Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Erdvinio atskyrimo sutankinimo sistemų efektyvumas

Baigiamasis magistro projektas

Jonas Mikušauskas

Projekto autorius

doc. dr. Paulius Tervydis

Vadovas

Kaunas, 2019

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Erdvinio atskyrimo sutankinimo sistemų efektyvumas

Baigiamasis magistro projektas

Elektronikos inžinerija (6211EX012)

Jonas Mikušauskas

Projekto autorius

doc. dr. Paulius Tervydis

Vadovas

doc. dr. Vitas Grimaila

Recenzentas

Kaunas. 2019

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Jonas Mikušauskas

Erdvinio atskyrimo sutankinimo sistemų efektyvumas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Jono Mikušausko, baigiamasis projektas tema „Erdvinio atskyrimo sutankinimo sistemų efektyvumas“ 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)

Mikušauskas, Jonas. Erdvinio atskyrimo sutankinimo sistemų efektyvumas. Magistro baigiamasis projektas / vadovas doc. dr. Paulius Tervydis; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Elektronikos inžinerija, inžinerijos mokslai.

Reikšminiai žodžiai: Erdvinio atskyrimo sutankinimas, SDM, optinių pliūpsnių komutavimas, OBS, šviesolaidis, skaidula, šerdis, optika, efektyvumas.

Kaunas 2019. 53 p.

Santrauka

Baigiamojo darbo tikslas – sukurti erdvinio atskyrimo sistemų, kuriose atliekamas optinių pliūpsnių komutavimas, efektyvumo įvertinimo metodiką. Darbe yra išanalizuojami SDM sistemų ypatumai, atliekama mokslinės literatūros analizė. Apžvelgiami įvairūs komponentai, kurie gali būti panaudoti daugiašerdžių skaidulų (SDM) sistemose.

Tyrinėjant mokslinę literatūrą buvo pastebėta, jog nėra atliktų tyrimų, kuriuose analizuojami optinių pliūpsnių komutavimo (OBS) per daugiašerdes skaidulas atvejai. Šios technologijos pagrindinis uždavinys – optinių pliūpsnių surinkimo (buferizavimo ir nukreipimo) procesas, kuomet kraštiniuose tinklo mazguose į pliūpsnius (duomenų blokus) surenkami ateinantys paketai iš įvairių interneto šaltinių (maršrutizatorių). Pagrindinė OBS tinklų problematika – pliūpsnių nesuderinamumas (pliūpsnių kolizija) kai du ar daugiau pliūpsnių vienu metu „reikalauja“ to paties kanalo ar bangos ilgio viename išėjime. Norint išvengti pliūpsnių kolizijos, gali būti panaudojami įvairūs sprendimai bangos ilgio, laiko ir erdvės atžvilgiu. Daugiašerdžių skaidulų tinkluose pliūpsnių kolizija nuspręsta spręsti pritaikant skaidulos vėlinimo linijas (buferius) ir perduodant pliūpsnius atskiromis šerdimis.

Siekiant kompleksiškai įvertinti tokių sistemų funkcionalumą buvo sudaryta jų efektyvumo įvertinimo metodika.

Norint įvertinti SDM sistemų su OBS efektyvumą buvo sudaryti analitiniai Markovo procesų modeliai (2, 3 ir 4 šerdžių). Tačiau didėjant šerdžių skaičiui sudėtingėja Markovo modelis, todėl norint ištirti SDM sistemas su daugiau šerdžių turinčiomis skaidulomis buvo sudarytas imitacinis modelis.

Taikant sudarytus modelius, buvo gauta, jog prie mažesnių apkrovų efektyviau duomenų pliūpsnius perduoti mažiau šerdžių turinčiomis skaidulomis. Prie didesnių apkrovų – skaidulomis, turinčiomis daugiau šerdžių. Nustatyta jog didinant buferio talpą šerdys tampa tolygiau užimtos, pliūpsniai tolygiau pasiskirsto tarp šerdžių dėl to didėja efektyvumas. Taip pat buvo nustatyti maksimalūs daugiašerdžių skaidulų ilgiai, kuriuos viršijus efektyvumas tampa lygus 0.

Mikušauskas, Jonas. Efficiency of Space Division Multiplexing System. Master's Final Degree Project / supervisor Assoc. Prof. Dr. Paulius Tervydis; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): Electronics Engineering, engineering science.

Keywords: Space Division Multiplexing, SDM, optical burst switching, OBS, fiber optics, multicore, efficiency.

Kaunas, 2019. 53 p.

Summary

The aim of this project is to develop a methodology for the assessment of the efficiency of the space division multiplexing (SDM) system in which optical burst switching is performed. The paper analyses scientific literature and peculiarities of SDM systems. An overview of the various components that can be used in multicore fiber (SDM) systems was made.

During the research of scientific literature, it was found that no studies have been conducted to investigate cases of optical burst switching (OBS) through multicore fibers. The main task of this technology is the aggregation of packets (buffering and redirecting) from different Internet sources (routers) on the peripheral network nodes into optical data bursts. The main problem of OBS networks is the collision of optical bursts, when two or more bursts require the same channel or wavelength at one output. Various solutions for wavelength, time, and space can be used to avoid collision of optical bursts. In order to solve these collisions in multicore fiber networks, it was decided to apply fiber delay lines (buffers) and transmitting the bursts to separate cores in multicore fiber networks.

In order to evaluate the functionality of such systems, a methodology for assessing their effectiveness was developed.

Analytical models of Markov's processes (2, 3 and 4 cores) were designed to evaluate the effectiveness of the SDM systems with OBS. However, as the number of cores increases, Markov's model becomes more complex, so a simulation model was created to investigate SDM systems with fibers with more cores.

Using the models made, it was found, that when having smaller loads, it was more efficient to transfer data bursts through fibers with fewer cores. At higher loads – fibers with more cores. It was found that by increasing the buffer capacity, the cores become more evenly occupied, the bursts are more evenly distributed between the cores, resulting in increased efficiency. Also, the maximum lengths of multicore fibers over which efficiency becomes equal to 0 is set.

.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

BHP – pliūpsnių antraštės paketas (angl. *Burst Header Packet*);

CMCF – sujungta daugiašerdė skaidula (angl. *Coupled Multicore Fiber*);

DRA – Ramano optinis stiprintuvas (angl. *Distributed Raman amplifier*);

DSP – skaitmeninis signalų apdorojimas (angl. *Digital Signal Processing*);

EDFA – erbiu legiruotos skaidulos optinis stiprintuvas (angl. *Erbium-Doped Fiber Amplifier*);

FDL – skaidulos vėlinimo linijos (angl. *Fiber Delay Lines*);

FMF – kelių modų skaidula (angl. *Few-mode Fiber*);

IoT – daiktų internetas (angl. *Internet of Things*);

LCoS – skystieji kristalai ant silicio (angl. *Liquid Crystal-on-Silicon*);

LAN – vietinis tinklas (angl. *Local Area Network*);

MAN – municipalinis tinklas (angl. *Metropolitan Area Network*);

MCF – daugiašerdė skaidula (angl. *Multicore Fiber*);

MDM – modų sutankinimas (angl. *Mode Divison Multiplexing*);

MEMS – mikroelektronikos mechaninės sistemos (angl. *Microelectronic Mechanical Systems*);

NF – triukšmo faktorius (angl. *Noise Figure*);

OBS – optinių pliūpsnių komutavimas (angl. *Optical Burst Switching*);

OCS – optinis grandinių komutavimas (angl. *Optical Circuit Switching*);

OPS – optinis paketų komutavimas (angl. *Optical Packet Switching*);

PDM – poliarizacijos sutankinimas (angl. *Polarization division multiplexing*);

QoS – paslaugos kokybė (angl. *Quality of Service*);

RSMSA – maršrutizacijos, erdvės, moduliavimo lygio ir spektro paskirstymas (angl. *Routing, Space, Modulation-level and Spectrum Assignment*);

SDM – erdvinio atskyrimo sutankinimas (angl. *Space Division Multiplexing*);

SMF – vienmodė optinė skaidula (angl. *Single-mode Fiber*);

SMUX – erdvinis multiplekseris (angl. *Spatial Multiplexer*);

TDM – laikinis sutankinimas (angl. *Time Division multiplexing*);

UMCF – nesujungta daugiašerdė skaidula (angl. *Uncoupled Multicore Fiber*);

WDM – bangos ilgių sutankinimas (angl. *Wavelength Division Multiplexing*);

WSS – bangų ilgių selektyvus komutavimas (angl. *Wavelength Selective Switching*);

XT – signalo persidengimas tarp šerdžių (angl. *inter-core crosstalk*).

Įvadas

Žmogaus darbas šiais laikais yra beveik neįsivaizduojamas be kompiuterių, telefonų ir kitų įrenginių, kurie vienaip ar kitaip apdoroja ar perduoda skaitmeninę informaciją. Šiuolaikiniam žmogui telekomunikacijos tapo viena iš neatsiejamų gyvenimo dalių. Visiems vartotojų poreikiams patenkinti reikalingos vis didesnės perdavimo spartos. Paprastam vartotojui tiesiog patikrinti elektroninį paštą ar perskaityti naujienas internete nebeužtenka. Internetas suteikia galimybę dalintis didelės apimties duomenimis, žiūrėti filmus ar televiziją, pirkti prekes ar paslaugas ir taip toliau.

Norint pilnai išnaudoti visas funkcijas, kurias suteikia internetas, reikia didinti tinklo perdavimo spartas. Prognozuojama jog 2020 metais interneto ryšį turės net 30 milijardų įrenginių [1]. Toks įrenginių bei vartotojų augimas daro didelę įtaką ir perduodamų duomenų kiekio augimui. Todėl dėl nuolat augančio perduodamų duomenų kiekio bus pasiekta teorinė (apie 100Tb/s) perdavimo sparta per vienmodę optinę skaidulą. Tokia duomenų perdavimo sparta vienmodėje skaiduloje negali būti viršyta dėl netiesinių efektų ir ribojamos optinio signalo galios. Norint išspręsti šia problemą mokslinėje literatūroje pastaraisiais metais plačiau nagrinėjamos erdvinio atskyrimo sutankinimo (angl. *Space Division Multiplexing* (SDM)) perdavimo sistemų galimybės, siekiant padidinti maksimalią perdavimo spartą optinėje skaiduloje. [2]

Darbo tikslas – sukurti erdvinio atskyrimo sistemų, kuriose atliekamas optinių pliūpsnių komutavimas, efektyvumo įvertinimo metodiką.

Darbo uždaviniai:

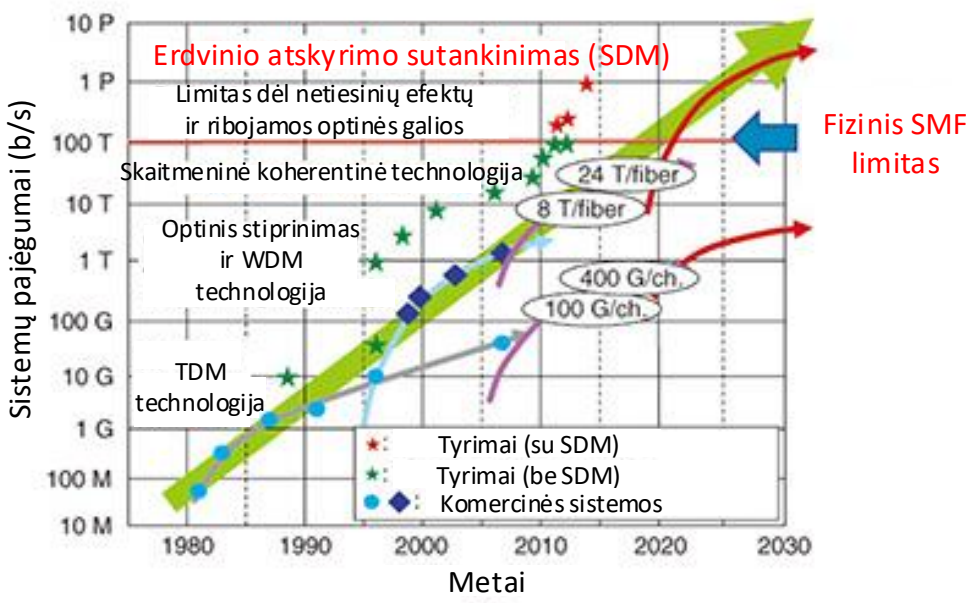
1) atlikti literatūros, kurioje nagrinėjami erdvinio atskyrimo sutankinimo ir optinių pliūpsnių komutavimo sistemų klausimai analizę;
2) sudaryti erdvinio atskyrimo sutankinimo sistemų efektyvumo įvertinimo metodiką;
3) sudaryti sistemų su daugiašerdėmis skaidulomis modelius;
4) įvertinti erdvinio atskyrimo sutankinimo sistemų parametrų priklausomybes;
5) įvertinti erdvinio atskyrimo sutankinimo sistemų efektyvumą.

