyti kolizijų mechanizmą taip, kad jo naudojimas nebūtų nuostolingas eterio laiko atžvilgiu. Keliama prielaida, kad užimamas eterio laikas RTS/CTS funkcionalumui su visais papildomais laiko intervalais (angl. „overhead“), privalo būti mažesnis nei pasitaikančių kolizijų užimamas eterio laikas. Išpildžius šią prielaidą, parenkant kolizijų valdymo mechanizmo konfigūraciją, laikoma, kad parinkta konfigūracija yra efektyvi.

3 lentelė. Analizės modulio apskaičiuojamos statistikos

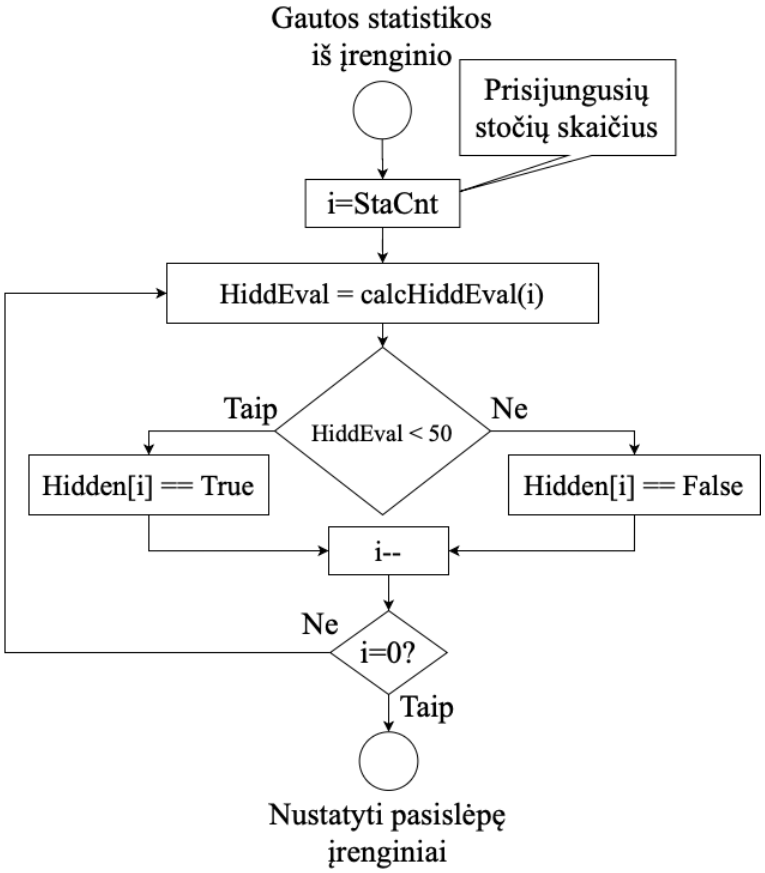
Pavadinimas	**Aprašymas**
$STAConc_i$	Einamosios stoties eterio laiko konkurencijos įvertinimas [1 – konkuruoja; 0 – nekonkuruoja];
N_{Concur}	Konkuruojančių stočių dėl eterio laiko kiekis vieno periodo metu;
P_{DC}	Tiesioginės kolizijos (angl. „direct“) tikimybė su esamu RTS/CTS nustatymu;
P_{SC}	Neplanuotos kolizijos (angl. „staggered“) tikimybė su esamu RTS/CTS nustatymu;
At_{col}	Kolizijų sukeltos papildomos eterio laiko sąnaudos (angl. „collision airtime“);
At_{rts}	RTS/CTS sukeltas papildomas eterio laiko vėlinimas;

Efektyvios konfigūracijos parinkimo sąlygos įgyvendinimui atliekami skaičiavimai (3 lentelė.). Šie skaičiavimai reikalingi eterio laiko sunaudojimo nustatymui, taip pat vertinant kolizijų tikimybę nenaudojant ir naudojant RTS/CTS kontrolės kadrus.

2.3.3.1. Paslėptos stoties sąlygos vertinimas

Priklausomai nuo to ar tinkle yra paslėpta stotis, naudojama skirtinga kolizijos tikimybės skaičiavimo schema. Jei tinkle visos stotys girdi vienos kitas, laisvo kanalo nustatymo mechanizmas veikia patikimai. Tokiu atveju kolizijos įvyksta tik jei atlaisvėjus kanalui, dvi ar daugiau stočių, pasirenka identišką atsitraukimo intervalą, t. y. įvyksta atsitraukimo sinchronizacija, kuri sukelia tiesiogines

kolizijas (angl. „direct collision“). Kitu atveju, jei tinkle yra paslėpta stotis, tuomet kolizijos tikimybės skaičiavimus papildo dar viena sąlyga. Visi paslėptos stoties siunčiami kadrai prieigos taško kryptimi gali sukelti kolizijas. Tokio tipo kolizijos vadinamos neplanuotomis kolizijomis (angl. „staggered collision“). Paslėptų įrenginių nustatymo veiksmų seka pateiktas 13 pav.



13 pav. Paslėptų įrenginių skaičiaus nustatymo veiksmų seka

Paslėptos stoties vertinimas atliekamas kiekvieno periodo metu gavus statistikas iš prieigos taško. Pirmiausia apskaičiavus kiekvienos prisijungusios stoties komunikacijos kokybės $HiddEval$ (1) įvertį. Jei gautas rezultatas yra mažesnis už 50 ($HiddEval < 50$), stotis pažymima kaip paslėpta (angl. „hidden“).

$$\boldsymbol{HidEval} = \left(\overline{\boldsymbol{PER}} \times \mathbf{0.2}\right) + \left(\overline{\boldsymbol{Eval}} \times \mathbf{0.6}\right) + \left(\left(\left(\frac{\sum_{i=1}^{n} \boldsymbol{signal}}{\boldsymbol{n}} + \mathbf{90}\right) \times \mathbf{2}\right) \times \mathbf{0.12}; \right); \qquad (1)$$

čia $\overline{PER}$– stoties perduotų/priimtų klaidų ir pakartojimų santykių vidurkis (angl. „packer error rate“), $\overline{Eval}$ – stoties techninių galimybių ir esamo ryšio kokybės įvertis (2.3.3.1.2), $Signal$ – AP girdimas stoties signalas (jei stotis palaiko keletą lygiagrečių signalų, tokiu atveju skaičiuojamas visų signalų vidurkis).

Paslėptos stoties įvertinimą lemia paketų klaidų santykis, perdavimo spartos kokybė ir signalų stipriai. Šie kintamieji sudarė atitinkamas galutinio įverčio santykines dalis, kurios buvo nustatytos testavimo metu. Paslėptos stoties įvertis skaičiuojamas kiekvienai asocijuotai stočiai. Naudojant šį rodiklį parenkama atitinkamas kolizijos tikimybės skaičiavimo schema.

2.3.3.1.1. Duomenų kadrų klaidų santykis

Kadro klaidos santykis PER (angl. „packet error rate“) vertina sėkmingai perduotų kadrų skaičių įvykusių perdavimo klaidų ir kadro pakartojimų atžvilgiu. PER yra tik dalinai tikslus, nes pasak tinklo plokštės gamintojo, fizinio lygmens techninė įranga neturi galimybės grąžinti tikslaus nesėkmingų perduotų kadrų skaičiaus.

Perdavimo kadrų klaidų santykis (PER_{tx}) (2) priklauso nuo sėkmingo persiuntimų skaičiaus ($tx\_dataFrame$) ir siuntimo pakaitrojimai ($tx\_retries$). Sėkmingai perduoti/priimti kadrai ar kadrų eilės, laikomos tokios, kurios buvo patvirtintos ACK pažymėtu kadru.

$$\boldsymbol{PER_{tx} = \frac{tx\_dataFrame}{tx\_dataFrame + tx\_retries} \times 100}; \quad (2)$$

čia $tx\_dataFrame$ – perduotų duomenų kadrų kiekis [n], $tx\_retries$ – pakartotinai išsiųstų kadrų kiekis [n].

PER_{rx} (3) parodo nesėkmingai priimtų kadrų skaičių, kurių nepavyko iššifruoti ar patikrinti (nesutapo kontrolinė suma). Padidėjęs priimtų kadrų klaidų santykis gali indukuoti suprastėjusias tinklo sąlygas.

$$\boldsymbol{PER_{rx} = \frac{rx\_dataFrame}{rx\_dataFrame + rx\_decap + rx\_decryptcrc + rx\_tkipivc + rx\_wepfail} \times 100}; \quad (3)$$

čia $rx\_dataFrame$ – iš viso priimtų duomenų kadrų kiekis [n], rx_decap - stoties priimtų nepavykusių dekapsuliuoti kadrų skaičius [n], $rx\_decryptcrc$ – stoties priimtų nepavykusių iššifruoti dėl nesutapusios CRC (angl. „checksum“) patikros [n], $rx\_tkipivc$ – stoties priimtų kadrų su nepavykusia ICV patikra (TKIP) [n], $rx\_wepfail$ – stoties priimtų neteisingai iššifruotų WEP (angl. „check wireless enhancesd protection“) apsaugos mechanizmo kadrų skaičius [n].

Perduotų ir priimtų kadrų klaidų santykiai (PER_{rx} ir PER_{rx}) matuojami procentais. Kai visi kadrai yra sėkmingai perduoti ar priimti šie kintamieji įgauna maksimalias vertes (100 proc.) atitinkamai. Sumažėjus šių santykių reikšmėms galima teigti, kad tinklo sąlygos suprastėjo.

2.3.3.1.2. Techninių galimybių ir esamo ryšio kokybės įvertis

Stoties techninių galimybių ir esamo ryšio kokybės įvertis $linkEval$ parodo stoties perdavimo spartą ir galimybių išnaudojimą esamu laiko momentu. Santykio vertinimo metu (4), lyginama maksimali palaikoma stoties fizinė sparta ($MaxBitRate$) atsižvelgiant į paskutinio perduoto kadro fizinę spartą (angl. „bitrate“). Atitinkamai apskaičiuojama perdavimo (angl. „downlink“) ir priėmimo (angl. „uplink“) ryšio sparta.

$$\boldsymbol{linkEval_{tx/rx} = \frac{BitRate_{tx/rx} \times 100}{MaxBitRate}}; \quad (4)$$

čia $BitRate$ – paskutinio sėkmingai perduoto kardo sparta, kuria bendrauja AP ir STA [Mb/s], $MaxBitRate$ – nustatyta maksimali STA palaikoma sparta [Mb/s].

Atsižvelgiant į techninius stebimų stočių parametrus, priimtus iš prieigos taško, nustatoma maksimali stoties perdavimo ir priėmimo palaikoma sparta. Vertinami parametrai: operuojamas fizinio lygmes

dažnis $band$ [5Ghz arba 2Ghz]; fizinio lygmens protokolas $mode$; naudojamas kanalo plotis $ChWidth$ ir antenų konfigūracijos parametrai $PhyMode$. Maksimali palaikoma sparta taip pat priklauso nuo įrenginių signalo praradimo apsaugos intervalo (angl. „guard interval“). Esant galimybei t. y. jei klientas palaiko aukštesnius nei 802.11 a/b/g fizinio lygmens standartus, naudojamas trumpas apsaugos intervalas SGI (400ns). Kitu atveju laikoma, kad kadrai perduodami naudojant ilgą LGI (800 ns) apsaugos intervalą (802.11n/ac). Stoties techninių galimybių ir periodo metu naudojamos spartos, vertinimas suteikia daugiau informacijos apie BSS tinklo kokybę ir įrenginių pozicijas.

2.3.3.2. Eterio laiko konkurencijos sąlygos vertinimas

Be paslėpto įrenginio sąlygos, kanale vykstanti stočių eterio laiko konkurencija yra dar vienas WLAN veiksnys turintis įtakos tinklo kolizijų atsiradimui. Jei su prieigos tašku yra asocijuotos nemažiau kaip dvi sotys, kurios konkuruoja dėl eterio laiko, tokiu atveju atsiranda kadrų kolizijų tikimybė. Jei viena stotis sugeneruoja visą periodo kadrų srautą, laikoma kad visi kadrų praradimai nutiko dėl nepalankių kanalo sąlygų.

Eterio laiko konkurencijai įvertinti naudojami tinklo plokštės lygmens perduodamų ir priimamų kardų skaitikliai. Šie skaitikliai parodo visų tinklo plokštės asocijuotų stočių perduotų ir priimtų duomenų kiekį per matavimo periodą. Keliama prielaida, kad stotys konkuruoja dėl eterio laiko jei ji sugeneruoja bent 30 proc. visų tinklo plokštės kadrų prisijungusiam stočių kiekiui (5).

$$\boldsymbol{STAConc_i} = \begin{cases} \mathbf{0}, & \boldsymbol{if}\ \left(\dfrac{\boldsymbol{dataFrames_{sta}}}{\boldsymbol{dataFrames_{ti}}} > \dfrac{\mathbf{1}}{\mathbf{30}\ \times \boldsymbol{N_{sta}}}\right) \\ \mathbf{1}, & \boldsymbol{if}\ \left(\dfrac{\boldsymbol{dataFrames_{sta}}}{\boldsymbol{dataFrames_{ti}}} < \dfrac{\mathbf{1}}{\mathbf{30} \times \boldsymbol{N_{sta}}}\right) \end{cases}; \tag{5}$$

čia $STAConc_i$ – stoties konkurencingumo vertė [1 arba 0], $dataFrames_{sta}$ – einamosios stoties perduotų/priimtų duomenų kadrų kiekis; $dataFrames_{ti}$ – tinklo plokštės perduotų/priimtų duomenų kadrų kiekis; N_{sta}- prisijungusių stočių skaičius.

$STAConcur_i$ gali įgyti reikšmę 1 arba 0, ši reikšmė žymi, kad vertinama stotis sta_i periodo metu perdavė santykinai didesnį duomenų kiekį nei esant lygiam duomenų perdavimo pasiskirstymui tarp asocijuotų stočių n_{sta}. Susumavus $BSAConcur_i$ vertinimo rezultatus (6) gaunamas stočių kiekis N_{Concur} galimai konkuruojančių dėl eterio laiko. Stočių konkuravimo indeksas N_{Concur} yra saugomas duomenų dėkle ir bus naudojamas apskaičiuoti kolizijos tikimybės reikšmės.

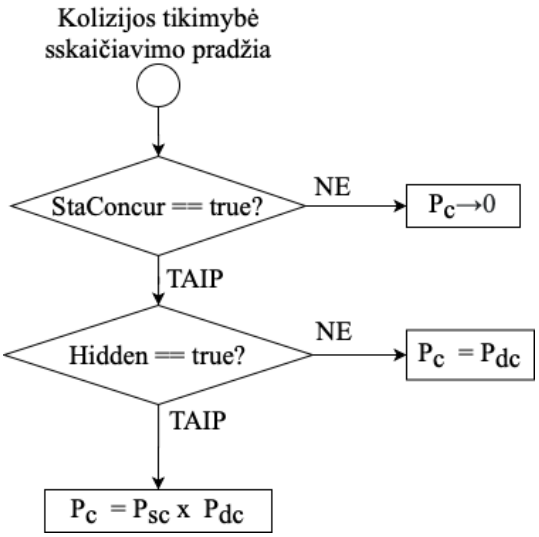
$$\boldsymbol{N_{Concur}} = \begin{cases} \boldsymbol{true},\ \boldsymbol{if}\left(\sum_{\boldsymbol{i=n}}^{\boldsymbol{n_{sta}}} \boldsymbol{STAConc_i} > \mathbf{2}\right) \\ \boldsymbol{false},\ \boldsymbol{if}\left(\sum_{\boldsymbol{i=n}}^{\boldsymbol{n_{sta}}} \boldsymbol{STAConc_i} < \mathbf{2}\right) \end{cases} \tag{6}$$

čia N_{Concur} – konkuruojančių stočių kiekis vieno periodo metu, $STAConc_i$ – stoties konkurencingumo vertė [1 arba 0].

2.3.3.3. Kolizijos tikimybė

Kolizijų tikimybė parodo galimą kolizijų kiekį vieno periodo laiko intervale. Tikimybės rodiklis apskaičiuojamas kiekvieno periodo metu, analizuojant paskutinio laiko tarpu gautas statistikas. Tikimybės apskaičiavimas skiriasi priklausomai nuo eteryje konkuruojančių stočių skaičiaus ir paslėpto įrenginio sąlygos.

Tinkle nutinkančios duomenų susidūrimo tipo kolizijos skirstomos į dvi kategorijas – neplanuotos kolizijos SC (angl. „staggered collisions“) ir tiesioginės kolizijos DC (angl. „direct collisions“). Kolizijos skaičiavimo schemos parinkimas pavaizduotas 14 pav.



14 pav. Kolizijos tikimybės skaičiavimo pasirinkimo algoritmas

Neplanuotos kolizijos (SC) įvyksta kai stotis inicijuoja duomenų perdavimą tuo metu, kai kanalas yra užimtas prieigos taško pusėje. Tokios kolizijos nutinka kai tinkle yra paslėpta stotis. Paslėpta stotis bando perduoti kadrus kiekvieną kartą kai užfiksuojamas laisvas kanalas. Dėl paslėptos stoties nuotolio nuo kitų BSS stočių ar skirtingos perdavimo galios, neteisingai įvertinama laisvo kanalo sąlyga. Dėl šios priežasties kolizijų skaičiavimo metu vertinama paslėpto įrenginio sąlyga ir tikimybė skaičiuojama atitinkamai.

Tiesioginės įvyksta (DC) kai dvi stotys inicijuoja duomenų perdavimą tuo pačiu metu, tokios kolizijos tinkle įvyksta nepriklausomai nuo paslėptos stoties sąlygos. Visos kitos tinkle įvykstančių kadrų praradimo priežastys priskiriamos su kanalo kokybe susijusioms problemoms. Tokiu atveju kolizijos tikimybė dėl susidūrusių kadrų yra minimali ir atsirandančios kolizijos buvo sukeltos nepalankių kanalo sąlygų ar aplinkos triukšmų.

