VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

Belaidžių jutiklių tinklo modeliavimas

Atliko:
Povilas Stračinskis parašas

Vadovas:
dr. Valdas Rapševičius

Vilnius
2018

Turinys

Santrauka

Daugelyje sričių yra vis plačiau naudojami belaidžių jutiklių tinklai. Belaidžių jutiklių tinklai sudaryti iš daugelio tarpusavyje bendraujančių jutiklių. Šių jutiklių turimi energijos resursai labai riboti, todėl būtina optimizuoti jų veikimą, prieš taikant šias technologijas plačiu mastu.

Šio darbo tikslas - sukurti bevielių jutiklių tinklo modelį ir ištirti lošimų teorijos pritaikymo galimybes, optimizuojant tinklo energetines charakteristikas.

Raktiniai žodžiai Belaidžių jutiklių tinklai, lošimų teorija, modeliavimas, energetinis efektyvumas.

Summary

Wireless sensor network modeling

Wireless sensor networks are increasingly used in many areas. These types of networks are made up of many sensors, which can communicate information between each other. Because of the limited energy resources that these sensors have, their operation must be optimised before deploying these systems at a large scale.

In this thesis, a game theoretic model for the optimisation of energy efficiency is proposed. Principles of cooperative and non-cooperative game theory are used to model the communication between wireless nodes. We present a credit based algorithm for cooperation between nodes in the system.

The goal of this thesis is to create a wireless sensor network model, and to optimize energy efficiency of the network, using game theory.

Keywords Wireless sensor networks, game theory, modeling, energy efficiency.

Įvadas

Šiuolaikiniai elektronikos inžinerijos pasiekimai bei proveržiai telekomunikacijų ir informatikos srityse leido atsirasti modernioms belaidžių jutiklių tinklų technologijoms. Artimoje ateityje, vystantis daiktų internetui, milijonai belaidžių jutiklių bus naudojami duomenų surinkimui ir perdavimui. Daiktų internetas taps daugybės paslaugų pagrindu, taigi belaidžių tinklų patikimumas ir pasiekiamumas turės didelę svarbą [14].

Belaidžių jutiklių tinklai (BJT) sudaryti iš tarpusavyje bendraujančių mazgų. Šiuo atveju, mazgai - tai modernūs jutikliai, kurie tipiškai susideda iš radio siųstuvo-imtuvo, mikrovaldiklio, maitinimo grandinės bei energijos šaltinio. Jutikliai turi unikalius identifikatorius ir gali siųsti bei gauti duomenis belaidžiame tinkle. Jutiklių energijos šaltinio dydis, atminties kiekis, skaičiuojamoji galia ir ryšio pralaidumas yra labai riboti. Tai lemia mažas jų dydis ir nedidelė jų kaina. Dėl šių charakteristikų BJT technologijos susiduria su aibe unikalių problemų.

Belaidžiai jutikliai, atlikdami užduotų parametrų stebėseną, tarpusavyje dalinasi informacija, bandydami ją prasiųsti iki duomenų kolektoriaus (angl. *sink*). Kadangi BJT daugiausiai energijos sąnaudų sudaro duomenų apsikeitimas, būtina naudoti optimalius maršrutizavimo algoritmus. Taip pat svarbu tausoti jutiklio energijos resursus, todėl reikia riboti jo aktyvaus veikimo trukmę iki kaip galima mažesnės.

Beladžių jutiklių tinkluose, pavieniai mazgai neturi juos supančio tinklo topologijos modelio. Jiems tenka jį susikurti autonomiškai. Tokių mazgų tinklai dar vadinama *ad hoc* tinklais. Kai naujas jutiklis prijungiamas į tinklą, jis praneša apie savo buvimą, bei klausosi pranešimų iš savo kaimynų, taip gaudamas informaciją apie duomenų apsikeitimo galimybes. Žinodamas šias galimybes, jutiklis gali prisitaikyti prie esamos situacijos pagal nustatytus parametrus ir algoritmus.

Šiame darbe bus kuriamas bevielių jutiklių tinklo modelis. Šis modelis privalės turėti galimybę simuliuoti visų pagrindinių OSI tinklo modelio sluoksnių veikimą. Jutikliams siunčiant duomenų paketus, bus atsižvelgiama į jutiklių lokaciją bei energetinius parametrus, tokius kaip energijos likutis ir bei signalo stipris. Modelis taip pat palaikys programuojamą jutiklių logiką. Kiekvienas jutiklis galės priimti sprendimus pagal jo vidinę aplikaciją ir individualius veikimo parametrus.

Vienas didžiausių iššūkių BJT pritaikymo srityje - energetinio efektyvumo ir ilgos tinklo gyvavimo trukmės užtikrinimas [14]. Dauguma modernių BJT sudaryti iš mažus energijos resursus turinčių mazgų, o kiekvienas iš jų turi dvigubą rolę - matuoti fizinius aplinkos parametrus bei persiųsti kitų jutiklių pamatuotus duomenis iki galutinio jų surinkimo taško. Taigi, jutikliai turi du skirtingus tikslus - kuo ilgiau išlikti funkcionuojantys bei užtikrinti reikiamą serviso kokybę (angl. *QOS*).

Bevielių jutiklių tinklams didėjant ir tampant vis labiau sudėtingais, atsiranda naujų tokių tinklų efektyvumo modelių poreikis. Vienas iš galimų įrankių modeliuojant BJT - lošimų teorija (angl. *game theory*). Lošimų teorija - tai mokslo sritis, tyrinėjanti strateginį sprendimų priėmimą [15]. Lošimų teorija gali būti galingas įrankis modeliuojant individualių tinklo mazgų tarpusavio santykius bevielių jutiklių tinklų sistemose. Šiose sistemose jutikliai dalijasi baigtiniu resursu - radijo dažniu. Palikti savo valiai, jutikliai stengiasi užimti kuo didesnę dalį šio resurso, taip sukeldami trikdžius sistemoje. Naudojant lošimų teoriją, galima patobulinti jutiklių veikimo logiką bei pakreipti tinklo veikimą stabilaus ir efektyvaus veikimo link.

Lošimų teorija darbe bus pritaikoma modeliuojant tarpusavio komunikaciją tarp atskirų tinklo mazgų. Kadangi informacijos perdavimas tinkle reikalauja ne tik individualių jutiklių optimalaus veikimo, bet ir kooperacijos tarp jų, tyrime bus naudojami kooperatyvinių ir nekooperatyvinių lošimų principai. Bus taikoma ekonominio lošimo schema, kurioje jutikliai yra skatinami bendradarbiauti tarpusavyje.

Maršrutizavimas yra labai svarbi belaidžių jutiklių tinklų projektavimo dalis. Norėdami sukurti realistišką BJT modelį, sukursime galimybę naudoti modernų maršrutizacijos būdą, vadinamą DSR (angl. *Dynamic Source Routing*). Šis maršrutizavimas yra pritaikytas asinchroniškoms jutiklių tinklų sistemoms.

Skyriuje *Belaidžių jutiklių tinklai* aptariama modernių belaidžių jutiklių architektūra, pateikiama jutiklių klasifikacija pagal matuojamus dydžius ir naudojamas technologijas. Taip pat aprašomos problemos, su kuriomis susiduriama, projektuojant BJT sistemas. Galiausiai, apibrėžiamos projektuojamo simuliatoriaus veikimo sąlygos.

Skyrius *Lošimų teorija* sutelktas ties lošimų teorijos principų pateikimu ir jos taikymo BJT nagrinėjimu. Jame ištiriamos lošimų teorijos taikymo belaidžių jutiklių tinkluose galimybės, bei nustatomos galimos problemos, su kuriomis gali būti susiduriama, modeliuojant sistemą kaip lošimą.

Tinklo modeliavimo dalyje nagrinėjamas sukurtas simuliatorius, leidžiantis atlikti tikslius bandymus su jutiklių tinklais, modeliuoti didelį kiekį jutiklių ir matuoti tinklo charakteristikas. Šiame skyriuje išsamiai aptariamos pagrindinės simuliatoriaus dalys.

Pasitelkiant darbe kuriamo modelio galimybes, skyriuje *Tinklo optimizacija* pateikiami BJT energetinio efektyvumo optimizavimo žingsniai. Pritaikomos skirtingos maršrutizavimo schemos, bei tiriamas lošimų teorijos taikymo BJT naudingumas. Galiausiai, pateikiami bandymų rezultatai.

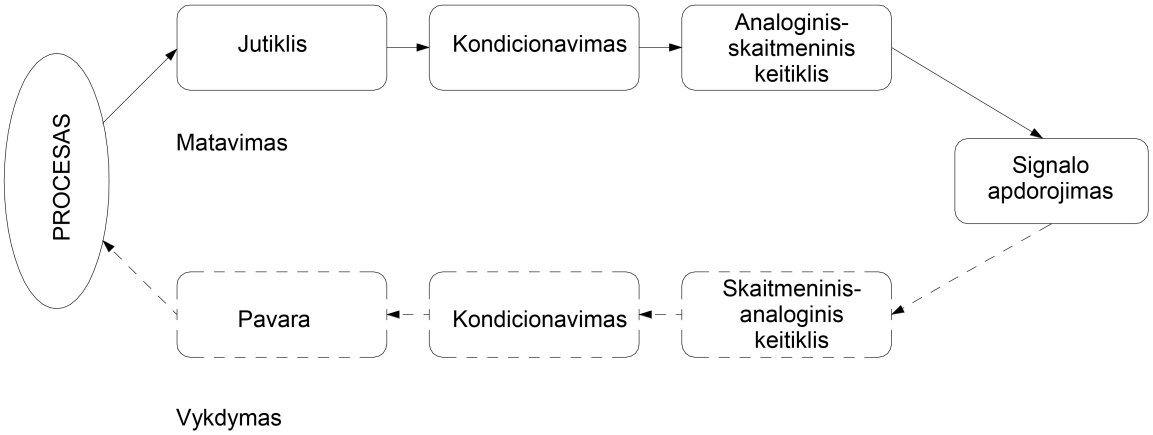
Šio baigiamojo magistrinio darbo tikslas - sukurti bevielių jutiklių tinklo modelį, bei pritaikyti lošimų teoriją, optimizuojant jutiklių tinklų veikimo charakteristikas.

1. Belaidžių jutiklių tinklai

1.1. Belaidžiai jutikliai

1.1.1. Belaidžiai jutikliai automatikoje

Jutikliais vadinami prietaisai, skirti informacijos apie objektų ir terpių fizikinių parametrų dydžius, matavimui. Jutikliai - tai signalų keitikliai, kurie konvertuoja pamatuotus fizikinius realaus pasaulio duomenis ir užfiksuotus įvykius į elektrinius signalus, kurie gali būti perduodami į programuojamus valdiklius ir išanalizuojami. Jutikliai plačiai naudojami automatizuotose procesų valdymo sistemose. Šiose sistemose modernūs belaidžiai jutikliai atlieka labai svarbią duomenų surinkimo ir perdavimo funkciją. Vienas esminių automatinio valdymo principų yra įvairiais jutikliais išmatuoti valdomo proceso darbo parametrus ir, naudojant grįžtamąjį ryšį, juos perduoti aktuatoriams, kurie atlieka fizinių parametrų korekcijas, priklausomai nuo matuojamo dydžio nuokrypio nuo norimos vertės. Bendrinė tokių sistemų veikimo schema pavaizduota (1 pav.). Jutikliui pamatavus tam tikrą fizikinį dydį (temperatūrą, slėgį ir t.t.), šie duomenys perduodami į signalų apdorojimo įtaisus, kuriuose signalai gali būti filtruojami bei sustiprinami. Po apdorojimo analoginiai signalai yra transformuojami į skaitmeninius, naudojant analoginius-skaitmeninius keitiklius.



1 pav. Duomenų surinkimas ir aktuacija uždaroje valdymo grandinėje.

Didelė dalis belaidžių jutiklių tinklų susieti su vykdikliais, kurie leidžia valdyti fizinius proceso parametrus. Iš jutiklių atsiųsti skaitmeniniai signalai yra apdorojami programuojamuose valdikliuose, iš kurių pagal pamatuotas vertes siunčiami valdymo signalai į vykdiklius. Vykdikliais gali būti siurbliai vandens rezervuaruose, relės, sujungiančios elektros grandines ir kiti automatikos komponentai. Modernūs belaidžių jutiklių tinklai (BJT) gali persiųsti iš valdiklio gautas komandas į šiuos vykdiklius, taip sukuriant uždarą valdymo grandinę.

1.1.2. Jutiklių klasifikacija

Egzistuoja daug jutiklių rūšių. Projektuojamoms sistemoms jutikliai parenkami pagal konkrečius dydžius, kurie bus matuojami, pavyzdžiui, temperatūra, drėgmė, šviesis, spaudimas. Lentelėje nr. 1 pateikiama matuojamų fizinių parametrų santrauka su atitinkamomis

matavimo technologijomis. Jutikliai klasifikuojami pagal tai, ar jiems veikti reikalingi išoriniai energijos šaltiniai. Jei jutiklis be išorinio energijos šaltinio neveikia, toks jutiklis vadinamas *aktyviniu*. Tokie jutikliai spinduliuoja energiją (šviesą, elektromagnetines bangas) tam, kad pamatuotų atsaką į jas. Kita jutiklių grupė - *pasyviniai* jutikliai, naudoja energiją iš matuojamos aplinkos ir ją konvertuoja į signalus.

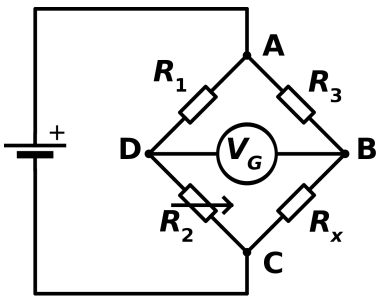
1 lentelė. Jutiklių topologija

Tipas	**Jutikliai**
Temperatūros	Termistoriai, termoporos
Spaudimo	Barometrai, manometrai
Optiniai	Fotodiodai, fototranzistoriai, fotorezistoriai
Judesio	Akselerometrai, giroskopai
Akustiniai	Mikrofonai, pjezoelektriniai rezonatoriai
Mechanikos	Pjezoelektriniai jutikliai
Elektromagnetizmo	Holo jutikliai, magnetometrai
Cheminiai	pH matuokliai, elektrocheminiai jutikliai, dujų jutikliai
Drėgmės	Kondensatoriniai ir rezistoriniai jutikliai, higrometrai
Radiacijos	Geigerio-Miulerio skaitikliai
Rac	Fotodiodai, fototranzistoriai, jonizacijos detektoriai

Jutikliai taip pat gali būti klasifikuojami pagal veikimo metodą, kuriuo fizinių reiškinių parametrai verčiami į elektrinius signalus. *Varžiniai* jutiklių veikimas paremtas laidininko elektrinės varžos ρ pokyčiais, besikeičiant aplinkos parametrams, tokiems, kaip temperatūra. Laidininko varža R randama taip:

$$R = \frac{l \times \rho}{A} \tag{1.1}$$

čia l - laidininko ilgis, o A yra laidininko skerspjūvio plotas.
Vienas labiausiai paplitusių varžinių jutiklių - Wheatstone'o tiltelis (2 pav.).



2 pav. Wheatstone'o tiltelio elektrinė schema.