1. SDM technologijos apžvalga

Pastaruoju metu optiniuose ryšiuose pradėtas nagrinėti erdvinio atskyrimo naudojimas. Jis gali būti laikomas paskutiniu neišnagrinėtu fiziniu aspektu optiniuose ryšiuose, o atitinkamos technologijos vadinamos erdvinio atskyrimo sutankinimu (SDM). SDM tai sutankinimo (multipleksavimo) technologija, kai naudojamas fizinis duomenų perdavimo atskyrimas norint vienu metu perduoti skirtingus duomenų srautus [3].

Nors erdvinių kanalų naudojimas, siekiant pagerinti sistemos pajėgumus, vis dar yra laikomas inovacija optiniame pasaulyje, tačiau SDM yra plačiai taikomas elektroniniuose ir bevieliuose pasauliuose, kurie stipriai daro įtaką mūsų gyvenimams. Pavyzdžiui, PCI (periferinė komponentų sąsaja) – tai yra kelių lygiagrečių skaitmeninių signalų prievadai. PCI sujungia motinines plokštes su išorinėmis vaizdo garso ar tinklo plokštėmis. HDMI (didelės raiškos multimedijos sąsaja) ir DVI (skaitmeninė vizuali sąsaja) pasižymi 2-dimensijų skaitmeninio prievado struktūra, palaiko aukštos kokybės vaizdo įrašų siuntimą į ekranus. Taip pat, mūsų kasdieniame gyvenime neišvengiama Wi-Fi, kuri paremta MIMO technologija (siųstuvas ir imtuvas turi daugiau nei po vieną anteninį įrenginį), kuri pagrįsta daugybe antenų, siekiant užtikrinti mobiliojo interneto prieigą [4].

Maksimalus vienmodės skaidulos (SMF) siunčiamas pajėgumas padidėjo apytiksliai 1000, 100 ir 10 kartų, naudojant įvairias multipleksavimo technologijas: laikinio kanalų multipleksavimo (TDM) ir bangų ilgių multipleksavimo (WDM). 1.1 paveiksle pateikiamos optinių tinklų perdavimo spartų galimybės [5].



1 pav. Optinių tinklų perdavimo spartų palyginimas [5]

1.1. Raida

Optiniai kabeliai vystėsi palaipsniui per dešimtmečius, vieninteliai dideli pokyčiai – perėjimas nuo daugiamodžio iki vienmodžio kabelio skirstomuosiuose tinkluose. Tam reikėjo keisti lazerio šaltinius (iki 1310 ir 1550 nm) ir patobulinti jungčių technologijas bei kabelio dizainą [6].

Perėjimas prie optinių kabelių su daugiašerdėmis skaidulomis paskatins didelį papildomų gaminių augimą (stiprintuvai, jungtys, skaidulų virinimo įrenginiai). Tai potencialiai brangesnis procesas nei pačių skaidulų gaminimas. Perėjimas prie naujų gaminių priklausys nuo to, ar ši nauja optinių skaidulų technologija bus visuotinai priimta. Telekomunikacijų operatoriams įdiegus SDM sistemas, optinių kabelių gamybos ir montavimo pramonėje būtų galima tikėtis didelio pakilimo [6].

1.2. Ateities poreikis

Petabitinės sistemos yra nukreiptos į pagrindinių interneto tiekėjų tinklų pagrindą, veikiantį kartu su naujais multipleksavimo ir signalizavimo būdais. Bendras Japonų „NEC“ ir JAV „Corning“ korporacijų darbas sugeneravo sistemą, galinčią perduoti informaciją esant bendram greičiui virš 1 Pb/s. Tačiau 2015 metais duomenys, kurie buvo perduodami šiais tinklais gali būti išmatuojami šimtais petabitų per mėnesį. Tai reiškia, kad 1 Pb/s optinėms sistemos gali pasirodyti kaip per didelės investicijos [6].

Tačiau tai gali pasikeisti. Pavyzdžiui, Jungtinėje Karalystėje vidutinis dabartinis interneto greitis yra apie 20 Mb/s. Tačiau šis greitis nuolatinai didėja. Pavyzdžiui, Europos sąjunga siekia, kad iki 2020 metų bent 50% interneto vartotojų turėtų didesnes nei 100 Mb/s spartas. Pramogų tiekėjai, tokie kaip „Netflix“ pradėjo tiekti „Ultra HD“ (4K raiškos) turinį, kuris reikalauja apie 25 Mb/s spartos. Padauginus šį skaičių iš vis didėjančio optinių tinklų vartotojų skaičiaus bei pridėjus 4G tinklų plėtros reikalavimus optiniams tinklams, Pb/s sistemų poreikis pagrindiniame tinkle stipriai išauga [6].

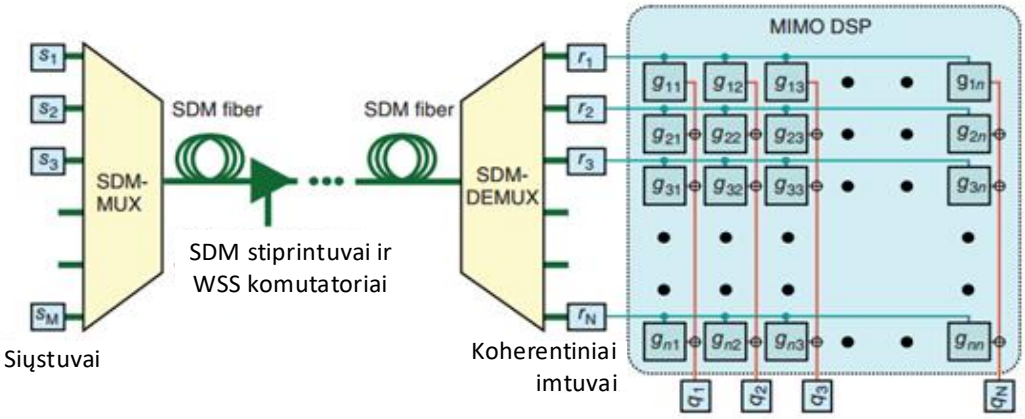
1.3. Naujausios technologijos

140,7 Tbit/s transmisija per 7 šerdžių MCF 7326 km ilgio yra rekordinis SDM perdavimo atstumas. Visuose naujausiuose SDM sistemos bandymuose, MDM buvo realizuojami su FMF (kelių modų skaidulomis) arba MMF (daugiamodėmis skaidulomis). Didžiausias pasiektas spektro efektyvumas vienai skaidulos šerdžiai MDM yra 32 bitai/s/Hz [7], tai yra beveik 3 kartus daugiau už maksimalų pajėgumą, kurį SMF gali tiekti tuo pačiu atstumu. MDM demonstravimas per įprastą MMF dar labiau sustiprina praktines SDM galimybes, nes seniau naudoti MMF gali būti naudojami iš naujo. 2014 m. Gruodžio mėn. Pabaigoje kombinuoti SDM ir WDM bandymai, kurių perdavimo nuotolis buvo ilgesnis nei 1000 km, buvo demonstruoti tik su MCF [4].

1.4. SDM komponentai

Tam, kad SDM būtų priimtas, jis turi suteikti reikšmingą ekonominį privalumą lyginant su SMF sistemų dubliavimu. Bendri SDM komponentai gali būti naudojami kaip N SMF komponentų pakeitimas. Pavyzdžiui, N šerdžių SDM stiprintuvas galėtų pakeisti N atskirų SMF stiprintuvų, naudojant vieną skaidulą sustiprinti N signalų [8].

Pagrindinė MIMO perdavimo per SDM šviesolaidį koncepcija: keli skirtingi signalai yra perduodami į SDM šviesolaidį naudojant erdvinius multiplekserius, o gauti signalai atkuriami naudojant MIMO skaitmeninį signalų apdorojimą. SDM perdavimo principinė schema pateikiama 2 paveiksle [8].



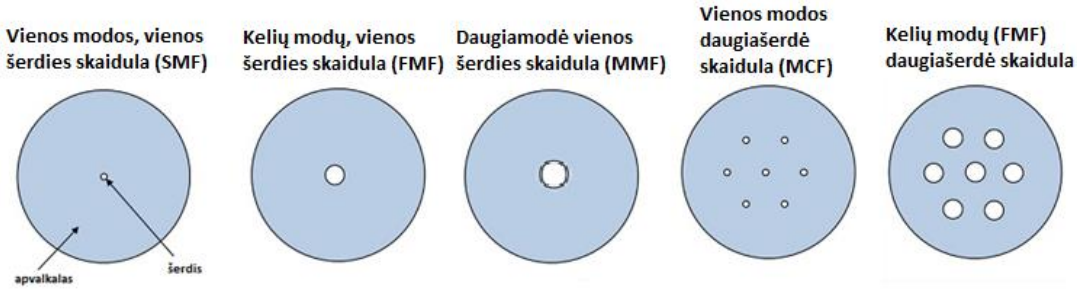
2 pav. SDM principinė schema [8]

SDM sistemose yra daug komponentų kaip ir SMF sistemose, įskaitant stiprintuvus ir bangų ilgių nukreipimo ir perjungimo elementus. Kai kurie komponentai, pavyzdžiui laisvosios erdvės prietaisai, dažnai gali būti pritaikyti iš SMF sistemų su minimaliais pakeitimais. Pavyzdžiui izoliatoriai, MCF ir WSS filtrai. SDM taip pat reikalauja naujų komponentų, kurių nėra SMF sistemose. Tokių kaip erdviniai multiplekseriai (SMUX), kurie yra reikalingi signalams perjungti į SDM šviesolaidį, optinio erdvinio kanalo ekvalaizeriai, taip pat N x N MIMO procesorius, kuris gali būti pagamintas visomis optinėmis ar elektroninio skaitmeninio apdorojimo priemonėmis. Šie komponentai SMF sistemoms suteikia papildomą kompleksiškumą, todėl jų naudojimas turėtų būti kuo mažesnis. Tačiau daugelis iš jų, tokie kaip erdviniai multiplekseriai ar MIMO procesoriai yra reikalingi tik įvedant ir (arba) išvedant skaidulų liniją [8].

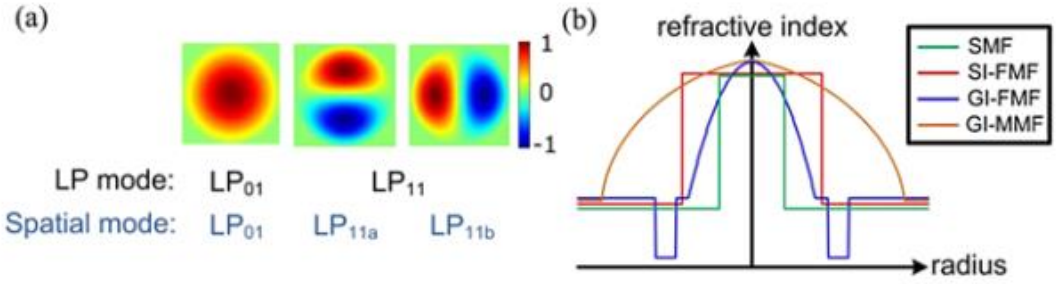
1.4.1. SDM šviesolaidis

3 paveiksle pateikiami skirtingi optinės skaidulos tipai. SMF veikia žemiau ribinio (angl. *cutoff*) dažnio ir nukreipia tik pagrindinę LP_{01} modą. Todėl SMF teikia vieną erdvinį kanalą su dviem ortogonaliomis poliarizacijos būsenomis [4].