2.3.3.3.1. Tiesioginės kolizijos tikimybė

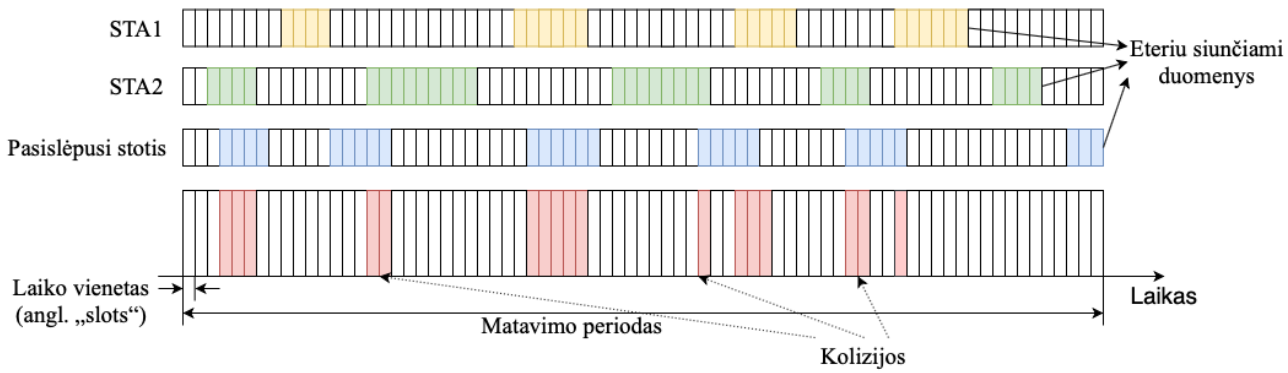
Analizės metu nenustačius paslėptų stočių, manoma, kad įrenginiai perduodami duomenis eteriu girdi vienas kitą. Tokiu atvejų CSMA/CA mechanizmas veikia korektiškai nustatydamas laisvą kanalą ir perduodamų duomenų kolizijos įvyksta tik dėl atsitraukimo sinchronizacijos [9]. Tokio tipo kolizijos yra priskirtos prie tiesioginių kolizijų (angl. „direct collision probability“, DC).

$$\boldsymbol{P_{DC}} = \mathrm{N}\left(\frac{1}{CW\_min}\right)\left(1-\left(\frac{1}{CW\_min}\right)\right)^{N-1}; \quad (7)$$

čia P_{DC} – tiesioginių kolizijų tikimybė vieno periodo metu; CW_min – konkurencinio lango minimali vertė, N – periodo metu prisijungusių stočių skaičius.

2.3.3.3.2. Neplanuotos kolizijos tikimybė

Neplanuotos kolizijos (angl. „staggered collision“, P_{SC}) tikimybė skaičiuojama tuo atveju jei tinkle yra paslėptas įrenginys ir stotys konkuruoja dėl eterio laiko. Šio tipo kadrų kolizijos įvyksta kai paslėpta stotis inicijuoja komunikaciją su prieigos tašku kai kanalas yra užimtas AP pusėje. Neplanuotos kolizijos (SC) gali įvykti jei stotis siųsdama kadrus įsiterpia į kitų tinklo įrenginių komunikaciją (DCF tinkluose STA su AP) arba kai paslėptos stoties išsiųstas kadras esant laisvam kanalui pasiekia priimantį (angl. „receiving“) įrenginį, kai kanalas jau yra užimtas.



15 pav. Neplanuotos kolizijos situacijos modelis

Reguliuojamų kolizijų tikimybė (P_{SC}) apskaičiuojama (8) susumavus visų tinkle paslėptų stočių kolizijos tikimybes. Kadangi paslėpta stotis gali sukelti koliziją įsiterpusi į kitų įrenginių duomenų perdavimo procesą, tikimybė vertinama atsižvelgiant į paslėptos ir visų kitų to periodo stočių perduotų duomenų išnaudotus eterio laikus (9). Stoties eterio laikas $Airtime_i$ matuojamas laiko vienetais (angl. „time slots“). Toks tikimybės skaičiavimas remiasi prielaida, kad bet kuris paslėptos stoties naudotas laiko vienetas galėjo sukelti duomenų koliziją.

$$\boldsymbol{P_{SC}} = \sum_{i=n_{hid}}^{i=1}\left(\frac{\boldsymbol{Airtime_i[slots]}}{\left(\sum_{j=n}^{j=1}\boldsymbol{Airtime_j[slots]}\right)-\boldsymbol{Airtime_i[slots]}}\right); \quad (8)$$

čia P_{SC} – neplanuotų kolizijų tikimybė vieno periodo metu, $Airtime_i$ – paslėptos stoties naudotas eterio laiko vienetų kiekis periode, $Airtime_j$ – einamosios stoties sunaudotas eterio laiko vienetų kiekis periode. Stoties sunaudotas eterio laikas apskaičiuojamas pagal (9) formulę.

$$\boldsymbol{Airtime_{sta}} = \frac{\bar{\boldsymbol{t}}_{\boldsymbol{tx/rx}}}{\boldsymbol{t_{slot}}}\ \boldsymbol{[slots]}; \quad (9)$$

čia $Airtime_{sta}$ – stoties naudotas eterio laiko vienetų kiekis periode, $\bar{t}_{tx/rx}$ – perduotų ar priimtų kadrų sugaištas eterio laikas vieno periodo metu, t_{slot} – vieno laiko vieneto užimamas laikas [µs].

Apskaičiavus vidutinį kadro dydį galima įvertini apytikslį stoties užimamo eterio laiką (10). Kadro erdvėje trukmę nuo STA iki AP ir atvirkščiai lemia perduodamo kadro ilgis $\bar{L}_{tx/rx}$, (šiuo atveju ilgio vidurkis) ir kadro perdavimo sparta.

$$\bar{\boldsymbol{t}}_{\boldsymbol{tx/rx}} = \frac{\bar{\boldsymbol{L}}_{\boldsymbol{tx/rx}}}{\boldsymbol{Rate}_{\boldsymbol{tx/rx}}}; \tag{10}$$

čia $\bar{t}_{tx/rx}$ – stoties išnaudotas eterio laikas vieno periodo metu; $\bar{L}_{tx/rx}$ – perduotų arba priimtų kadrų ilgis vieno periodo metu; $Rate_{tx/rx}$ – duomenų kadrų sparta.

Kadro ilgis, bei duomenų perdavimo sparta naudojama apskaičiuojant laiko trukmę vienam kadrui keliauti, nuo siuntėjo iki gavėjo. Kiekvieno statistikų priėmimo periodo pradžioje suskaičiuojama kiekvienos stoties perduotų duomenų kadro ilgio vidurkio reikšmė baitais (11).

$$\bar{\boldsymbol{L}}_{\boldsymbol{tx/rx}} = \frac{\boldsymbol{data}_{\boldsymbol{tx/rx}}}{\boldsymbol{N}_{\boldsymbol{tx/rx}}}; \tag{11}$$

čia $\bar{L}_{tx/rx}$ – MPDU dydžio vidurkis matuojamas [B], $data_{tx/rx}$ – perduotas ar priimtas duomenų kiekis baitais, $N_{tx/rx}$ – perduotų arba priimtų duomenų kadrų kiekis.

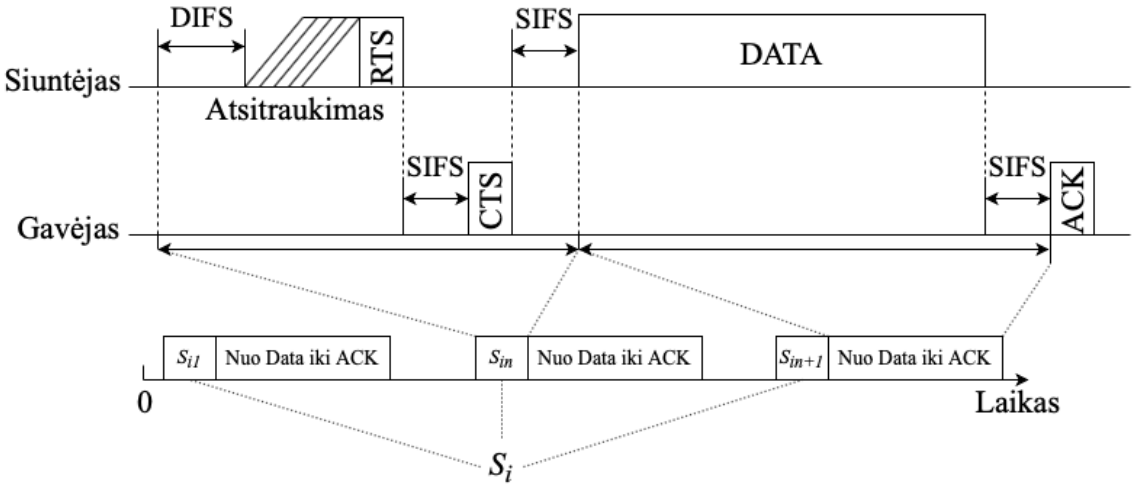
2.3.3.4. Įvykusios kolizijos eterio laiko įvertinimas

Atsižvelgiant į kolizijos tikimybę apskaičiuojama, kokia eterio laiko dalis periode buvo papildomai išnaudota dėl įvykusių kolizijų. Apskaičiavimas atliekamas viso sugaišto eterio laiko ir apskaičiuoto kolizijos tikimybės santykiu.

2.3.3.5. Kanalo rezervacijos trukmė

Kai BSS tinkle naudojamas RTS/CTS, kaip minėta analizėje, šis mechanizmas reikalauja kanalo rezervacijos prieš užimant kanalą. Analizėje nustatyta, kad tai sukelia papildomas eterio laiko sąnaudas. Kanalo rezervacijos trukmę *Si* (16 pav.) sudaro valdymo kadrų ir tarpkadrinių intervalų eterio laikas (12). Šis matmuo kinta priklausomai nuo naudojamo dažnio, kuris lemia DIFS intervalo ilgį, bei spartos, kuria yra perduodami kontrolės kadrai. RTS yra universalus kontrolės kadras, kuris privalo būti siunčiamas visiems radijo erdvės dalyviams, todėl jis perduodamas mažiausia BSS stočių palaikoma sparta.

Eterio rezervacijos trukmę (16 pav.) sudaro laiko intervalas nuo atlaisvėjusio kanalo iki duomenų kadro perdavimo pradžios. Į šį intervalą patenka: DIFS trukmė; atsitiktinio atsitraukimo pasirinktas laikas; RTS kadro sunaudotas eterio laikas; SIFS intervalo trukmė ir CTS kadro sunaudotas eterio laikas.



16 pav. Kanalo rezervacijos S_i trukmės sudedamosios dalys

CTS kontrolės kadrą suprasti privalo BSS zonos stotys, todėl šio tipo valdymo kadrai perduodami mažiausia BSS palaikoma perdavimo sparta. Kanalo rezervacijos trukmė apskaičiuojama (12) susumavus visų sudedamųjų daliu išnaudotą eterio laiką.

$$\boldsymbol{S_i = t_{DIFS} + t_{RTS} + t_{SIFS} + t_{CTS};} \tag{12}$$

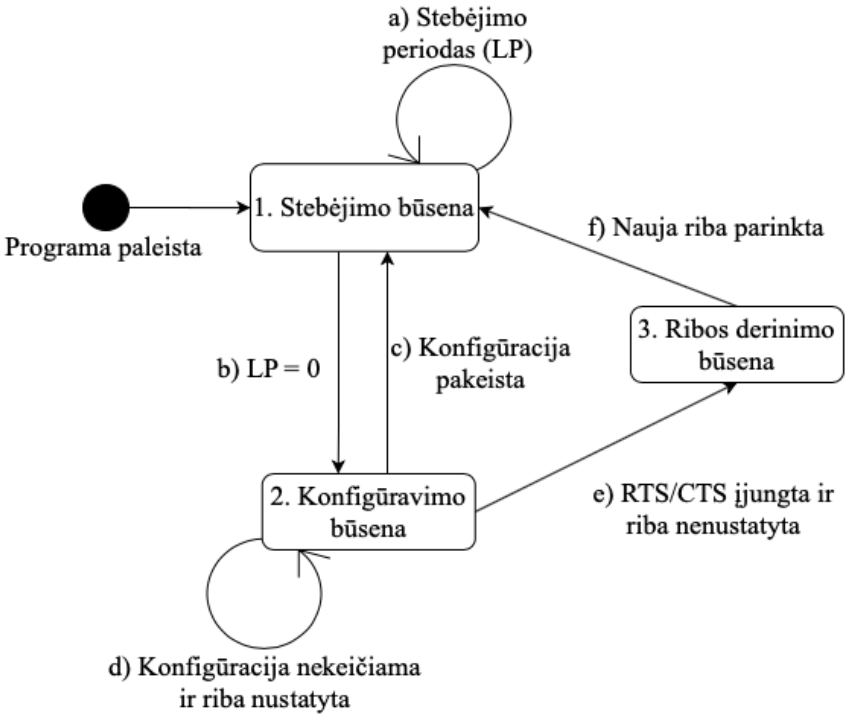
čia t_{DIFS} - DIFS trukmė, t_{RTS} – RTS kadro sunaudotas eterio laikas; t_{SIFS} – SIFS intervalo trukmė; t_{CTS} – CTS kadro sunaudotas.

Informacijos srauto sulėtėjimas, sukeltas dėl kanalo rezervacijos S_i, apskaičiuojamas kiekvieno periodo metu. Skaičiuojama naudojant rezervacijų kiekį periodo metu bei vienos rezervacijos užimtą eterio laiką. $S_c = n * S_i$; kai n – kanalo rezervacijų kiekis per ciklą ir S_i – kanalo rezervacijos trukmė.

2.3.4. Sprendimo priėmimo modulis

Kolizijų valdymo metodo modulis, kuriame vykdomas įrenginio kanalo prieigos mechanizmo konfigūracijos ir aktyvacijos ribos parinkimas, vadinamas – sprendimo priėmimo moduliu. Sprendimo priėmimo modulio įvesties duomenys yra tiesiogiai imami iš analizės modulio registruojamo istorinių duomenų dėklo. Modulis seka esamą prieigos taško konfigūraciją operatyvioje atmintyje. Sprendimo priėmimo modulio darbo procedūra inicijuojama iš karto po analizės modulio apskaičiuotų reikšmių patalpinimo į duomenų dėklą. Modulio veiklos principas yra sekti registruojamas istorinių duomenų dėklo reikšmių vidurkius ir generuoti įrangos konfigūraciją.

Sprendimo priėmimo modulis veikia keisdamas būsenas (17 pav.) į stebėjimo būseną, konfigūravimo būseną ir ribos derinimo būseną.



17 pav. Sprendimo priėmimo modulio būsenų diagrama

Stebėjimo būsena (1) skirta įvertinti naujai pritaikytos konfigūracijos įtaką. Kadangi konfigūracijos pakeitimas yra ilgas procesas eterio laiko atžvilgiu (~ 10s), todėl dažnas konfigūracijos keitimas gali sukelti per didelį radijo tylos (angl. „radio silence“) periodą, kuris turi neigiamą įtaką WLAN tinko paslaugos kokybei. Stebėjimo būsena tęsiasi (a) kol kintamasis žymintis apsimokymo periodą LP (angl. „learning period“) pasiekia nulinę reikšmę (b). LP programos paleidimo pradžioje yra nustatomas parinkto slankaus lango SW ilgis.

Konfigūravimo būsenoje (2) metodas siekia įvertinti RTS/CTS mechanizmo naudą esamai tinklo būklei. Nusprendus nekeisti konfigūracijos (d) sprendimo priėmimo modulis išlieka konfigūravimo būsenoje. Nusprendus pakeisti konfigūraciją (c), būsena keičiama į stebėjimo (1), kol nusistovės pastovūs tinklo parametrai (t. y. pasibaigs stebėjimo periodas).

Ribos derinimo būsena (3) aktyvuojama tuo metu, kai RTS/CTS mechanizmas yra įjungtas, tačiau efektyvi RTS/CTS aktyvacijos riba nėra nustatyta (e). Sprendimo priėmimo modulis patenka į šią būseną kol ribos kitimo žingsnis pasiekia minimalią reikšmę. Tokiu atveju ribos derinimas yra baigtas ir modulis po stebėjimo periodo grįžta į konfigūracijos būsenos ciklą.

2.3.4.1. RTS/CTS įjungimo sprendimas

Vienas iš sprendimo priėmimo modulio tikslų yra nustatyti ar esamai tinklo būklei stebėtu laikotarpiu yra naudinga naudoti RTS/CTS kontrolės mechanizmą ar ne. Tai atliekama su prielaida, kad tinklo būklė stebėtu laikotarpiu išliks nepakitusi.

Daromas sprendimas atsižvelgiant į stebėto istorinio laikotarpio perduotų duomenų ir kolizijų tikimybės rodiklius (4 lentelė.).

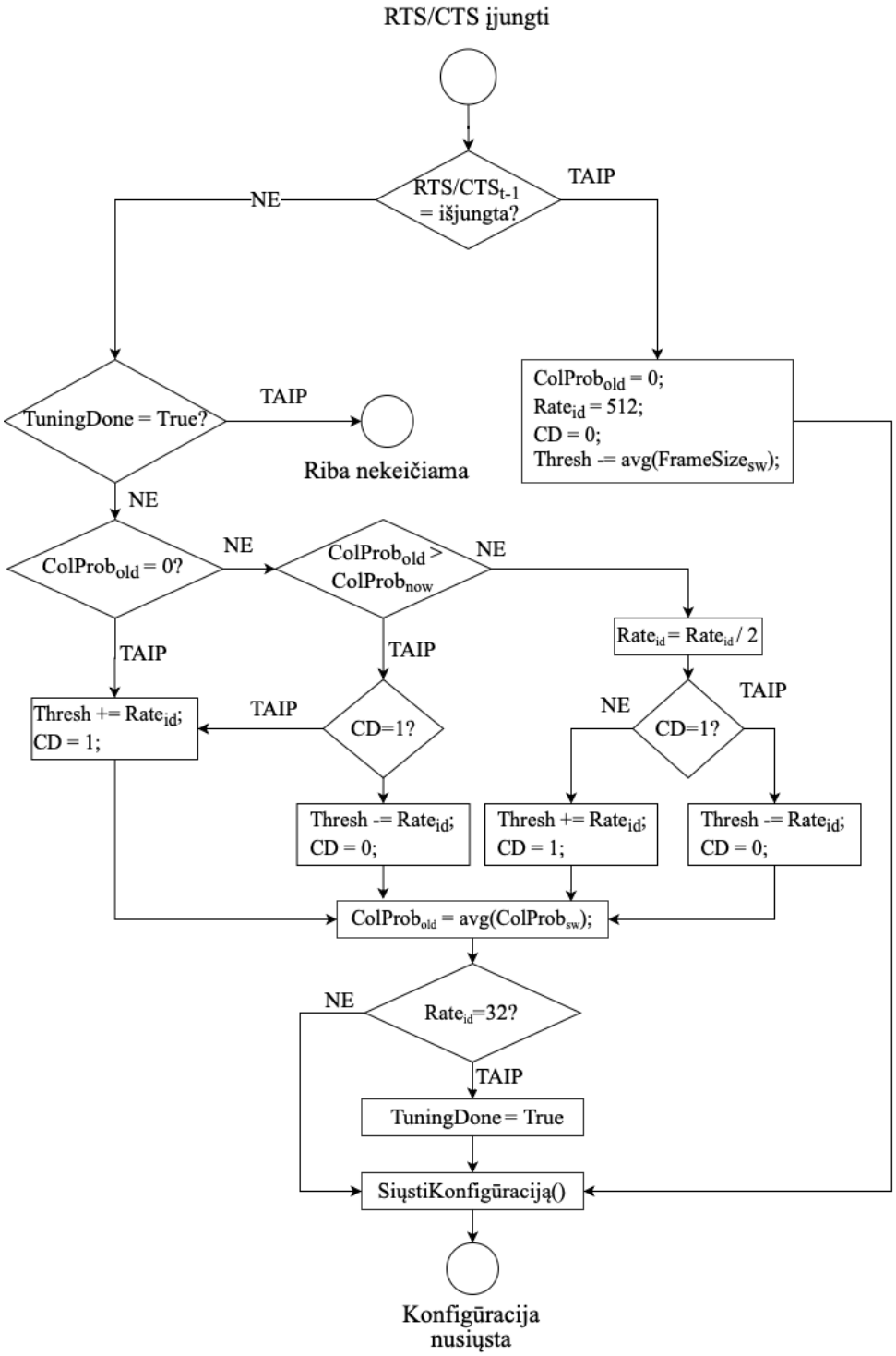
4 lentelė. RTS/CTS mechanizmo įjungimo/išjungimo algoritmas

Naudojamas algoritmas
Jei ($Cost_{RTS/CTS\ on} > Cost_{RTS/CTS\ off}$):
išjungti RTS/CTS
kitu atveju:
įjungti RTS/CTS

Jei įvykusių kolizijų sukeltas kanalo vėlinimas didesnis už RTS/CTS valdymo kadrų tuo pačiu laikotarpiu apskaičiuotą kanalo vėlinimą, tokiu atveju RTS/CTS kontrolės kardai įjungiami. Priešingu atveju RTS/CTS kontrolės kadrai nenaudojami (neįjungiami).