Šio tiltelio schemoje R_1, R_2 ir R_3 yra žinomo dydžio rezistoriai (R_2 varža yra reguliuojama), o rezistoriaus R_x varža yra nežinoma. Kai varžų santykis R_2/R_1 yra identiškas santykiui R_x/R_3, matuojama įtampa V_G yra lygi 0 V. Tačiau, kai varža R_x pakinta (pavyzdžiui, pakitus temperatūrai), pakinta varžų balansas, tarp taškų D ir B atsiranda potencialų skirtumas $V_G > 0$. Ši priklausomybė gali būti išreikšta 1.2 išraiška:

$$V_G = E \times (\frac{R_x}{R_3 + R_x} - \frac{R_2}{R_1 + R_2}) \tag{1.2}$$

čia E - akumuliatoriaus elektrovara. Wheatstone'o tilteliai plačiai naudojami matuoti temperatūrą, įtempio jėgą, šviesį ir kitus dydžius.

Antra didelė jutiklių grupė - *kondensatoriniu* principu veikiantys jutikliai. Jie naudojami matuoti judesį, atstumą, pagreitį, elektrinius laukus, cheminę sudėtį. Kondensatoriai kaupia elektros energiją tarp dviejų elektrodų sukurtame elektriniame lauke. Elektrodus skiria dielektrinis sluoksnis, kurio skvarba žymima ε. Kondensatoriaus talpa apskaičiuojama taip:

$$C = \frac{\varepsilon \times A}{d} \tag{1.3}$$

kur A yra elektrodų paviršiaus plotas, o d - atstumas tarp jų. Pasikeitus kažkuriam iš šių parametrų, kondensatoriaus talpa pasikeis. Pavyzdžiui, jei vieną iš elektrodų paveiks spaudimo jėga, atstumas d sumažės, taip padidinant talpą C. Analogiškai, padidėjus aplinkos temperatūrai ar drėgmei, dielektrinė skvarba sumažės, taigi kondensatoriaus talpa pasikeis.

Indukciniai jutikliai paremti elektrinės indukcijos principu, kuris aprašo reiškinį, kurio metu kintančioji elektros srovė sukuria elektromagnetinę jėgą. Induktyvumo dydį nulemia jutiklio dydis (skerspjūvio plotas, ritės ilgis), vijų skaičius ritėje, bei šerdies elektromagnetinė skvarba. Šerdžiai judant ritėje, pasikeičia induktyvumas. Indukciniai jutikliai dažnai naudojami atstumo, jėgos, spaudimo, temperatūros ir pagreičio matavimui.

Paskutinė jutiklių grupė yra *pjezoelektriniai* jutikliai. Jie remiasi kai kurių medžiagų (kristalų, keramikos) savybe sukurti įtampą, kai jos deformuojamos išorinės jėgos poveikiu. Pagrindinis tokių jutiklių privalumas - pjezoelektrinis efektas yra atsparus elektromagnetinių laukų ir radiacijos poveikiui.

1.2. Belaidžių jutiklių tinklai

Tradicinėse telemetrijos sistemose jutikliai laidais tiesiogiai jungiami prie signalus apdorojančių įrenginių. Tačiau tai lemia didelę instaliacijos kainą, kai jutikliai yra dideliu atstumu nuo duomenų surinkimo įrenginių, bei mažą sistemos atsparumą gedimams. Nutrūkus jutiklio laidininkui, vienintelė išeitis yra aptarnaujančiam personalui pakeisti visą laidininką nauju. Dėl šių priežasčių vis plačiau naudojami belaidžių jutiklių tinklai, kurie gali veikti nuotoliniu būdu, dažnai pavojingose arba neprieinamose aplinkose. Kadangi belaidžiai jutikliai gali būti mobilūs, esant poreikiui, jie gali būti nesunkiai perkeliami į kitą lokaciją.

Balaidžiai jutikliai BJT sistemose turi ne tik jutiminį komponentą, bet ir integruotą procesorių, komunikacijos modulį bei vidinę atmintį. Tokie jutikliai atsakingi ne tik už duomenų surinkimą, bet ir kitų jutiklių duomenų persiuntimą, informacijos apdorojimą, tinklo analizę ir maršrutizavimą. Kai spiečius tarpusavyje susijusių jutiklių matuoja didelio masto sistemos ar aplinkos parametrus, jie sudaro belaidžių jutiklių tinklą (BJT). Jutikliai komunikuoja ne tik tarpusavyje, bet ir su bazinių stočių prieigos taškais (3 pav.). Bazinėse stotyse atliekama duomenų analizė, saugojimas, pagal gautus duomenis suformuojamos valdymo komandos.

1.2.1. Panaudojimo sritys

Belaidžių jutiklių tinklai yra naudojami teleinformacijos surinkimui ir perdavimui, matavimui ir procesų valdymui. Ši technologija paplitusi gamtinės aplinkos tyrimuose, karyboje, urbanistiuose matavimuose. Pagrindinės BJT pritaikomo sritys yra šios:

- ***Karyba.*** Kadangi BJT gali būti sudaryti iš tankiai pasklidusių, pigių jutiklių, jie yra labai tinkami naudoti karo lauke. Priešui sunaikinus kai kuriuos iš tinklo jutiklių, bendras tinklo veikimas nėra sutrikdomas. Šie tinklai yra naudojami savų karinių pajėgų (karių, amunicijos, transporto) monitoringui, karo lauko žvalgybai, priešiškų pajėgų ir aplinkos žvalgybai, cheminių ir biologinių atakų aptikimui.

- ***Aplinkos tyrimai ir gamtosauga.*** Belaidžiai jutikliai vis plačiau naudojami gamtiniuose tyrimuose. Viena to priežasčių - šie jutikliai sėkmingai veikia atokiose vietovėse, kuriose nėra elektros energijos tinklų, ir atšiauriose sąlygose, kuriose reikia įpatingo tiklo atsparumo. Keletas panaudojimo pavyzdžių pateikiama žemiau:

  - *Oro taršos matavimas.* BJT buvo sėkmingai panaudoji jau keliuose miestuose (Stokholme, Londone), matuojant pavojingų žmonėms dujų koncentraciją ore.
  - *Miško gaisrų aptikimas.* Belaidžių jutiklių tinklai gali būti instaliuojami miškuotose vietovėse, siekiant kuo greičiau užfiksuoti gaisro pradžią. Ankstyvas gaisro aptikimas yra kritiškas, nuo to gali priklausyti, ar ugniagesiams pavyks suvaldyti gaisrą.
  - *Nuošliaužų aptikimas.* Jutikliai gali užfiksuoti nuošliaužų atsiradimo požymius, joms dar neprasidėjus.
  - *Vandens kokybės matavimas.* Vandens kokybė gali būti analizuojama užtvankose, upėse, ežeruose, taip pat požeminiuose vandens rezervuaruose. Belaidžiai jutikliai gali būti naudojami giliai po vandeniu, kur kitos matavimų priemonės būtų sunkiai įgyvendinamos.
  - *Pavojingų gamtos reiškinių aptikimas.* Belaidžiai jutikliai gali realiu laiku aptikti pavojingus gamtos reiškinius (vandens lygio kilimas, žemės drebėjimas).

- ***Sveikatos monitoringas.*** Medicinoje gali būti naudojami keli tipai belaidžių jutiklių: implantuoti, nešiojami ir esantys aplinkoje. Implantuojami jutikliai yra mediciniškai įterpiami žmogui po oda. Nešiojami prietaisai gali matuoti žmogaus kūno poziciją, jo vietą (koordinates), šie jutikliai gali būti naudojami sergančių pacientų priežiūrai ligoninėse ar namuose.

- ***Pastatų monitoringas.*** Belaidžių jutiklių tinklai gali būti naudojami pastatų (dangoraižių, tiltų) monitoringui ir struktūrinių pažeidimų aptikimui.

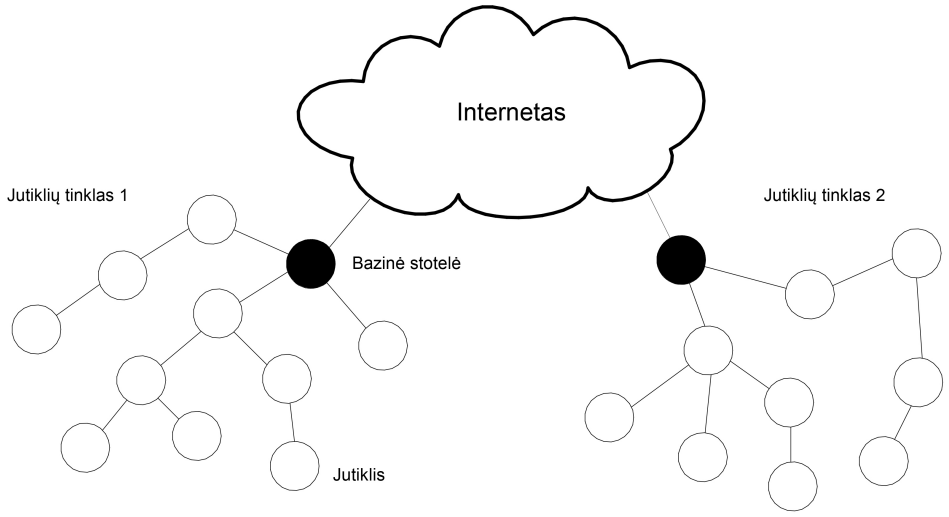
- ***Žemdirbystė.*** Belaidžių jutiklių tinklai sėkmingai naudojami moderniuose žemdirbystės ūkiuose. Jie yra naudojami temperatūros ir drėgmės kontroliavimui šiltnamiuose. Kai temperatūra ar drėgmė nukrenta/pakyla iki tam tikro lygio, šiltnamio savininkas gali būti informuojamas apie tai SMS žinute arba el. paštu.

- ***Pramonė ir inžinerinės sistemos.*** BJT vis plačiau pritaikomi inžinerinėse sistemose. Bevielių jutiklių tinklų pritaikymas lemia didėjantį šių sistemų produktyvumą ir efektyvumą. Jutikliai aptinka gedimus, matuoja temperatūrą, vibracijas ir elektrinius dydžius. Pagrindiniai BJT pritaikymai pramonėje yra šie:

  - *Įrenginių monitoringas.* Jau sukurti bevieliai daugumos standartinių automatikoje naudojamų jutiklių atitikmenys, naudojami įrenginių darbo parametrų matavimui ir gedimų aptikimui. Modernių belaidžių jutiklių naudojimas gali būti

ne tik ekonomiškai naudingesnis, tačiau ir sukuria galimybes atsirasti naujiems technologiniams sprendimams.

– *Duomenų centrų monitoringas.* Dėl ribotos erdvės serverių narveliuose vis dažniau naudojami belaidžiai jutikliai, kurie matuoja temperatūrą. Kadangi standartuose rekomenduojama vienam narveliui naudoti iki šešių temperatūros jutiklių, belaidžiai jutikliai turi pranašumą prieš tradicinius jutiklius, sujungiamus laidais.
– *Duomenų surinkimas.* Belaidžių jutiklių tinklai naudojami duomenų surinkimui vėjo ir saulės jėgainėse, vandens saugyklose ir kitose infrastruktūrose. Surinkti duomenys yra perduodami į duomenų atvaizdavimo sistemas, kurios realiu laiku gali parodyti svarbią statistinę informaciją.



3 pav. Belaidžių jutiklių tinklų komunikacija su bazine stotimi.

1.2.2. Privalumai ir trūkumai

Belaidžių jutiklių technologijoms atsiranda vis daugiau panaudojimų sričių dėl akivaizdžių pastarųjų privalumų, tačiau, kaip ir kitos technologijos, ši turi ir svarbių trūkumų. Toliau pateikiame pagrindinius BJT privalumus ir trūkumus.

Privalumai:

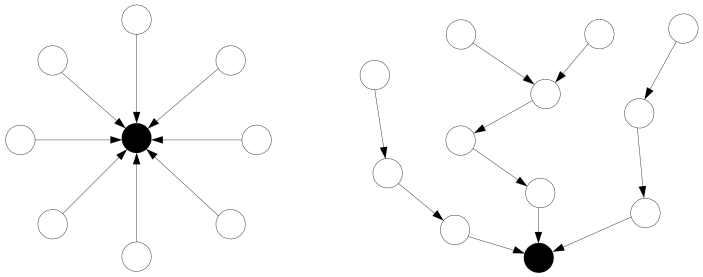
- Sistemos gali būti diegiamos be iš anksto numatytos infrastruktūros.
- BJT gali būti sėkmingai naudojami nepasiekiamose vietovėse ir pavojingose sąlygose.
- Nebrangus šių technologijų diegimas.
- Signalų perdavimui nereikalingi laidai.
- Tinklai gali būti lengvai plečiami arba mažinami pagal poreikius.

Trūkumai:

- Mažas saugumo laipsnis. Nors pastaruoju metu daug akademinės bendruomenės dėmesio yra sutelkta ties daiktų interneto saugumo užtikrinimu, jutiklių tinklai vis dar gali tapti lengvu taikiniu kibernetinėje erdvėje.
- Maža greitaveika. Palyginus su tinklais, perduodančiais informaciją laidais, BJT komunikacijos vyksta žymiai lėčiau.
- Radijo ryšio trikdžiai. Komunikaciją tarp jutiklių gali sutrikdyti dideli atstumai, storos sienos ir stiprūs elektromagnetiniai laukai.

1.2.3. Tinklo topologija

Kai jutiklių radijo modulių transliavimo nuotolis yra pakankamai didelis ir visi tinklo jutikliai gali transliuoti savo duomenis tiesiogiai į bazinę stotį, galima juos sujungti žvaigždės topologija (4 pav.). Tokiame tinkle, kiekvienas mazgas gali perduoti duomenis bazinei stočiai vienu persiuntimu. Tačiau jutiklių tinklai dažnai yra pasklidę dideliame plote, o radijo ryšio transliavimo stipris turi būti kuo mažesnis, siekiant sumažinti energijos sunaudojimą. Dėl to belaidžių jutiklių tinkluose dažniausiai naudojama tinklelio topologija. Taip išdėstyti jutikliai turi ne tik matuoti ir perduoti savo duomenis, tačiau ir tarnauti kaip tarpinės stotelės kitiems tinklo mazgams, t.y. kooperuoti su kitais jutikliais.



4 pav. Tinklo topologijos. Žvaigždės topologija kairėje, tinklelio - dešinėje.

1.3. Apribojimai

Nors belaidžių jutiklių tinklai turi daug panašumų į tradicinius kompiuterinius tinklus, tačiau juose susiduriama su unikaliomis problemomis ir apribojimais, specifiškais šioms technologijoms. Toliau bus aptariami svarbiausi BJT apribojimai, į kuriuos atsižvelgiama projektuojant darbe kuriamą modelį.

1.3.1. Energija

Belaidžiai jutikliai tipiškai turi mažus elektros energijos šaltinius. Dažniausiai jais tarnauja baterijos, kurioms išsikrovus, jas reikia keisti arba įkrauti (pavyzdžiui, saulės energija). Kai kuriems jutikliams nei vienas iš šių variantų netinka, pasibaigus energijai, jutikliai tampa nebetinkami naudojimui. Nuo to, ar jutiklio baterija yra įkraunama, priklauso jo panaudojimo galimybės. Jutiklis turi išlikti gyvybingas per visą numatytą darbo periodą arba iki kol

baterija galės būti įkrauta. Darbo periodas numatomas pagal panaudojimo specifiką, pavyzdžiui, mokslininkams atliekant ledkalnių judėjimo matavimus, jutikliai turi veikti mėnesius ar net metus, tuo tarpu karo lauke jutikliai gali būti reikalingi keletai valandų ar dienų.

1.3.2. Autonomija

Belaidžių jutiklių tinklai privalo veikti aukšto patikimumo sistemose, esančiose atokiose vietovėse bei pavojingose aplinkose. Tinklelio topologija lemia didelį tinklo fizinį saugumą, tačiau dažnai šiose aplinkose nėra galimybės reguliariai aptarnauti ir taisyti gedimus, todėl BJT sistemos turi būti autonomiškos.