FMF skaidulos šerdis yra šiek tiek didesnė, tai leidžia skaidula perduoti keletą modų. Tiesiškai poliarizuotos (LP) modos yra gerai žinomos ir dažniausiai naudojamos, tačiau MDM erdvinės modos naudojamos dažniau, nes erdvinių modų skaičius yra būtent erdvinių kanalų skaičius, kurių kiekvienas turi dvi statmenas poliarizacijos būsenas. Dėl erdvinių modų degeneruotos LP modos laikomos dvejomis skirtingomis modomis, kitaip tariant – dviem erdviniais kanalais. Pavyzdžiui, LP_{11} moda LP modų apibrėžime yra degeneruota ir turi dvi erdvines modas LP_{11a} ir LP_{11b} . Viena pasukta 90 laipsnių kampu kitos atžvilgiu. Tai pavaizduota 4 paveiksle [4].



3 pav. Skirtingų optinių skaidulų iliustracija

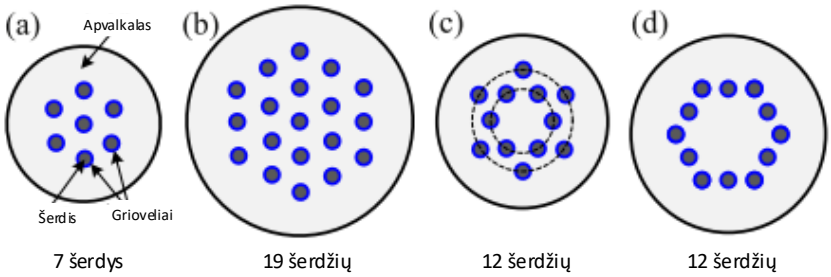


4 pav. (a) Ryšys tarp LP ir erdvinių modų ir (b) įvairių skaidulų tipų lūžio rodiklių profiliai. [4]

MDM perdavimo metu FMF skaiduloje gali būti naudojami tiek pakopinio lūžio rodiklio SI (angl. *step-index*), tiek ir gradientinio lužio rodiklio GI (angl. *graded-index*) profiliai (žiūrėti 4 paveikslo (b)). SI lūžio rodiklio profilis leidžia sumažinti modų persidengimą (angl. *mode coupling*). SI-FMF sklaidos greičių skirtumai tarp skirtingų modų yra gana dideli, todėl sunku pasiekti nedidelį MDGD (angl. *modal differential group delay*) vėlinimą ilgu atstumu (<20 ps/km). Kuomet modų persidengimas (angl. *crosstalk*) yra nedidelis – kiekviena moda gali būti individualiai aptikta, MIMO paremtų imtuvų sudėtingumas gali būti gerokai sumažintas, o MDGD tampa nereikšmingas. Praktiniam tokio tipo FMF naudojimui reikalingas tikslus skaidulų sujungimas ir aukštos kokybės FMF komponentai, tokie kaip erdviniai multiplekseriai (SMUX), kad pasirinktinai praleistų ir aptiktų kiekvieną modą [4].

MCF (daugiašerdė skaidula) sujungia daugelį šerdžių viename apvalkale. MCF gali būti suskirstyta į dvi grupes priklausomai nuo to ar šerdys yra sujungtos viena su kita CMCF (angl. *Coupled-MCF*), ar nesujungtos UMCF (angl. *Uncoupled-MCF*) [4].

Kiekviena UMCF šerdis gali būti laikoma atskiru erdviniu kanalu. Daugiašerdės skaidulos pavaizduotos 5 paveiksle.

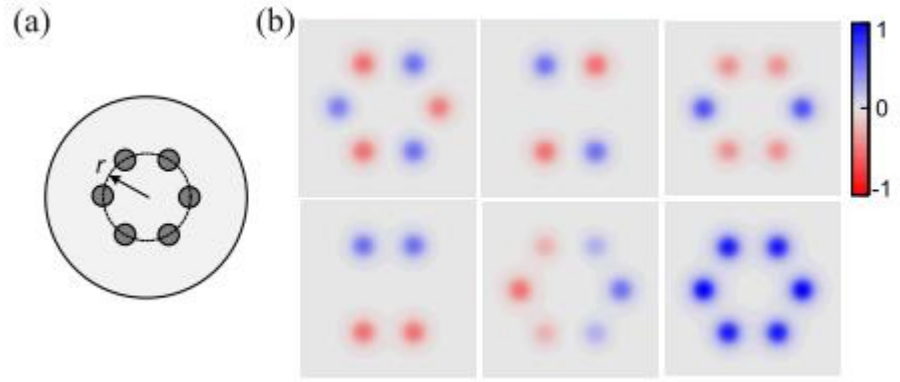


5 pav. UMCF skaidulos (a) 7, (b) 19, (c,d) 12 šerdžių [4]

Optiniuose tinkluose su didelio ilgio (>10km) kabeliais buvo pademonstruotas didelio pralaidumo signalo perdavimas per 7 šerdžių, 12 šerdžių ir 19 šerdžių skaidulas. 7 šerdžių ir 19 šerdžių UMCF skaidulos yra pagrįstos šešiakampe glaudžiai supakuota struktūra HCPS (angl. *hexagonal close-packed structure*). Siekiant sumažinti persidengimą tarp šerdžių (angl. *crosstalk*), buvo pasiūlytos griovelių ir skylių (trench-assisted and hole-assisted) struktūros. Grioveliai ir skylės aplink šerdis šerdies išsklaidomos šviesos dalį. Taip pat buvo pademonstruotos, dviejų tipų 12 šerdžių skaidulos, kuriose šerdys išdėstytos dvigubo žiedo struktūra (two-pitch structure (TPS)) [9] ir vieno žiedo struktūra (one ring structure(ORS)). Šios skaidulos atitinkamai pavaizduotos 5 paveikslo (c) ir (d). Svarbus UMCF pranašumas yra tai, kad komerciniai poliarizacijos atskyrimo koherentiniai imtuvai gali būti tiesiogiai naudojami signalui atkurti. Siekiant palaikyti signalo perdavimą dideliais atstumais, UMCF turi pasiekti mažą skaidulų slopinimą, didelį veiksmingą plotą (A_{eff}), kad būtų sumažinti netiesiniai efektai skaiduloje, mažą signalo persidengimą tarp šerdžių (angl. *inter-core crosstalk*) ir apvalkalo skersmenį, kad būtų užtikrintas mechaninis patikimumas. Taip pat yra svarbus ir mastelio klausimas. 19 šerdžių UMCF apvalkalo skersmuo apie 200 µm, kuris yra gana jautrus net ir mažo spindulio sulenkimams. Jei apvalkalas turi būti padidintas siekiant integruoti daugiau šerdžių su nežymiu tarpusavio persidengimu, skaidulos mechaninis patikimumas bus rimta problema [4].

CMCF yra kitas MCF skaidulų tipas, kuris gali turėti stipriai arba silpnai sujungtas šerdis. CMCF „supermodos“ yra generuojamos sujungtoje šerdyje, kur kiekviena „supermoda“ gali būti laikoma vienu erdviniu kanalu. 6 ir 7 paveiksluose parodytos schemos ir modeliuojamos „supermodos“ profiliai 3 šerdžių ir 6 šerdžių CMCF atitinkamai.

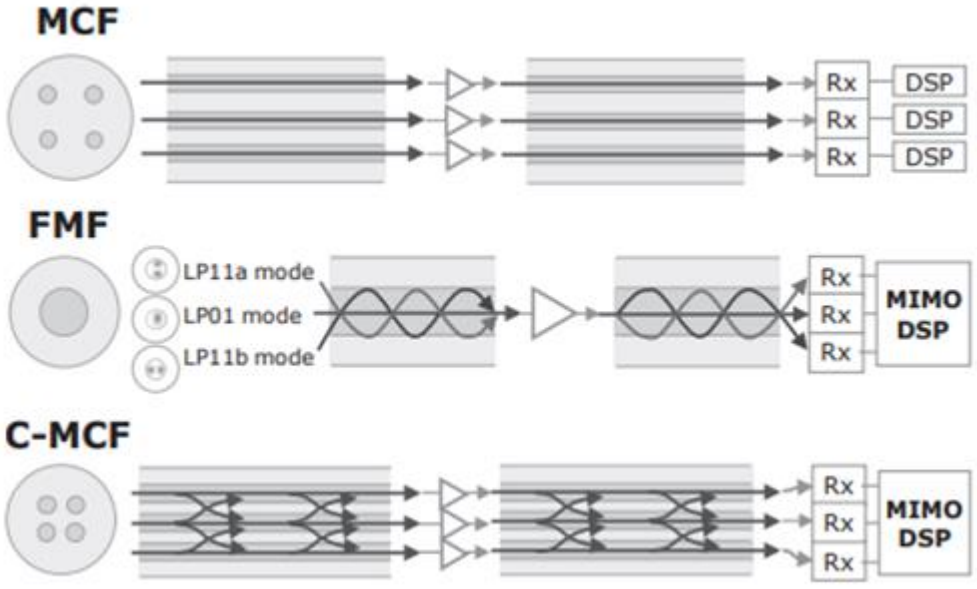
6 pav. (a) 3 šerdžių CMCF schema ir (b) simuliuojami „supermodos“ profiliai. Spalva rodo santykinę izoliuotų modų fazę [4]



7 pav. (a) 6 šerdžių CMCF schema ir (b) simuliuojami „supermodos“ profiliai. Spalva rodo santykinę izoliuotų modų fazę [4]

Verta pažymėti, kad izoliuotų modų fazės, kurių superpozicija sukelia „supermodas“, ne visada yra vienodos. Tai yra pažymėta 6 ir 7 paveiksluose skirtingomis spalvomis, kurios nurodo sudedamąsias izoliuotas modas. Apskritimo spinduliai, *r*, kur yra šerdys, yra atitinkamai 17 µm ir 28 µm. Palyginus su FMF, CMCF skaidulos gali būti suprojektuotos su didesne A_{eff} ir todėl yra labiau atsparios netiesiškumo efektuose pasireiškiančiuose skaidulose. 4200 km ir 1705 km bendra SDM ir WDM transmisija buvo pasiekta su 3 šerdžių ir atitinkamai 6 šerdžių CMCF skaidula. Tai yra daug ilgesnis atstumas pasiektas nei su 3 modų ir 6 modų FMF skaidulomis. Taip pat, CMCF skaidulose MDGD didėja proporcingai skaidulos ilgio kvadratinei šakniai, o ne tiesiškai, kaip nustatyta FMF. Panašiai į transmisiją per FMF skaidulą, MIMO paremtas DSP yra reikalingas, kad CMCF galėtų visiškai atkurti signalus, tačiau su mažesniu skaičiavimo sudėtingumu dėl mažesnio MDGD. Palyginti su UMCF, CMCF turi didesnį erdvinio panaudojimo efektyvumą ir gali puikiai tilpti į standartinį 125 µm apvalkalą.

8 paveiksle pateikiamas UMCF, FMF ir CMCF skaidulų palyginimas [11]



8 pav. UMCF, FMF ir CMCF skaidulų palyginimas [11]

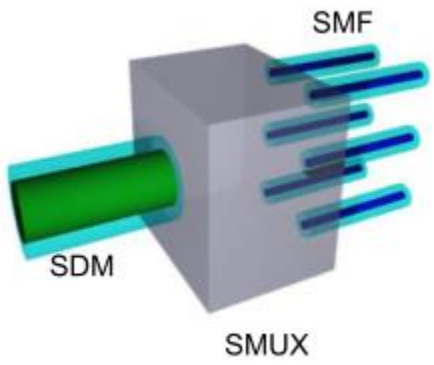
1 lentelėje yra pateikiamas FMF ir dviejų tipų MCF palyginimas.