2.3.4.2. Aktyvacijos ribos apskaičiavimas

Kolizijos mechanizmo aktyvacijos riba (RT) žymi minimalų kadro dydi [Baitais]. Perduodant kadrą į eterį, kurio ilgis didesnis už aktyvacijos ribą (RT), stotis privalo rezervuoti kanalą. Aktyvacijos ribos kitimo rėžis [0; 2346] baitais nustatoma atsižvelgiant į istorinį kolizijos tikimybės apskaičiavimo rodiklį. Ribos apskaičiavimas atliekamas tik tuo metu, jei RTS/CTS mechanizmas yra įjungtas. Aktyvacijos riba apskaičiuojama tam tikra veiksmų seka (18 pav.).



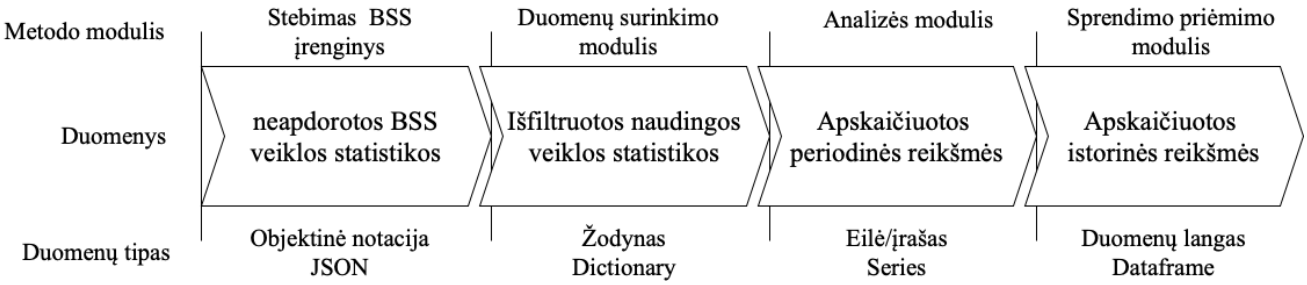
18 pav. Aktyvacijos ribos apskaičiavimo veiksmų seka

Ribos derinimo (angl. „threshold tuning“) būsenoje keičiama aktyvacijos ribos reikšmė atsižvelgiant į kolizijos tikimybės rodiklį. Pirmą kartą įjungus RTS/CTS mechanizmą aktyvavimo riba prilyginama vidutiniam istorinio laikotarpio perduotų kadrų ilgiui. Riba didinama tol, kol kolizijos tikimybė mažėja. Jei padidinus ribą kolizijos tikimybė padidėjo, tada ribos keitimo žingsnis mažinamas pusiau ir aktyvacijos riba mažinama. Procesas kartojamas kol kitimo žingsnis yra didesnis už 32. Tai reiškia, kad ribos kitimo kryptis buvo pakeista (iš didinimo į mažinimą ir atvirkščiai) 4 kartus ir surasta optimali RTS/CTS ribos konfigūracija prie kurios kolizijos tikimybės vidurkis yra mažiausias.

Pasikeitus kolizijos tikimybės vidurkiui po sėkmingo ribos parinkimo, riba pažymima kaip nesuderinta ir derinimo procesas pradedamas iš pradžių.

2.4. Duomenų modelis

Projektuojamo metodo duomenų išgryninimo procesas (19 pav.) vyksta kiekviename metodo modulyje. WLAN tinklo prieigos taškas periodiškai generuoja JSON tipo objektinės notacijos pranešimą. Pranešimo informacija filtruojama duomenų surinkimo modulyje ir toliau perduodama analizės moduliui kaip žodynas ({„raktas“: „reikšmė“}). Analizės modulis atlieka reikalingus skaičiavimus ir suformuoja įrašą pažymėtą laiko žyme. Toliau šis įrašas talpinama duomenų dėkle taip suformuojant istorinių duomenų struktūrą (angl. „data frame“). Remiantis duomenų lange esančiais istoriniais įrašais sprendimo priėmimo modulis atlieka skaičiavimus ir gr1žina rezultatą.



19 pav. Duomenų migracijos modelis metodo gyvavimo cikle

Pagrindinė metodo manipuliuojama serverio pusės duomenų struktūra yra duomenų langas arba lentelė. Lentelėje saugomos analizės metu apskaičiuotos veiklos parametrų reikšmės. Vienas lentelės įrašas žymi vieno periodo tinklo būklės apskaičiuotus veiklos parametrus. Lentelė veikia FILO (angl. „first in last out“) dėklo principu.

Dėklas turi iš anksto apibrėžtą dydį – slankų langą SW (angl. „sliding window“), kuris atitinka istorinių įrašų laikotarpį. Dėkle saugomų istorinių įrašų trukmė priklauso nuo raportavimo greičio. Dėklas yra pildomas analizės modulio skaičiavimais kol persipildo (įrašų kiekis tampa didesnis už dėklo dydį). Kai lentelė užsipildo paskutinis gautas įrašas yra šalinamas taip išlaikant slenkantį duomenų langą. Duomenų kadro (angl. „data frame“) struktūra naudojama dėklo funkcijai realizuoti.

2.5. Projektinės dalies išvados

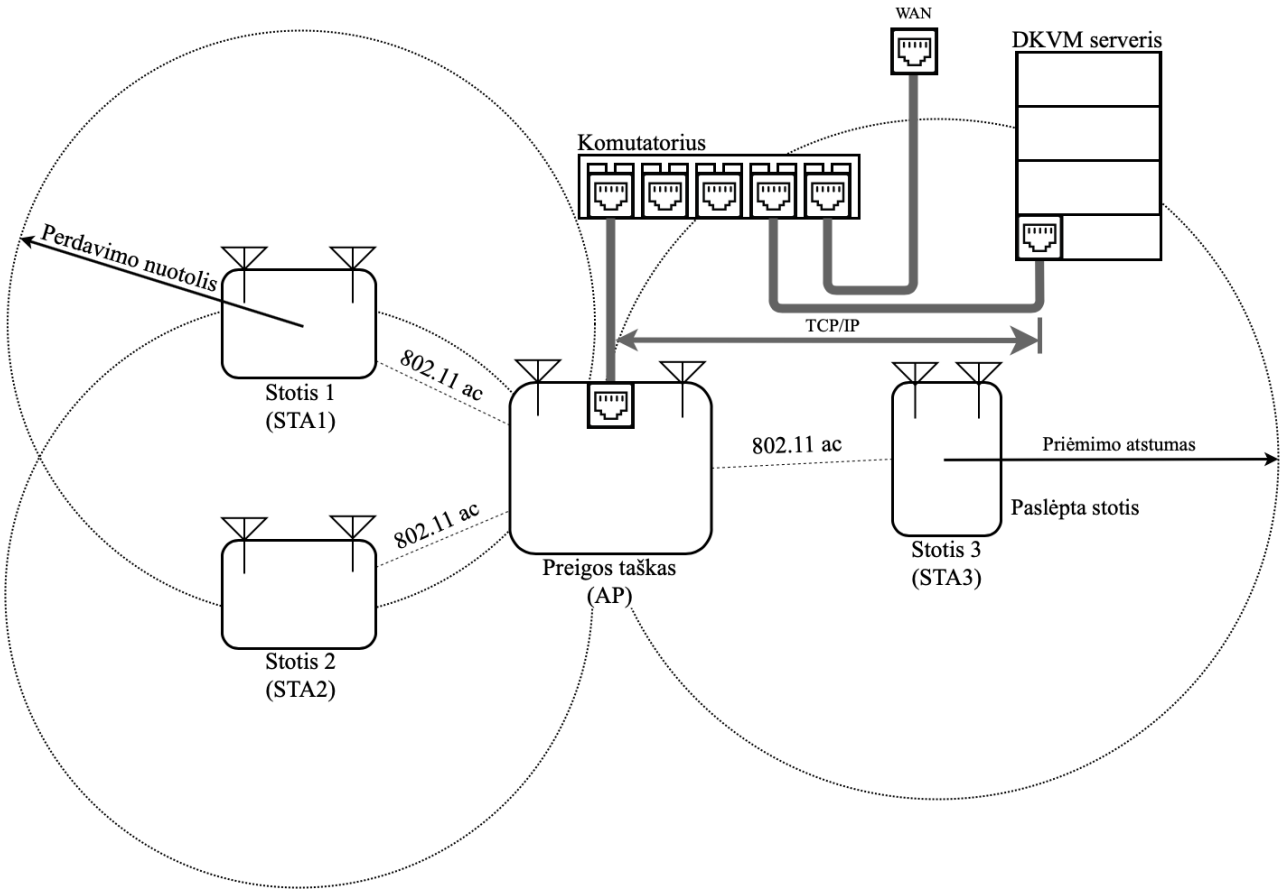
Baigiamojo darbo projektinėje dalyje, įvertinus ir išanalizavus paskirstytosios koordinuotos funkcijos kanalo prieigos mechanizmo trūkumus, suprojektuotas dinaminis kolizijų valdymo metodas. Metodas tikslas – stebint belaidžio tinklo charakteristikas, valdyti paskirstytosios koordinuotosios funkcijos tinklo kanalo prieigos mechanizmą nustatant optimalią RT ribą, užtikrinančią maksimalų duomenų pralaidumą ir efektyvų eterio laiko išnaudojimą.

Dinaminio kolizijų valdymo metodo architektūrą sudaro WLAN prieigos taškas ir dedikuotas serveris. Prieigos taške veikiantis agentas atlieka dvi pagrindines funkcijas: periodiškai perduoda surinktas statistikas į dedikuotąjį serverį ir priima naują konfigūraciją. Serveryje įvertinamas esamo tinklo įrenginio konfigūracijos efektyvumas, lyginant kolizijos tikimybę ir kanalo rezervacijos sąnaudas eterio laiko atžvilgiu. Periodiniai skaičiavimai kaupiami duomenų lange, kur yra analizuojami sprendimo priėmimo modulio. Kiekvieno duomenų patalpinimo į dėklą metu

įvertinamas stebimo laikotarpio kolizijų kiekis panaudojant kolizijų tolerancijos ribą. Nustačius, kad kolizijų kiekis viršija tolerancijos ribą, nusiunčiama nauja konfigūracija prieigos taškui. Prieigos taškas pasikeičia gautą konfigūraciją ir atnaujina veiklą tinkle. Suprojektuotas metodas leidžia reaguoti į pasikeitusias tinklo sąlygas dinamiškai valdant prieigos taško konfigūraciją.

3. Dinaminio kolizijos valdymo metodo prototipas

Prototipo dalyje aprašomas suprojektuoto dinaminio kolizijų valdymo metodo realizuotas prototipas. Aprašoma naudota programinės įrangos architektūra (3.1), pateikiamas realizuotas informacinis modelis (3.2), prototipo realizacijos priemonės (3.3), aprašoma naudota techninė įranga (3.4), pateikiamos prototipo realizacijos išvados (3.5).



20 pav. Dinaminio kolizijų valdymo metodo koncepcinė diagrama

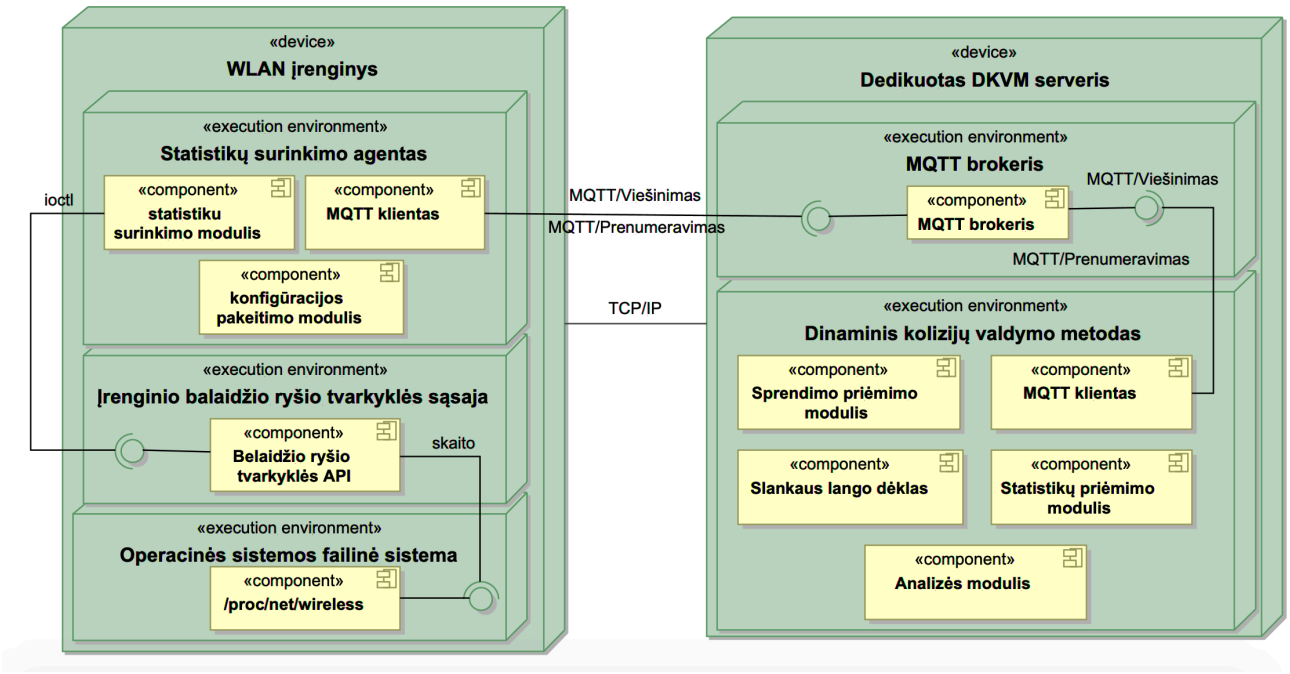
Paveiksle (20 pav.) pavaizduotas prototipo koncepcinis modelis su komponentų tarpusavio komunikacijos ryšiais. Prototipą sudaro 3 stotys, vienas prieigos taškas ir DKVM serveris, kuriame realizuotas suprojektuotas metodas. Stotys (STA1-3) su prieigos tašku yra asocijuotos (AP) belaidžio ryšio 802.11ac technologija. Prieigos taškas per komutatorių siunčia statistikas į serverį, TCP/IP protokolų rinkiniu, per komutatorių.

3.1. Dinaminio kolizijų valdymo prototipo architektūra

Sudarytas dinaminio kolizijų valdymo prototipo architektūrinis modelis (21 pav.). Metodo architektūra susideda iš dviejų įrenginių: belaidžio ryšio įrenginio ir dedikuoto serverio. WLAN įrenginyje veikia statistikų surinkimo agentas, kurį sudaro 3 pagrindiniai komponentai:

1. **Statistikų surinkimo modulis** – periodiškai vykdo sisteminius šaukinius (angl. „system call“), per WLAN tvarkyklės sąsają, naudojant operacinės sistemos failus;

2. **MQTT klientas** – statistikų surinkimo modulio grąžintas statistikas perduoda SSL/TLS (TCP/IP) kanalu į dedikuotą DKVM serverį, priima iš dedikuoto serverio MQTT brokerio publikuotą RTS/CTS konfigūraciją ir perduoda konfigūracijos pakeitimo moduliui;
3. **Konfigūracijos pakeitimo modulis** – pritaiko naują konfigūraciją, gautą iš dedikuotojo serverio per MQTT klientą.