1.3.3. *Ad hoc* veikimas

Daugelyje BJT sistemų jutikliai neturi iš anksto suplanuotų lokacijų. Jutiklio generuojamos informacijos kiekis ir tipas gali keistis, priklausomai nuo jo instaliacijos vietos, aplinkinių jutiklių skaičiaus ir artumo, bei kitų parametrų. Jutikliai gali būti mobilūs, palikti tinklą ir grįžti į jį, jutikliams išsikrovus, tinklas privalo pasigydyti, t.y. rasti naujus maršrutus duomenims perduoti. Šie reikalavimai lemia tai, jog BJT negalima pritaikyti tradicinių maršrutizavimo algoritmų.

1.3.4. Radijo ryšys

Belaidžių jutiklių tinkluose esantys mazgai dalijasi tuo pačiu radijo ryšio kanalu. Tai lemia, jog BJT veikia tokie neigiami reiškiniai kaip radijo bangų interferencija, difrakcija, bei susilpnėjimas keliaujant per kliūtis.

1.4. Modelio veikimo sąlygos

Realistiškam ir tiksliam bevielių jutiklių tinklų simuliavimui reikalingas kompleksinis tinklo modelis. Šis modelis turi įgyvendinti visus OSI tinklo modelio sluoksnius. Dėl to BJT simuliatorius turi įgyvendinti šiuos reikalavimus:

1. Turėti galimybę tiksliai atkartoti radijo bangų sklidimą erdvėje. Tai simuliuos OSI fizinio ryšio sluoksnį. Radijo modelis turi simuliuoti duomenų paketų siuntimą/gavimą, taip pat paketų kolizijas.

2. Tinklo mazgai turi turėti lokacijas. Pagal tai bus apskaičiuojama, ar jie gali bendrauti tarpusavyje.

3. Kadangi mazguose bus naudojami lošimų teorijos logika, mazgai turi turėti galimybę priimti sprendimus.

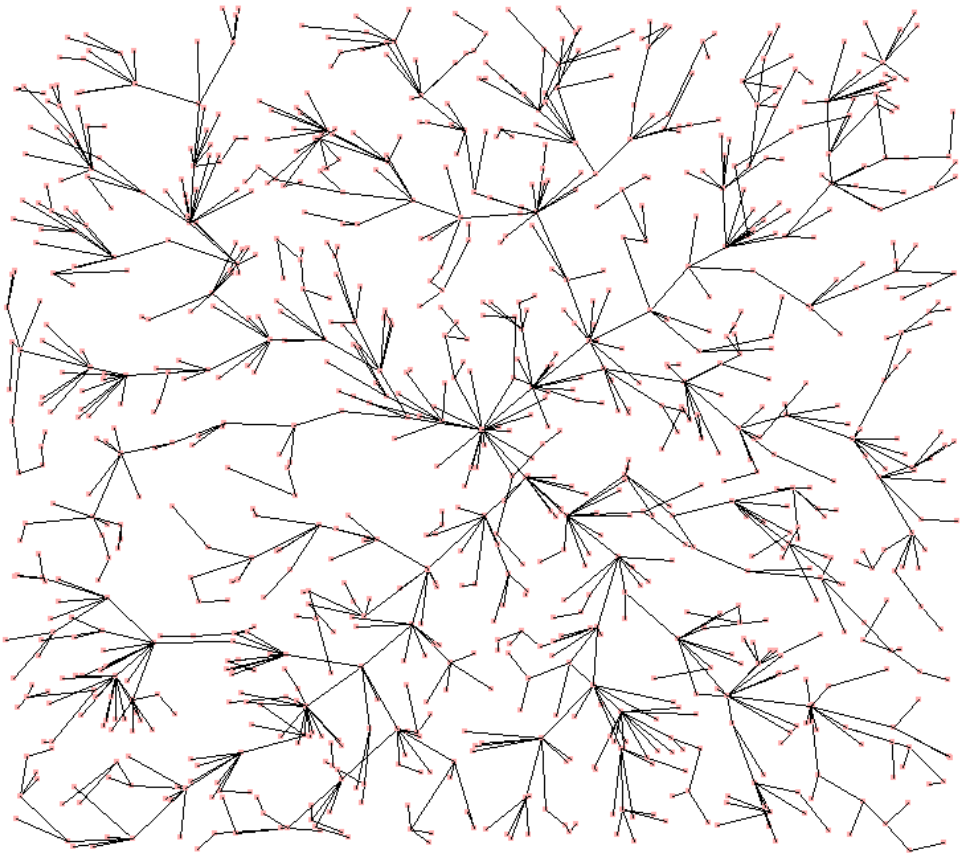
4. Modelis turi turėti galimybę grafiškai atvaizduoti modeliuojamos sistemos veikimo reprezentaciją.

2. Maršrutizavimas

Belaidžių jutiklių tinkle efektyvus marštutizavimas yra būtinas. Kadangi jutikliai yra išdėstomi tinklelio topologijos principu, pasiekti tinklo efektyvumą yra sudėtingas uždavinys. Be to, priešingai, nei tradiciniuose kompiuterių tinkluose, BJT dažniausiai nesiremiama IP adresavimu tinklo lygmenyje, taigi nesiremiama daugeliu šiose technologijose turimų privalumų. Šiame skyrelyje aptariami pagrindiniai maršrutizavimo BJT metodai, pritaikomi šiame darbe.

2.1. Tvindymo maršrutizavimas

Paprasčiausia maršrutizavimo strategija yra *tvindymas* (angl. *flooding*) [6]. Jutiklis siunčia duomenų paketus savo artimiausiems kaimynams, kurie analogiškai juos persiunčia saviems kaimynams, kol, galiausiai, visi tinkle esantys jutikliai gauna duomenis. Darbe kuriamo modelio su tvindymo maršrutizavimu veikimas pateikiamas (5 pav.). Jei tarp jutiklio ir bazinės stotelės tinkle egzistuoja nepertraukiamas maršrutas, tvindymo maršrutizavimas užtikrina, kad duomenys bus persiųsti (neatsižvelgiant į neigiamus veiksnius, galinčius sutrikdyti duomenų siuntimą).



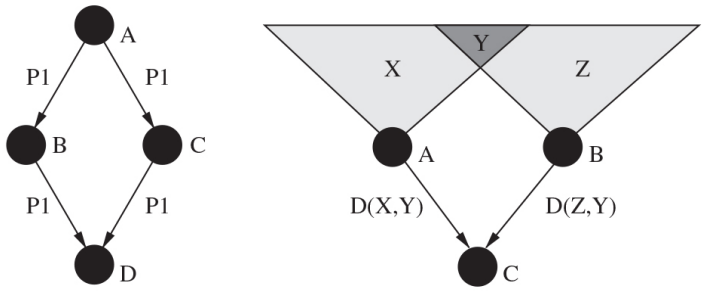
5 pav. Tvindymo maršrutizavimas. Viduryje esantis jutiklis išsiunčia vieną žinutę, kuri pasiekia visus kitus simuliacijoje esančius jutiklius.

Pagrindinis šio maršrutizavimo privalumas yra paprastumas. Didžiausias trūkumas - smarkai apkraunamas tinklas. Tam, kad tos pačios žinutės nekeliautų tinkle ilgiau, nei to reikia, būtina riboti duomenų paketo persiuntimų skaičių. Jis turi būti pakankamai didelis,

kad duomenys nukeliautų iki galutinio gavėjo, tačiau tiek mažas, jog žinutės neužsiliktų tinkle per ilgai. Siunčiamos žinutės taip pat turi turėti laiko žymes, pagal kurias jutikliai galėtų spręsti, ar duomenys yra pasenę. Tačiau, net įgyvendinus šiuos reikalavimus, tvindymo maršrutizavimas turi svarbių trūkumų:

1. **Duomenų duplikacija**. Jutiklis, gavęs naują žinutę, ją persiunčia visiems kaimynams, neatsižvelgiant į tai, ar kaimynai šią žinutę jau yra matę. Taip yra švaistomi energijos resursai. Ši problema iliustruojama (6 pav.). Mazgas *A* išsiunčia duomenų paketą *P1* kaimynams *B* ir *C*. *B* persiunčia gautą paketą savo kaimynui *D*, kaip ir mazgas *C*. Net jei jutiklis *D* atmes antrą kartą gautus duomenis, energija bus iššvaistyta siunčiant juos iš *C* į *D*.

2. **Persidengimas**. Jutikliai gali matuoti duomenis arti vienas nuo kito, todėl atsiranda radijo ryšio persidengimo problema. (6 pav.) dešinėje jutikliai *A* ir *B* dalijasi matuojamu regionu *Y*, todėl abu siunčia tuos pačius duomenis jutikliui *C*. Dėl to sistema patiria bereikalingus energijos nuostolius. Sprendimas šiai problemai sudėtingesnis nei prieš tai aptartąjai, nes jis reikalauja koordinacijos ir kooperacijos tarp arti esančių jutiklių.

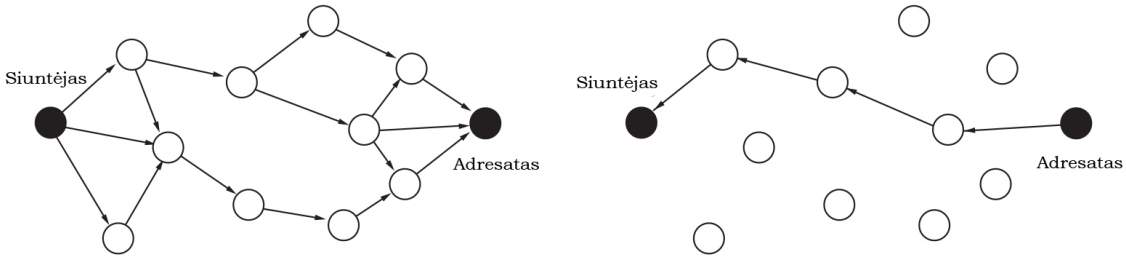


6 pav. Duomenų duplikacijos ir persidengimo problemos.

Tam, kad padidinti tinklo pralaidumą ir sumažinti besidubliuojančių žinučių persiuntimų skaičių, reikia, kad jutikliai gautus duomenis persiųstų ne visada. Tam pasiekti taikomas tikimybių teorija paremtas algoritmas, kurį pritaikydamas jutiklis pasirenka, persiųsti gautus duomenis (tikimybė *p*), ar juos atmesti (tikimybė *1-p*) [6]. Taip galima sutaupyti didelį kiekį tinklo energijos resursų, bei sumažinti duomenų duplikacijos problemos padarinius. Tačiau, pritaikius šį sprendimą, padidėja duomenų praradimo tikimybė.

2.2. Dinaminis maršrutizavimas

Dinaminis maršrutizavimas (angl. *Dynamic Source Routing*, trumpinys DSR), yra maršrutizavimo algoritmas, pritaikytas belaidžių jutiklių sistemoms. Pagrindinis šio algoritmo mechanizmas yra optimalaus maršruto radimas prieš siunčiant duomenis adresatui (7 pav.).



7 pav. Maršruto radimo procesas

Kiekvienas jutiklis turi atskirą maršrutų lentelę visiems jam aktualiems maršrutams saugoti. Jutiklis inicijuoja maršruto aptikimo procesą, kai jis neturi galiojančio maršruto iki reikiamo adresato [16]. Maršrutas randamas taip:

1. Duomenų siuntėjas išsiunčia maršruto aptikimo užklausos žinutę RREQ (angl. *route request*) savo kaimynams. Šioje žinutėje yra siuntėjo ir adresato adresai, šuolių iki adresato skaitiklis bei laiko žymė. Paskutinis parametras RREQ žinutėje yra visų šią žinutę persiuntusių jutiklių adresai.

2. Kiekvienas tarpinis jutiklis, gavęs maršruto radimo žinutę, prie jos prideda savo adresą ir ją persiunčia toliau. Jei jutiklis gauna jau matytą žinutę RREQ, ji atmetama.

3. Gavęs RREQ žinutę, numatytasis duomenų adresatas suformuoja atsako žinutę RREP (angl. *route response*), kurioje yra patalpinamas gautasis žinutės maršrutas, t.y. visų žinutę persiuntusių jutiklių adresai. Ši žinutė išsiunčiama.

4. Žinutę RREP persiunčia tik tie jutikliai, kurių adresai yra užkoduoti žinutės maršruto skiltyje. Visi kiti jutikliai ją atmeta.

5. RREP žinutė su maršrutu iki adresato pasiekia pradinį siuntėją. Jis atnaujina savo maršrutų lentelę naujai gautu maršrutu, ir pradeda siųsti duomenų žinutes DATA šiuo maršrutu.

6. Gavęs duomenų žinutę, jutiklis patikrina, ar jo adresas yra tarp žinutėje esančio maršruto adresų. Jei taip, jis persiunčia žinutę toliau. Jei ne, žinutė atmetama.

ID	Siuntėjo nr.	Adresato nr.	Maršrutas iki adresato	Siunčiami duomenys

8 pav. DSR maršrutizavimo DATA tipo žinutės struktūra

Tokiu būdu duomenys tinkle siunčiami tik egzistuojančiais gyvybingais maršrutais. Duomenų paketų kolizijos tikimybė tampa labai maža, kadangi žinutes persiunčia tik aptiktame maršrute esantys jutikliai [16].

3. Lošimų teorija

Lošimų teorija gali būti apibrėžta kaip konfliktų, konkurencijos bei bendradarbiavimo situacijų loginė analizė. Šios situacijos gali būti vadinamos lošimais [15]. Lošimų teorija daro prielaidą, kad kiekvienas lošėjas sieks maksimizuoti savo išlošį. Šitaip atsiranda poreikis kvantifikuoti išlošį, siejant jį su „ekonominės naudos" samprata.

3.1. Pagrindiniai lošimų teorijos principai

Lošimų teorijos tyrinėjimo objektas yra strateginio lošėjų sąveikavimo analizė. Lošimų teorija iki šiol buvo plačiai taikoma ekonomikoje, politikos moksluose, psichologijoje ir biologijoje. Lošimų teorija nagrinėja žaidimus, kurių galutinis rezultatas priklauso nuo kiekvieno žaidėjo ėjimų. Lošėjų galimi ėjimai vadinami *strategijomis* [15]. Lošimas susideda iš trijų pagrindinių komponentų: žaidėjų imties, individualių strategijų ir ėjimų imties, bei išlošių imties. Lošimo dalyviai yra lošėjai, kurie kiekviename žaidimo etape daro ėjimus. Šie ėjimai visada nulemia lošimo baigtį, kuri yra skirtinga kiekvienam lošėjui. Ėjimai įtakoja ne tik juos darantį lošėją, bet ir kitus lošėjus. Svarbu apskaičiuoti, kiek lošėjas gali išlošti ir kaip. Lošimų teorijoje visi lošėjai laikomi protingais, sugebančiais analizuoti ir prieiti logiškas turima informacija grįstas išvadas.

3.2. Lošimų tipai

3.2.1. Nulinės sumos ir nenulinės sumos lošimai

Patys paprasčiausi yra dviejų asmenų lošimai, kuriuose lošėjų interesai yra tiesiogiai priešingi: vienam kažką išlošiant, kitas tiek pat pralošia. Tokie lošimai vadinami *nulinės sumos* lošimais. Tokioje situacijoje žaidėjų derybos neturi prasmės.