1 lentelė. FMF ir dviejų tipų MCF palyginimas [11]

	Erdvinis naudojimo efektyvumas	**DSP kompleksiškumas**	**Netiesiškumo tolerancija**	**Mastelis**
FMF/MMF	Aukštas	Aukštas	Vidutinė	Didelis
CMCF	Vidutinis	Vidutinis	Vidutinė/Aukšta	Vidutinis
UMCF	Žemas	Žemas (≈ SMF)	Žema (≈ SMF)	Mažas

1.4.2. Erdviniai multiplekseriai (SMUX)

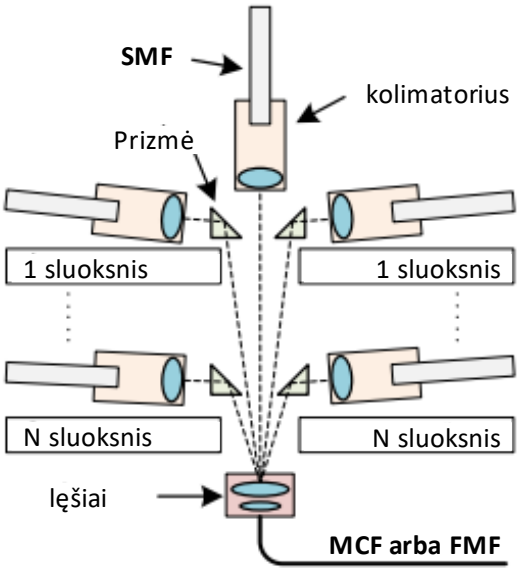
Erdviniai multiplekseriai (SMUX) yra nauji ir būtini SDM komponentai. 9 paveiksle pateikiama SMUX schema, kurios pagrindinė funkcija yra perduoti optinį signalą iš SMF skaidulų į SDM šviesolaidinio kabelio modas arba atskiras šerdis. [4]



9 pav. SMUX schema [4]

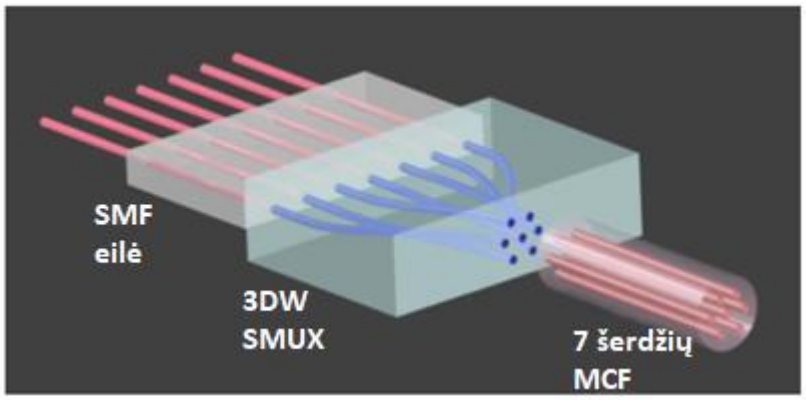
SMUX struktūroje panaudotos prizmės ir veidrodėliai pritaikyti tam, kad perkeltų daugybę kolimuotų optinių spindulių arčiau vienas kito tam tikrose vietose. Papildoma vaizdo gavimo optika naudojama siekiant optimizuoti lygiagrečių optinių spindulių dydį, kad jis atitiktų SDM skaidulą. Dėl optikos reversiškumo, ta pati struktūra gali būti naudojama ir demultipleksavimui.

Erdvinio multiplekserio, kuris buvo suprojektuotas sujungti 19 šerdžių UMCF skaidulą schema pavaizduota 10 pav. Skaidulų kolimatoriai ir prizmės yra išdėlioti aplink SMUX centrinę ašį taip, kad būtų galima pridėti daugiau sluoksnių.



10 pav. SMUX veikimo principas [4]

Trijų dimensijų bangolaidžių įtaisų gamyba paremta femto sekundžių lazerio impulsais, kurie sutelkti į lydyto silicio dioksido substratą. Bangolaidžių šerdys ir apvalkalai pagaminti iš gryno lydyto silicio dioksido. Tai leidžia lokaliai keisti stiklo lūžio rodiklį taip, kad būtų sukurti bangolaidžiai 3D. 11 paveiksle pateiktas 3DW SMUX eskizas, skirtas 7 šerdžių MCF skaidulos sujungimui. 3DW SMUX turi septynis vienos modos bangolaidžius, išdėstytus linijoje viename gale, kurie leidžia efektyviai prijungti prie standartinio SMF.



11 pav. 3DW SMUX, skirtas sujungimui su 7 šerdžių MCF [4]

Lazeriu išraižyti bangolaidžiai buvo ištirti dar 1996. Visai neseniai buvo realizuoti lazeriu išraižyti venmodžiai bangolaidžiai su mažais sklidimo nuostoliais, maždaug 0.3 dB/cm esant 1550 nm bangai [10]. Tačiau lūžio rodiklio pokyčiai, kuriuos sukelia lazerinis išraižymas, yra ribojami. Aukštesnis lūžio rodiklis, pvz. $\Delta n > 6 \times 10^{-3}$ buvo pasiektas per papildomą gamybos procesą [12], [13], tačiau bangolaidžių vienodumas ir sklaida nebuvo išsamiai aptariami / ištirti. 3DW įrenginiai su žemu Δn gali tinkamai veikti su MCF skaidulomis, tačiau FMF atveju – reikia didesnio Δn. [4]

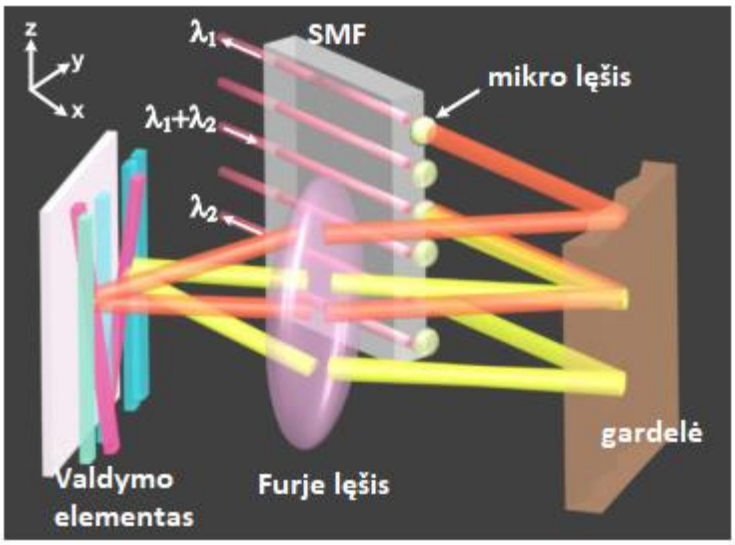
1.4.3. Optiniai stiprintuvai

Nors optinės skaidulos šiais laikais gali būti gaminamos su mažu slopinimu (maždaug 0,2 dB/km), stiprintuvai, tokie kaip erbiu legiruotos skaidulos optiniai stiprintuvai (EDFA) ir Ramano optiniai stiprintuvai (DRA), vis tiek neišvengiami, norint kompensuoti optinių tinklų nuostolius, ypač linijoms, kurių atstumas didesnis nei 100 km. [4]

Stiprintuvo schema, naudojant DRA ir EDFA buvo įrodyta [10], kur 9-12 dB DRA stiprinimas, su mažesniu kaip 1 dB triukšmo faktoriumi (NF) buvo realizuotas per 75 km 7 šerdžių MCF skaidulą. Papildomi SMF EDFA stiprintuvai buvo naudojami tam, kad pilnai kompensuoti skaidulų nuostolius.

1.4.4. Bangos ilgių selektyvūs komutatoriai (WSS)

WSS (angl. *Wavelength selective switch*) tai 1 x *N* optinis įrenginys, kuris viename bendrame įvesties prievade priima kelis bangos ilgius ir leidžia bet kokį bangos ilgio kanalą dinamiškai nukreipti į bet kurį iš *N* išėjimo prievadų. WSS galima naudoti ir atvirkščiai – kombinuoti bangų ilgių kanalus. WSS yra pagrindinis komponentas, skirtas perkonfigūruojamiems optiniams multiplekseriams [12]. SMF WSS schema parodyta 12 pav.



12 pav. SMF WSS schema [4]

SMF WSS susideda iš:

1. Skaidulų ir mikro objektyvų įvestyje / išvestyje.
2. Difrakcinės gardelės, kuri skleidžia bangos ilgio kanalus horizontalioje plokštumoje (x-y plokštuma 12 paveiksle).
3. Furjė lęšio, kuris paverčia kampinius poslinkius spindulio poslinkiais ant šviesos valdymo elemento.
4. Šviesos valdymo elementas, kuris gali būti mikroelektronikos mechaninė sistema (MEMS) [13], [14] arba skystasis kristalas ant silicio (LCos) [15], vertikaliai valdomiems spinduliams.

Tradiciniame SMF WSS kiekvienas bangos ilgis gali būti keičiamas atskirai. Tyrimai, bandantys perduoti šią koncepciją SDM, individualiai perjungiant erdvinius kanalus, parodė, jog šis metodas

akivaizdžiai yra ribotas dėl tarpusavio modų persidengimų [16]. Taip pat, WDM ir MDM yra dvi iš esmės skirtingos sąvokos: Persidengimas WDM gali būti nereikšmingas, o MDM pasižymi stipriu linijiniu persidengimu tarp lygiagrečių modų. Atsižvelgiant į našumo reikalavimus, tokius kaip nereikšmingas persidengimas ir didelė spektrinė sklaida, visi erdviniai kanalai viename bangos ilgyje turėtų būti laikomi vienu objektu, kuris turi būti perjungiamas. Todėl turėtų būti bendrai nukreiptas per WSS suderinamą su SDM, panašų į SMF atvejį [4].

[17], [18] šaltiniuose demonstruojamas WSS, kai naudojama 7 šerdžių MCF skaidula, SMUX, kad demultipleksuoti signalą į septynis lygiagrečius signalus per septynias SMF skaidulas. Septyni lygiagretūs signalai perduodami į komercinę WSS, kurioje yra daugiau nei 21 SMF prievadai (7 šerdys x (1 įprastas prievadas + 2 išvesties prievadai)), bendrai valdomi septyniais išvesties SMD prievadais, kuriuos jungia kitas SMUX į 7 branduolių MCF. Siekiant sumažinti šviesos spindulio kampą ir persidengimą buvo pasiūlyta pridėti pertvarkymo bloką tarp SMUX ir SMF įėjimo / išvesties sekcijos, kuri tarpusavyje perduoda signalus iš skirtingų erdvinių kanalų į gretimus vienos modos prievadus [18].

1.4.5. Skaidulų suvirinimo aparatai ir jungtys

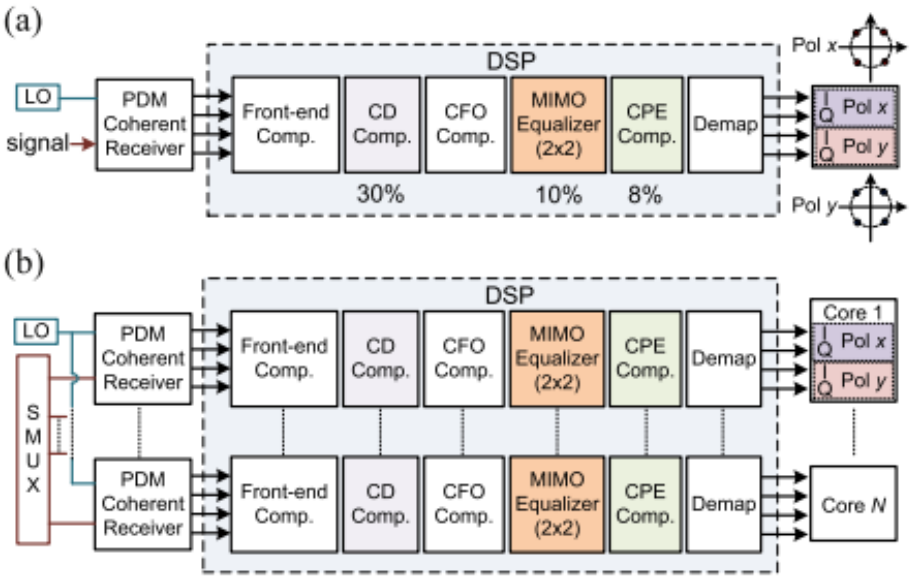
Skaidulų suvirinimo aparatai yra naudojami norint suvirinti dvi skaidulas, kaitinant abu skaidulų galus, juos ištirpdant ir sujungiant. Optinių skaidulų suvirinimas yra būtinas įrankis optinių tinklų priežiūrai ir montavimui. Todėl, ar aukštos kokybės SDM skaidulų sujungimas gali būti veiksmingai įgyvendintas yra kritinis kriterijus, ar SDM galiausiai galėtų tapti praktiniu sprendimu [4].