21 pav. Dinaminio kolizijų valdymo prototipo diegimo modelis

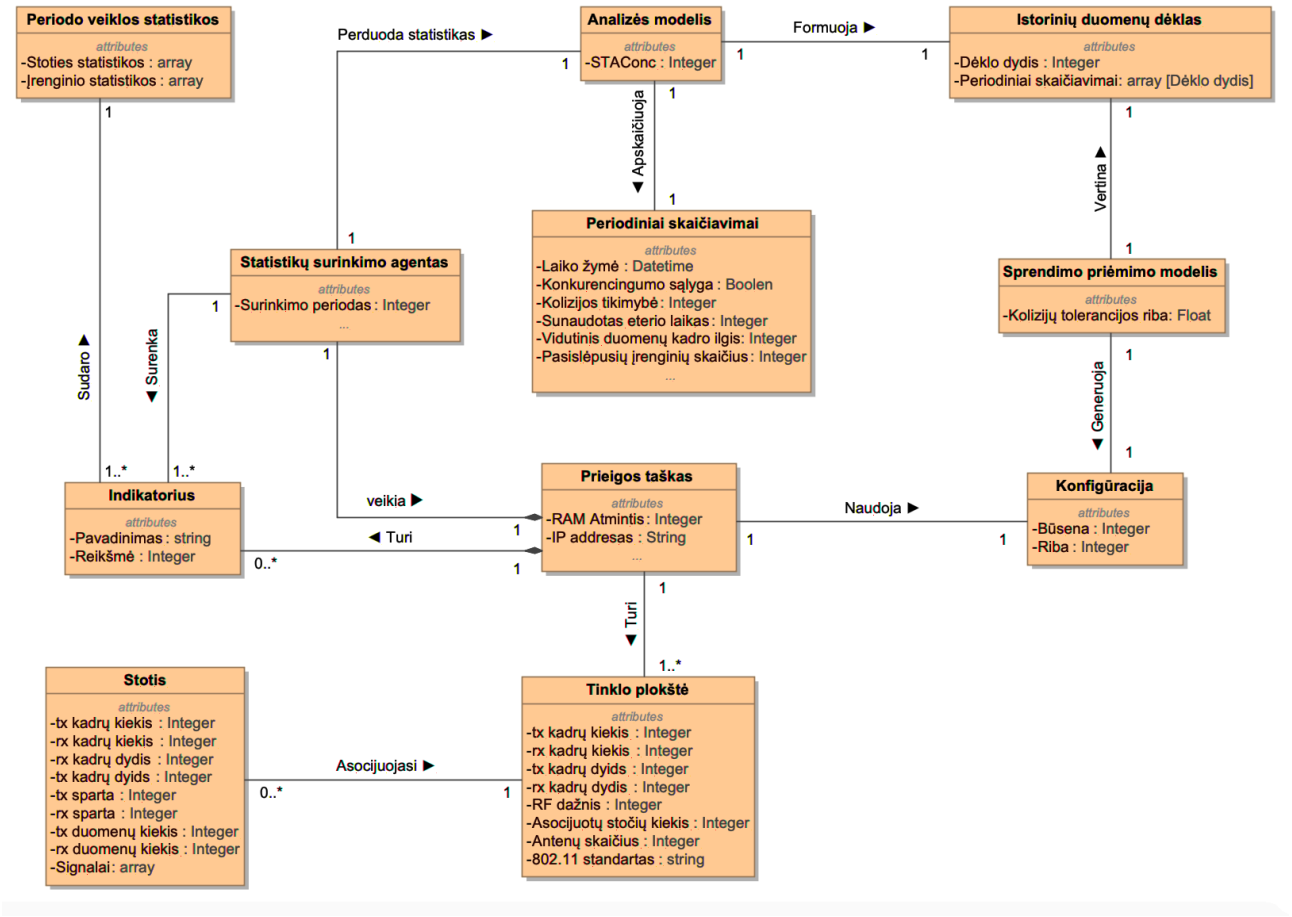
Dedikuotą DKVM serverį sudaro dvi sudedamosios dalys:

1. **MQTT brokeris** – užtikrina duomenų mainus viešinimo/prenumeravimo principu į nustatytą temą, užtikrina duomenų konfidencialumą SSL/TSL šifravimo paslauga;
2. **Dinaminis kolizijų valdymo metodas** – realizuotas sukurto metodo funkcionalumas, kurį sudaro 5 pagrindinės dalys:
   2.1. **MQTT klientas** – priima WLAN įrenginio periodiškai publikuojamas statistikas iš MQTT brokerio, publikuoja sprendimo priėmimo modulio grąžintą konfigūraciją, skirtą WLAN įrenginiui.
   2.2. **Statistikų priėmimo modulis** – transliuoja publikuotas WLAN įrenginio statistikas (iš JSON formato) į vieno lygmens žodyno (angl. „dictionary“) duomenų tipą, filtruoja naudingas statistikas ir inicijuoja duomenų analizės modulį, perduodamas apdirbtas statistikas;
   2.3. **Analizės modulis** – atlieka vieno periodo reikšmės skaičiavimus su priimtomis statistikomis, kurios yra priimtos iš WLAN įrenginio. Suformuoja eilės (angl. „series“) duomenų tipo vektorių ir patalpina jį į slankaus lango dėklą;
   2.4. **Slankaus lango dėklas** – duomenų klasė, talpinanti stebimo periodo analizės modulio apskaičiuotus duomenis. Pasirūpina seniausių duomenų šalinimu iš dėklo, kai šis perpildomas;
   2.5. **Sprendimo priėmimo modulis** – atlieka stebimo periodo reikšmės skaičiavimus ir atitinkamai atlieka įrenginio konfigūracijos parinkimo bei *RT* ribos nustatymo funkcijas.

Perduoda suformuotą konfigūraciją MQTT klientui, kuri vėliau yra perduodama WLAN įrenginiui.

3.2. Dinaminio kolizijų valdymo prototipo informacinis modelis

Šiame poskyryje pateikiamas realizuoto prototipo informacinis modelis, išreikštas koncepciniu duomenų modeliu. Informaciniame modelyje (22 pav.) pavaizduoti realizuotos dinaminio kolizijų valdymo metodo prototipo ryšiai, jungiantys suprojektuoto metodo informacinius komponentus.



22 pav. Dinaminio kolizijų valdymo metodo koncepcinis duomenų modelis

Prisijungusi stotis atlikdama duomenų perdavimo ir priėmimo operacijas keičia prieigos taške registruojamas statistikas. Statistikų surinkimo agentas perduoda statistikas į serveryje veikiantį kolektorių. Kolektorius apdirba statistikas ir perduoda jas analizės moduliui. Analizės modulis atlieka periodinius skaičiavimus ir talpina rezultatus istorinių duomenų dėkle. Sprendimo priėmimo modulis vertina sukauptas istorinio duomenų dėklo reikšmes ir generuoja prieigos taško konfigūraciją, kuri nustato kanalo prieigos mechanizmo veiklos taisykles.

3.3. Prototipo realizacijos priemonės

Prototipo realizacijai buvo naudotos šiame skyriuje aprašytos programavimo technologijos ir atviro kodo bibliotekos. Prieigos taško statistikų surinkimo agentas yra realizuotas C kalba. Agentas naudoja (5 lentelė.) „Paho MQTT C“ biblioteką komunikacijai su serveriu ir „IOCTL“ biblioteką komunikacijai su tinklo plokštės tvarkyklės sąsaja.

5 lentelė. Prieigos taške naudotos bibliotekos

biblioteka	Funkcionalumas darbe	Paskirtis
eclipse paho MQTT C	AP pusės MQTT klientas	MQTT komunikacijos biblioteka
ieee80211_ioctl	Statistikų surinkimo iš WLAN tvarkyklės	Komunikacija su tvarkykle

Serverio pusės statistikų analizės ir konfigūracijos generavimo programa realizuota „python“ kalba. Programa naudoja „Paho MQTT python“ ir „SSL“ bibliotekas bendravimui su brokeriu ir duomenų šifravimui. Istorinių duomenų dėklas realizuotas naudojant „Pandas“ biblioteką.

6 lentelė. Prototipo serverio pusėje naudotos bibliotekos

biblioteka	Funkcionalumas darbe	Paskirtis
pandas	Istorinių duomenų dėklas	Aukšto lygmens duomenų analizės biblioteka
paho-mqtt 1.5.0	Serverio pusės MQTT klientas	MQTT komunikacijos biblioteka
ssl	MQTT žinučių šifravimas	Kriptografinė biblioteka
json	Žinučių apdirbimas	Java objektu biblioteka

Be aukščiau aprašytų bibliotekų, serveryje įdiegtas MQTT brokeris „Mosquitto“, kuris suteikia pranešimų apsikeitimo platformą prototipo realizacijai. MQTT brokeris veikia viešinimo/ prenumeravimo principu. Norintis perduoti duomenis naudotojas privalo būti autentifikuotas brokerio vartotojas, kuris vykdo publikaciją į iš anksto apibrėžtą temą. Visa komunikacija yra šifruota SSL/TLS paslauga naudojant sertifikatus.

3.4. Naudotos techninės įrangos charakteristikos

Prototipe naudojamas prieigos taškas palaiko 2.4GHz ir 5GHz dažnio spektro komunikaciją 802.11n/ac protokolu (7 lentelė.). Prieigos taškas yra tiesioginės jungties (angl. „direct attach“) architektūros ir turi 3 LAN sąsajas. Prietaise veikia „Linux“ operacinė sistema su 4.14 versijos branduoliu.

7 lentelė. Prototipe naudoto prieigos taško techninės charakteristikos

LAN sąsajos	2.4 GHz WLAN standartas	5 GHz WLAN standartas	Operacinė sistemas
3	2x2 11a/b/g/n	2x2 11n/ac	Linux 4.14

Visos trys stotys pasižymi identiškomis arba panašiomis techninės įrangos savybėmis, kurios nedaro įtakos stebėjimų rezultatams. Stotys palaiko tik 5GHz dažnio spektrą ir taip pat yra tiesioginės jungties architektūros. Stotys turi 2 LAN sąsajas ir naudoja „linux“ operacinę sistemą.

8 lentelė. Prototipe naudotų stočių techninės charakteristikos

LAN sąsajos	5 GHz WLAN standartas	Operacinė sistema
2	2x2 11n/ac	Linux 4.14

Visa darbo metu naudojama WLAN techninės įrangos tvarkyklė kompiliuojama iš pirminio kodo (angl. „source code“). WLAN techninei įrangai valdyti naudojamos įvairių versijų komercinės „Atheros“ tvarkyklės.

3.5. Prototipo realizacijos išvados

Baigiamojo darbo metu realizuotas sukurto dinaminio kolizijų valdymo metodo prototipas, kuris yra sudarytas iš prieigos taško ir dedikuoto DKVM serverio. Prieigos taške buvo realizuotas statistikas surenkantis agentas, kuris periodiškai siunčia statistikas MQTT (TLS/SSL) protokolu į brokerį, veikiantį dedikuotame DKVM serveryje. Serveryje statistikas priimanti DKVM programa realizuota „python“ kalba. Programą sudaro duomenų kolektorius, kurio MQTT klientas priima statistikas iš MQTT brokerio paskirtos temos analizės modulis, kuris apskaičiuoja periodines reikšmes ir talpina į istorinių duomenų dėklą bei sprendimo priėmimo modulis, kuris analizuoja istorinius duomenis generuodamas prieigos taško konfigūraciją. Sugeneruota konfigūracija perduodama prieigos taškui, kuris pritaiko atsiųstą konfigūraciją.

4. Dinaminio kolizijų valdymo metodo tyrimas

Vertinant sukurtą dinaminį kolizijų valdymo metodą (DKVM), buvo sudaryta tyrimo metodika, kurios metu stebimi belaidžio tinklo prieigos taško (AP) kokybiniai parametrai (bendra tinklo sparta, paketų klaidų santykis, naudotas eterio laiko vidurkis). Tyrimas atliekamas naudojant apibrėžtus scenarijus, vertinant sukurto metodo efektyvumą. Tyrimo rezultatai palyginami su analitinėje dalyje analizuotais metodais.

4.1. Tyrimo metodika

Tiriant realizuotą DKVM, atliekamas tinklo stebėjimas, keičiant prisijungusių stočių padėtį. Sudaromi eksperimentų scenarijai, apibrėžiantys stočių migravimo taisykles. Vienu atveju stotys perkeliamos, taip sudarant paslėpto įrenginio sąlygas, kitu atveju stotys išlaiko stacionarias pozicijas. Identiškos sąlygos kartojamos keičiant duomenų kolizijos valdymo konfigūraciją.

4.1.1. Tyrimo įrankiai

Tyrimo metu generuojamas UDP paketų srautas panaudojant programinę įrangą „Nepim“. Paketų generatorius buvo sukurtas laidinių tinklų srauto generacijai, kur didžiausias galimas kadro ilgis (MTU) yra 1500 baitų (programinė įranga nenumato galimybės keisti siunčiamų kadrų dydį). „Nepim“ programinė įranga sudaro sąlygas vienu metu generuoti duomenų srautą daugiau nei vienam klientui, todėl, paleidus srauto generatorių vienu metu visose stotyse, tinkle susidaro eterio laiko konkuravimo sąlyga.

4.1.2. Metodo konfigūracija

Tyrimo rezultatus nulemia keturi pagrindiniai sukurto metodo parametrai: įrenginio statistikų perdavimo periodas, slankaus duomenų lango dydis, kolizijos tikimybės aktyvacijos riba ir stebėjimo periodas. Kiekvienas scenarijus tiriamas varijuojant šiais parametrais, siekiant nustatyti efektyviausią metodo konfigūraciją.

Testavimo metu nustatyta, kad naudojamos mechanizmo aktyvacijos algoritmo reakcijos laikas yra per ilgas. Sukūrus tinkle situaciją, kai perduodami paketai susiduria, tinklo įrangos konfigūracija pakeičiama tik praėjus SW ilgio laiko tarpui. Mažinant reakcijos laiką buvo nuspręsta naudoti kolizijos tolerancijos ribą R_{col}. Ribą galima automatiškai koreguoti sekant kolizijų tikimybę tinkle, tačiau tyrime naudotos geriausius rezultatus pelniusios kolizijos tolerancijos ribas. Lentelėje (9 lentelė.) pateikti metodo konfigūruojami parametrai, nulemiantys realizuoto metodo reakcijos laiką.

9 lentelė. Dinaminio kolizijų valdymo metodo konfigūracija

Žymuo	**Paaiškinimas**
T	statistikų perdavimo periodas [Hz];
SW	slankaus lango (istorinių duomenų dėklo) dydis;
R_{col}	kolizijų tikimybės tolerancijos riba;
LP	stebėjimo periodas (kiekis).

Statistikų perdavimo periodas (T) ir slankus langas (SW) yra susiję kintamieji, nulemiantys istorinių stebinių dažnį. Esant mažai T reikšmei, istorinių duomenų langas užsipildo greičiau. T taip pat dalyvauja kolizijos tikimybės skaičiavime, nustatant vieno periodo laiko vienetų (angl. „time slot“) kiekį. Stebėjimo periodas (LP) nustato periodų kiekį, kurį metodas nesiims veiksmų po atliktos konfigūracijos pakeitimo. Tai yra skirta tam, kad pakeitus konfigūraciją būtų atnaujintas istorinių stebinių langas su pasikeitusiais tinklo parametrais. Priklausomai nuo perdavimo periodo, kinta slankaus duomenų lango saugomos laiko trukmės ilgis.

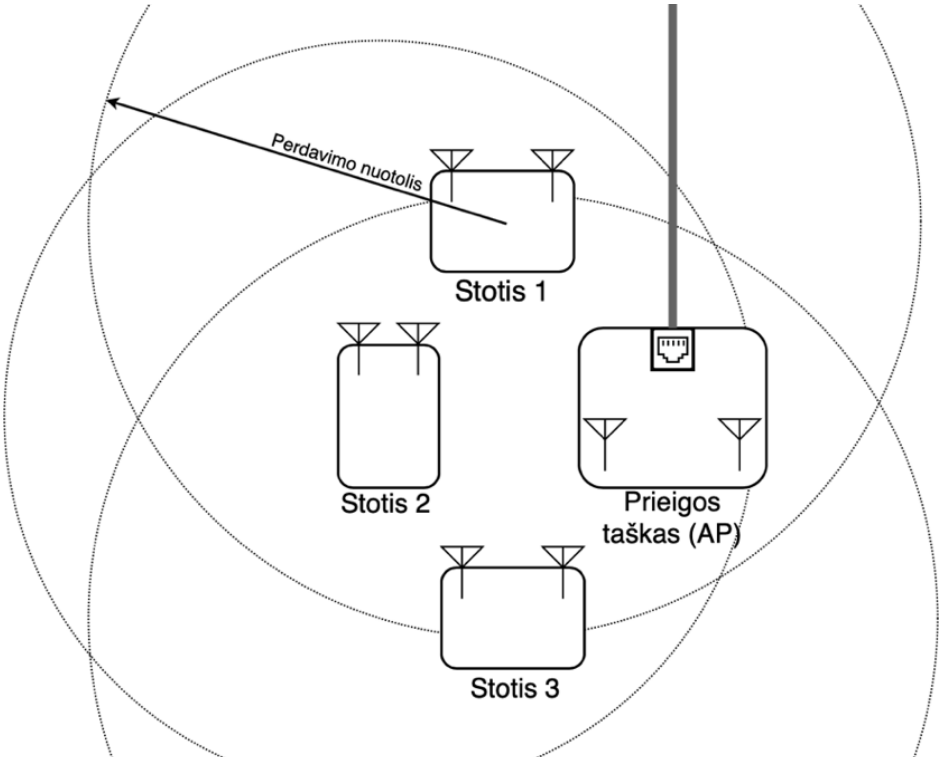
4.1.3. Tyrimo scenarijai

Esamas kolizijų valdymo metodas (RTS/CTS) su statinėmis aktyvacijos ribos RT reikšmėmis ir sukurtas dinaminis kolizijų valdymo metodo prototipas vertinami trijuose skirtinguose scenarijuose (10 lentelė.). Pirmais dviem scenarijais ($SC1$ ir SC2) siekiama ištirti paslėpto įrenginio įtaką belaidžio ryšio kokybiniams parametrams, tokiems kaip tinklo sparta, paketų klaidų ir pakartojimų santykis, sunaudojamas eterio laikas vienam duomenų kadrui. Trečiuoju scenarijumi siekiama nustatyti ar keičiantis tinklo sąlygoms naudinga pritaikyti tinklo konfigūraciją ir koks yra sukurto metodo reakcijos laikas.

10 lentelė. Tyrimo scenarijai

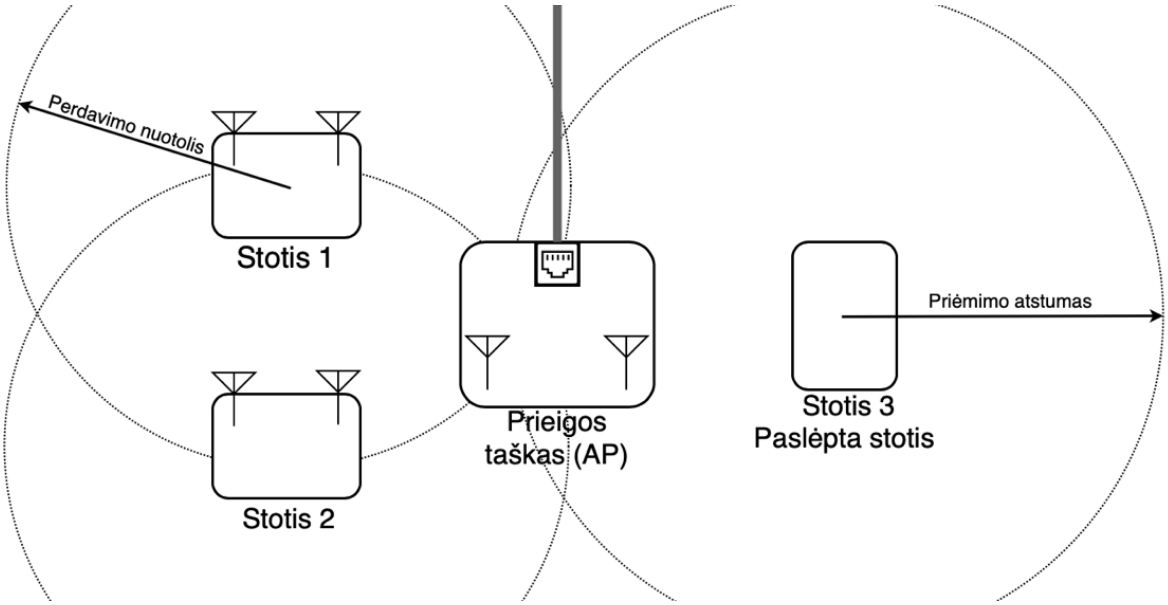
Žymuo	**Paslėptų stočių kiekis**	**Aprašymas**
$SC1$	0	Visos stotys yra stacionarios, jų buvimo vietos nėra keičiamos;
$SC2$	1	Visos stotys yra stacionarios, jų buvimo vietos nėra keičiamos;
$SC3$	0→1	Viena stotis matavimo metu perkeliama į poziciją, kurioje kitų stočių perduodamų duomenų srautas nebūtų girdima. (sudaromos paslėpto įrenginio sąlygos)

Pirmieji eksperimentai atliekami esant stacionarioms prisijungusių stočių pozicijoms. Eksperimentai atliekami esant paslėptam įrenginiui ($SC2$) ir kai paslėpto įrenginio nėra ($SC1$). Atliekamas vienas dinaminis tyrimas ($SC3$), kurio metu viena stotis perkeliama į paslėpto įrenginio poziciją.