Šiais atvejais ieškoma geriausių, arba dominuojančių, strategijų, kai tam tikros situacijos yra geriausios (optimalios) lošėjams. Nulinės sumos lošimų atveju bendradarbiaujant nieko neįmanoma gauti abiem pusėms iš karto, nes lošėjų interesai yra diametriškai priešingi. Skirtingai analizuojamos situacijos, kai lošėjai renkasi kartu nežinodami, ką pasirinko priešininkas, ir kai lošimo dalyviai žaidžia paeiliui, žinodami ankstesnį priešininko žingsnį. Nulinės sumos lošimo atveju lošėjas, kuris pirmasis turi pradėti pasirinkimų seriją, automatiškai atsiduria prastesnėje situacijoje - tai gerai iliustruoja „žinojimo" svarbą konkurencinėje pasirinkimo situacijoje.

Nenulinės sumos lošime išlošio funkcija turi būti duota kiekvienam lošėjui, ji nebėra kito lošėjo išlošio funkcija su minuso ženklu. Tuomet atsiranda aibė strategijų, kai lošimų teorija priartėja prie racionalaus pasirinkimo realiame pasaulyje. Lošimuose susidaro sąlygos bendradarbiavimui tarp jo dalyvių. Paprastesniais atvejais galima rasti elegantiškus sprendimus, nes atsiranda „pusiausvyros" taškai. Nors jie ir nėra idealūs, arba kitaip - Pareto optimalūs, kai kurie sprendimai yra priimtiniausi tuo atveju, jei kiekvienas lošimo dalyvis vadovaujasi savo individualiu racionaliu pasirinkimu . Šiame darbe toliau bus nagrinėjami nenulinės sumos lošimai.

3.2.2. Kooperatyviniai ir nekooperatyviniai lošimai

Egzistuoja dvi svarbiausios lošimų klasės - nekooperatiniai ir kooperatiniai lošimai [8]. Pirmuoju atveju, lošėjas priima sprendimus nederindamas savo veiksmų su kitais lošėjais ir visiškai neturėdamas informacijos apie jų sprendimus. Antruoju atveju, lošėjai veikia tarpusavyje susitarę ir derybos dėl lošimo baigties sudaro lošimo esmę. N-dalyvių lošimai yra unikalūs keletu požiūrių: papildomų lošėjų atsiradimas leidžia atsirasti koalicijoms, t.y. atskiriems susitarimams tarp lošėjų. Vienas svarbiausių iškylančių klausimų yra laimėjimų „dalybos" tarp lošėjų, arba kitaip - išlošių pasiskirstymas.

3.2.3. Statiniai lošimai

Statiniai lošimai - tai tokie lošimai, kuriuose visi žaidėjai priima sprendimus tuo pat metu, nieko nežinodami apie kitų žaidėjų strategijas. Jei lošėjas visada pasirenka tą pačią strategiją, kuri apsprendžia lošėjo ėjimą bet kurioje žaidimo situacijoje, ta strategija vadinama *grynąja*. Jei žaidėjas renkasi strategiją atsižvelgdamas į išlošio tikimybę - ta strategija vadinama *maišyta*. Maišytą strategiją galima užrašyti taip: $\delta_i \in \delta(S_i)$ kur $\delta(s_i)$ yra tikimybė pasirinkti s_i ($\delta(S_i) \geq 0$).

$$\sum_{s_i \in S_i} \delta_i(s_i) = 1 \tag{3.1}$$

Jei maišyta strategija $\delta_i(a_i) = 0$ visoms strategijoms išskyrus vieną, ta strategija yra grynoji strategija.

3.2.4. Dinaminiai kartotiniai lošimai

Dinaminiuose lošimuose lošėjai daro ėjimus iš eilės, turėdami informacijos apie kitų lošėjų ėjimus. Bent vienas lošėjas turi daugiau nei vieną ėjimą, o ėjimų tvarka yra svarbi. Lošėjai priima sprendimus, remdamiesi savo buvusiais ir priešininkų ėjimais, kurie apibrėžia jo dabartinę poziciją. Jei lošimas turi daug periodiškų ėjimų su tomis pačiomis sąlygomis, jis vadinamas kartotiniu. Šio darbo praktinėje dalyje nagrinėjamas modelis yra būtent toks - jutikliai siunčia informaciją daugybę kartų, siekdami ją perduoti galutiniam kolektoriui.

3.3. Lošimų išraiška

Lošimai paprastai yra atvaizduojami strategine forma, kur kiekvienas lošėjas turi tik padaryti vieną ėjimą, ir jis nieko nežino apie kitų žaidėjų jau padarytus ėjimus. Visi lošėjai atlieka ėjimus tuo pačiu metu. Lošėjų sprendimai daryti ėjimus vadinami *strategijomis*. Lošimą G strategine forma galima užrašyti taip:

$$G =< P, S, F > \tag{3.2}$$

kur:

$P = \{1, 2, ..., n\}$ yra lošėjų aibė;

$S = \{S_1, S_2, ..., S_n\}$ yra lošėjų strategijų aibė, po rinkinį kiekvienam lošėjui;

$F = \{F_1, F_2, ..., F_n\}$ yra veiksmingumo funkcijų aibe, po rinkinį kiekvienam lošėjui.

Lošimo baigtys vadinamos strategijomis. Strategijų aibė susidaro iš vektorių $s = (s_1, s_2, ..., s_n)$, kurių kiekvienas elementas reiškia vieną lošėjo ėjimą. Kiekvienam lošėjui pasirinkus ėjimus, iš jų susidaranti strategijų aibė apibrėžia žaidimo baigtį. Visų strategijų aibėje apibrėžtos lošėjų veiksmingumo funkcijos $F = (f_1(s), f_2(s), ..., f_n(s))$. Kiekvienas lošėjo strategijos pasirinkimas vadinamas ėjimu. Strategijos apibrėžia specifiškus veiksmus, kuriuos turi atlikti lošėjas kiekviename lošimo etape. Lošėjo norima pasiekti žaidimo baigtis yra vadinama *išlošiu*. Išlošis yra kiekvieno lošėjo žaidimo baigties kiekybinė išraiška. Išlošių matrica tiesiogiai apibūdina kiekvieno lošėjo išlošius, susiklosčius vienokiam ar kitokiam pasirinktų strategijų deriniui. Lošimo tikslas kiekvienam lošėjui yra rasti pasiekti didžiausią išlošį.

3.3.1. Nešo pusiausvyra

Lošimų teorijoje vyrauja keletas būdų nustatyti, kurios strategijos ar veiksmai leidžia lošėjui sėkmingai užbaigti lošimą. Vienas tokių būdų - Nešo (angl. *Nash*) pusiausvyros radimas. Dviejų žaidėjų strategijų porą vadiname Nešo pusiausvyra, jei vieno žaidėjo pasirinkimas yra optimalus, kai žinomas kito pasirinkimas, o pastarojo pasirinkimas yra optimalus, kai žinomas pirmojo pasirinkimas. Nešo pusiausvyrą galima interpretuoti kaip tokią spėjimų dėl kiekvieno lošėjo pasirinkimo porą, kai, išaiškėjus kito asmens pasirinkimui, nė vienas iš lošėjų savo elgsenos keisti nenori.

3.4. Lošimų teorijos pritaikymas BJT

Belaidžių komunikacijų srityje, radijo dažnis yra ribotas resursas, kuriuo dalijasi visi tinkle esantys įrenginiai [1]. Efektyvus radijo dažnio panaudojimas priklauso nuo sistemos architektūros bei individualių jutiklių parametrų (signalų siuntimo galia, ryšio pralaidumas ir kt.).

Neseniai žaidimų teorija pradėta naudoti sunaudojamų tinklo resursų kiekio optimizacijai paskirstytuose kompiuterių tinkluose. Tuos pačius principus galime pritaikyti ir bevielių jutiklių tinklams. Norint pritaikyti lošimų teoriją bevielių jutiklių tinklų optimizacijos problemoms spręsti, būtina susieti anksčiau aptartus lošimų teorijos aspektus su BJT elementais. Taigi, lošėjais vadinsime tinklo mazgus. Strategijomis vadinsime jutiklių loginius veiksmus, kurie gali būti priimami, pavyzdžiui signalo siuntimo galios keitimas, savo duomenų siuntimas arbo kito jutiklio duomenų persiuntimas, duomenų paketo sunaikinimas ir kiti. Veiksmingumo funkcijos modelyje atspindės gaunamą paslaugų kokybės vertę (angl. *QoS*), pasirinkus tam tikrą strategiją. Išlošis turi teigiamai veikti tinklo charakteristikas, tokias kaip SINR [1], pralaidumas, radijo spektro išnaudojimas ir kitas. Išlošis yra kiekybinė veiksmingumo funkcijos išraiška, atsižvelgiant į paslaugos kokybės užtikrinimą.

Vienas iš iššūkių lošimų teorijos pritaikyme BJT - teisingų veiksmingumo funkcijų apibrėžimas jutiklių logikoje. Kadangi jutikliai iš prigimties yra savanaudiški lošėjai, reikia taikyti papildomas priemones, kad jų pasirenkami veiksmai pagerintų sistemos veikimą, o ne jį apsunkintų. Kaip pavyzdys gali būti duomenų paketo išsiuntimo svarba. Jei duomenų svarba yra didelė, jutikliui nusprendus jų nepersiųsti, sistema gali prarasti reikalingą informaciją.

Jutiklių tinklo mazgai turi ribotą energijos kiekį. Lošimų teorijos perspektyvoje, energija žymi veiksmų *kainą*. Kiekvienas duomenų perdavimas kainuoja tam tikrą kiekį energijos, o tai įtakoja lošėjų pasirenkamas strategijas. Modeliuojant žaidimą, reikia taikyti optimalią kainų parametrizaciją, norint pasiekti adekvačias jutiklių reakcijas į aplinkos veiksnius. Tai

reiškia, kad kainos turi keistis dinamiškai pagal esamą situaciją. Kaip pavyzdį galima pateikti vėliau darbe nagrinėjamą situaciją, kai lošėjai su mažesniu energijos likučiu moka didesnę kainą, nei lošėjai su pilnomis baterijomis.

Duomenų persiuntimas nuo siuntėjo iki adresato turi kooperacinio lošimo bruožų - jutikliai turi bendradarbiauti, siekdami bendro tikslo. Gyvybingų kaimyninių jutiklių palaikymas individualiam jutikliui reiškia tai, jog jis gaus daugiau žinučių ir galės užsidirbti didesnį balansą, jas persiųsdamas. Todėl jutikliui svarbu gauti kuo daugiau informacijos apie kaimyninius jutiklius, o konkrečiai - jų likutinį energijos lygį. Duomenys apie aplinkinius jutiklius gaunami kartu su iš jų atsiunčiamomis žinutėmis. Taip kiekvienas jutiklis yra informuojamas apie kitų lošėjų padėtį žaidime, todėl kuriamame modelyje duomenų persiuntimo lošimą galima laikyti pilnos informacijos lošimu.

3.5. Kredito ir reputacijos mechanizmai

Kaip anksčiau aptarėme, jutiklius palikus savo valiai, jie elgiasi savanaudiškai, bandydami užimti kuo didesnę ryšio spektro dalį bei išsiųsti kuo daugiau savo duomenų. Belaidžių jutiklių baterijos ištekliai yra labai vertingi, todėl jutikliai yra labai stipriai motyvuoti nepersiųsti kitų jutiklių duomenų. Toks jutiklių savanaudiškas elgesys trikdo bendrą sistemos veikimą. Todėl, norint priversti jutiklius bendradarbiauti, svarbu tinkle įdiegti gero elgesio iniciatyvos mechanizmą, kuris verstų jutiklius bendradarbiauti ir persiųsti duomenų paketus.

Šie metodai literatūroje yra dažniausiai skaidomi į dvi grupes [4]:

1. Reputacijos mechanizmai.

2. Kredito mechanizmai.

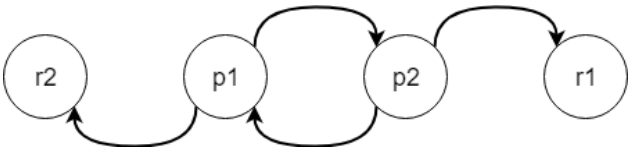
Reputacijos mechanizmai paremti atmintimi apie kaimyninių jutiklių elgesį. Sistemos mazgai prisimena savo kaimynų elgesį. Jei šie kooperuoja, jų reputacija kyla, jutikliai noriai su jais bendradarbiauja. Tačiau tie jutikliai, kurie visada elgiasi savanaudiškai, susikuria blogą reputaciją, kaimynai nebepersiunčia šių jutiklių duomenų, kol, galiausiai, šie yra izoliuojami nuo likusio tinklo. Tokia schema reikalauja, kad jutikliai prisimintų didesnį informacijos kiekį apie savo kaimynus, bei implikuoja didesnį koordinacijos laipsnį, nei kredito schema. *Kredito mechanizmai* naudoja kreditus kaip atlygį už gerą elgesį. Kiekvienas jutiklis gauna užmokestį už atliktą darbą - žinutės perdavimą, bei gali mokėti kitiems jutikliams už savo duomenų paketų perdavimą. Taip jutikliai gali naudoti savo sukauptus kreditus duomenims persiųsti. Kadangi šio tipo schema nereikalauja įsikišimo iš išorės, ją pasirinkome taikyti šiame darbe. Šiek tiek modifikuotoje kredito schemoje, kiekvienas jutiklis už bendradarbiavimą gauna kreditus, o už savanaudišką elgesį juos turi sumokėti. Jutikliai bando gauti kuo daugiau kreditų. Taip tinkle yra skatinamas bendradarbiavimas tarp mazgų.

Šiame darbe toliau naudosime kredito schemą, kadangi ji lengviau įgyvendimama ir yra lankstesnė nei reputacijos schema.

3.6. Lošimai bevielių jutiklių tinkle

3.6.1. Persiuntimo dilemos lošimas

Kadangi daugumoje belaidžių jutiklių tinklo situacijų žaidėjai (šiuo atveju -jutikliai), turi dalintis bendrą išteklių (radijo bangų dažnį), šis darbas orientuotas į lošimų, kuriuose žaidėjai nebendradarbiauja, teoriją. Bendradarbiavimo lošimai reikalauja papildomo informacijos apsikeitimo bei susitarimų tarp žaidėjų, tad sumodeliuoti tokią sistemą yra daug sudėtingiau.



9 pav. Persiuntimo dilema

Pirmas mūsų modeliuojamas lošimas bus vadinamas "Persiuntimo Dilema". Priimsime, jog lošime dalyvauja du jutikliai kaip žaidėjai *p1* ir *p2*. Kiekvienas iš jų nori persiųsti savo duomenų paketą gavėjui, atitinkamai *r1* ir *r2* (9 pav.). Laikas yra suskirstytas lygaus ilgio intervalais. Per vieną laiko intervalą žaidėjas gali atlikti tik vieną veiksmą. Tam, kad gavėjas *r1* gautų duomenų paketą, jį turės persiųsti jutiklis *p2*, ir atvirkščiai, gavėjas *r2* duomenis gaus iš jutiklio *p1*.

Lošimas turi turėti atlygį laimėtojui. Jį pažymėsime skaičiumi 1. Jei žaidėjas *p1* persiunčia *p2* paketą, tai kainuoja žaidėjui *p1* kainą c, $0 < c < 1$. Kaina šiuo atveju yra atlikti skaičiavimai bei energija, suvartota duomenų siuntimo metu. Persiųsdamas duomenis, *p2* sukuria duomenų maršrutą tarp *p2* ir *r2*, o tai suteikia *p2* atlygį, vertą 1. Ėjimo pelningumas - tai atlygio ir kainos skirtumas. Darome prielaidą, kad lošimas yra simetriškas ir tos pačios taisyklės galioja žaidėjui *p2*. Šio lošimo dilema tokia: Kiekvienam žaidėjui yra naudinga ištrinti duomenų paketą, kurį jis turėtų persiųsti, kadangi tai išsaugotų dalį jo baterijos energijos. Tačiau jei kitas žaidėjas priims tokį patį sprendimą, žaidėjo *p1* paketas, kurį šis norėtų perduoti gavėjui, butų taip pat ištrintas. Abu žaidėjai galėtų išlošti, jei bendradarbiautų ir perduotų vienas kito siunčiamus duomenis. Taigi, atsiranda dilema.