Virinimo metu siekiant sulygiuoti visas MCF skaidulos šerdis, suvirinimo aparatai turi turėti rotacinį skaidulų sulygiavimą. Buvo įrodyta, kad ir optinės skaidulos šoninio vaizdo ir vaizdo iš galo stebėjimas gali įgalinti sulyginimą sukant skaidulas. Šoninio vaizdo stebėjimui – tuščiaviduriai žymekliai, kurių didelis lūžio rodiklis buvo pridedamas prie 7 šerdžių MCF skaidulos, kad padidintų matomumą. Galinio vaizdo stebėjimui naudojamas fotoaparatas, skirtas abiem skaidulos galams stebėti, aptikti šerdžių pozicijas ir sulygiuoti atitinkamas šerdis. Kadangi MCF skaidulų apvalkalo dydis paprastai didesnis, įprasti suvirinimo aparatai negali užtikrinti vienodo šilumos pasiskirstymo, dėl kurio skirtingos šerdys gali būti suvirinamos su skirtingais nuostoliais [19].

7 šerdžių MCF jungtis su 0.13 dB vidutiniu slopinimu ir < -45 dB vidutiniu atspindžiu buvo sukurtas naudojant slankiojantį mechanizmą, siekiant eliminuoti deformacijų ir fizinio kontakto įtakas, kad sumažinti nuostolius. MCF jungties veikimas yra panašus į atitinkamų SMF jungčių veikimą [4].

1.5. Skaitmeninis signalų apdorojimas

Vienas svarbus didelio pajėgumo SDM sistemų faktorius yra didžiulis elektronikos ir DSP (Skaitmeninio signalų apdorojimo) vystymas. Norint aptikti ir atrinkti priimtus signalus, naudojami optiniai koherentiniai imtuvai ir greitaeigiai realaus laiko osciloskopai. Optinių imtuvų, palaikančių SMF, UMCF su N šerdimis schemos yra pateikiamos 13 paveiksle.



13 pav. Optinio imtuvo DSP schema skirta (a) SMF ir (b) UMCF su N šerdžių [4]

Optinis imtuvas susideda iš optinio poliarizavimo sutankinimo (PDM), koherentinių imtuvų ir DSP blokų. Skaitmeninėje srityje DSP kompensuoja priekio-galo (angl. *front-end*) netolygumus, chromatinę dispersiją, laiko poslinkį ir nešlio dažnio poslinkį. Nešlio fazės įvertinimas yra realizuotas po laiko ar dažnio srities MIMO ekvalaizerio. UMCF aplikacijoms, kaip parodyta 13 paveikslo (b), DSP blokas SMF gali būti dubliuojamas, kad kiekviena šerdis atskirai atskirtų mišrius PDM signalus [20].

1.6. Šerdžių tarpusavio persidengimas

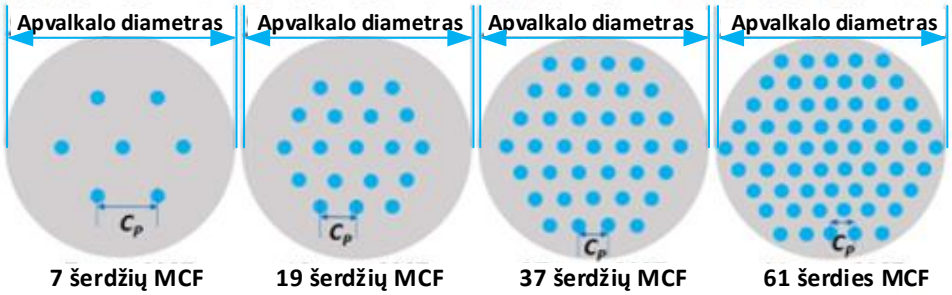
Nors ir MCF tinklo perdavimo pajėgumai pasiekė 1 Pbps [21], tačiau viena svarbiausių problemų tokio tipo tinkluose yra signalo persidengimas tarp šerdžių (tai yra perdavimo signalo trukdžiai, kai signalas iš vieno perdavimo kanalo pasiekia ir kitus kanalus). Per daug šerdžių perduodami signalai sąveikauja vienas su kitu, taip vienas kitam tapdami triukšmu. Tai sąlygoja duomenų perdavimo pajėgumo ir perdavimo atstumo mažėjimą optiniuose tinkluose [22].

Persidengimas tarp šerdžių (angl. *crosstalk*) pasireiškia didinant daugiašerdžių skaidulų perdavimo spartas. Buvo atliktas tyrimas [22] siekiant sumažinti šį persidengimą tarp skirtingų šerdžių. Tyrime taikytas metodas buvo šerdžių paskirstymas, t. y. šerdžių valdymo ir signalo spektro padalijimas. Šiame metode lygiagretinami du algoritmai. Pirma atliekamas šerdžių prioritizavimo algoritmas, pagrįstas daugiašerdės skaidulos struktūra. Antra, atliekamas šerdžių klasifikavimo algoritmas, pagrįstas reikiamo pralaidumo sąsaja. Tyrimo rezultatai parodė, kad šis metodas veikė gerai ir veiksmingai sumažino persidengiančių spektrų skaičių, todėl sumažino persidengimą tarp šerdžių iki 59,76%. Buvo padaryta išvada, jog siūlomas metodas gali sumažinti persidengimo problemą daugiašerdžių skaidulų (MCF) tinkluose [22].

[21] šaltinyje naudojama pirmojo pritaikymo (angl. *first-fit*), atsitiktinio, klasifikavimo ir pasiūlymo algoritmas tam, kad būtų galima įvertinti šerdies persidengimo vertes. Klasifikavimo ir pasiūlymo algoritmų naudojimas parodė gerus rezultatus mažinant persidengimą MCF. Tačiau algoritmai vis tiek generuoja reikšmingą šerdžių persidengimo vertes, ir šio tyrimo rezultatai nepateikia persidengimo kiekvienai šerdžiai atskirai. [22] Tyrime siūlomas tinklo pralaidumo valdymas, pritaikytas MCF šerdims, neatsižvelgiant į daugiašerdžių skaidulų struktūrą ar šerdžių skaičių [22].

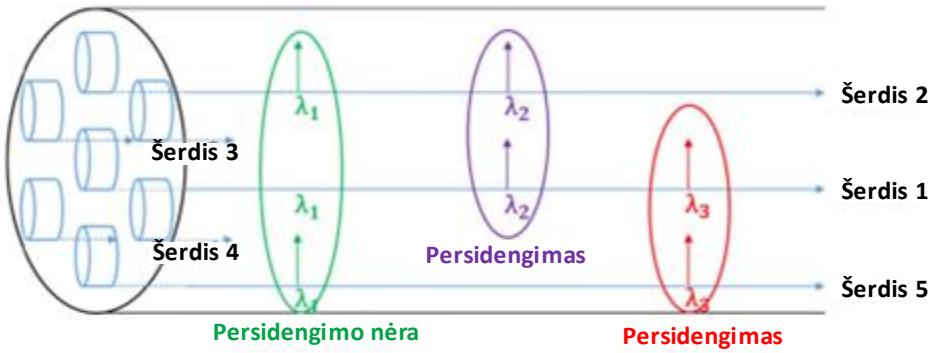
[21] šaltinio tyrimai rodo, kad tarpusavio šerdžių persidengimas priklauso nuo signalo spektro ir šerdžių išsidėstymo. Buvo daroma prielaida, kad to paties dažnio signalai yra perduodami vienu metu visose skaidulos šerdyse. [22] tyrimas nagrinėja persidengimo problemą MCF skaidulose atsižvelgiant į tuo metu naudojamą šerdį ir dažnį. Be to, [22] tyrime siūlomi šerdžių valdymo metodai, skirti sumažinti persidengimą ir spektro valdymas siekiant sumažinti fragmentaciją. Siūlomas metodas [22] grindžiamas dviem algoritmais asocijuotais su persidengimu ir fragmentacija – šerdžių prioritizavimo algoritmu, pagrįstu MCF struktūra ir šerdžių klasifikavimo algoritmu, pagrįstu reikalingu ryšiui pralaidumu.

Moksliniuose tyrimuose ir eksperimentuose dažniausiai homogeniškas MCF su šešiakampiu išdėstymu. Pagrindiniai tokio MCF bruožai yra tai, kad atstumas tarp gretimų šerdžių (angl. *core pitch*) yra vienodas. Tai pateikiama 14 pav.



14 pav. MCF pavyzdžiai. (*Cp* – atstumas tarp šerdžių) [23]

15 paveiksle pavaizduojama, kaip MCF galimai susidaro interferencija gretimose šerdyse perduodant vienodo bangos ilgio signalus.



15 pav. Persidengimo MCF skaiduloje pavyzdys [23]

Statistinis vidutinis persidengimas homogeninėje MCF skaiduloje, kuris gali būti taip pat laikomas galios nutekėjimu iš vienos šerdies į kitą, išreiškiamas 1 formule [23] .

$$h(\kappa, Cp) = 2 * \frac{\kappa^2 * R}{\beta * Cp} \quad (1)$$

Parametrai naudojami formulėse pateikiami 2 lentelėje

2 lentelė. FMF ir dviejų tipų MCF palyginimas [23]

n	Gretimų šerdžių skaičius
κ	Sąryšio (coupling) koeficientas
β(m^{-1})	Propagacijos konstanta (konstanta)
R(m)	Lenkimo spindulys (konstanta)
Cp(m)	Atstumas tarp gretimų šerdžių
L(m)	Sujungimo ilgis

Atsižvelgiant į susietos galios teoriją, homogeniškos MCF šerdžių persidengimas, kai naudojama vienakryptė transmisija, gali būti išreiškiama 2 formule [23]:

$$\mathrm{XT} = \frac{n - ne^{-(n+1)*2*h*L}}{1 + ne^{-(n+1)*2*h*L}} \tag{2}$$

Čia skaitiklis ir vardiklis atitinkamai reiškia kaimyninių šerdžių signalo galią ir tikslinės šerdies signalo galią, atsižvelgiant į jų energijos nutekėjimą [23].

Siekiant modeliuoti XT supresiją (ΔXT_{-dB}) dvikryptėje transmisijoje [23] šaltinyje galios mažinimo koeficientas P_r yra apibrėžiamas, kad būtų atsižvelgta į persidengimą ir tarp šerdžių kuriomis signalas perduodamas priešinga kryptimi ($P_r = 10^{(-\Delta XT-dB/10)}$). Šis įnašas gali būti modeliuojamas dauginant esamos XT formulės (2 formulė) skaitiklį iš P_r. Gaunama (3) formulė ir .P_r skaičiuojamas iš (4) formulės [23].

$$XT = \frac{P_r * (n - ne^{-(n+1)*2*h*L})}{1 + ne^{-(n+1)*2*h*L}} \tag{3}$$

$$P_r = \frac{S\alpha_R}{2\alpha}\left[\frac{e^{\alpha L} - e^{-\alpha L}}{\alpha L} - 2e^{-\alpha L}\right] \tag{4}$$

Šiose lygtyse n reiškia gretimų šerdžių skaičių, kuriomis perduodamas signalas yra priešingos krypties nei skaičiuojamos šerdies. S, α_R, α atitinka Reilėjaus sklaidos komponento atkūrimo faktorių atgaline kryptimi, susilpnėjimo (attenuation) koeficientą gaunamą iš Reilėjaus sklaidos ir skaidulos slopinimo koeficientą [23].