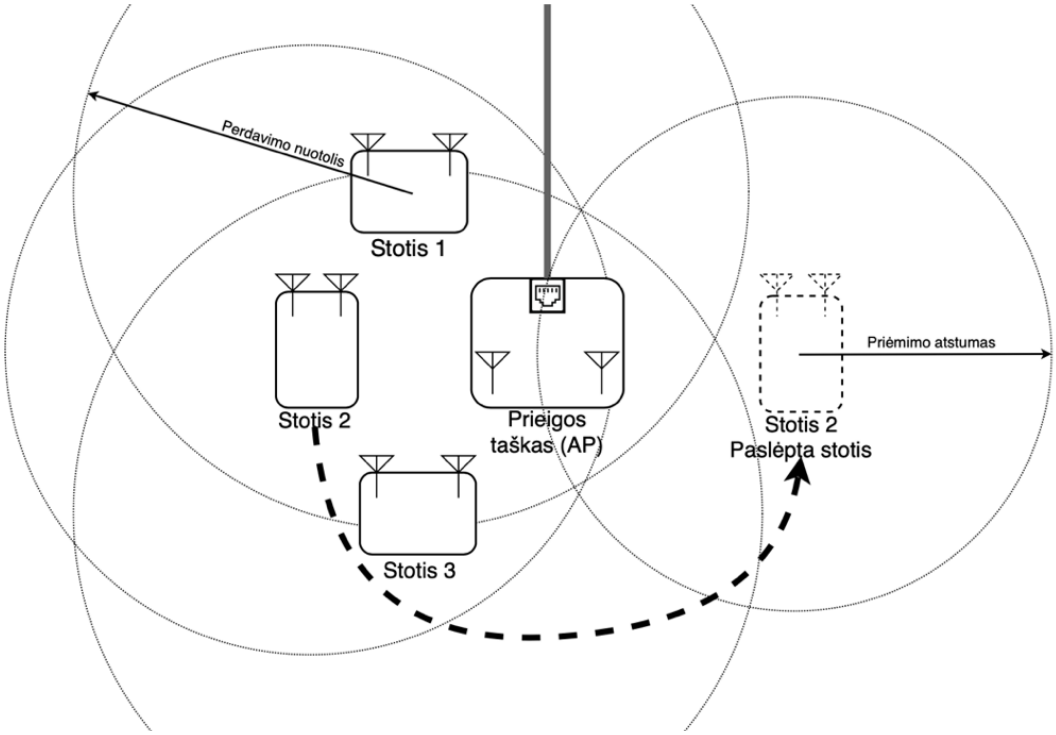
23 pav. Scenarijus 1: nėra paslėptų stočių

Pirmame scenarijuje ($SC1$) visi BSS įrenginiai girdi visų tinklo įrenginių siunčiamus duomenis, t. y. nėra paslėptų įrenginių (23 pav.).



24 pav. Scenarijus 2: viena paslėpta stotis

Antrame scenarijuje ($SC2$) sudaroma paslėpto įrenginio sąlyga. Viena stotis padedama tokiu atstumu, kad negirdėtų kitų stočių siunčiamų kadrų prieigos taškui (24 pav.). Tai realizuojama sumažinus stočių (STA1 ir STA2) perdavimo galią (angl. „transmit power“).



25 pav. Scenarijus 3: stoties pozicijos pakeitimas

Trečiuoju scenarijumi ($SC3$) atliekamas įrenginio pozicijos pakeitimas matavimo metu. Pusę numatyto matavimo laiko naudojamas (SC1), kai tinkle nėra paslėptų įrenginių. Kitą pusę pakeičiamas scenarijus, sudarant paslėptos stoties sąlygas ($SC2$). Taip siekiama nustatyti DKVM reagavimą į tinklo pasikeitimus.

4.1.4. Vertinamos statinės aktyvacijos ribos

Vertinamos esamo kolizijų valdymo mechanizmo RTS/CTS konfigūracijos kartu su sukurtu metodu. Lentelėje (11 lentelė.) pateiktos visos tyrimo metu vertinamos konfigūracijos.

11 lentelė. Vertinamos kolizijų valdymo metodo konfigūracijos

Kolizijų valdymo konfigūracija	**Aktyvacijos riba**	**Paaiškinimas**
RTS/CTS išjungta	0	Tinklas veikia įprastu režimu;
RTS/CTS įjungta	1	Kanalo rezervacija naudojama kadrams, kurių ilgis yra didesnis už 1 baitą.
RTS/CTS įjungta	10	Kanalo rezervacija naudojama kadrams, kurių ilgis yra didesnis už 10 baitų.
RTS/CTS įjungta	100	Kanalo rezervacija naudojama kadrams, kurių ilgis yra didesnis už 100 baitų.
RTS/CTS įjungta	200	Kanalo rezervacija naudojama kadrams, kurių ilgis yra didesnis už 200 baitų.
RTS/CTS įjungta	400	Kanalo rezervacija naudojama kadrams, kurių ilgis yra didesnis už 400 baitų.
RTS/CTS įjungta	800	Kanalo rezervacija naudojama kadrams, kurių ilgis yra didesnis už 800 baitų.
Tyrimo metu generuojamo srauto kadrų ilgis yra 1500 baitų;		
RTS/CTS įjungta	1600	Kanalo rezervacija naudojama kadrams, kurių ilgis yra didesnis už 1600 baitų.

RTS/CTS įjungta	2364	Kanalo rezervacija naudojama kadrams, kurių ilgis yra didesnis už 2364 baitų.
DKVM	Dinaminė	RTS/CTS įjungimą/išjungimą ir aktyvacijos ribą parenka sukurtas metodas.

Kai riba yra 2364 baitai (maksimali) RTS/CTS mechanizmas nerezervuoja eterio jokiems duomenų kadrams, tačiau RTS paketai tinkle yra siunčiami radijo perimetre esantiems prieigos taškams, todėl kontrolės kadrų kiekis išlieka didesnis lyginant su išjungta RTS/CTS konfigūracija.

4.1.5. Tyrimo rezultatų vertinimas

Tyrimo rezultatų vertinimui naudojami iš prieigos taško atsiųsti tinklo veiklos parametrai (12 lentelė.). Perduotų duomenų kiekis ir perduotų duomenų sparta yra jautriausi WLAN tinkle įvykstančioms kolizijoms, be to šie kintamieji yra registruojami realiu laiku iš WLAN įrenginio, todėl tiksliausiai atspindi naudojamo kolizijos metodo efektyvumą.

12 lentelė. Tyrimo vertinimo kriterijai

Žymuo	**Paaiškinimas**
P_{col}	vidutinė kolizijų tikimybė stebėtu laikotarpiu
$RATE_{tx/rx}$	Vidutinė visų klientų perdavimo/priėmimo sparta stebėtu laikotarpiu
$PER_{tx/rx}$	Vidutinė perduotų/priimtų paketų pakartojimo ir klaidų santykis
N_{DATA}	išsiųstų/priimtų duomenų kadrų kiekis vieno periodo metu
N_{CTRL}	išsiųstų/priimtų kontrolės kadrų kiekis vieno periodo metu
t_{air}	Eterio laikas, išnaudotas duomenų ir kontrolės kadrų perdavimui su tarpkadriniais intervalais
t_{ctrl}	Eterio laikas, išnaudotas kontrolės kadrams su tarp-kadriniais intervalais [ACK, RTS, CTS]
RT	Metodo parinkta RTS/CTS aktyvacijos riba

4.2. Stebėjimų rezultatai

Pirmiausia atliekamas tyrimas vertinant tinklo pralaidumą, kai nenaudojamas RTS/CTS mechanizmas. Šio tyrimo rezultatai lyginami su suprojektuoto dinaminio kolizijų valdymo metodo tyrimo rezultatais. Kiekvienas matavimas atliekamas 3 kartus ir rezultatų lentelėje pažymimas matavimų vidurkis, taip siekiant stabilizuoti reikšmes.

4.2.1. Tiriamų scenarijų vertinimas nenaudojant RTS/CTS mechanizmo

Lentelėje (13 lentelė.) pateikti stebimų tinklo parametrų rezultatai visais tyrimo scenarijais, kai RTS/CTS mechanizmas yra išjungtas.

13 lentelė. Tyrimo rezultatai nenaudojant RTS/CTS

Nr.	$\boldsymbol{SC}$	$\boldsymbol{T/SW}$	$\boldsymbol{P_{col}}$	$\boldsymbol{RATE_{tx}/RATE_{rx}}$	$\boldsymbol{PER_{tx}/PER_{rx}}$	$\boldsymbol{N_{DATA}}$	$\boldsymbol{N_{CTRL}}$	$\boldsymbol{t_{air}[s]}$	$\boldsymbol{t_{ctrl}}$
1.	SC1	0.1/30	0.04%	348/330	92.52/99.89	172324	5167	2.7572	0.0416
2.	SC2	0.1/30	10.71%	228/107	82.63/99.34	80464	18358	1.2875	0.1478
3.	SC3	0.1/30	5.41%	280/251	90.17/99.3	124277	10591	1.9885	0.0853

4.2.2. Tyrimo scenarijų vertinimas naudojant RTS/CTS su 1 baito riba

Lentelėje (14 lentelė.) pateikti stebimų tinklo parametrų rezultatai visais tyrimo scenarijais, kai RTS/CTS riba yra 1 baitas.

14 lentelė. Tyrimo rezultatai naudojant RTS/CTS su 1 baito riba

Nr.	SC	T/SW	P_{col}	$RATE_{tx}$ /$RATE_{rx}$	PER_{tx} /PER_{rx}	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}
1.	SC1	0.1/30	0 %	331/345	85.70/99.89	164826	74363	4.7849	1.1836
2.	SC2	0.1/30	0,37%	236/133	72.94/99.38	84385	53598	2.4524	0.8600
3.	SC3	0.1/30	0 %	282/286	84.38/99.63	126329	74135	3.6442	1.1704

4.2.3. Tyrimo scenarijų vertinimas naudojant RTS/CTS su 10 baitų riba

Lentelėje (15 lentelė.) pateikti stebimų tinklo parametrų rezultatai visais tyrimo scenarijais, kai RTS/CTS riba yra 10 baitų.

15 lentelė. Tyrimo rezultatai naudojant RTS/CTS su 10 baitų riba

Nr.	SC	T/SW	P_{col}	$RATE_{tx}$ /$RATE_{rx}$	PER_{tx} /PER_{rx}	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}
1.	SC1	0.1/30	0 %	310/338	85.99/99.52	170855	75096	4.9651	1.2029
2.	SC2	0.1/30	0,37%	238/177	69.32/99.51	80817	68995	2.3463	1.1048
3.	SC3	0.1/30	0 %	281/279	83.01/99.67	124633	80031	3.5927	1.2650

4.2.4. Tyrimo scenarijų vertinimas naudojant RTS/CTS su 100 baitų riba

Lentelėje (16 lentelė.) pateikti stebimų tinklo parametrų rezultatai visais tyrimo scenarijais, kai RTS/CTS riba yra 100 baitų.

16 lentelė. Tyrimo rezultatai naudojant RTS/CTS su 100 baitų riba

Nr.	SC	T/SW	P_{col}	$RATE_{tx}$ /$RATE_{rx}$	PER_{tx} /PER_{rx}	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}
1.	SC1	0.1/30	0 %	328/313	85.42/99.80	160861	73864	3.9647	1.1486
2.	SC2	0.1/30	2.25%	235/204	75.00/99.48	81704	68309	2.0355	1.0968
3.	SC3	0.1/30	1.02%	279/251	83.58/99.66	121279	79419	3.0116	1.2384

4.2.5. Tyrimo scenarijų vertinimas naudojant RTS/CTS su 200 baitų riba

Lentelėje (17 lentelė.) pateikti stebimų tinklo parametrų rezultatai visais tyrimo scenarijais, kai RTS/CTS riba yra 200 baitų.

17 lentelė. Tyrimo rezultatai naudojant RTS/CTS su 200 baitų riba

Nr.	SC	T/SW	P_{col}	$RATE_{tx}$ /$RATE_{rx}$	PER_{tx} /PER_{rx}	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}
1.	SC1	0.1/30	0.41%	312/341	86.01/99.70	164728	79928	4.0923	1.2546

2.	SC2	0.1/30	2.26%	236/202	74.52/99.50	82164	66872	2.0464	1.0747
3.	SC3	0.1/30	1.51%	280/234	83.05/99.58	115227	74195	2.8552	1.1749

4.2.6. Tyrimo scenarijų vertinimas naudojant RTS/CTS su 400 baitų riba

Lentelėje (18 lentelė.) pateikti stebimų tinklo parametrų rezultatai visais tyrimo scenarijais, kai RTS/CTS riba yra 400 baitų.

18 lentelė. Tyrimo rezultatai naudojant RTS/CTS su 400 baitų riba

Nr.	SC	T/SW	P_{col}	$RATE_{tx}$ /$RATE_{rx}$	PER_{tx} /PER_{rx}	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}
1.	SC1	0.1/30	0%	316/344	85.60/99.74	161694	77490	4.0073	1.2268
2.	SC2	0.1/30	2.03%	233/129	71.88/99.44	83706	50832	2.1070	0.8172
3.	SC3	0.1/30	1.10%	275/280	84.71/99.59	123574	76567	3.0965	1.2309

4.2.7. Tyrimo scenarijų vertinimas naudojant RTS/CTS su 800 baitų riba

Lentelėje (19 lentelė.) pateikti stebimų tinklo parametrų rezultatai visais tyrimo scenarijais, kai RTS/CTS riba yra 800 baitų.

19 lentelė. Tyrimo rezultatai naudojant RTS/CTS su 800 baitų riba

Nr.	SC	T/SW	P_{col}	$RATE_{tx}$ /$RATE_{rx}$	PER_{tx} /PER_{rx}	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}
1.	SC1	0.1/30	0.39%	312/338	85.22/99.85	161097	77539	3.9991	1.2209
2.	SC2	0.1/30	2.44%	237/194	74.26/99.41	86260	70816	2.1423	1.0992
3.	SC3	0.1/30	1.13%	276/287	84.01/99.84	123319	75947	3.0905	1.2210

4.2.8. Tyrimo scenarijų vertinimas naudojant RTS/CTS su 1600 baitų riba

Lentelėje (20 lentelė.) pateikti stebimų tinklo parametrų rezultatai visais tyrimo scenarijais, kai RTS/CTS riba yra 1600 baitų.

20 lentelė. Tyrimo rezultatai naudojant RTS/CTS su 1600 baitų riba

Nr.	SC	T/SW	P_{col}	$RATE_{tx}$ /$RATE_{rx}$	PER_{tx} /PER_{rx}	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}
1.	SC1	0.1/30	0.04%	333/335	81.19/96.57	155931	60757	2.4949	0.9619
2.	SC2	0.1/30	11.0%	230/172	73.99/98.13	82706	64384	1.3234	1.0323
3.	SC3	0.1/30	5.66%	270/235	84.48/99.72	123406	81087	1.9746	1.2756

4.2.9. Tyrimo scenarijų vertinimas naudojant RTS/CTS su 2346 baitų riba

Lentelėje (21 lentelė.) pateikti stebimų tinklo parametrų rezultatai visais tyrimo scenarijais, kai RTS/CTS riba yra 2346 baitų. Tai yra maksimali galima *RT* ribos reikšmė.

21 lentelė. Tyrimo rezultatai naudojant RTS/CTS su 2346 baitų riba

Nr.	SC	T/SW	P_{col}	$RATE_{tx}$ $/RATE_{rx}$	PER_{tx} $/PER_{rx}$	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}
1.	SC1	0.1/30	0.05%	333/347	83.69/96.59	146947	81105	2.6339	1.303
2.	SC2	0.1/30	11.32%	220/198	75.73/99.37	91331	60888	1.4614	0.9775
3.	SC3	0.1/30	6.63%	271/285	84.13/99.80	124618	86541	1.9939	1.3586

4.2.10. Tyrimo scenarijų vertinimas naudojant sukurtą DKVM

Tyrimas atliekamas trimis scenarijais, naudojant skirtingus DKVM parametrus (22 lentelė.). Matavimai atliekami su periodu $T = 0.1\ Hz$, t. y. įrenginys perduoda statistikas į serverį kas 10s. Naudojamas slankaus lango dydis $SW = 30$ ir apsimokymo periodas $LP = 5$. Metodas vertinamas su $\boldsymbol{R_{col}} = 5$ proc. ir $\boldsymbol{R_{col}} = 3$ proc. kolizijos tolerancijos ribomis. Šios ribos parinktos dėl testavimo metu pelnytų reikšmingiausių rezultatų.

22 lentelė. DKVM tyrimo metu naudoti parametrai

Nr.	$T\ [Hz]$	SW	LP	R_{col}
1.	0.1	30	5	5 %
2.	0.1	30	5	3 %

4.2.10.1. DKVM vertinimas (SC1)

Pirmajame scenarijuje tinkle nėra paslėptų stočių, todėl DKVM prie skirtingų kolizijos tolerancijos ribos verčių elgiasi taip pat – RTS/CTS mechanizmas nėra įjungiamas. Lentelėje (23 lentelė.) pateikti stebėjimo rezultatai.

23 lentelė. Sukurto metodo tyrimo rezultatai (SC1)

Nr.	R_{col}	T/SW	P_{col}	$RATE_{tx}$ $/RATE_{rx}$	PER_{tx} $/PER_{rx}$	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}	RT
1.	3%	0.1/30	0.04%	345/343	91.82/99.91	165773	46455	2.6327	0.4134	Išjunkta
2.	5%	0.1/30	0.04%	340/346	91.81/99.90	168964	48764	2.7231	0.4537	Išjunkta

4.2.10.2. DKVM vertinimas (SC2)

Antrame scenarijuje, esant vienai paslėptai stočiai, naudojant $R_{col} = 3$ proc., įjungiamas RTS/CTS mechanizmas su aktyvacijos riba $RT = 741$. Kai $R_{col} = 5$ proc. RTS/CTS, mechanizmas taip pat įjungiamas su RT =1120. Lentelėje (24 lentelė.) pateikti stebėjimo rezultatai.

24 lentelė. Sukurto metodo tyrimo rezultatai (SC2)

Nr.	R_{col}	T/SW	P_{col}	$RATE_{tx}$ $/RATE_{rx}$	PER_{tx} $/PER_{rx}$	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}	RT
1.	3%	0.1/30	2.21%	240/134	73.05/99.37	84093	41461	1.9709	0.6456	741
2.	5%	0.1/30	4.90%	240/135	73.52/99.40	84996	43432	2.0321	0.7121	1120

4.2.10.3. DKVM vertinimas (SC3)

Trečiajame scenarijuje buvo keičiama vienos stoties padėtis. Esant $R_{col} = 3$ proc. įjungiamas RTS/CTS mechanizmas su aktyvacijos riba $RT = 780$. Kai $R_{col} = 5$ proc., RTS/CTS mechanizmas taip pat įjungiamas su RT =1072. Lentelėje (25 lentelė.) pateikti stebėjimo rezultatai.