Kaip minėjome anksčiau, kiekvienas ėjimas užima visą laiko intervalą. Kiekvienas žaidėjas turi pasirinkti, perduoti (F) duomenis kitam žaidėjui, ar juos ištrinti (D). Šis sprendimas apibrėžia žaidėjo strategiją. Jei abu žaidėjai prasiųstų vienas kito duomenis, abu gautų pelną, kitaip nei abiems sunaikinus duomenų paketus. Šias strategijas apibrėšime matrica (10 pav.).

p1\p2	F	D
F	(1-c,1-c)	(-c,1)
D	(1,-c)	(0,0)

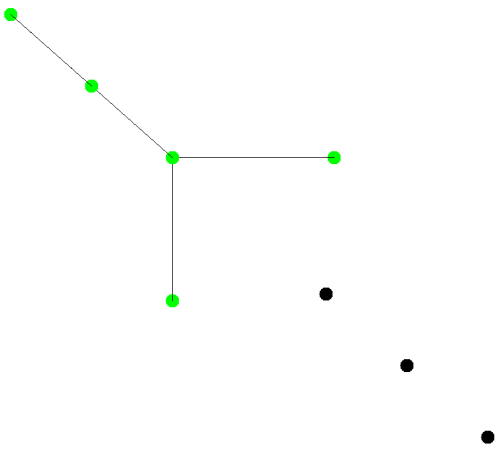
10 pav. Persiuntimo dilema strategine forma. *p1* - eilučių žaidėjas, *p2* - stulpelių. Kiekvienas žaidėjas turi po dvi strategijas: perduoti (F) arba atmesti (D) kito siųstus duomenis. Kiekviename lauke pirma vertė - *p1* išlošis, antra - *p2* išlošis.

Matricoje žaidėjo *p1* ėjimai yra eilutėse, *p2* - stulpeliuose. Kiekvienas matricos laukelis žymi galimą abiejų žaidėjų strategijų kombinaciją. Laukelių viduje yra atitinkamų žaidėjų veiksmų pelningumo išraiškos. Šį lošimą galima išspręsti griežto dominavimo metodu [15]. Jei bandytume išspręsti lošimo dilemą žaidėjo *p1* požiūriu, *D* strategija visada pelningesnė už *F* strategiją. Šiuo atveju tai *dominuojanti* (vyraujanti) strategija [15]. Tai reiškia, jog galima eliminuoti pirmąją matricos eilutę, nes racionalus žaidėjas niekada nepasirinks šios strategijos. Analogiškai, žaidėjo *p2* požiūriu, reikia eliminuoti pirmąjį matricos stulpelį. Taip gauname lošimo sprendimą - (*D,D*) ir jo pelno vertes (*0,0*). Vadinasi, šie du pasirinkimai vyrauja, jie nustelbia alternatyvas - taigi yra vyraujančių strategijų pusiausvyra.

Kadangi lošime kiekvienam iš žaidėjų egzistuoja vyraujanti strategija, galime numatyti, jog būtent šiuo pusiausvyros susidarymu lošimas ir baigsis. Taip yra todėl, kad ši lošimo baigtis yra Nešo pusiausvyra - nei vienas lošėjas negaus didesnio atlygio, pakeitęs savo strategiją, jei jos nekeis priešininkas. Nors strategijos (*F,F*) veda prie pelningesnio rezultato abiems žaidėjams, jų tarpusavio nepasitikėjimas veda prie nenaudingos lošimo pabaigos. Kadangi šis lošimas yra simetrinis, be išorinės motyvacinės schemos įsikišimo jis yra neveiksmingas optimizuojant BJT. Dėl to praktiniame modelyje šio lošimo patobulinimui naudojama kredito motyvacinė schema, aptarta skyriuje 3.4.

3.6.2. Ryšio kanalo lošimas

Kitas mūsų nagrinėjamas lošimas supažindina su radijo dažnio užimtumo problema. Tarkime, kad lošime dalyvauja du žaidėjai - *p1* ir *p2*. Lošėjai tuo pačiu metu nori išsiųsti duomenų paketus bendram gavėjui *r*. Abu lošėjai yra vienodu atstumu nuo gavėjo, o taip pat labai arti vienas kito, todėl atsiranda radijo bangų interferencija. Tai galima patikrinti, panaudojant sukurtą BJT simuliatorių. Išdėstome jutiklius taip, kad dešinėje esantis gavėjas galėtų gauti duomenis iš dviejų siuntėjų, esančių šalia vienas kito. Jiems lygiagrečiai siunčiant žinutę, atsiranda duomenų kolizija, paketas sugadinamas ir gavėjas negauna duomenų.



11 pav. Ryšio kanalo lošimo simuliacija

Darome prielaidą, kad kiekvienu laiko intervalu abu žaidėjai sprendžia, ar siųsti duomenų paketą. Kaip ir praeitame lošime, duomenų paketo siuntimas kainuoja $0 < c \ll 1$. *p1* sėkmingai perduos duomenis *r*, jei *p2* nesiųs duomenų, kitaip atsiras duomenų paketų kolizija. Jei duomenys sėkmingai perduodami, *p1* gauna išlošį, vertą 1.

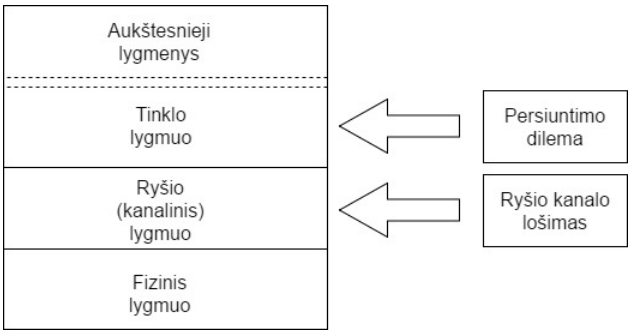
Kiekvienas lošėjas turi dvi galimas strategijas: siųsti duomenis (T) arba nesiųsti duomenų

(Q). Kadangi radijo dažnis bendras, abiems siunčiant duomenis, atsiranda paketų kolizija. Žaidimo strateginė forma matoma 12 pav.

p1/p2	Q	T
Q	(0,0)	(0, 1-c)
T	(1-c, 0)	(-c, -c)

12 pav. Ryšio kanalo lošimas strategine forma. *p1* - eilučių žaidėjas, *p2* - stulpelių. Kiekvienas žaidėjas turi dvi strategijas: siųsti (T) arba nesiųsti (Q) duomenų. Kiekviename lauke pirma vertė - *p1* išlošis, antra - *p2* išlošis.

Iškart galima pastebėti, jog šiame lošime nėra dominuojančios strategijos. Pagal anksčiau aptartus kriterijus, randame Nešo pusiausvyros taškus: *(Q,T)* ir *(T,Q)*. Jie ir yra šio lošimo sprendimai.



13 pav. Lošimai OSI sluoksniuose.

4. Literatūros apžvalga

Belaidžių jutiklių panaudojimo galimybės yra plačios - pagal [18], tai transporto priemonių sekimas, industrinių įrenginių stebėsena. Šaltinyje [11] taip pat minimos karinės technologijos, gamtosauga, energetikos sektorius. Dėl šio plataus panaudojimo spektro, ši sritis pritraukia vis daugiau akademinės bendruomenės dėmesio. Šiame skyriuje trumpai apžvelgsime mokslinius straipsnius, tematiškai susijusius su su šiuo magistro baigiamuoju darbu.

Egzistuojančioje su belaidžių jutiklių tinklų modeliavimu susijusioje mokslinėje literatūroje yra nagrinėjamos maršrutizavimo problemos, ieškoma būdų, kaip sutaupyti jutiklių baterijų energiją, taip pat kiti tinklo veiklos aspektai. Q. Wang ir kitų darbe bandyta sukurti realistišką energetinį belaidžių jutiklių sistemos modelį [17]. Autoriai nagrinėjo atskirų jutiklio sudedamųjų komponentų poveikį bendroms jutiklio energijos sąnaudoms, bei pasiūlė tinklo modelį, kuriame atsižvelgiama į kiekvieno jutiklio sandarą. Šaltinyje [2] taip pat analizuojami energetiniai belaidžių jutiklių sistemų rodikliai, o konkrečiai - heterogeniniuose tinkluose susidarančių spiečių efektus bendram energijos suvartojimui. Buvo pateiktos optimalaus spiečių skaičiaus nustatymo gairės.

Belaidžių jutiklių tinklai dažniausiai sudaryti tinklelio topologijos (angl. *mesh network*) pagrindu. Optimalaus duomenų paketų maršrutizavimo algoritmo parinkimas yra labai svarbus efektyviai tinklo veiklai. Maršrutizavimo algoritmai apžvelgiami šaltinyje [4]. Jame išskiriami populiariausių algoritmų privalumai ir trūkumai, atsižvelgiama į energetinį efektyvumą.

Šiame darbe belaidžių jutiklių tarpusavio ryšiams modeliuoti pasitelkiama lošimų teorija. Ši matematikos šaka dažniausiai naudojama ekonomikoje, tačiau ji vis plačiau pritaikoma kompiuterių tinkluose. Neseniai lošimų teorija pradėta taikyti ir belaidėse kompiuterių technologijose. Šaltinyje [13] aprašoma, kaip įvairūs sistemos ryšiai dinaminiuose (angl. *ad hoc*) belaidžiuose tinkluose gali būti modeliuojami kaip lošimai, bet pateikiamos tik teorinės tokio modeliavimo galimybės. Lošimų teorijos taikymas belaidžiuose tinkluose taip pat nagrinėjamas šaltinyje [5]. Šiame darbe lošimų teorija taikoma maršrutizavimo optimizavimui, parenkant optimalų duomenų persiuntimo kelią. Darbe kuriamas maršrutizavimo algoritmas, kurio esmė yra Nešo pusiausvyros skaičiavimas, pasirenkant duomenų siuntimo taikinius.

Vienas iš artimiau su atliekamo darbo tema susijusių yra slovėnijoje atliktas tyrimas [10]. Šiame darbe mokslininkai bandė pritaikyti žaidimų teoriją tinklo ryšio pralaidumui pagerinti. Kiekvienam tinklo mazgui skaičiuojamas optimalus signalo stipris, naudojant potencinių žaidimų principus. Darbo autoriai kūrė bangų transliavimo modelį, kuriame, keičiant transliavimo stiprį, bandoma rasti Pareto optimalų Nešo ekvilibriumą. Darbe atliekamas kartotinio lošimo bandymas, kurio metu iš dalies sėkmingai randami tinklo optimalaus pralaidumo parametrai.

Lošimų teorija *Ad hoc* tinkluose taikoma ir darbe [3]. Šiame straipsnyje teoriškai nagrinėjamos galimybės pritaikyti lošimų teoriją MANET (angl. *Mobile ad hoc network*) tinkluose. Autoriai siūlo modelį, kuris gali būti taikomas jutiklių, esančių *Ad hoc* tinkle, MAC sluoksnyje. Pateikiami ir pagrindžiami būdai, kaip pritaikti lošimų teoriją, siekiant tolygaus radijo ryšio pasidalijimo tarp jutiklių. Autoriai pritaiko kooperacinį ir nekooperacinį lošimus, pateikiami matematiniai įrodymai šių lošimų sprendimams. Darbe taip pat yra trumpa tyrimų dalis, kurioje parodomas pasiūlyto modelio efektyvumas.

Belaidžių jutiklių tinklų energetinis efektyvumas yra yra tiriamas [19] straipsnyje. Šiame darbe autoriai pristato algoritmą, paremtą jutiklių grupavimu į spiečius (andl. *cluster*). Spiečiai išsirenka savo lyderius pagal likusios energijos kiekį. Spiečių energijos lygis yra sekamas, siekiant sumažinti komunikacijos energijos nuostolius. Jutiklių radijo signalų stipris taip pat yra koreguojamas, siekiant energetinio efektyvumo. Autoriai vadina savo algoritmą "HEED". Eksperimentinėje dalyje parodoma, kaip "HEED" pritaikymas sistemose gali pratęsti jų gyvavimo trukmę.

Energetinį BJT optimalumą taip pat tyrinėjo M. Cardei, M. T. Thai ir kiti savo straipsnyje "Energy-efficient target coverage in wireless sensor networks" [7]. Autoriai siūlo išrinkti iš jutiklių aibės tam tikrą jų skaičių ir juos aktyvinti, o likusius laikyti miego režime. Aktyvių jutiklių parinkimas atliekamas, naudojant tiesinį programavimą. Darbe pateikiami simuliacijos rezultatai parodo šio metodo efektyvumą.

Darbo [9] autoriai, bandė pritaikyti lošimų teoriją BJT kibernetinio saugumo srityje. Principas, pasirinktas lošimų teorijos taikymui, panašus į mūsų pasirinktą šiame darbe. Jie taiko lošimų teoriją, modeliuodami komunikacijas tarp tinklo jutiklių. Šiame modelyje, individualaus jutiklio ir atakuojančio elemento sąveika nagrinėjama kaip dviejų lošėjų nekooperacinis lošimas. Deja, nors autoriai teigia, jog jų siūlomas atakos aptikimo modelis yra veiksmingas, darbe neatliekami eksperimentai, patvirtinantys arba paneigiantys jų išvadas.

5. Tinklo modeliavimas

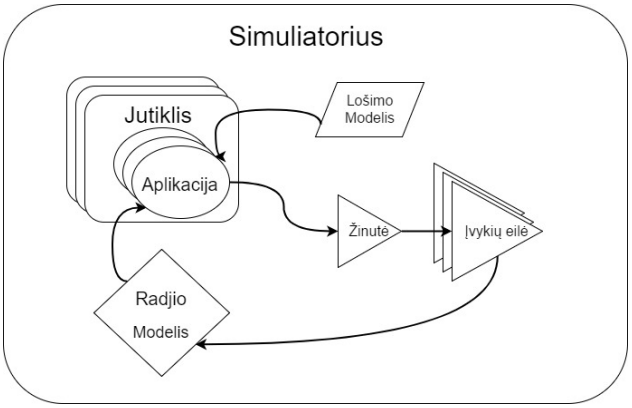
Šiame skyriuje aptarsime kuriamą bevielių jutiklių tinklų modelį. Jis yra rašomas Java programavimo kalba. Modelyje buvo pasiremta bevielių jutiklių tinklo simuliavimo įrankiu "Probabilistic Wireless Network Simulator", sukurtu "Vanderbilt" universitete, "Institute for Software Integrated Systems" institute. Jis leido sukurti pasirenkamą kiekį belaidžių jutiklių trimatėje erdvėje ir simuliuoti radijo ryšius tarp jų. Šiame darbe simuliatorius stipriai modifikuojamas ir patobulinamas. Implementuojami maršrutizavimo algoritmai, modelio grafinio atvaizdavimo galimybės, kuriami jutiklių loginiai komponentai, kuriuose pritaikoma lošimų teorija, bei atliekami kiti pakeitimai.

Modelis veikia įvykių eilės principu. Tai reiškia, jog simuliatoriuje žinutės gali būti siunčiamos asinchroniškai. Simuliacijoje jutiklius galima išdėstyti atsitiktine tvarka arba paskirti jiems pasirinktas koordinates. Jutiklių žinučių siuntimo dažnis taip pat gali būti keičiamas.