Kadangi MCF daugiau nei 1 šerdis gali būti priskirta perduoti signalus bet kuria kryptimi, reikia apibendrinti lygties formą, kad būtų įtrauktas gretimų šerdžių nešančių signalą ta pačia (5 lygtis) ir priešinga (6 lygtis) kryptimis persidengimas [23].

$$XT_{same} = \frac{n_1 - n_1 e^{-(n+1)*2*h*L})}{1 + ne^{-(n+1)*2*h*L}} \tag{5}$$

$$XT_{opposite} = \frac{P_r * (n_2 - n_2 e^{-(n+1)*2*h*L})}{1 + ne^{-(n+1)*2*h*L}} \tag{6}$$

Lygtyse n_1 ir n_2 reiškia gretimų šerdžių skaičių, kuriomis perduodamas signalas ta pačia arba priešinga kryptimis ($n = n_1 + n_2$). Persidengimas abejomis kryptimis išreiškiamas (7) formule [23].

$$XT_{hex} = XT_{same} + XT_{opposite} = \tag{7}$$
$$= \frac{n_1 - n_1 e^{-(n+1)*2*h*L} + P_r * n_2 - P_r * n_2 e^{-(n+1)*2*h*L})}{1 + n_1 e^{-(n+1)*2*h*L} + n_2 e^{-(n+1)*2*h*L}}$$

Esant pastoviam atstumui tarp gretimų šerdžių ir gretimų šerdžių skaičiui, persidengimas yra proporcingas sujungimo ilgiui (angl. *link distance*). Tai parodo, kad persidengimo mažinimas dvejomis kryptimis gali padidinti palaikomą skaidulos ilgį [23].

2. Optinių pliūpsnių komutavimas (OBS)

Yra atlikta daug mokslinių tyrimų, susijusių su erdvinio atskyrimo sutankinimo sistemomis. MCF tinklai yra jautrūs persidengimui tarp šerdžių (angl. *inter-core crosstalk* (XT)) [24]. Siekiant panaikinti šią problemą didelių distancijų MCF tinklams, kurie garantuoja signalo kokybę tinkamai paskirstydami spektrinius išteklius, buvo pasiūlyti maršrutizavimo ir spektro paskirstymo (RSA) algoritmai [25]. Reikšmingas XT sumažėjimas, t.y. bent 20 dB, taip pat buvo pasiektas perduodant priešingų krypčių optinius signalus gretimose MCF šerdyse [26]. Net ir dvikrypčio perdavimo privalumai naudojant MCF buvo išnagrinėti sudarinėjant prioritetų algoritmus, kurie, atsižvelgiant į skaidulos šerdžių išsidėstymą, paskirsto resursus minimalizuojant patiriamus persidengimus tarp skirtingų šerdžių [27].

Viena iš technologijų, numatančių efektyvų juostos pločio panaudojimą – optinis grandinių komutavimas (angl. *Optical Circuit Switching* – OCS). Ši technologija remiasi bangos ilgio maršrutizavimu, sudarydama perduodamiems duomenims „galas-galas“ sujungimus. OCS technologija yra patogi balso duomenų perdavimui, tačiau visiškai netinkama pliūpsnių paraiškų srauto DWDM sistema perdavimui. Optinis paketų komutavimas (angl. *Optical Packet Switching* – OPS) būtų tinkamas tokių pliūpsninių paraiškų srauto perdavimui, tačiau ši technologija yra per daug sudėtinga ir ekonomiškai neperspektyvi. Taigi, norint perduoti DWDM pliūpsninių paraiškų srautą, buvo sukurta optinių pliūpsnių komutavimo (OBS) technologija [28].

Mokslinėje literatūroje nėra išsamiai išanalizuota optinių pliūpsnių perdavimo per daugiašerdes skaidulas.

2.1. OBS technologija

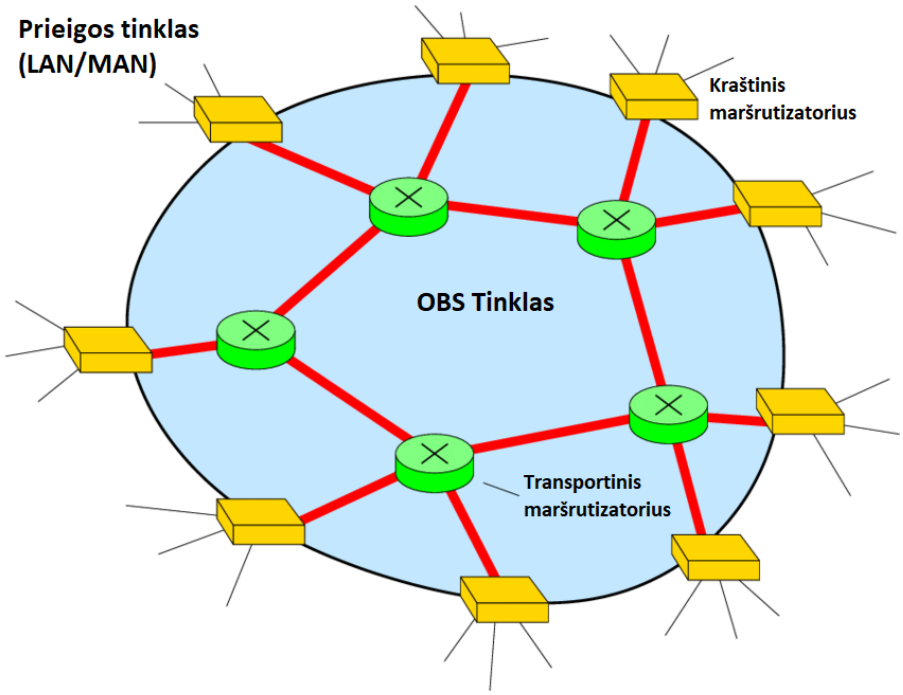
IP paketai, turintys bendrų savybių ir bendrą galutinį tikslą, OBS tinkluose praeina per tą patį įėjimo pernašos tinklo maršrutizatorių. Šiame maršrutizatoriuje paketai yra sudedami į pliūpsnius, kur kiekvienas pliūpsnis yra atskirai komutuojamas ir maršrutizuojamas. Valdymo pranešimas yra svarbiausias persiunčiant kiekvieną pliūpsnį. Valdymo pranešimo tikslas yra informuoti tarpinius mazgus apie ateinančius duomenų pliūpsnius – tokiu būdu mazguose konfigūravimas nustatomas taip, kad pliūpsniai išeitų per reikiamus išėjimo prievadus. Pirminis mazgas, gavęs valdymo pranešimą po prastovos laiko, nelaukia patvirtinimo apie sujungimo sudarymą, o pradeda duomenų pliūpsnių siuntimą [29].

Pliūpsnių antraštės paketas (BHP) saugo prastovos laiko, duomenų pliūpsnio trukmės (ilgio), pliūpsnių siuntimo dažnumo, QoS ir pliūpsnio pernešimui reikalingo duomenų kanalo informaciją. Dėl BHP perduodamos informacijos nereikalingas buferis tarpiniame mazge. Kai nėra buferio, pliūpsniai gali panaudoti ne vieno kanalo pralaidumo resursus. Tačiau jie vis tiek gali būti prarandami nepasiekę paskirties adreso dėl išėjimo prievadų trūkumo arba pliūpsnių kolizijų [29].

Optinių pliūpsnių komutavimas užtikrina aukštą kanalų panaudojimą optiniame kelyje „galas-galas“ (angl. *end-to-end*). Rezerviniai resursai yra laikomi kiekvieno atskiro pliūpsnio komutavimui ir persiuntimui, kiekviename tinklo komutatoriuje ir išėjimo prievade – tokiu būdų resursai gali būti efektyviau paskirstyti [29].

2.2. OBS tinklo architektūra

OBS tinklą sudaro kraštiniai bei transportiniai maršrutizatoriai, sujungti optinėmis skaidulomis. Kraštiniai maršrutizatoriai atsakingi už pliūpsnių surinkimą ir paskirstymą, transportiniai – už maršrutizavimą ir antraščių apdorojimą [30].



16 pav. OBS tinklas [30]

Paketai surenkami kraštiniuose maršrutizatoriuose į optinius pliūpsnius pagal paskirties adresą bei reikalavimus paslaugos kokybei ir pan. Kraštinis maršrutizatorius yra atsakingas už pliūpsnių tolimesnį perdavimą. Surinkti pliūpsniai yra perduodami OBS tinklu į paskirties adresą (nenaudojant vėlinimo linijų tarpiniuose maršrutizatoriuose). Išėjime (kraštiniame maršrutizatoriuje) pliūpsniai yra išskaidomi į paketus ir toliau persiunčiami į paskirties adresą.

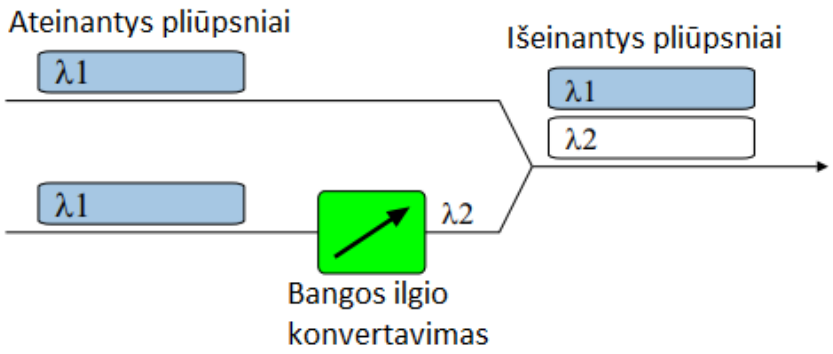
2.3. Optinių pliūpsnių perdavimo problematika

OBS technologijos pagrindinis uždavinys – pliūpsnių surinkimo procesas, kuomet kraštiniuose tinklo mazguose į pliūpsnius (duomenų blokus), surenkami ateinantys paketai iš įvairių interneto šaltinių (maršrutizatorių). Paketai linkę uždelsti, todėl laikas, skirtas paketus surinkti į pliūpsnius, suvėlinamas. Pliūpsnių surinkimo algoritmuose turi būti įskaičiuotas ir leistinas paketų vėlinimo nuokrypis, tam, kad būtų užtikrintas efektyvus perdavimas [28].

Pliūpsnių, esančių tinklo centre dislokuotuose mazguose, nesuderinamumas yra dar vienas svarbus faktorius kalbant apie OBS. Nesuderinamumas (pliūpsnių kolizija) atsiranda, kai du ar daugiau pliūpsnių vienu metu „reikalauja“ to paties kanalo ar bangos ilgio viename išėjime [28].