25 lentelė. Sukurto metodo tyrimo rezultatai (SC3)

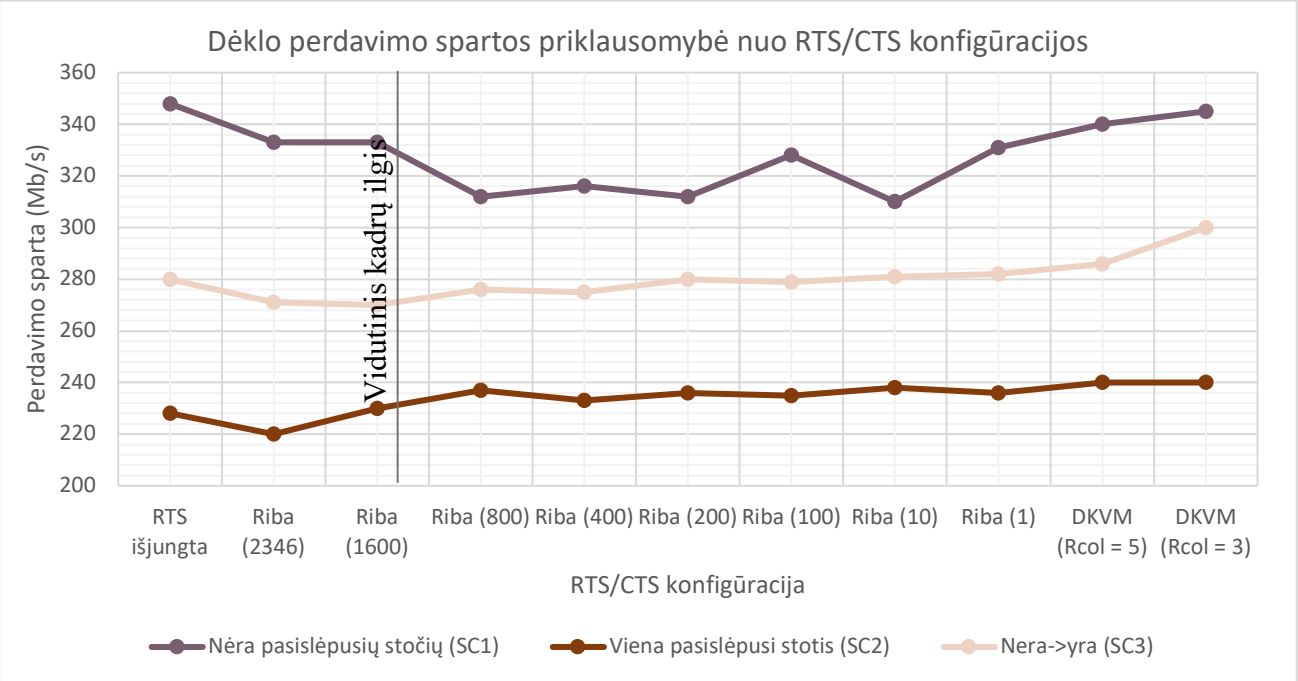
Nr.	R_{col}	T/SW	P_{col}	$RATE_{tx}$ /$RATE_{rx}$	PER_{tx} /PER_{rx}	N_{DATA}	N_{CTRL}	t_{air}	t_{ctrl}	RT
1.	3%	0.1/30	4.42%	300/245	86.93/98.69	117962	10596	1.9526	0.1004	780
2.	5%	0.1/30	1.20%	268/221	86.23/99.76	126440	12550	2.2499	0.1959	1072

4.3. Tyrimo rezultatų apibendrinimas

Atliktas tyrimas vertinant visų scenarijų tinklo būklę, naudojant įvairias kolizijų valdymo metodo konfigūracijas. Tyrimo rezultatai rodo, kad tinkle esanti paslėpta stotis sukelia fizinės spartos sumažėjimą. Naudojant RTS/CTS mechanizmą fizinė tinklo sparta yra pagerinama, tačiau išauga eterio laiko sąnaudos. Parodyta, kad sukurtas DKVM sumažina eterio laiko sąnaudas ir pagerina fizinę tinklo spartą, valdydamas RTS/CTS konfigūraciją pagal esamas tinklo sąlygas.

4.3.1. Paslėptos stoties įtaka WLAN tinklo kokybei

Rezultatai rodo, kad esant paslėptai stočiai WLAN tinkle, fizinė tinklo sparta sumažėja, lyginant su tinklo sparta, išmatuota, kai tinkle paslėptų stočių nėra. Remiantis dėklo perdavimo spartos priklausomybės nuo RTS/CTS konfigūracijos grafiku (26 pav.), galima teigti, kad paslėpta stotis sukelia žymų vidutinės fizinės spartos sumažėjimą.

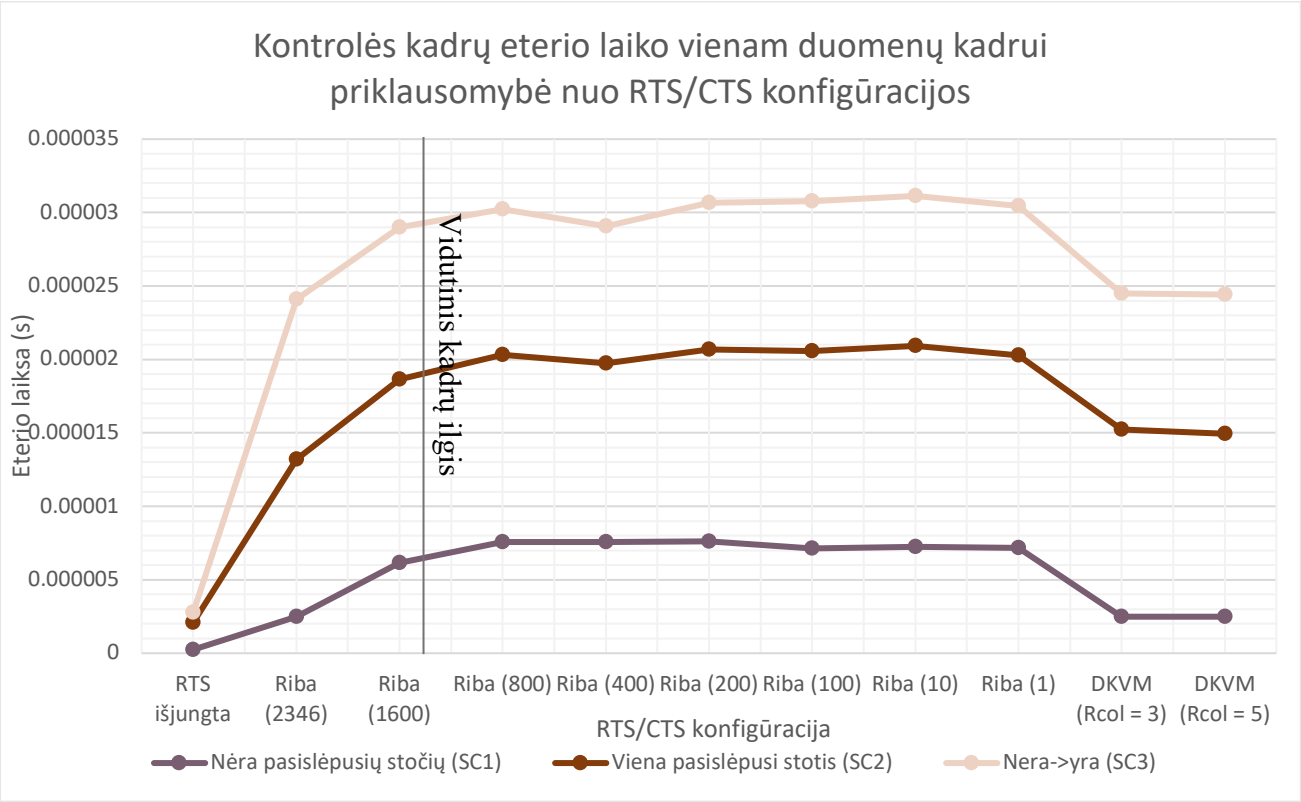


26 pav. Dėklo perdavimo spartos priklausomybė nuo RTS/CTS konfigūracijos grafikas

Paslėpta stotis antrajame scenarijuje sukėlė ~52 proc. spartos suprastėjimą, kuris siekė iki 120 Mb/s.

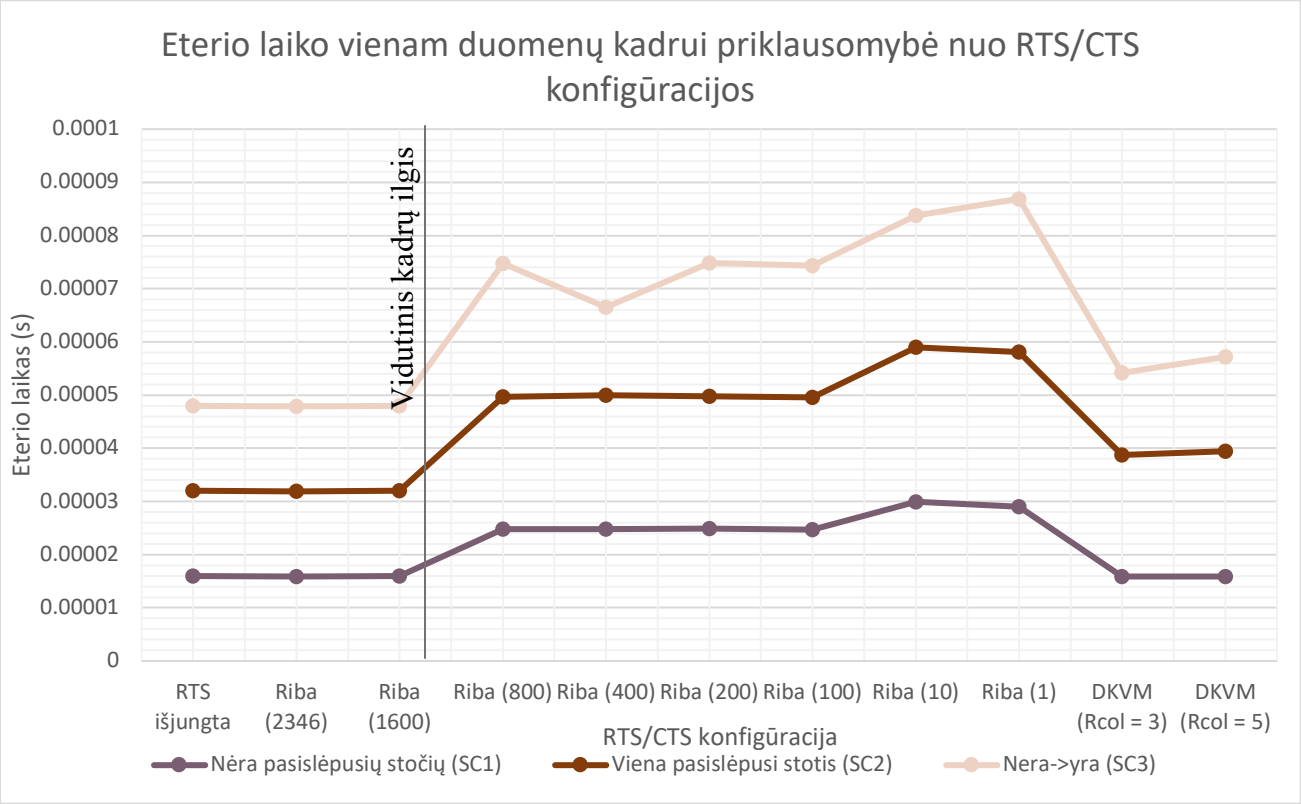
4.3.2. RTS/CTS mechanizmo įtaka tinklo kokybei

Tyrimo rezultatai patvirtina hipotezę, iškeltą analizės metu, kad RTS/CTS analizės mechanizmas sukelia papildomas eterio laiko sąnaudas. Eterio laiko vienam duomenų kadrui grafikas (27 pav.) rodo RTS/CTS mechanizmo aktyvacijos ribos dydžio priklausomybę nuo eteriu siunčiamų kontrolės kadrų užimamo eterio laiko. Esant išjungtam RTS/CTS mechanizmui, kontrolės kadrų užimamas eterio laikas yra mažiausias. Visą kontrolės kadrų užimtą eterio laiką nenaudojant RTS/CTS sudaro ACK kadrai. Įjungus mechanizmą, kontrolės kadrų užimamas eterio laikas išauga iki 24 kartų. Reikia įvertinti ir tai, kad esant paslėptai stočiai, padidėja kadrų kolizijų ir pakartotinių persiuntimų skaičius, todėl ištransliuotų kadrų skaičius išauga, didindamas kontrolės kadrų užimamą eterio laiką.



27 pav. Kontrolės kadrų eterio laiko vienam duomenų kadrui priklausomybės nuo konfigūracijos grafikas

Naudojant RTS/CTS kanalo rezervaciją, tyrimo scenarijuose naudojamas kanalo eterio laikas išauga iki 80 proc.. Taip pat naudojant RTS/CTS, duomenų perdavimo pralaidumas sumažėja (26 pav.) iki 17 proc., priklausomai nuo naudojamos aktyvacijos ribos.

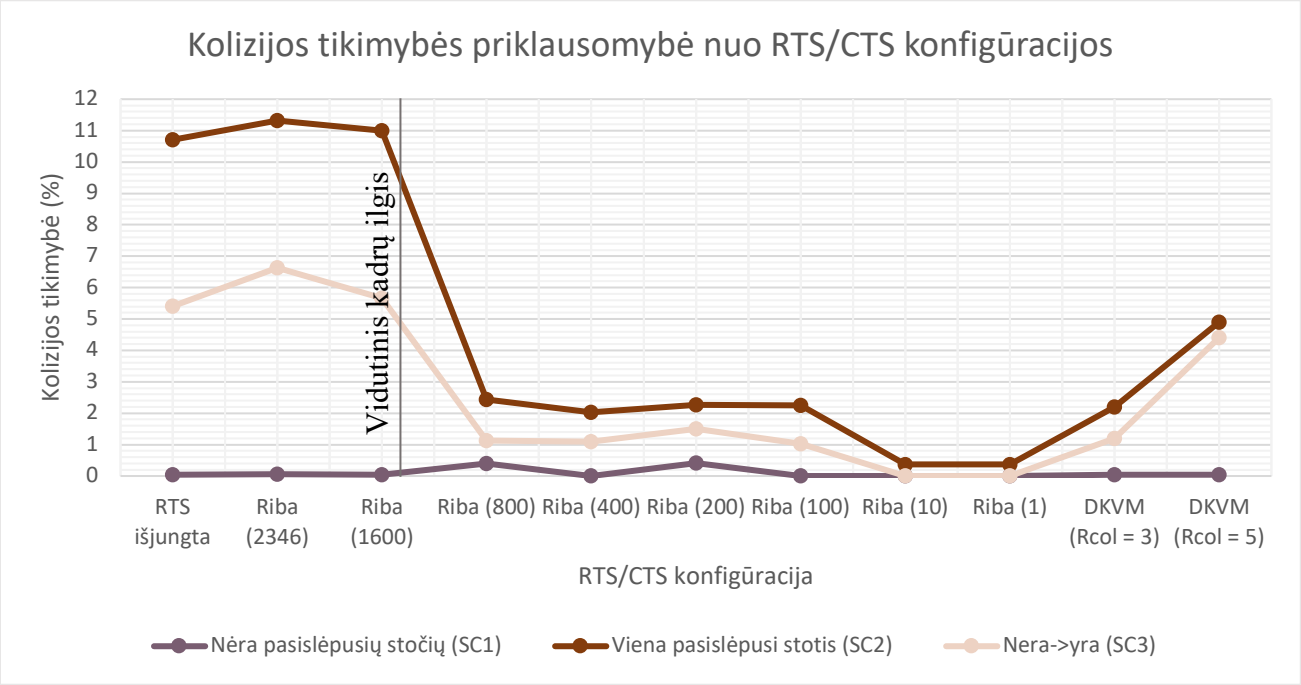


28 pav. Eterio laiko vienam duomenų kadrui priklausomybės nuo konfigūracijos grafikas

Skaičiuojant eterio laiką vienam duomenų kadrui (28 pav.) matomas padidėjimas, kai RTS/CTS aktyvacijos riba yra mažesnė už vidutinį perduodamų kadrų ilgį. Tyrimo rezultatai rodo RTS/CTS mechanizmo naudą fizinės spartos atžvilgiu, kai tinkle yra paslėpta stotis, tačiau taip pat matoma, kad sunaudojamas ilgesnis eterio laikas tokiam pat kiekiui duomenų perduoti.

4.3.3. Dinaminio kolizijų valdymo metodo nauda

Suprojektuotas metodas pirmajame scenarijuje (26 pav.) ($SC1$) neįjungia kolizijų valdymo mechanizmo, todėl pasiekia aukščiausia fizinę spartą. Tuo tarpu, statinės ribos konfigūracijos sukelia papildomas eterio laiko sąnaudas, taip sumažindamos fizinę tinklo spartą. Taip pat, scenarijuje $SC3$ keičiant stoties padėtį tinkle, DKVM išlaiko aukštesnę fizinę spartą istorinio duomenų lango ribose, parodant sukurto metodo pranašumą, esant tinklo dinamiškumui.



29 pav. Kolizijos tikimybės priklausomybės nuo RTS/CTS konfigūracijos grafikas

Kolizijos tikimybės priklausomybės nuo RTS/CTS konfigūracijos grafike (29 pav.) matoma, kad DKVM pasirenka efektyvią RTS/CTS aktyvacijos ribą, mažindamas vidutinę istorinio duomenų lango kolizijos tikimybę. Naudojant mažesnę kolizijos tolerancijos ribą, ($R_{col} = 5$ proc.) RTS/CTS mechanizmas yra įjungiamas anksčiau, taip išlaikydamas mažesnę vidutinę kolizijos tikimybę istoriniame duomenų lange.

4.4. Tyrimo išvados

Atliktas tyrimas, vertinantis analitinėje dalyje iškeltas prielaidas. Iš analizuotų šaltinių nustatyta, kad paslėpta stotis WLAN tinkle gali sumažinti tinklo spartą. Taip pat nustatyta, kad paslėptos stoties problemai spręsti naudojamas RTS/CTS mechanizmas padidina eterio laiko sąnaudas, perduodamas didesnį kontrolės kadrų kiekį. Suprojektuoto dinaminio kolizijų valdymo metodo tikslas – sumažinti kolizijos tikimybę esant paslėptai stočiai tinkle, efektyviai išnaudojant eterio laiką.