Tinklo simuliatorius modeliuoja visų tinklo komunikacijų ir aplikacijos lygmenų svarbiausius aspektus. Modeliuojant radijo ryšį, atsižvelgiama į bangų stiprį bei trikdžių dydį. Supaprastintas MAC (angl. *Medium Access Control*) modelis bendrauja su aplikacijos lygmeniu per įvykių eilę.

Kuriamas bevielių jutiklių modelis susideda iš šių pagrindinių komponentų:

- **Simuliatorius** - bazinė klasė, per kurią paleidžiama/stabdoma simuliacija. Simuliatorius inicijuoja sistemos įvykių apdorojimą. Simuliatoriuje laikomi visi jutiklių objektai, radijo modelis, įvykių eilė.
- **Radijo modelis** - Simuliuoja radijo bangų sklidimą erdvėje. Žino apie visų jutiklių lokacijas bei radijo ryšius tarp jų. Sprendžia, kurios žinutės pasiekia jutiklius.
- **Jutiklis** - tai bazinis simuliacijos mazgas, turintis savo aplikacijos objektą, per kurį jis atlieka visą užprogramuotą logiką.
- **Aplikacija** - jutiklio programa, kuri valdo duomenų apsikeitimo veiksmus, naudodama lošimų teorijos logiką.

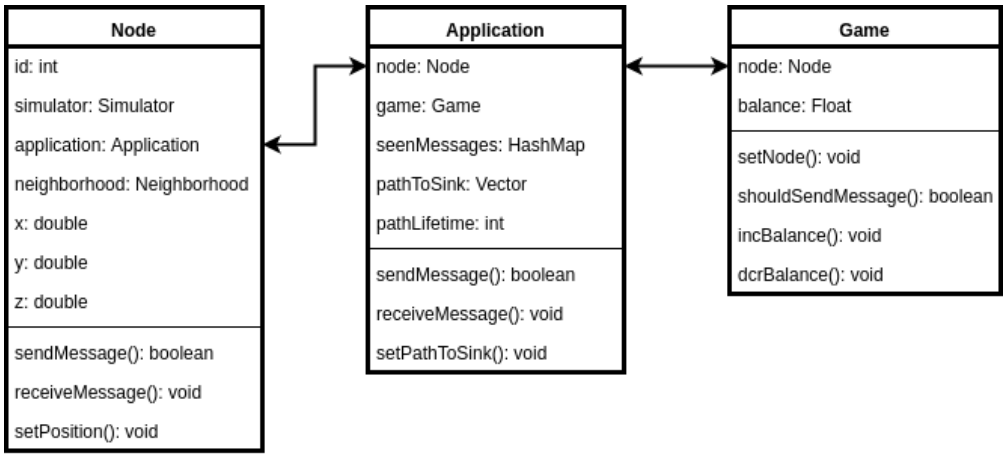


14 pav. Simuliatoriaus modelio schema

Šiame skyriuje aptariama pagrindinių simuliatoriaus komponentų veikimo principai. Toliau išsamiau paaiškinama kiekvieno šių komponentų struktūra ir tarpusavio ryšiai.

5.1. Jutiklis

Mazgas (*Node*) yra jutiklio reprezentacinis objektas simuliacijoje. Jis gali būti pozicionuojamas trimatėje erdvėje pagal nustatomas koordinates. Pagal šias koordinates yra nustatoma, kurie jutikliai gali komunikuoti tarpusavyje pagal 5.2 išraišką. Jutikliai gali būti stacionarūs, arba jų koordinatės gali būti kintančios [12]. Pastaruoju atveju, sistema veikia lėčiau, nes po pozicionavimo turi būti iš naujo inicializuojamos jutiklių kaimynystės grupės.



15 pav. Jutiklio objekto UML schema.

Kiekvienas tinklo mazgas yra surištas su kaimyniniais jutikliais per *Neighborhood* klasės objektą. Simuliacijos pradžioje kiekvienam jutikliui yra nustatoma jo kaimyninių jutiklių grupė pagal atstumą tarp jų. Į kaimynystę patenka arčiausiai jutikliui erdvėje esantys mazgai.

Jutiklio loginis elementas yra jo aplikacija, kuri atitinka procesoriaus komponentą realiuose jutikliuose. Kiekvienas jutiklis turi savo aplikaciją. Sukūrūs visus reikiamus jutiklio objektur, jie yra patalpinami į simuliacijos objektą, kuris yra modelio kertinis akmuo, surišantis visus komponentus.

Aplikacijoje užkoduojama žinučių apsikeitimo logika. Kiekvienas jutiklis prieš siųsdamas žinutę, patikrina, ar radijo kanalas yra laisvas. Jeigu taip, žinutė persiunčiama per radijo modelį artimiausiems jutikliams. Jeigu radijo kanalas užimtas, jutiklis palaukia nustatytą laiko tarpą ir bando dar kartą.

Jutiklio aplikacijoje gyvuoja lošimo objektas (*Game*). Ši klasė turi metodus balanso valdymui bei veiksmų strategijos apskaičiavimui. Jutikliui gavus naują žinutę, kviečiamas šios klasės metodas, kuris apskaičiuoja, ar jutiklis turi persiųsti žinutę.

5.2. Įvykių eilė

Žinutės modelyje yra perduodamos per asinchronišką įvykių eilę. Tai įvykdoma, kuriant įvykio objektus (*Event*) ir dedant juos į eilę. Atėjus numatytam įvykio laikui, jis yra įvykdomas ir pašalinamas iš eilės. Žinutės siuntimo tvarka yra tokia:

1. Tikrinamas ryšio kanalas. Jei jutiklis šiuo metu gauna duomenis, žinutės siuntimas atidedamas.

2. Kai ryšio kanalas laisvas, kuriamas siuntimo pradžios įvykis ir dedamas į įvykių eilę su jam skirtu įvykdymo laiku.

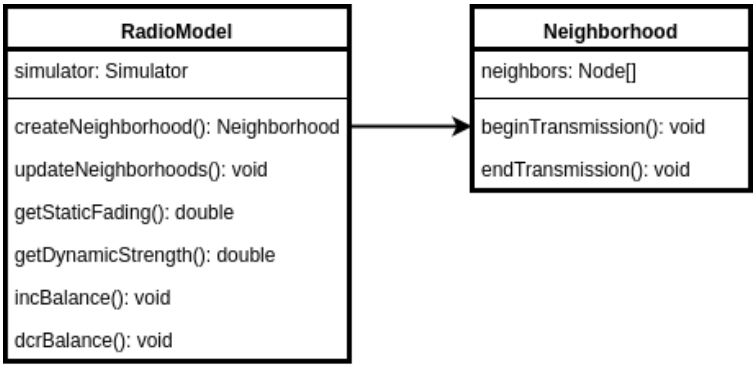
3. Atėjus įvykio laikui, jis išimamas iš eilės ir įvykdomas. Šiame žingsnyje generuojamas radijo signalo triukšmas, kuris gali sutrikdyti kitų žinučių siuntimą.

4. Pasibaigus nustatytam žinutės siuntimo laikui, į įvykių eilę dedamas žinutės siuntimo pabaigos įvykis.

5. Atėjus žinutės siuntimo pabaigos laikui, pažymimas sėkmingas žinutės išsiuntimas, žinutė perduodama gavėjui.

5.3. Radijo bangų modelis

Simuliatoriuje įgyvendintas radijo modelis leidžia persiųsti informaciją tarp sistemos mazgų, o taip pat simuliuoti duomenų paketų kolizijas. Jei du ar daugiau arti esančių mazgų tuo pačiu laiko intervalu siunčia duomenis, yra tikimybė, jog šie bus sugadinti, o informacija prarasta.



16 pav. Radijo modelio objekto UML schema.

Kiekvieno žinutės siuntimo įvykio pradžioje radijo bangų sklidimo modelis apskaičiuoja kiekvieno sistemoje esančio jutiklio siunčiamo signalo stiprumą. Remiantis šia informacija yra apskaičiuojamos duomenų gavimo sąlygos ir randamos duomenų kolizijos.

5.3.1. Nustatymai

Atskiri tinklo mazgai gali būti sukonfigūruoti turėti specifinius veikimo parametrus. Šie parametrai naudojami programos veikimo metu signalų interferencijoms apskaičiuoti. Toliau pateikiami šiame darbe naudojami jutiklių nustatymai, apibrėžiantys žinučių siuntimo trukmę ir kitas tinklo charakteristikas:

- Minimalus laukimo laikas prieš siunčiant: 128 ms - tai laiko intervalas, kurį jutiklis palaukia, gavęs užklausą siųsti žinutę.
- Minimalus atsitraukimo po nesėkmingo siuntimo laikas: 100 ms - tai laiko intervalas, kurį jutiklis lauks, kol atsilaisvins užimtas radijo kanalas.
- Žinutės siuntimo laikas: 100ms - tiek jutiklis užtrunka, siųsdamas žinutę.
- Leistinas triukšmo koeficientas: 2.0 - tai reiškia, jog jutiklis gaus žinutę, jei signalo stipris viršys triukšmo lygį du kartus.

5.3.2. Modelis

Signalo stipris tarp siuntėjo ir gavėjo gali būti nustatomas, pasitelkiant signalo silpnėjimo erdvėje (pagal atstumą ir signalo stiprį) funkciją 5.1.

$$P_{gav.ideal.} = P_{siunt}\frac{1}{1+d^{\gamma}} \tag{5.1}$$

kur $P_{gav.ideal.}$ yra gavėjo imtuvą pasiekiančio signalo stipris idealiomis sąlygomis,
P_{siunt} - siunčiamo signalo stipris,
d yra atstumas tarp siuntėjo ir gavėjo, γ yra nykimo parametras, kuris tipiškai turi reikšmes $2 \leq \gamma \leq 4$.

Tačiau, tikrovėje, signalai susiduria su įvairiomis kliūtimis, kurios simuliatoriuje modeliuojamos, į 5.1 įvedant atsitiktinius trikdžių parametrus. Siuntėjui i siunčiant žinutę gavėjui j, gauto signalo stipris $P_gav.$ skaičiuojamas taip:

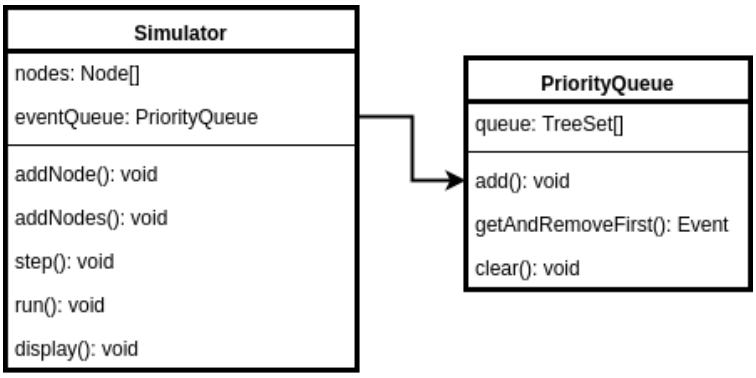
$$P_{gav.} = P_{gav.ideal.}(d_{ij}) \times [1 - \alpha(d_{ij})] \times [1 - \beta(t)] \tag{5.2}$$

Kintamasis α priklauso tik nuo atstumo tarp jutiklių, todėl simuliatoriuje jis skaičiuojamas tik pasikeitus šiam atstumui. β priklauso nuo žinutės siuntimo laiko, todėl jo vertė perskaičiuojama kiekvieno siuntimo pradžioje.

Signalai sėkmingai perduodami, kai gauto signalo stipris yra didesnis už trikdžių stiprį. Jutikliai negali vienu metu gauti dviejų signalų. Jei vienu metu gaunami du ir daugiau signalų, atsiranda signalų interferencija ir duomenys gali būti sugadinami.

5.4. Simuliatorius

Simuliatoriaus klasė *Simulator* yra viena pagrindinių šiame darbe. Ji atsakinga už sąryšį tarp radijo modelio, jutiklių ir įvykių eilės. Simuliatoriaus objektas laiko nuorodas į visus modelio jutiklius. Per šios klasės metodą paleidžiamas modelio veikimas. Taip pat simuliatoriaus klasė turi metodą, skirtą BJT veikimo atvaizdavimui.



17 pav. Simuliatoriaus ir įvykių eilės UML schema.

6. Tinklo optimizacija

Šiame skyriuje aptariama atlikta bevielių jutiklių tinklo optimizacija. Sudaromas tinklo modelis bandymams, pritaikomi skirtingi maršrutizavimo algoritmai, lošimų teorijos sprendimai bei kitos optimizacijos, bei pamatuojami ir pateikiami rezultatai.

6.1. Pradinės sąlygos

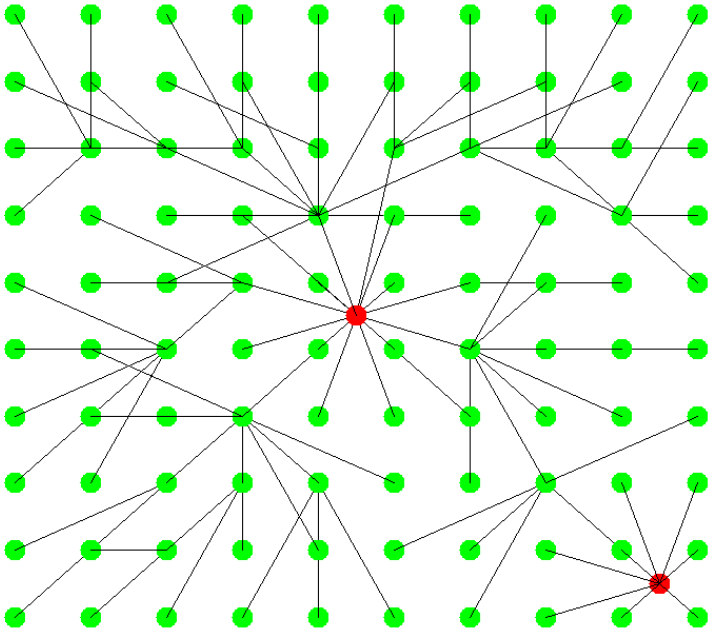
Norėdami išbandyti ilgalaikį jutiklių darbo režimą, panaudojame modelį daugkartiniam žinučių siuntimui. Siuntėjas t tam tikru kintančiu dažniu siunčia žinutes gavėjui r. Iš viso bandoma persiųsti 1000 žinučių. Kiekvienas žinutės išsiuntimas kainuoja ją siunčiančiam jutikliui kainą c. Jutikliai turi pradinį įkrovos lygį, žymimą E, kur $E = 100 \cdot c$. Taigi, idealiomis sąlygomis kiekvienas jutiklis gali išsiųsti 100 žinučių prieš išsikraudamas. Mazgų t ir r energijos resursus laikysime nesibaigiančiais.

2 lentelė. Bandymų pradinės sąlygos

Sąlyga	**Dydis**
Plotas	100x100 m
Siuntėjai/Gavėjai	2
Jutikliai	100
Jutiklio baterijos talpa	1000 mA·h
Žinutės siuntimo kaina	10 mA·h
Siunčiama žinučių	1000

6.2. Jutiklių išdėstymas

Simuliacijoje sudarytas bevielių jutiklių tinklas iš 102 mazgų. 100x100m plote kas 10m atstumu išdėstyti 100 standartinių jutiklių. Šie jutikliai tarnaus kaip agentai duomenų persiuntimui. Duomenis siunčiantis mazgas, kurį žymėsime t, padedamas simuliuojamo ploto viduryje. Jis bandys perduoti duomenis iki duomenų gavėjo, žymimo r, kuris yra padedamas dešiniąjame apatiniame simuliacijos ploto kampe. Simuliacijos išbandymui, išsiunčiama viena žinutė iš siuntėjo t. Rezultatai matomi 18 pav.