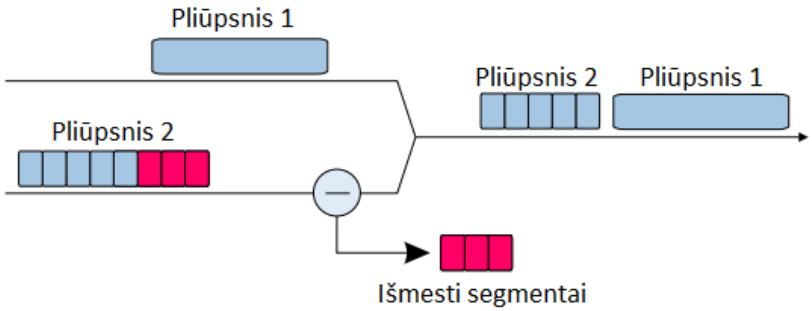
Norint neprarasti pliūpsnių, gali būti panaudojami įvairūs sprendimai bangos ilgio, laiko ir erdvės atžvilgiu. Pliūpsnių kolizijų apėjimo būdai:

- bangos ilgių konvertavimas. Tai yra paprastas kolizijų sprendimo būdas OBS tinkluose. Pagrindinis trūkumas – bangos ilgių konverterių brangumas;



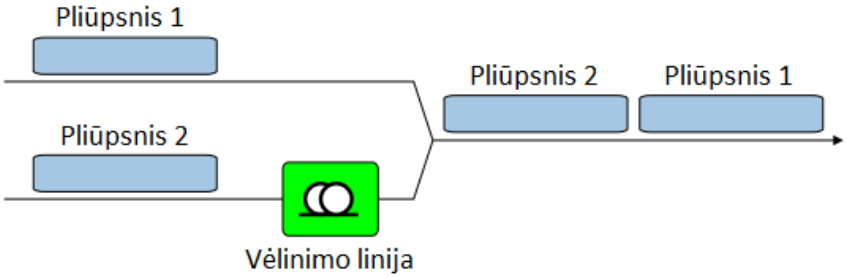
17 pav. Bangos ilgio konvertavimas [31]

- pliūpsnių segmentavimu. Kiekvienas pliūpsnis padalijamas į segmentus ir, pliūpsnių kolizijos atveju, prarandama tik tam tikra dalis pliūpsnio (pradžios ar pabaigos segmentai);



18 pav. Pliūpsnių segmentavimas [31]

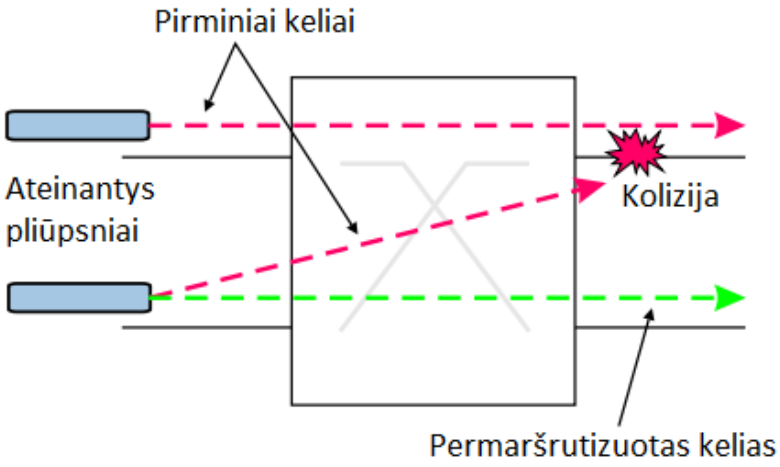
- skaidulos vėlinimo linijomis (FDL). Užlaiko kai kurių pliūpsnių aptarnavimą, taip valdant galimas kolizijas;



19 pav. Skaidulos vėlinimo linijos [31]

- aplinkiniu maršrutizavimu. Pliūpsniai komutavimo matricoje persiunčiami į kitą išėjimo prievadą (skaidulą arba skaidulos šerdį). Šiam būdui įgyvendinti nereikia papildomų įrenginių, todėl jis sąlyginai pigus. Pagrindinė problema, kad pakeitęs kryptį pliūpsnis turi pasiekti tikslą,

nors ir per kitą mazgo išėjimo kryptį. Esama tinklo topologija ir taikomos maršrutizavimo strategijos labiausiai apibrėžia šio metodo efektyvumą;



20 pav. Aplinkinis maršrutizavimas [31]

Jei neįgyvendinamas nė vienas iš pliūpsnių kolizijų apėjimo būdų – pliūpsnis paprasčiausiai yra prarandamas.

MCF tinkluose pliūpsnių koliziją būtų galima spręsti pritaikant skaidulos vėlinimo linijas (buferius) ir aplinkinį maršrutizavimą – perduodant pliūpsnius atskiromis šerdimis. Todėl toliau bus modeliuojami daugiašerdžių skaidulų su optiniu pliūpsnių komutavimu tinklai.

3. Daugiašerdžių skaidulų tinklų su OBS efektyvumo įvertinimas

Šio skyrelio esmė – nustatyti daugiašerdžių skaidulų su optinių pliūpsnių komutavimu efektyvumą. Kad tai būtų įgyvendinta, pirmiausia turi būti apibrėžta efektyvumo įvertinimo metodika. Tuomet programiškai bus kuriami pliūpsnių perdavimo per MCF skaidulas modeliai.

3.1. Erdvinio atskyrimo sutankinimo sistemų efektyvumo įvertinimo metodika

Tam, kad būtų įvertintas erdvinio atskyrimo sutankinimas, turi būti nustatyti tam tikri kriterijai. Efektyvumas priklausys nuo pliūpsnių blokavimo tikimybės, šerdžių užimtumo, pliūpsnių pasiskirstymo tarp kanalų ir persidengimo tarp gretimų kanalų.

Todėl *bendrąjį erdvinio atskyrimo sutankinimo sistemos efektyvumą* galima išreikšti tokia formule:

$$E = \sum_{k=1}^{N_k}(1 - P_b)U_k\, P_k H(XT) \quad (8)$$

Čia N_k – šerdžių skaičius skaiduloje.

Pliūpsnių blokavimo tikimybė (P_b) – tai kintamasis, kuris parodo, kokia tikimybė, kad sistema perduotas pliūpsnis bus prarandamas (blokuojamas). Pliūpsnis yra prarandamas kuomet visos skaidulos šerdys yra užimtos ir užpildytos visos vietos buferyje.

Šerdžių užimtumas (U_k). Šis kintamasis parodo, kokią santykinę laiko dalį bus užimta tam tikra šerdis, prie tam tikro sistemos apkrautumo. Kuo didesnis sistemos apkrautumas tuo didesnis šerdžių užimtumas.

Pliūpsnių pasiskirstymas tarp kanalų (P_k). Tai yra sistema perduodamų pliūpsnių pasiskirstymas tarp skirtingų šerdžių – tikimybė, kad pliūpsnis bus nukreiptas į k-ąją šerdį.

Persidengimas skaiduloje tarp šerdžių (XT). Tai yra perdavimo signalo trukdžiai, kai signalas iš vienos šerdies pasiekia kitas šerdis. Įvertinant persidengimo įtaką efektyvumui, bus naudojama Hevisaido funkcija:

$$H(n) = \begin{cases} 0, & n < 0; \\ 1, & n \geq 0. \end{cases} \quad (9)$$

Persidengimo vertės bus lyginamos su kritinėmis vertėmis ($H(XT_{kr} - XT)$). Kritinės daugiašerdžių optinių skaidulų persidengimų vertės prie tam tikrų moduliacijų yra nustatytos [23] šaltinyje.

Tam, kad galima būtų apskaičiuoti pliūpsnių blokavimo, šerdžių užimtumo, pliūpsnių pasiskirstymo tarp kanalų tikimybes bei persidengimo vertes reikia paruošti erdvinio atskyrimo OBS sistemų su daugiašerdėmis skaidulomis modelius.

3.2. Markovo procesų Modeliavimas

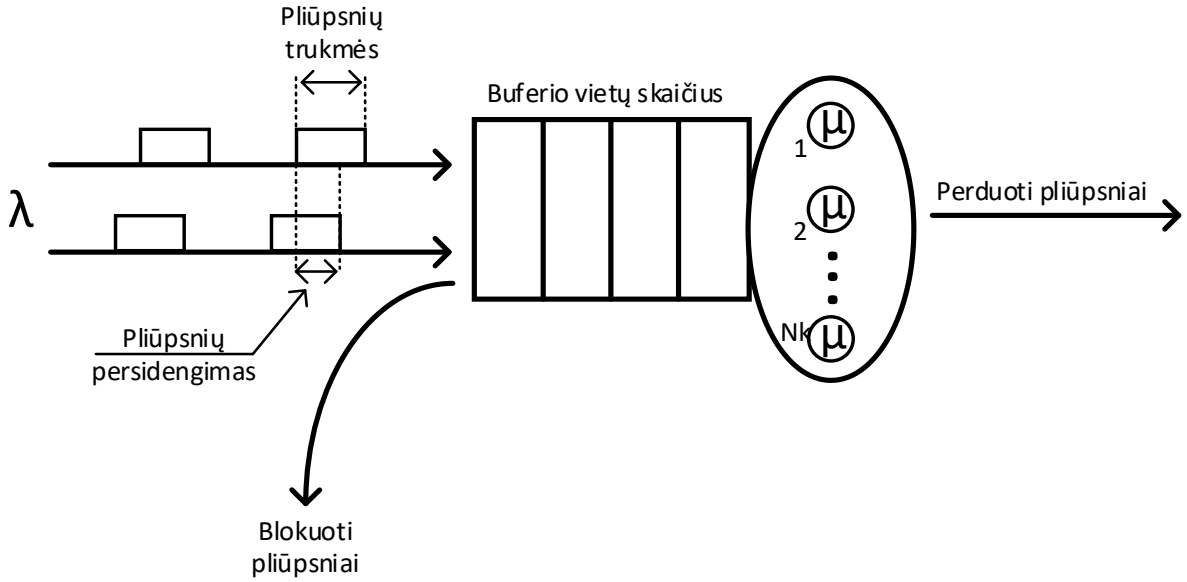
Taikant matematinius modelius bus nagrinėjama MCF tinklų OBS pliūpsnių kolizija, pritaikant optines vėlinimo linijas (buferius) ir aplinkinį maršrutizavimą – perduodant pliūpsnius atskiromis šerdimis.

3.2.1. SDM sistemos su dvejų šerdžių skaidula Markovo procesų modelis

Pagal Markovo procesus, sudaromas perdavimo per dvejų šerdžių skaidulą būsenų grafas. Grafas sudarytas iš būsenų, kurias nusako dų skaitmenys: pirmas skaitmuo simbolizuoja pirmosios šerdies

užimtumą, antras skaitmuo – antrosios. Tai reiškia, kad būsena 00 simbolizuoja, jog nė viena šerdimi nėra perduodamas signalas, būsena 11 simbolizuoja, jog abejomis šerdimis yra perduodamas signalas, būsena 10 simbolizuoja jog signalas perduodamas pirmąja šerdimi, antroji laisva, o būsena 01 – priešingai.

Principinė modelio schema pateikiama 21 pav.



21 pav. Principinė modelio schema

Markovo grandinėje bus naudojami kintamieji λ (optinių pliūpsnių atėjimo į sistemą intensyvumas) ir μ (pliūpsnių perdavimo intensyvumas). Kintamieji aprašomi (10) ir (11) lygtimis.

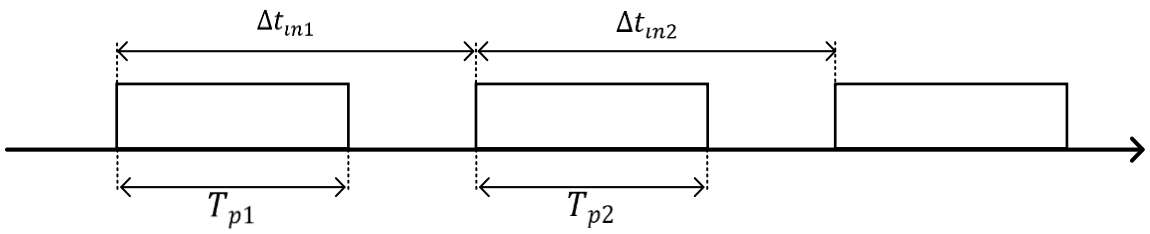
$$\lambda = \frac{1}{\overline{\Delta t_{in}}} \tag{10}$$

čia $\overline{\Delta t_{in}}$ – vidutinis laikotarpis tarp pliūpsnių pradžios laiko momentų.

$$\mu = \frac{1}{\overline{T_p}} \tag{11}$$

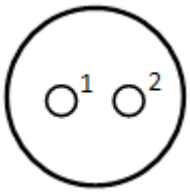
čia $\overline{T_p}$ – vidutinė optinių pliūpsnių trukmė

Optiniai pliūpsniai su laikiniais parametrais grafiškai pavaizduoti 22 pav.