Atlikto tyrimo metu parodyta, kad paslėpta stotis WLAN tinkle sumažina fizinės tinklo spartos vidurkį iki 52 proc.. Esamas kolizijų valdymo mechanizmas sumažina prarandamą tinklo spartą, tačiau du kartus padidina eterio laiko sąnaudas vienam duomenų kadrui.

Tyrimo rezultatai parodė, kad sukurtas dinaminis kolizijų valdymo metodas pasirenka RTS/CTS aktyvacijos ribą, efektyviai išnaudojant eterio laiką ir sumažinant kolizijų tikimybę. Tirtas sukurtas metodas su skirtingomis kolizijos tolerancijos reikšmėmis tinkle išlaikė aukščiausią tyrimo metu išmatuotą spartą. Tiriant DKVM efektyvumą esant paslėptam įrenginiui nustatyta, kad metodas pasirenka RTS/CTS aktyvacijos ribą, maksimaliai sumažinančią kolizijos tikimybę, sunaudojant minimalias eterio laiko sąnaudas. Tiriant sukurto metodo reagavimą į tinklo sąlygų pasikeitimą nustatyta, kad metodas, pritaikydamas prieigos taško konfigūraciją, padidina tinklo spartą istorinio duomenų lango ribose.

5. Išvados

Analitinėje dalyje atlikta kolizijų belaidžiuose paskirstytosios koordinuotosios funkcijos tinkluose analizė, kurios metu apibrėžtos kolizijų atsiradimo priežastys. Analizuojant esamus protokolo kolizijų valdymo mechanizmus nustatyta, kad naudojamas mechanizmas sumažina tinkle įvykstančių kolizijų kiekį, tačiau drastiškai padidina eterio laiko sąnaudas. Protokolas apibrėžia statinę valdymo ribą mechanizmo aktyvacijai, tačiau esant unikalioms tinklo sąlygoms kiekviename BSS tinkle, riba nėra efektyvi. Iškeltas baigiamojo darbo tikslas – suprojektuoti, realizuoti ir ištirti metodą, valdantį kanalo prieigos mechanizmą ir nustatantį optimalią RT ribą užtikrinančią maksimalią tinklo spartą ir efektyvų eterio laiko išnaudojimą.

Projekto metu sukurtas dinaminis kolizijų valdymo metodas, kuris stebi belaidžio tinklo būklę iš prieigos taško perspektyvos ir keičia kanalo mechanizmo būseną bei aktyvacijos ribą. Suprojektuoto metodo architektūrą sudaro belaidžio tinklo prieigos taškas, kuris registruoja tinklo įrenginių ir asocijuotų stočių veiklos statistinius duomenis ir dedikuotas serveris, kuris analizuoja sukauptas statistikas ir parenka efektyviausią kolizijų valdymo mechanizmo konfigūraciją.

Prototipo dalyje aprašyta suprojektuoto kolizijų valdymo metodo prototipo realizacija. Realizuotas statistikų surinkimo agentas prieigos taške, kuris periodiškai surenka veiklos statistikas ir perduoda jas į serverį. Serverio pusėje realizuotas kolektorius, analizės ir sprendimo priėmimo moduliai, kurių pagalba pagal priimtas tinklo įrangos statistikas sugeneruojama kolizijų valdymo metodo konfigūracija. Realizuoti visi suprojektuoti komponentai, išpildant pilną dinaminio kolizijų valdymo metodo funkcionalumą.

Atlikto tyrimo metu buvo vertinama esamo kolizijų valdymo metodo įtaka tinklo spartai ir kanalo laiko sunaudojimui. Tyrimo rezultatai parodė, kad sukurtas dinaminis kolizijų valdymo metodas parenka prieigos taško konfigūraciją efektyviai išnaudojant eterio laiką ir sumažinant kolizijų tikimybę. Tirtas sukurtas metodas su skirtingomis kolizijos tolerancijos reikšmėmis tinkle išlaikė aukščiausią tyrimo metu išmatuotą spartą. Tiriant dinaminio kolizijų metodo efektyvumą esant paslėptai stočiai, metodas pasirenka RTS/CTS aktyvacijos ribą, maksimaliai sumažinančią kolizijos tikimybę, sunaudojant minimalias eterio laiko sąnaudas. Tiriant sukurto metodo reagavimą į tinklo sąlygų pasikeitimą nustatyta, kad metodas pritaikydamas prieigos taško konfigūraciją, padidina tinklo spartą istorinio duomenų lango ribose.

Esant didelei kanalo apkrovai belaidžiuose tinkluose susidaro situacijos, kai įvykstančios kolizijos sukelia didesnį informacijos srauto sulėtėjimą nei esamos kanalo rezervacijos mechanizmo pristatomos eterio laiko sąnaudos. Įdiegus sukurtą dinaminį kolizijų valdymo metodą galima aptikti tokius scenarijus ir parinkti tinkama konfigūraciją kiekvienai unikaliai tinklo topologijai. Šis metodas padeda maksimaliai išnaudoti eterio talpą išvengiant nereikalingo duomenų srauto sulėtėjimo ir sumažinant kolizijų tikimybę.

6. Šaltiniai

[1] I. S. I. U. o. S. California, „RFC: 793 TRANSMISSION CONTROL PROTOCOL,“ Marina del Rey, California, 1981.

[2] I. S. f. I. technology, „Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) SpecificationsSponsored by the LAN/MAN Standards CommitteeIEEE 3 Park Avenue New York, NY 10016-5997 USAIEEE Computer SocietyIEEE Std 802.11™-2016(Revision of IEEE Std 802.11-2012,“ IEEE, New York, 2016.

[3] erahia, Eldad ir Stacey, R.., „Next generation wireless LANs: 802.11n and 802.11ac.,“ 2013.

[4] P. CHATZIMISIOS, A. C. BOUCOUVALAS ir V. VITSAS, „Effectiveness of RTS/CTS handshake in IEEE 802.11a WLANs,“ Departament of Information Technology, Educational Institution, Thessaloniki, Thessaloniki, Greece, 2004.

[5] M. KRISHAN ir A. ZAKHOR, Throughput Improvement in 802.11 WLANs Using Collision Probability Estimates in Link Adaptation, 2019.

[6] H. WU, S. CHENG, Y. PENG, K. LONG ir J. MA, „IEEE 802.11 Distributed Coordinated Function (DCF): Analysis and Enhancement,“ National Key Lab of Switching Technology and telecommunication Networks, Bejing, 2002.

[7] I. S. f. I. technology, „Part 3: Carrier sense multiple access withcollision detection (CSMA/CD) access method and physical layer specifications,“ IEEE, New York, 2002.

[8] C. Hu, H. Kim ir J. C. Hou, „An Analysis of the Binary Exponential Backoff Algorithm in Distributed MAC Protocols,“ University of Illinois at Urbana-Champaign.

[9] A. RAHMAN ir P. GBURZYNSKI, „Hidden Problems with the Hidden Node Problem,“ IEEE, Alberta, Canada, 2006.

[10] M. N. KRISHNAN, S. POLLIN ir A. ZAKHOR, „Local Estimation of Probabilities of Direct and Staggered Collision in 802.11 WLANs,“ Department of EECS,U.C.Berkeley, 2009.

[11] P. VEERARAGHAVAN, G. KHOMAMI ir F. P. FONTAN, „The Relation Between the Probability of Collision-Free Broadcast Transmission in Wireless Network and the Stirling Number of Second Kind,“ Departament of computer Science and Information Technology, La Trobe Univercity of Vigo, Australia, 2018.

[12] T. SUGIMOTO, N. KOMURO , H. SEKIYA ir S. SAKATA, „Maximum Throughput Analysis for RTS/CTS-used IEEE 802.11 DCF in Wireless Multi-hop Networks,“ 2010.

[13] P. KYASANUR ir N. H. VAIDYA, „Selfish MAC Layer Misbehavior in Wireless Networks,“ International Conference on Dependable Systems and Networks, 2004.

[14] R. KAUR ir R. SHARMA, „An Analysis of RTS/CTS Mechanism for data Transfer In Wireless Network: A Reviev,“ Deptt. Of Electronics and Communication Engineeringm Punjiabi University, Punjab, India, 2016.

[15] Y. EDALAT, K. OBRACZKA ir B. AMIRI, „A Machine Learning Approach for Dynamic Control of RTS/CTS in WLANs,“ įtraukta *EAI International Conference on Mobile and Ubiquitious Systems: Computing, Networking and Services*, New York, 2018.

Priedai

1 priedas. Prieigos taško statistikos registravimo agento kodo fragmentai

agent.c:

```
+ #define GATHER_STATS        SCRIPT_LOCATION"gather.lua"
+ #define SUBTOPIC_STATS      "stats"
+       ...
+ static int agent_run(void){
+       struct timespec current_time;
+       int cur_sec_period = 0;
+            if (current_time.tv_sec - cur_sec_period) > 10) {
+                  cur_sec_leo = current_time.tv_sec;
+                  publish(SUBTOPIC_STATS, GATHER_SCRIPT);
+                  if (state.secure)
+                        publish_secure ();
+             }
+ }
```

stats.c:

```
+ #define STATS_FILE            "/tmp/stats.json"
+ struct ieee80211_nodestats *ns = &stats.is_stats;
+ json_t *node = json_array();
+ char filename[256];
+       ...
+ gather_node(){
+       float txPERk, rxPERk;
+         txPERk = ((float)ns->ns_tx_data_success / (ns->ns_tx_data_success + ns->ns_tx_retry))*100;
+       rxPERk = ((float)ns->ns_rx_data / (ns->ns_rx_data + (ns->ns_rx_decap +
+                                 ns->ns_rx_decryptcrc + ns->ns_rx_tkipicv + ns->ns_rx_wepfail))) *100;
+      json_object_set(stats, "txBytes", json_integer(txPERk))
+      json_object_set(stats, "txBytes", json_integer(rxPERk))
+      json_object_set(stats, "txBytes", json_integer(ns->ns_rx_data))
+      json_object_set(stats, "txBytes", json_integer(ns->ns_rx_decap))
+      json_object_set(stats, "txBytes", json_integer(ns-> ns->ns_rx_tkipicv))
+      json_object_set(stats, "txBytes", json_integer(ns->ns_rx_decryptcrc))
+      json_object_set(stats, "txBytes", json_integer(ns->ns_rx_wepfail))
+       ...
+      snprintf(filename, sizeof(filename), STATS_FILE);
+      json_t *vaps = json_load_file(filename, 0, &error);
+ }
```

gather.lua:

```
+       ...
+ local ATHTOOL_DATA   = "/tmp/athtool"
+ local STATS_FILE           = "/tmp/stats.json"
+ local output = {}
+       ...
+ local function read_stats()
+       local device = {}
+       local f, err = io.open(ATHTOOL_DATA)
+       if err then
+            print(string.format("Failed to open a file: %s", ATHTOOL_DATA))
+            return -1
+       end
+       while true do
+            local line = f:read()
+
+            if line == nil then
+                 break
+            end
+            if string.find(line, "Rx PHY errors") then
+                 device.rx_phy_err           = get_number(line)
+            elseif string.find(line, "Rx CRC errors") then
+                 device.rx_crc_err           = get_number(line)
+            elseif string.find(line, "excess retries") then
+                 device.tx_ex_ret                 = get_number(line)
+            -- Radio statistics (FRAME as PDU)
+            -- Total frames per radio:
+            elseif string.find(line, "ast_rx_packets") then
+                 device.rx_total_frame       = get_number(line)
+            elseif string.find(line, "ast_tx_packets") then
+                 device.tx_total_frame       = get_number(line)
+            elseif string.find(line, "Tx Data Packets  ") then
+                 device.tx_data_frame        = get_number(line)
+            elseif string.find(line, "Rx Data Packets  ") then
+                 device.rx_data_frame        = get_number(line)
+            elseif string.find(line, "Rx Mgmt Frames") then
+                 device.rx_mgmt_frame             = get_number(line)
+            elseif string.find(line, "ast_tx_mgmt") then
+                device.tx_mgmt_frame        = get_number(line)
+            ...
+       device.time = os.time()
+       output.device = device
+
+       return 0
+       ...
+ -- Calculate changes per period
+  local function calculate_delta()
+       -- client found in historical data
+       if not cli.rxBytesd then
+            cli.rxBytesd = 0
+            cli.txBytesd = 0
+       end
+       if old_peers then
+            for _, oldcl in pairs(old_peers) do
+                 if cli.mac == oldcl.mac then
+                      cli.rxBytesd = cli.rxBytes - oldcl.rxBytes
+                      cli.txBytesd = cli.txBytes - oldcl.txBytes
+                     cli.txFramed = cli.txPackets - oldcl.txPackets
+                     cli.txmgmtd = cli.txmgmt - oldcl.txmgmt
+                     ...
+                     cli.txPacketsSuccessd = cli.txPacketsSuccess - oldcl.txPacketsSuccess
+                     cli.rxdecrypterrd = cli.rxdecrypterr - oldcl.rxdecrypterr
```

```
+                           cli.rxerrorsd = cli.rxerrors - oldcl.rxerrors
+                       end
+                   end
+           end
+  end
+
+function _M.gather_radio()
+           -- get node stats
+           init()
+
+           -- calculate deltas
+           calculate_delta()
+
+           -- dump all stats to json
+           dump (output)
+
+           -- write json table to a json file
+           utils.write_json(output, STATS_FILE)
+  end
+
+ return _M
```

2 priedas. Serveryje realizuoto sukurto metodo kodo fragmentai

app.py:

```
+ #!/usr/bin/env python
+ import sys
+ import signal
+ import collector as mqtt
+ import decision
+ import pandas as pd
+ import database as db
+ import processing as proc
+ import tracker as trc
+
+ def signal_handler(sig, frame):
+     print('You pressed Ctrl+C!')
+     client.loop_stop() #stop the loop
+     sys.exit(0)
+ # 1 day - 720 datapoints
+ limit = 50 #2 days
+ signal.signal(signal.SIGINT, signal_handler)
+ # initiate configuration tracker class
+ trc.Node()
+ mqttc = mqtt.MyMQTTClass()
+ rc = mqttc.run()
```

decision.py:

```
+ import json
+ import context
+ import paho.mqtt.publish as publish
+ import log
+ import tracker as trc
+ import pandas as pd
+ import ssl
+ import method as mt
+ import paho.mqtt.client as mqtt
+
```

```
+ CA = "./crt/ca.crt"
+ IP = ""
+ PORT = 8883
+ TIMEOUT = 60
+ STATS = "node/3553043635/3620611251/XXXXXXXXX/stats"
+ CONFIG = "node/3553043635/3620611251/XXXXXXXXX/config"
+
+ def decision(df):
+     MetDf = df.get_all_peer()
+     SW = df.get_SW()
+
+     if len(MetDf) < df.get_LP():
+         if (len(MetDf) == 1) and (not df.get_first()):
+             apply_startup(df, True, 800)
+             df.set_first()
+         log.error("STATE:Learning period (%d/%d)"%(len(MetDf),df.get_LP()))
+     else:
+         config = df.get_current_config()
+         NPCNT5g = df.get_NPCNT()
+         IDRATE5g = df.get_IDRATE()
+         THR5g = int(config["g5thr"])
+         change5g = False
+         tracker = trc.Node()
+         if float(MetDf["CollProb5gCurr"].mean())*100 > df.get_CollTollThr5g():
+             log.debug("Collision tolerance exceeded 5g")
+             # Now 5g RTS/CTS is off
+             if (int(config.config) == 0) or (int(config.config) == 1):
+                 # Were the changes made recently?
+                 # No changes made recently
+                 #print(df.get_ChTimer5g())
+                 if df.get_ChTimer5g() == 0:
+                     log.error("STATE: Configurating (%d)"%(df.get_ChTimer5g()))
+                     change5g = True
+                     df.start_ChTimer5g()
+                     # For first enabling turn thr to avg pck size
+                     THR5g = int(MetDf["AvgFrameSize"].mean())  # need to get packet size here
+                     log.error("Setting THR = %d"%(int(MetDf["AvgFrameSize"].mean())))
+                 # changes were made need to w8
+                 else:
+                     log.error("STATE: Learning (%d/%d)"%(df.get_ChTimer5g(), df.get_LP()))
+                     #decrease counter
+                     df.dec_ChTimer5g()
+             # Now 5g RTS/CTS is on
+             else:
+                 # Were the changes made recently?
+                 # No changes made recently
+                 if df.get_ChTimer5g() == 0:
+                     log.error("STATE: Threshold tunning (%d)"%(df.get_ChTimer5g()))
+                     df.start_ChTimer5g()
+                     change5g = True
+                     THR5g = THR5g - IDRATE5g
+                     if THR5g < 0:
+                         THR5g = 1
+                     IDRATE5g = check_idrate(IDRATE5g)
+                     df.set_IDRATE(IDRATE5g)
+                 # changes were made need to w8
+                 else:
+                     log.error("STATE: Learning (%d/%d)"%(df.get_ChTimer5g(), df.get_LP()))
+                     df.dec_ChTimer5g()
+         #cihange5g = False
+         if change5g == False:
+             NPCNT5g = NPCNT5g + 1
```

```
+        if change5g == True:
+            NPCNT5g = 0
+            new = {'datetime': pd.Timestamp.now(), 'config': 0, 'g2thr': 0, 'g5thr': 0}
+            # so i neet to change 5g config but leave 2g as it is
+            CurrConf = config.config
+            new["g5thr"] = THR5g
+            new["config"] = 2 # only 5g
+            publish(on5g(THR5g))
+            df.update_current_config(new)
+            log.error(new)
+        if (MetDf["Hidden5g"].iloc[0] == False):
+            if ((int(config.config) == 2) or (int(config.config) == 3)) and (MetDf["CollProb5gCurr"].mean()*100 < 1):
+                log.error("Turning off RTS/CTS")
+                new = {'datetime': pd.Timestamp.now(), 'config': 0, 'g2thr': 0, 'g5thr': 0}
+                df.update_current_config(new)
+                publish(alloff())
+        df.set_NPCNT(NPCNT5g)
+        if NPCNT5g > SW:
+            RTS/CTS 5 g is on
+            if (int(config.config) == 3) or (int(config.config) == 2):
+                THR2g = check_thr(THR2g + IDRATE2g)
+        trc.show_config(config)
+
+ # treshold boudary check
+ def check_thr(thr):
+     if thr < 1:
+         thr = 1
+     elif thr > 2346:
+         thr = 2346
+     return thr
+
+ # treshold boudary check
+ def check_idrate(idrate):
+     if idrate < 1:
+         idrate = 1
+     elif idrate > 300:
+         idrate = 300
+     return idrate
+
+ def alloff():
+     with open("files/alloff.json", 'r') as f:
+         data = json.load(f)
+     #log.error("Turning OFF RTS/CTS for 5g and 2g radio")
+     return json.dumps(data, separators=(',', ':'))
+
+ def on5g(g5):
+     with open("files/g5on.json", 'r') as f:
+         data = json.load(f)
+     if data["config"]["wireless"]["radio"][1]['ifname'] == 'wifi1':
+         data["config"]["wireless"]["radio"][1]['rts']['size'] = g5
+         #print(data["config"]["wireless"]["radio"][1]['rts'])
+         #print(json.dumps(data, separators=(',', ':')))
+     if data["config"]["wireless"]["radio"][0]['ifname'] == 'wifi1':
+         data["config"]["wireless"]["radio"][0]['rts']['size'] = g5
+         #print(data["config"]["wireless"]["radio"][0]['rts'])
+         #print(json.dumps(data, separators=(',', ':')))
+     #print("Turning OFF RTS/CTS for 2g radio")
+     return json.dumps(data, separators=(',', ':'))
+
+ def on2g(g2):
+     with open("files/g2on.json", 'r') as f:
+         data = json.load(f)
```

```
+
+    #print("Turning OFF RTS/CTS for 5g radio")
+    return json.dumps(data, separators=(',', ':'))
+
+ def allon(g2, g5):
+    with open("files/allon.json", 'r') as f:
+        data = json.load(f)
+
+    #print(data["config"]["wireless"]["radio"][1]['rts'])
+    #print("Turning ON RTS/CTS for 5g and 2g radio")
+    return json.dumps(data, separators=(',', ':'))
+
+ def publish(data):
+        mqttc = mqtt.Client()
+        mqttc.tls_set(ca_certs=CA,
+                      certfile=None,
+                      keyfile=None,
+                      cert_reqs=ssl.CERT_REQUIRED,
+                      tls_version=ssl.PROTOCOL_TLSv1_2)
+        mqttc.username_pw_set("master")
+        mqttc.connect("XXXXXXXX", 8883, 60)
+        mqttc.publish(CONFIG, data, qos=2)
+
+ def apply_startup(clas, STARTOFF, THR):
+    if STARTOFF:
+        log.error("STARTUP: Turning RTS/CTS off")
+        # turn all RTS/CTS off for learning period
+        publish(alloff())
+        # update config ....
+        new = {'datetime': pd.Timestamp.now(), 'config': 0, '2gthr': 0, 'g5thr': 0}
+        clas.update_current_config(new)
+    else:
+        log.error("STARTUP: Turning RTS/CTS on with %d"%(THR))
+        publish(on5g(THR))
+        new = {'datetime': pd.Timestamp.now(), 'config': 2, '2gthr': 0, 'g5thr': THR}
+        clas.update_current_config(new)
```