18 pav. Jutiklių išdėstymas simuliacijoje. Centre esantis jutiklis perduoda duomenis dešinėje apačioje esančiam gavėjui. Siuntėjo *t* išsiųsta žinutė pasiekia jos galutinį gavėją *r*, o taip pat ir visus kitus modelyje esančius mazgus.

Bandymų metu matuojame, kiek žinučių, išsiųstų iš siuntėjo t yra sėkmingai perduodama iki gavėjo r. Bandymas baigiasi, kai tarpiniai mazgai išsikrauna ir nebeperduoda žinučių iki mazgo r.

6.3. Tvindymo maršrutizavimas

Yra daug maršrutizavimo protokolų, pritaikytų bevielių jutiklių tinklams [4]. Paprasčiausias įgyvendinimui - tai "tvindymo" (angl. *flooding*) protokolas. Jo esmė yra, kad kiekvienas jutiklis, gavęs žinutę, ją iš karto persiunčia kitiems aplink esantiems jutikliams. Tačiau įgyvendinus tokį modelį ir jį išbandžius, greitai paaiškėja, jog jis praktiškai netinkamas naudojimui. Kadangi jutikliai gali gauti tą pačią žinutę iš kelių šaltinių, pasikartojančios žinutės yra publikuojamos ir persiunčiamos daug kartų tarp tų pačių jutiklių, pirmyn ir atgal, efektyviai užkemšant radijo kanalą, ir nebeleidžiant siųsti naujos informacijos. Bandymų metu nuo pirminio siuntėjo iki galutinio gavėjo pavyksta persiųsti tik kelias dešimtis žinučių, kol sistemoje įsivyrauja chaosas.

1 algoritmas. Gautos žinutės apdorojimas

input : message
if *žinutė jau buvo gauta* **then**
  return
else
  žinutė įsimenama
end
žinutė išsiunčiama kaimynams

Tinklas greitai užsikemša, nes siuntėjai aido principu gauna savo žinutes, ir jas pakartotinai siunčia, taip švaistydami radijo dažnio ir energetinius resursus. Norint naudoti "tvindymo" maršrutizavimą optimalesniam modeliavimui, reikia jį patobulinti. Geresnis maršrutizavimo algoritmas atrodo taip:

1. Jutiklis gauna duomenų paketą. Jis jį išsaugo savo atmintyje ateičiai ir persiunčia kitiems.

2. Jutiklio kaimynai gauna persiųstus duomenis ir patys juos siunčia toliau.

3. Jutiklis vėl gauna duomenų paketą, galimai tą patį, kurį ką tik siuntė. Jei tai tas pats paketas, kaip prieš tai išsaugotas - jutiklis jo nebepersiunčia.

Atlikus bandymus su minėtu modeliu, gauti šie **rezultatai**:

- Prasiunčiama vidutiniškai 100 žinučių.
- Išsikrauna visi jutikliai.

Simuliacijos metu persiunčiama tik 1/10 visų išsiųstų duomenų, galima teigti, jog tinklo veikimas nėra optimalus, ir jį būtina pagerinti. Kadangi kiekvienas jutiklis išsiunčia kiekvieną gautą naują žinutę, visi jutikliai yra iškraunami.

6.4. Dinaminis maršrutizavimas

Norėdami pagerinti tinklo veikimo charakteristikas, turime naudoti optimalesnį maršrutizavimo būdą. Kadangi modeliuojamas tinklas veikia asinchroniškai, t.y. nesiunčiami žinutės gavimo patvirtinimo signalai, buvo pasirinktas dinaminis maršrutizavimas, arba DSR, aptartas skyriuje 2.2. Šio maršrutizavimo algoritmo pagrindinis principas - maršruto aptikimas prieš siunčiant duomenis.

2 algoritmas. Žinučių siuntimo algoritmas, pritaikant DSR maršrutizavimą.

```
while i <= 1000 do
    if nerastas maršrutas arba esantis maršrutas nebegalioja then
        siunčiama RREQ žinutė ir laukiama atsakymo
        gaunamas RREP atsakymas, nustatomas maršrutas iki adresato
    else
        siunčiama DATA žinutė adresatui
        i++
    end
end
```

Atlikus bandymus su DSR maršrutizavimu, gauti tokie **rezultatai**:

- Prasiunčiama vidutiniškai 500 žinučių.
- Išsikrauna 90 jutiklių.

Pritaikant šį algoritmą, iš pradžių žinučių siuntimas vyksta sklandžiai, visa siuntėjo siunčiama informacija pasiekia adresatą. Modelis efektyviai aptinka trumpiausius maršrutus iki adresato. Tačiau, išsikrovus didžiąjai daliai jutiklių, maršruto nebepavyksta rasti, taigi žinučių siuntimas sustoja.

Didžiausia problema šiame bandyme yra paprasta charakteristika - arčiausiai siuntėjo esantys jutikliai gauna daugiausiai duomenų paketų, todėl jie pirmieji ir išsikrauna. Tam, kad išspręsti šią problemą, reikia, kad siuntėjo kaimynai ne visada persiųstų gautus duomenų paketus. Tai galima įgyvendinti atsitiktiniu būdu parenkant, kada siųsti gautus duomenis, o kada ne.

Naudojant DSR maršrutizavimą, persiunčiama 1/2 visų išsiųstų žinučių. Jei BJT panaudojimo sritis reikalauja didelio efektyvumo, šio maršrutizavimo pritaikymas gali žymiai pagerinti tinklo charakteristikas.

6.5. Lošimų teorijos taikymas

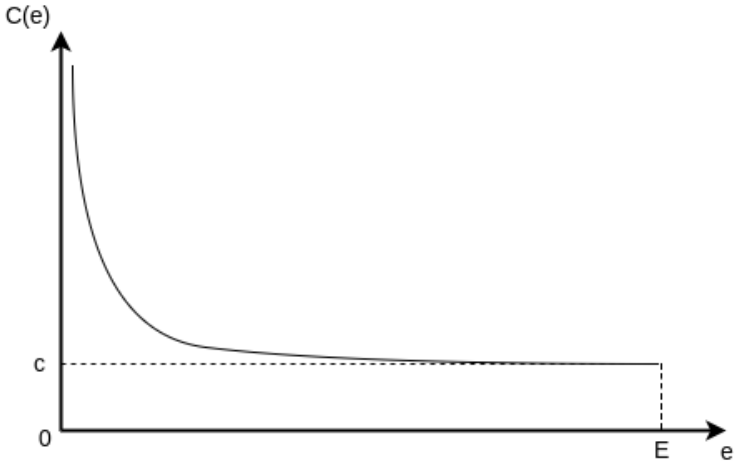
DSR maršrutizavimo bandymuose išryškėjusią artimiausių kaimynų išsikrovimo problemą galima spręsti, pasitelkiant lošimų teoriją. Jutikliams suteikiant loginių sprendimų galimybes, jie tampa savarankiški, ir gali spręsti, persiųsti gautus duomenis kitiems jutikliams, ar sutaupyti energijos kiekį kitų sąskaita. Modelyje lošimas yra dinaminio kartotinio tipo, t.y. jutikliai privalo atsižvelgti į savo ir kitų lošėjų praeitus ėjimus.

Atliekamuose bandymuose jutiklių aplikacijoms pritaikoma persiuntimo dilemos lošimo logika. Jei jutikliams būtų taikoma klasikinė lošimų teorijos logika, jutikliai niekada nepersiųstų gautų duomenų paketų, nes tai diktuoja šio lošimo Nešo pusiausvyra. Tam, kad sumažinti jutiklių savanaudiško elgesio poveikį, pritaikome tinklo mazgams motyvacinę kredito schemą.

Kiekvienas mazgas kaupia savo kredito balansą B. Jei mazgas išnaudoja visus savo kreditus, t.y. $B = 0$, jis privalo persiųsti duomenis, kad jį atstatytų. Persiųsdamas duomenis, mazgas kiekvieną kartą gauna išlošį 1. Išlošis prisideda prie mazgo balanso. Tačiau, tuo pat metu, persiųsdamas duomenis mazgas patiria energetinius nuostolius. Energetiniai nuostoliai C apskaičiuojami pagal 6.1 išraišką.

$$C = \frac{E - e}{E} + c \tag{6.1}$$

čia E - pradinis mazgo energijos lygis,
e - esamas energijos lygis, $e \leqslant E$,
c - pastovi minimali žinutės persiuntimo kaina.



19 pav. Žinutės persiuntimo kainos *C* priklausomybė nuo likusios energijos *e*.

Jutiklio sukauptas balansas turi būti atvirkščiai proporcingas jo stimului siųsti duomenis. Jutiklio gaunamas kreditas už sėkmingą žinutės persiuntimą prilyginamas minimaliai žinutės perdavimo kainai c. Taigi, strategija s, siųsti duomenis ar ne, randama taip:

$$s = \frac{1}{B} - \frac{E - e}{E} \tag{6.2}$$

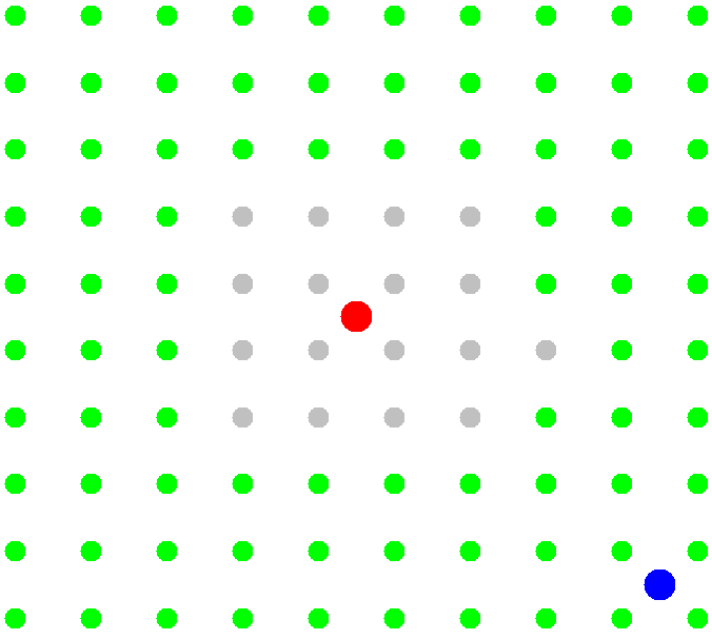
B - esamas balansas, $B \neq 0$.

Jei apskaičiuojama, jog $s > 0$, mazgas išsiunčia duomenis. Jei yra priešingai, mazgas nepersiunčia duomenų paketo. Pastaruoju atveju, mazgo balansas sumažėja -1. Įgyvendinus šią motyvacinę schemą, mazgai siunčia duomenis, kai turi daug energijos, arba kai jų balansas yra mažas. Kitais atvejais jie pereina į miego režimą ir taupo energiją.

Pritaikius lošimų teoriją tvindymo maršrutizavimo atveju, gaunami tokie **rezultatai**:

- Iki gavėjo r prasiunčiama vidutiniškai 190 žinučių.
- Paskutinė gauta žinutė nr. 207.
- Išsikrauna 38 jutikliai.

Gauti rezultatai rodo, jog galima pratęsti BJT gyvavimo trukmę, pritaikant lošimų teoriją. Jutikliai perduoda ne kiekvieną gautą žinutę, taip tausodami savo energiją. Perduodama beveik du kartus daugiau žinučių, nei praeitame bandyme. Duomenų praradimas atsiranda dėl to, kad arti siuntėjo esantys jutikliai siunčia duomenis tuo pat metu, taip keldami triukšmą radijo kanale (20 pav.).



20 pav. Modelis, veikiantis su lošimų teorijos algoritmu. Pilkai pažymėti išsikrovę jutikliai. Centre esantys jutikliai išsikrauna pirmiausiai, nes jie gauna visas siuntėjo t žinutes.

Formulė 6.2 neatsižvelgia į kaimyninių jutiklių situaciją, todėl galima ją dar optimizuoti, pritaikant kooperacinio lošimo principu paremtą koeficientą N. Jis turi turėti tolygiai augantį reikšmingumą lošimui einant į pabaigą, nes tada jutikliai turi mažą likusį energijos kiekį.

Kredito schema taip pat neišsprendžia to, jog pirmieji išsikraus aplink duomenų siuntėją t esantys jutikliai. Norint tai padaryti, reikia į formulę 6.2 įtraukti atsitiktinį dydį R, kuris turi veikti lošimo pradžioje, kol jutikliai turi didelį energijos kiekį ir žinutes turi persiųsti ne kiekvienas jas gaunantis jutiklis. Galutinė jutiklio strategijos apskaičiavimo išraiška yra tokia:

$$s(e) = \begin{cases} \frac{1}{B} - \frac{E-e}{E} - N^2, & \frac{e}{E} \leq \frac{1}{3} \\ \frac{1}{B} - \frac{E-e}{E} - R^2, & \frac{e}{E} \geq \frac{1}{3} \end{cases} \tag{6.3}$$

čia R - atsitiktinis dydis, $0 \leq R \leq 1$,
N - vidutinis likutinės energijos lygis, apskaičiuojamas taip:

$$N = \frac{\sum e_n}{n} \tag{6.4}$$

kai n - kaimyninių jutiklių skaičius, $n \geq 0$.

DSR maršrutizavime negalima taikyti lošimų teorijos visiems žinučių tipams, kadangi, maršrute esančiam jutikliui nusprendus nepersiųsti jam siunčiamos DATA tipo žinutės, būtų nutraukiamas visas maršrutas. Todėl Lošimų teoriją pritaikome tik RREQ tipo žinutėms. Gavę RREP ir DATA tipo žinutes, jutikliai jas visada perduoda. Taip sumažiname skaičiavimų kiekį jutikliuose iki minimumo, tausodami sistemos energiją.

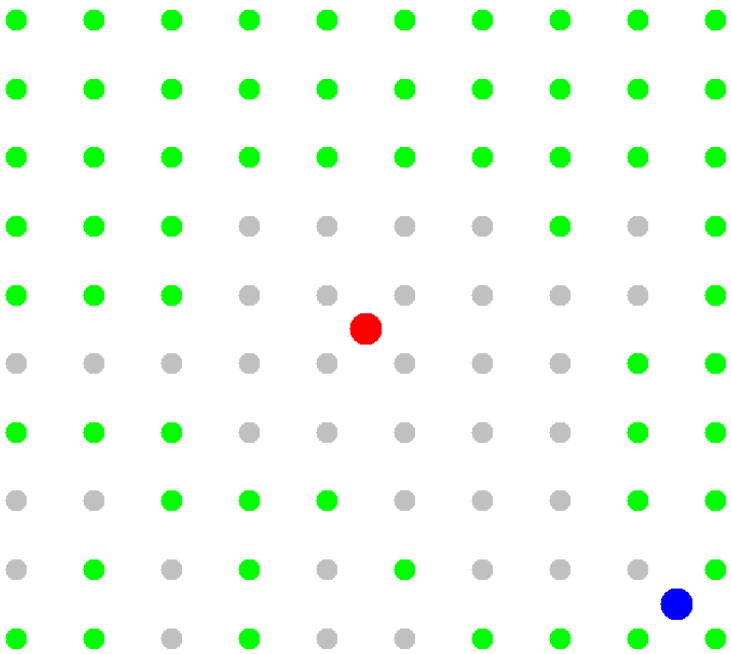
Jutikliui gavus naują žinutę, jis apskaičiuoja savo strategiją, remdamasis savo praeities veiksmais, bei savo kaimyninių lošėjų padėtimi. Pagal išraišką 6.3, jutikliai žaidimo pradžioje žinutes persiunčia rečiau, o pabaigoje - dažniau.