colector.py:

```
+ import context  # Ensures paho is in PYTHONPATH
+ import ssl
+ import paho.mqtt.client as mqtt
+ import log
+ import decision
+ import analysis
+ import tracker as trc
+ import time
+ import pandas as pd
+
+ config = trc.Node()
+
+ # if true starts RTS/CTS offf
+ # else starts on with THR
+ STARTOFF = False
+ THR = 1
+
+ CA = "./crt/ca.crt"
+ IP = "XXXXXXXX"
+ PORT = 8883
+ TIMEOUT = 60
+ STATS = "node/3553043635/3620611251/00193b14315b/stats"
+ CFG = "node/3553043635/3620611251/00193b14315b/config"
+
```

```
+ class MyMQTTClass(mqtt.Client):
+
+    def on_connect(self, mqttc, obj, flags, rc):
+        log.debug("rc: "+str(rc))
+
+    def run_publish(self, data):
+        log.error("Sending new configuration")
+        self.publish(CFG, data, qos=2)
+
+    def on_message(self, mqttc, obj, msg):
+        log.info("[MESSAGE] received ")
+        self.cnt = self.cnt + 1
+        analysis.message(msg)
+
+    def on_publish(self, mqttc, obj, mid):
+        log.debug("mid: "+str(mid))
+
+    def on_subscribe(self, mqttc, obj, mid, granted_qos):
+        log.debug("Subscribed: "+str(mid)+" "+str(granted_qos))
+
+    def on_log(self, mqttc, obj, level, string):
+        log.debug(string)
+
+    def run(self):
+        self.tls_set(ca_certs=CA,
+                     certfile=None,
+                     keyfile=None,
+                     cert_reqs=ssl.CERT_REQUIRED,
+                     tls_version=ssl.PROTOCOL_TLSv1_2)
+
+        self.username_pw_set("master")
+        self.connect("XXXXXXXXXX", 8883, 60)
+        self.subscribe("node/3553043635/3620611251/00193b14315b/stats", 0)
+        self.cnt = 0
+
+        rc = 0
+        while rc == 0:
+            rc = self.loop()
+        return rc
```

analyais.py:

```
+ import database as db
+ import pandas as pd
+ import processing as proc
+ import log
+ import numpy as np
+ import json
+ import time
+ import action
+ import tracker as trc
+
+ config = trc.Node()
+
+ def fix_date(df):
+    #unixepoch to datetime
+    df['devtime'] = pd.to_datetime(df['devtime'],unit='s')
+    #df = df.set_index('devtime')
+    return df
+
+ def message(msg):
+
+    cnf = config.get_current_config()
+    #trc.show_config(cnf)
```

```
+
+    # parse message
+    data = json.loads(msg.payload)
+
+    # 2g to DF
+    g2df = proc.proc_ath0(data["wireless"]["radio_2g"])
+    #print(g2df.columns)
+    # 5g to DF
+    #log.error(data["wireless"]["radio_5g"])
+    g5df = proc.proc_ath1(data["wireless"]["radio_5g"])
+    #print(g5df.columns)
+
+    devtime = data["devTime"]
+    # Peers to DF
+    if "peers" in data:
+        peerlist = list(data["peers"])
+        peerdf = pd.DataFrame(peerlist)
+        peers = proc.proc_peers(peerdf)
+        chan_reserv_latency(peers, g2df, g5df)
+    else:
+        log.error("No peers")
+
+    #config.shwo_stats_all()
+
+    # sprendimas
+    action.decision(config)
+ # @return
+ # CollProbability - Colision probability with current setting per cli
+ # CollCost - Colision Cost with current setting per cli
+ def CollisionCostCurr(row, DIFS, concur):
+    SIFS = 16
+    Si = DIFS + 288 # RTStt(608) + SIFS(16) + CTStt(93)
+
+    AirtimeRx = 0
+    AirtimeTx = 0
+    ProbTxOverHead = 0
+
+    TxOverHead = 0
+    TxDatapckSize = 0
+    RxDatapckSize = 0
+
+    Hidden5g = False
+
+    if (row["txFramed"] is None) or (row["rxdatapckd"] is None):
+        return 0, 0, 0, 0, 0, 0, 0, 0, 0
+
+    cnf = config.get_current_config()
+
+    ret = { 'RxCollProb': 0, 'RxCollProb': 0,
+            'RxCollCost': 0, 'RxCollCost': 0,
+            'rxPER': 0,      'txPER': 0
+          }
+
+    # TX Collision probability and cost
+    # No TX traffic = No cost
+   # print("Frames TX  = %d"%(int(row["txFramed"])))
+    if int(row["txFramed"]) > 0:
+
+        ret["txPER"] = row['txPER']
+        TxDatapckSize = division(float(row["txBytesd"]), float(row["txFramed"]))     # [b]
+        TxTravelTime = division(TxDatapckSize, float(row["txrate"]))              # [µs]
+        ProbTxTravelTime = 0                                    # [µs]
```

```
+        # different thresholds for different clients
+        if "5g" in row["band"]:
+                Hidden5g = isHidden(float(row['txPER']), float(row['rxPER']), float(row['txeval']), float(row['rxeval']),
int(row['signal0']))
+            # RTS/CTS is enabled
+            if (int(cnf.config) == 2) or (int(cnf.config) == 3):
+                thr = cnf.g5thr
+                # Vidutiis paketo dydis yra didesnis uz RIBA
+                # RTS/CTS nenaudojami
+                if TxDatapckSize < int(thr):
+                    #log.info("Tx Packet size is below threshold 5g")
+                    log.error("Nenaudoju rts %d"%(TxDatapckSize))
+                    TxOverHead = float(row["txFramed"]) * SIFS                  # [µs]
+                    ProbTxOverHead = float(row["txFramed"]) * SIFS              # [µs]
+                    ProbTxTravelTime = TxDatapckSize / float(row["txrate"])     # [µs]
+                # Vidutiis paketo dydis yra mazesnis uz RIBA
+                # RTS/CTS naudojami
+                else:
+                    log.error("Naudoju rts %d "%(TxDatapckSize))
+                    TxOverHead = (0.3 * float(row["txFramed"]) * Si) + (0.7 * float(row["txFramed"]) * SIFS)     # [µs]
+                    ProbTxOverHead = 0                                  # [µs]
+                    ProbTxTravelTime = 0                                # [µs]
+            # RTS/CTS is disabled
+            elif (int(cnf.config) == 0) or (int(cnf.config) == 1):
+                ProbTxOverHead = float(row["txFramed"]) * SIFS              # [µs]
+                ProbTxTravelTime = TxDatapckSize / float(row["txrate"])     # [µs]
+                TxOverHead = float(row["txFramed"]) * SIFS
+
+        AirtimeTx = TxTravelTime + TxOverHead                   # [µs]
+        #print("Airtime TX = %d"%(AirtimeTx))
+        ProbAirtimeTx = ProbTxTravelTime + ProbTxOverHead       # [µs]
+        AirSlotsTx = division(ProbAirtimeTx, 9)
+        ret["TxCollProb"] = division(AirSlotsTx, division(HEARTBEAT, 9)*1000000)
+        ret["TxCollCost"] = ProbAirtimeTx * ret["TxCollProb"]
+
+    if int(row["rxdatapckd"]) > 0:
+        ret["rxPER"] = row['rxPER']
+        #ret["txPER"] = row['txPER']
+        RxDatapckSize = division(int(row["rxBytesd"]), int(row["rxdatapckd"]))   # [b]
+        #log.error("RX Data packet size %0.2f"%(RxDatapckSize))
+        RxTravelTime = division(RxDatapckSize, int(row["rxrate"]))      # [µs]
+        toh = 0
+        ProbRxTravelTime = 0                                    # [µs]
+        RxOverHead = 0
+        ProbRxOverHead = 0
+
+        # different thresholds for different clients
+        if "5g" in row["band"]:
+            if int(row['rxPER']) < 70:
+                Hidden5g = True
+            # RTS/CTS is enabled
+            if (int(cnf.config) == 2) or (int(cnf.config) == 3):
+                thr = cnf.g5thr
+                # Vidutiis paketo dydis yra mazesnis uz RIBA
+                # RTS/CTS nenaudojami
+                if RxDatapckSize < int(thr):
+                    toh = overhead(int(row["rxdatapckd"]), int(row["rxrate"]), False)
+                    log.error("Nenaudoju rts %d"%(RxDatapckSize))
+                    #log.info("Rx Packet size is below threshold 5g")
+                    RxOverHead = int(row["rxdatapckd"]) * SIFS
+                    ProbRxOverHead = int(row["rxdatapckd"]) * SIFS              # [µs]
+                    ProbRxTravelTime = division(RxDatapckSize, int(row["rxrate"]))     # [µs]
```

```
+                else:
+                    log.error("Naudoju rts %d"%(RxDatapckSize))
+                    ProbRxOverHead = 0                                  # [µs]
+                    ProbRxTravelTime = 0                                # [µs]
+                    RxOverHead = (0.3 * float(row["rxdatapckd"]) * Si) + (0.7 * float(row["rxdatapckd"]) * SIFS)   # [µs]
+            # RTS/CTS is disabled
+            elif (int(cnf.config) == 0) or (int(cnf.config) == 1):
+                RxOverHead = int(row["rxdatapckd"]) * SIFS
+                ProbRxOverHead = int(row["rxdatapckd"]) * SIFS                  # [µs]
+                ProbRxTravelTime = division(RxDatapckSize, int(row["rxrate"]))    # [µs]
+        AirtimeRx = RxTravelTime + RxOverHead                       # [µs]
+        #print("Airtime RX = %d"%(AirtimeRx))
+        ProbAirtimeRx = ProbRxTravelTime + ProbRxOverHead           # [µs]
+        AirSlotsRx = division(ProbAirtimeRx, 9)
+        ret["RxCollProb"] = division(AirSlotsRx, division(HEARTBEAT,9)*1000000)
+        ret["RxCollCost"] = ProbAirtimeRx * ret["RxCollProb"]
+    Cost = ret["RxCollCost"] + ret["RxCollCost"]
+    Prob = ret["RxCollProb"] + (ret["RxCollProb"] * division(int(row["rxdatapckd"]), int(row["txFramed"])))
+
+ # Kanalo rezervacijos sukeltas velinimas priklausonuo
+ def chan_reserv_latency(df, ath0, ath1):
+
+    DIFS = 0
+
+    NConc = 0
+
+    Datapck5g = 0
+    DataBytes5g = 0
+
+    CollP5g = []
+    CollC5g = []
+    CollProbCurr5g = 0
+    CollCostCurr5g = 0
+    CollProbWithout5g = 0
+    CollCostWithout5g = 0
+
+    TxPck = 0
+    RxPck = 0
+    TxErr = 0
+    RxErr = 0
+    Airtime = 0
+
+    TxPER5g = 0
+    RxPER5g = 0
+    AvgFrameSize = 0
+    TxRate5g = 0
+    RxRate5g = 0
+    Hidden5g = False
+
+    Cli5g = 0
+
+    # count peer side errors
+    for index, row in df.iterrows():
+        row.fillna(0)
+            #log.info(row["mac"] + "(" + row["band"]  + ") - RX %0.1f("%(row["rxrate"])+ row["rxeval"] + "%%) TX
%s("%(row["txrate"]) + row["txeval"]+ "%%)")
+
+        # Rezervacijos delsa Dr()
+        if "5g" in row["band"]:
+            DIFS = 34
+
+            Cli5g = Cli5g + 1
```

```
+           Concur =  isConcurentioin(ath1["peercnt"], ath1["dtxdataframe"], row["txFramed"])
+
+                CollProbCurr, CollCostCurr, rxPER, txPER, rxRate, txRate, hidden, rxAirtime, txAirtime, pcksize =
CollisionCostCurr(row, DIFS, Concur)
+
+           if hidden:
+              Hidden5g = True
+
+           CollP5g.append(CollProbCurr)
+           CollC5g.append(CollCostCurr)
+
+           #CollProbWithout, CollCostWithout = CollisionCostWithout(row, DIFS)
+
+           #CollProbWithout5g = CollProbWithout5g + CollProbWithout
+           #CollCostWithout5g = CollCostWithout5g + CollCostWithout
+
+           AvgFrameSize = sumit(AvgFrameSize, pcksize)
+
+           TxPER5g = sumit(TxPER5g, txPER)
+           RxPER5g = sumit(RxPER5g, rxPER)
+           TxRate5g = sumit(TxRate5g, txRate)
+           RxRate5g = sumit(RxRate5g, rxRate)
+           Datapck5g = Datapck5g + int(row["txFramed"]) + int(row["rxdatapckd"])
+           DataBytes5g = DataBytes5g + int(row["txBytesd"]) + int(row["rxBytesd"])
+           Airtime = Airtime + rxAirtime + txAirtime
+           NConc = NConc + Concur
+
+     if NConc > 1:
+        if (Hidden5g):
+           CollProbCurr5g = sum(CollP5g)
+        else:
+           CollProbCurr5g = multiplyList(CollP5g)
+     else:
+        CollProbCurr5g = 0
+
+     TxPER5g = division(TxPER5g, Cli5g)
+     RxPER5g = division(RxPER5g, Cli5g)
+     AvgFrameSize = division(AvgFrameSize, Cli5g)
+
+                CtrlFrames, CtrlAirtime = ControllCalculations(int(ath1["dtxctlframe"]), int(ath1["drxctlframe"]),
int(row["lrxmgmtrate"]))

+     # Add new calculations
+     peer   = {'datetime': pd.Timestamp.now(),
+                        'CollProb5gCurr': CollProbCurr5g, 'CollProb5gWithout': CollProbCurr5g, 'CurrConfOverhead5g':
CollCostCurr5g, 'Overhead5gWithout': CollCostWithout5g,
+               'TxPER5g': TxPER5g, 'RxPER5g': RxPER5g, 'RxRate5g': RxRate5g, 'TxRate5g': TxRate5g, 'Hidden5g':
Hidden5g, 'DataPck5g': Datapck5g, 'DataBytes5g': DataBytes5g,
+                'Airtime': Airtime, 'CtrlFrames': CtrlFrames, 'CtrlAirtime': CtrlAirtime, 'AvgFrameSize': AvgFrameSize,
'NConc': NConc
+            }
+     config.update_peer_stats(peer)
+
+ #sanity check
+ def zero_check(x):
+     if x < 0:
+        return
+     else:
+        return x
+
+ def division(x,y):
+     try:
```

```
+        return x/y
+    except ZeroDivisionError:
+        return 0
+
+ def multiplyList(myList) :
+
+    # Multiply elements one by one
+    result = 1
+    for x in myList:
+         result = result * x
+    return result
+
+ def sumit(*args):
+    return sum(args)
+
+ # Control frames airtime
+ def ControllCalculations(txframe, rxframe, speed):
+    cnf = config.get_current_config()
+    SIFS = 16
+    CtrlFrames = zero_check(txframe) + zero_check(rxframe)
+    #RTS enabled
+    if (int(cnf.config) == 2) or (int(cnf.config) == 3):
+        #half ctrl = ACKtime = n * ACKtt()
+        CtrlAirtimeACK =  division(CtrlFrames, 2) * (312 / speed) + (division(CtrlFrames, 2) * SIFS)
+        CtrlAirtimeRTS =  division(CtrlFrames, 2) * (608 / speed) + (division(CtrlFrames, 2) * SIFS)
+        Airtime = CtrlAirtimeRTS + CtrlAirtimeACK
+    else:
+        Airtime = CtrlFrames * (312 / speed) + (division(CtrlFrames, 2) * SIFS)
+
+    return CtrlFrames, Airtime
+
+ # decides if client is hidden
+ def isHidden(txPER, rxPER, txEval, rxEval, signal):
+    if rxPER < 1 or txPER < 1:
+        return 0
+    number = ((division(txPER + rxPER, 2)) * 0.25) + (txEval*0.4) + (((signal + 90) * 2)* 0.35)
+    #log.info(number)
+    if number < 50:
+        return True
+    else:
+        return False
+
+ #checks if 2 devises concurent to transmit
+ def isConcurentioin(cnt, radata, stadata):
+    if radata < 50:
+        return 0
+    one = division(stadata, radata)
+    two = division(radata, cnt)
+    if  one > division(1, cnt) * 0.3:
+        return 1
+    else:
+        return 0
+
+ # calculate overhead
+ def overhead(frames, rate, rtson):
+    # Si = tRTS + SIFS + tCTS = RTS[b]/RAET + SIFS + CTS[b]/RATE = 160/6 + 16 + 112/6 = 61 us
+    Si = 61
+    SIFS = 16
+    if rtson:
+        #RTS on
+        # only ~30% packets need reservation
+        ovrhead = ((0.3 * frames) * Si) + ((0.7 * frames) * SIFS)
```

```
+         return ovrhead
+     else:
+         #RTS off
+         ovrhead = frames * SIFS
+         return ovrhead
```