3 algoritmas. DSR maršrutizavimas su lošimų teorija. Veiksmai jutikliui gavus RREQ žinutę.

gaunama RREQ žinutė
if *(žinutės laiko žymė nebegalioja)* **then**
    return
end
žinutės laiko žymė išsaugoma
skaičiuojama jutiklio veiksmų strategija
if *nuspręsta siųsti žinutę* **then**
    prie maršruto pridedamas jutiklio ID
    jutiklio kredito balansas pakeliamas +1
    RREQ žinutė transliuojama kaimynams
else
    lošimo kredito balansas sumažinamas –1
    žinutė atmetama
end

DSR maršrutizavimo atvejui pritaikome tokius patobulinimus komunikacijos logikoje:

- Jutiklių aplikacijos lygmenyje taikome lošimų teoriją.
- Jutikliai skaičiuoja veiksmų naudą pagal išraišką 6.3
- Lošimo pradžioje jutikliai verčiami dažniau atmesti žinutes, o pabaigoje - jas persiųsti, kooperuojant su kaimynais.



21 pav. DSR modelis, veikiantis su lošimų teorijos algoritmu. Pilkai pažymėti išsikrovę jutikliai. Išsikrauna ne tik arčiausiai siuntėjo esantys jutikliai.

Po pritaikytų optimizacijų, žinučių siuntimo rezultatai žymiai pagerėja (21 pav.). Daugiau žinučių pasiekia adresatą, o arčiausiai siuntėjo esantys jutikliai išsikrauna lėčiau. Po tinklą tolydžiau pasiskirsto krūvis, t.y. jutikliai bendradarbiauja, siunčia duomenis pasikeisdami. Taip pat atsiranda mažiau duomenų kolizijų, todėl daugiau žinučių pasiekia adresatą. Atlikus bandymus, gauti tokie **rezultatai**:

- Iki gavėjo r prasiunčiama vidutiniškai 600 žinučių.
- Paskutinė gauta žinutė nr. 770.
- Išsikrauna 40 jutiklių.

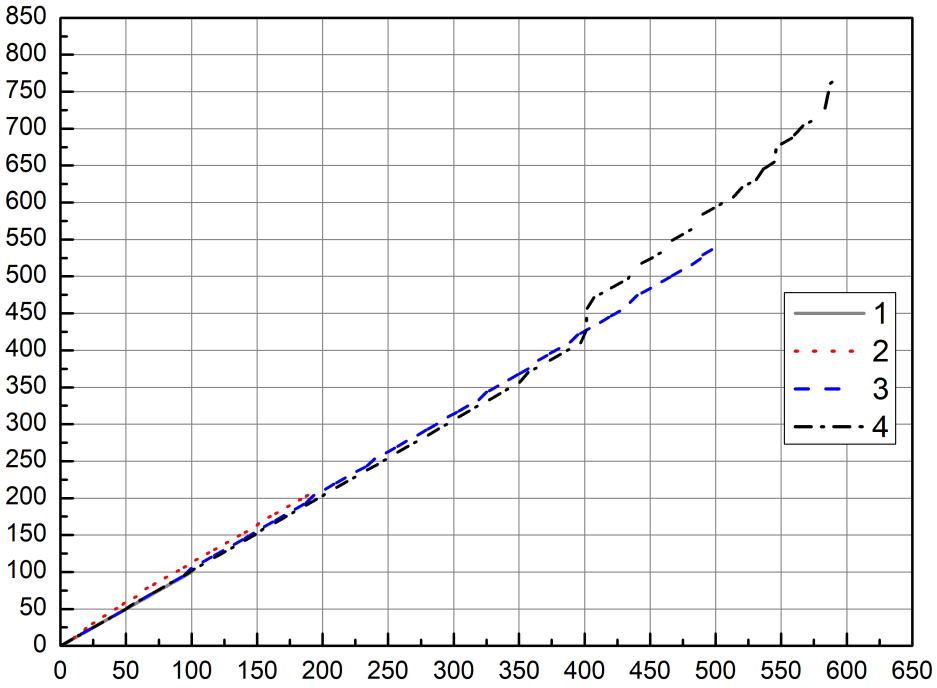
Šie rezultatai rodo, jog prasiunčiama 600% daugiau žinučių, nei būtų įmanoma, naudojant tvindymo maršrutizavimą. Su lošimų teorija DSR maršrutizavimo tinkle prasiunčiama 20% daugiau žinučių, nei be jos. Be to, smarkiai prailginama tinklo gyvavimo trukmė.

6.6. Bandymų rezultatai

Atlikus bandymus su skirtingais maršrutizavimo algoritmais, bei pritaikius skirtingus tinklo optimizavimo algoritmus, gauti rezultatai, pateikiami 3 lentelėje.

3 lentelė. Bandymų rezultatų santrauka

#	**Tinklo tipas**	**Perduota žinučių**	**Paskut. žin. nr.**	**Išsikrovę jutikliai**
1	Tvindymo maršrutizacija	100	100	100
2	Tvindymo maršrutizacija su kredito schema	190	207	38
3	DSR maršrutizacija	500	539	90
4	DSR maršrutizacija su kredito schema	600	768	40



22 pav. Rezultatų palyginimas. *x* ašyje - persiųstų žinučių skaičius, *y* ašyje - žinučių numeriai. Numeris *1* - Tvindymo maršrutizacija, *2* - Tvindymo maršrutizacija su lošimų teorija, *3* - DSR maršrutizacija, *4* - DSR maršrutizacija su lošimų teorija

Išvados ir rekomendacijos

Šiame darbe buvo sukurtas belaidžių jutiklių tinklo optimizacijos modelis, kuriame pritaikoma lošimų teorija. Darbe įrodoma, jog, panaudojant, lošimų teoriją, galima optimizuoti belaidžių sistemų energetinį efektyvumą. Sėkmingai pritaikoma DSR maršrutizavimo algoritmą. Pasiūloma ir įgyvendinama kreditinė motyvacinė schema, kurios efektyvumas įrodomas bandymais. Atlikus šį magistro baigiamąjį darbą, gauti tokie rezultatai:

- Jutikliai pagal lošimų teoriją yra "savanaudiški", kadangi duomenų persiuntimo lošimo Nešo ekvilibriumas yra abiejų lošėjų atsisakymas persiųsti duomenis.
- Naudojant primityvų tvindymo maršrutizavimą, bandymuose pasiekta 100 persiųstų žinučių. Panaudojus DSR maršrutizaciją, šis skaičius išaugo iki vidutiniškai 500 žinučių. Tinkamesnis maršrutizavimas šiuo atveju pagerino tinklo energetinį efektyvumą 5 kartus.
- Taikant lošimų teoriją ir tikimybinį žinučių persiuntimo modelį, yra prarandama dalis tarpinių žinučių. Atliktame bandyme su tvindymo maršrutizavimu, prarandama 17 žinučių iš 207 išsiųstų. Dėl šios priežasties, darbe siūlomi sprendimai labiau tinkami aplikacijoms, kuriose nėra kritiška užtikrinti visų duomenų paketų perdavimą.
- Siekiant priversti jutiklius kooperuoti tarpusavyje, darbe pritaikomas motyvacinis kredito algoritmas. Tvindymo maršrutizavimo atveju pasiekti 190% geresni rezultatai, o DSR maršrutizavimo - 20% geresni rezultatai, nei prieš taikant šį algoritmą.

Literatūros šaltiniai

[1] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of wireless sensor networks. Theory and practice.* Wiley, West Sussex, 2010.

[2] E.J. Duarte-Melo and M. Liu. Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks. *IEEE Global Telecommunications Conference*, 2002.

[3] Z. Fang and B. Bensaou. Fair bandwidth sharing algorithms based on game theory frameworks for wireless ad-hoc networks. *IEEE INFOCOM*, 2, 2004.

[4] A.E. Kamal and J.N. Al-Karaki. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6), 2004.

[5] R. Kannan and S.S. Iyengar. Game-theoretic models for reliable path-length and energy-constrained routing with data aggregation in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 22(6), 2004.

[6] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley and Sons, New Jersey, 2005.

[7] Yingshu Li M. Cardei, M. T. Thai and Weili Wu. Energy-efficient target coverage in wireless sensor networks,. *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 3, 2005.

[8] Martin J. Osborne. *A Course in Game Theory.* MIT Press, Cambridge, 1994.

[9] A. Patcha and Jung-Min Park. A game theoretic formulation for intrusion detection in mobile ad hoc networks. *International Journal of Network Security*, 2, 2006.

[10] E. Pertovt, T. Javornik, and M. Mohorčič. Game theory application for performance optimisation in wireless networks. *ELEKTROTEHNIŠKI VESTNIK. ENGLISH EDITION*, 5, 2011.

[11] K. Romer and F. Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(10), 2004.

[12] R.Sugihara and R. K. Gupta. Programming models for sensor networks: A survey. *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 3, 2005.

[13] V. Srivastava, J. Neel, A.B. Mackenzie, R. Menon, et al. Using game theory to analyze wireless ad hoc networks. *IEEE Communications Surveys and Tutorials*, 7(4), 2005.

[14] J. A. Stankovic. Research directions for the internet of things. *IEEE*, 1:3–9, 2014.

[15] S. Tadelis. *Game theory. An introduction.* Princeton University Press, Princeton, 2013.

[16] K.P. Vijayakumar, P. Ganeshkumar, and M. Anandaraj. Review on routing algorithms in wireless mesh networks. *International Journal of Computer Science and Telecommunications*, 3, 2012.

[17] Q. Wang, M. Hempstead, and W. Yang. A realistic power consumption model for wireless sensor network devices. *Journal of 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, 2006.

[18] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52:2292–2330, 2008.

[19] O. Younis and S. Fahmy. Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3, 2004.

Priedai

A. Programinis kodas

Modelio kodo fragmentas.

```
1 package md.mesh.net.routing.dsr;
2
3 import md.mesh.net.*;
4 import md.mesh.net.routing.MessageInterface;
5
6 import java.util.HashMap;
7 import java.util.Vector;
8
9 public class DsrWithGameRoutingApplication extends Application {
10
11     private HashMap<Integer, Long> seenMessages = new HashMap<Integer,
          Long>();
12     private Vector<Integer> pathToSink = new Vector<Integer>();
13
14     private long lastRreqMessageId = 0;
15     private long lastRrepMessageId = 0;
16
17     private int pathLifetime = 0;
18
19     /**
20      * Game instance for this application
21      */
22     private Game game;
23
24     public DsrWithGameRoutingApplication(Node node) {
25         super(node);
26         this.game = new Game((ZigBeeNode) node);
27     }
28
29     /**
30      * Stores the sender  from which it first receives the message, and
31      * passes the message.
32      *
33      * @param message The message that arrived.
34      * @param sender  The sender node of the last message.
35      */
36     @Override
37     public void receiveMessage(MessageInterface message, Node sender) {
38         DsrGameTestNode thisNode = (DsrGameTestNode) this.getNode();
39         DsrMessage dsrMessage = ((DsrMessage) message).copy();
40
41         if (this.seenMessages.containsKey(dsrMessage.getType()) && this.
               seenMessages.get(dsrMessage.getType()) >= dsrMessage.getId())
                {
42             return;
43         }
44
45         this.seenMessages.put(dsrMessage.getType(), dsrMessage.getId());
```

```
46
47          if (dsrMessage.getType() == DsrMessage.MSG_TYPE_RREQ) {
48              if (dsrMessage.getPath().contains(this.getNode().getId())) {
49                  return;
50              }
51              if (dsrMessage.getPath().contains(dsrMessage.getTargetId()))
                    {
52                  return;
53              }
54
55              if (thisNode.getType() == ZigBeeNode.TYPE_AGENT) {
56                  if (game.shouldNodeForwardMessageImproved()) {
57                      game.incBalance();
58                      thisNode.setSent(true);
59                  } else {
60                      game.dcrBalance();
61                      return;
62                  }
63                  dsrMessage.getPath().add(this.getNode().getId());
64                  if (dsrMessage.getLifetime() == -1 || (thisNode.
                        getEnergyLeft() < dsrMessage.getLifetime())) {
65                      dsrMessage.setLifetime(thisNode.getEnergyLeft());
66                  }
67                  this.sendMessage(dsrMessage);
68
69                  return;
70              }
71
72              if (thisNode.getType() == ZigBeeNode.TYPE_SINK) {
73                  dsrMessage.getPath().add(thisNode.getId());
74                  Vector<Integer> rrepPath = dsrMessage.getPath();
75                  long rrepId = this.lastRrepMessageId + 1;
76                  DsrMessage rrepMessage = new DsrMessage("rrep_#" +
                        rrepId, thisNode.getId(), 1, rrepId, DsrMessage.
                        MSG_TYPE_RREP, dsrMessage.getLifetime(), rrepPath);
77                  boolean rrepSent = this.sendMessage(rrepMessage);
78
79                  this.lastRrepMessageId++;
80
81                  return;
82              }
83
84              return;
85          }
86
87          if (dsrMessage.getType() == DsrMessage.MSG_TYPE_RREP) {
88              if (thisNode.getType() == ZigBeeNode.TYPE_AGENT) {
89
90                  this.sendMessage(dsrMessage);
91                  return;
92              }
93
94              if (thisNode.getType() == ZigBeeNode.TYPE_SOURCE) {
95                  if (this.pathToSink.isEmpty() || dsrMessage.getPath().
                        size() < this.pathToSink.size()) {
```

```
96                      this.setPathToSink(dsrMessage.getPath());
97                      this.pathLifetime = dsrMessage.getLifetime();
98                  }
99
100                 return;
101             }
102         }
103
104         if (dsrMessage.getType() == DsrMessage.MSG_TYPE_DATA) {
105             thisNode.setParent(sender);
106             if (thisNode.getType() == ZigBeeNode.TYPE_SOURCE || thisNode
                    .getType() == ZigBeeNode.TYPE_SINK) {
107                 if (thisNode.getType() == ZigBeeNode.TYPE_SINK ) {
108                     System.out.println(dsrMessage.getId());
109                 }
110                 return;
111             }
112             if (thisNode.getType() == ZigBeeNode.TYPE_AGENT) {
113                 if (!dsrMessage.getPath().isEmpty() && !dsrMessage.
                        getPath().contains(thisNode.getId())) {
114                     return;
115                 }
116             }
117
118                 boolean sendingSuccessful = this.sendMessage(dsrMessage)
                        ;
119                 if (sendingSuccessful) {
120                     game.incBalance();
121                     thisNode.setSent(true);
122                 }
123         }
124     }
125
126     public boolean sendMessage(DsrMessage dsrMessage) {
127         DsrMessage message = dsrMessage.copy();
128         this.seenMessages.put(message.getType(), message.getId());
129         return this.getNode().sendMessage(message, this);
130     }
131
132     public Vector<Integer> getPathToSink() {
133         return pathToSink;
134     }
135
136     public void setPathToSink(Vector<Integer> pathToSink) {
137         this.pathToSink = pathToSink;
138     }
139
140     public long getLastRreqMessageId() {
141         return lastRreqMessageId;
142     }
143
144     public void setLastRreqMessageId(long lastRreqMessageId) {
145         this.lastRreqMessageId = lastRreqMessageId;
146     }
147
```

```
148     public long getLastRrepMessageId() {
149         return lastRrepMessageId;
150     }
151
152     public void setLastRrepMessageId(long lastRrepMessageId) {
153         this.lastRrepMessageId = lastRrepMessageId;
154     }
155
156     public int getPathLifetime() {
157         return pathLifetime;
158     }
159
160     public HashMap<Integer, Long> getSeenMessages() {
161         return seenMessages;
162     }
163 }
```