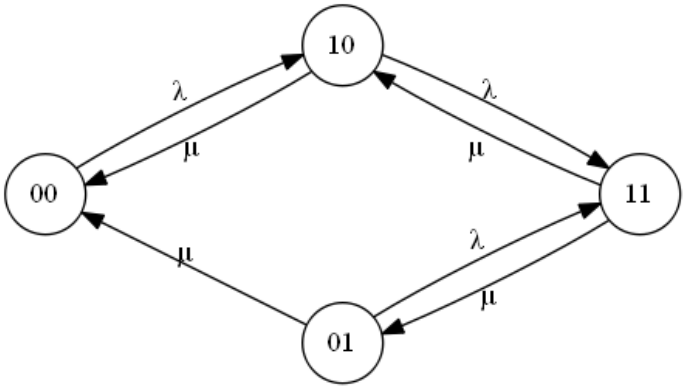
22 pav. Optinių pliūpsnių laikiniai parametrai

Modeliuojant, skaidulos šerdims suteikti prioritetai (1 žymi didžiausią prioritetą) pateikiami 23 paveiksle.



23 pav. Skaidulos šerdžių prioritetai

Naudojantis „python“ programavimo kalbos „graphviz“ programine įranga, buvo sumodeliuota grandinė pagal Markovo procesą pateikiama 24 pav. Programos kodas pateikiamas 1 darbo prieduose.



24 pav. Skaidulos šerdžių būsenų grafas, naudojant 2 šerdžių skaidulą be buferio

Tokiam būsenų grafui galima sudaryti tiesinių lygčių sistemą, kurią išsprendus galima apskaičiuoti sistemos būsenų tikimybes. Sudaryta lygčių sistema:

$$\begin{cases} -\lambda P_{00} + \mu P_{10} + \mu P_{01} = 0 \\ \lambda P_{00} - (\lambda + \mu) P_{10} + \mu P_{11} = 0 \\ -(\lambda + \mu) P_{01} + \mu P_{11} = 0 \\ -2\mu P_{11} + \lambda P_{10} + \lambda P_{01} = 0 \\ P_{00} + P_{10} + P_{01} + P_{11} = 1 \end{cases} \quad (12)$$

Iš lygčių sistemos išsireiškiama kiekvienos būsenos tikimybė.

Tikimybė, kad nebus užimta nei viena šerdis (P_{00}) pateikiama (13 lygtyje):

$$P_{00} : \frac{2\mu^2}{\lambda^2 + 2\mu(\lambda + \mu)} \quad (13)$$

Tikimybė, kad bus užimta tik antroji šerdis (P_{01}) pateikiama (14) lygtyje:

$$P_{01} : \frac{\lambda^2 \mu}{(\lambda + \mu)\left(\lambda^2 + 2\mu(\lambda + \mu)\right)} \quad (14)$$

Tikimybė, kad bus užimta pirmoji šerdis (P_{10}) pateikiama (15) lygtyje:

$$P_{10} : \frac{\lambda\mu(\lambda + 2\mu)}{(\lambda + \mu)\left(\lambda^2 + 2\mu(\lambda + \mu)\right)} \tag{15}$$

Tikimybė, kad bus užimtos abi šerdys (P_{11}) pateikiama (16) lygtyje:

$$P_{11} : \frac{\lambda^2}{\lambda^2 + 2\mu(\lambda + \mu)} \tag{16}$$

Turint kiekvienos būsenos išraiškas, bus modeliuojamas optinio perdavimo 2 šerdžių skaiduloje apkrautumas.

Tam, kad būtų sudarytas grafikas, bus apskaičiuojama tikimybė, kad užimta pirmoji šerdis (U_1). Ši tikimybė apskaičiuojama pagal (17) lygtį.

$$U_1 = P_{10} + P_{11} \tag{17}$$

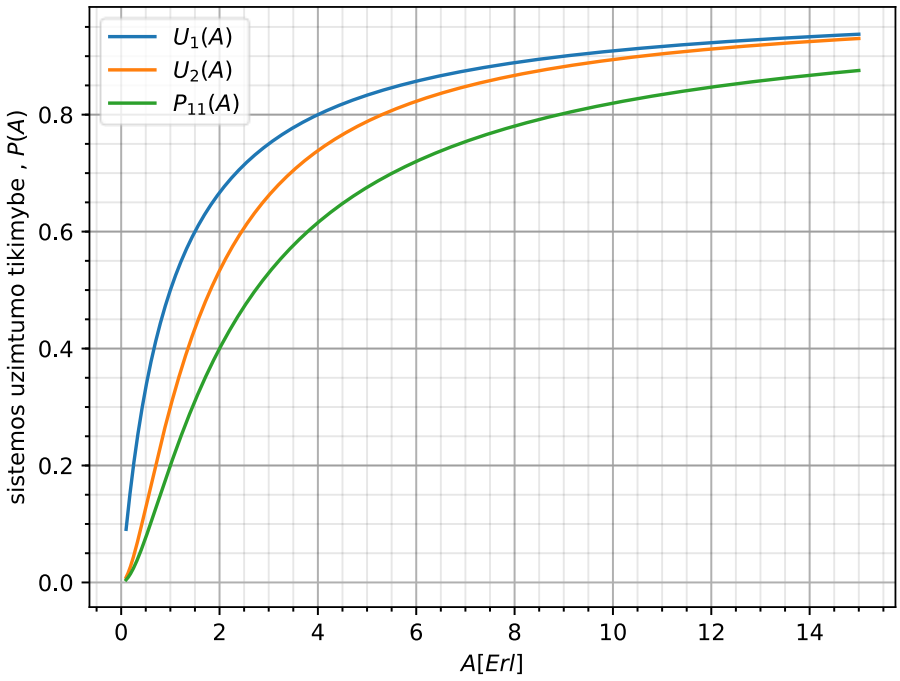
Tikimybė, kad užimta antra šerdis (U_2), apskaičiuojama iš (18) lygties.

$$U_2 = P_{01} + P_{11} \tag{18}$$

Sistemos apkrautumui atvaizduoti naudojamas kintamasis A (sukurtoji apkrova). Šis kintamasis matuojamas Erlangais (E). Tai yra perdavimo srauto tankio vienetas telekomunikacijų sistemose. Erlangas apibūdina bendrą srautą per laiko vienetą. Pavyzdžiui, jei per valandą sulaukiame 60 skambučių, kurie trunka po 5 minutes, tai tokios sistemos apkrautumas bus 5 Erlangai. Šiuo atveju sistemos apkrautumas bus lygus optinių pliūpsnių atėjimo į sistemą intesyvumo (λ) ir pliupsnių perdavimo intesyvumo (μ) santykiui (19 formulė).

$$A = \frac{\lambda}{\mu} \tag{19}$$

Sistemos apkrautumo grafikas, kuomet sistemos sukurtoji apkrova kinta nuo 0 iki 15 Erl pateikiamas 25 pav. Grafike $P_{11}(A)$ žymi sistemos blokavimo tikimybę, kuri efektyvumo įvertinimo metodikoje aprašyta kaip P_b.



25 pav. Sistemos su dvejomis šerdimis apkrautumo grafikas

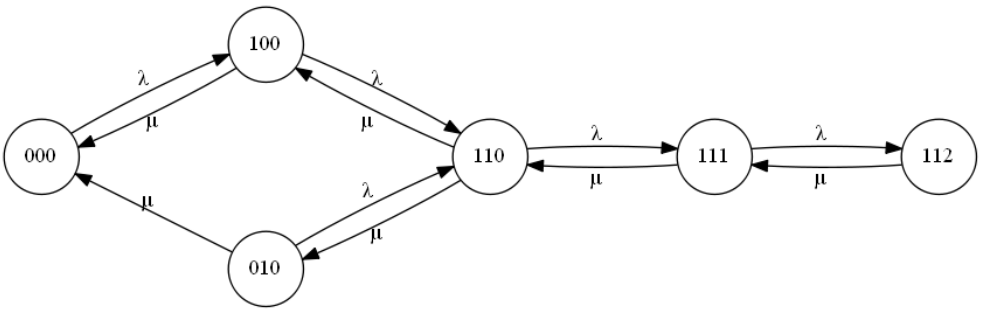
Iš grafiko galime pastebėti kaip sistemos visų šerdžių užimtumo tikimybė didėja, kuomet didėja sistemos apkrautumas.

Pavyzdžiui, kuomet sistemos sukurtoji apkrova A = 2 Erl, blokavimo tikimybė lygi net $P_{11}(A) =$ 0,4. Reiškia sistema su dviem šerdimis turi 40% tikimybę prie šio apkrautumo prarasti duomenų pliūpsnius.

3.2.2. SDM sistemos su dvejų šerdžių skaidula ir buferiu Markovo procesų modelis

Lygiai tokiu pačiu principu sumodeliuojamas ir dviejų šerdžių su buferiu Markovo procesas. Grafų būsenos simbolizuoja taip pat kaip ir prieš tai – pirmos ir antros šerdies užimtumą, tačiau atsiranda trečiasis skaitmuo, kuro reikšmė – buferio užimtumas. Buferis turės 2 vietas, todėl grafo būsenos trečiasis skaitmuo gali kisti nuo 0 iki 2. Skaidulos šerdžių prioritetai išlieka tokie pat, kaip ir buvo.

Sudarytas būsenų grafas pateikiamas 26 paveiksle.



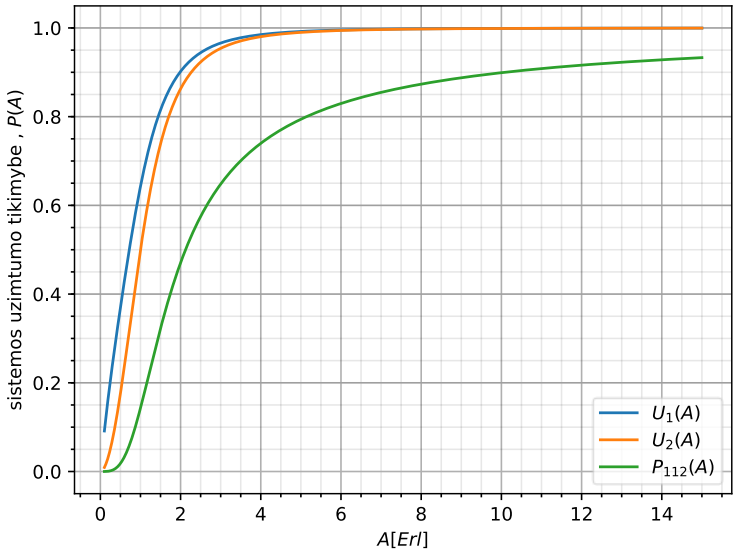
26 pav. Skaidulos šerdžių būsenų grafas, naudojant 2 šerdžių skaidulą su buferiu

Sudaryta lygčių sistema:

$$\begin{cases} -\lambda P_{000} + \mu P_{100} + \mu P_{010} = 0 \\ \lambda P_{000} - (\lambda + \mu) P_{100} + \mu P_{110} = 0 \\ -(\lambda + \mu) P_{010} + \mu P_{110} = 0 \\ -(\lambda + 2\mu) P_{110} + \lambda P_{100} + \lambda P_{010} + \mu P_{111} = 0 \\ -(\lambda + \mu) P_{111} + \lambda P_{110} + \mu P_{112} = 0 \\ -\mu P_{112} + \lambda P_{111} = 0 \\ P_{000} + P_{100} + P_{010} + P_{110} + P_{111} + P_{112} = 1 \end{cases} \quad (20)$$

Programos kodas kartu su gautomis tikimybių lygtimis pateikiamas 1 darbo prieduose.

Sistemos apkrautumo grafikas, kuomet sistemos sukurtoji apkrova kinta nuo 0 iki 15 Erl pateikiamas 27 pav. Grafike $P_{112}(A)$ žymi sistemos blokavimo tikimybę, kuri efektyvumo įvertinimo metodikoje aprašyta kaip P_b.



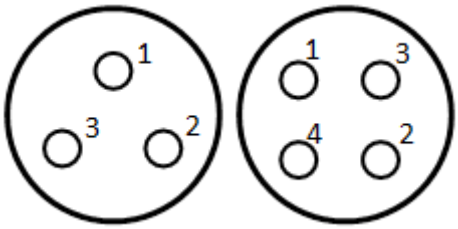
27 pav. Sistemos su dvejomis šerdimis apkrautumo grafikas

Lyginant tą pačią sistemą su buferiu ir be buferio, galima pastebėti, jog sistema su buferiu daug efektyviau išnaudoja šerdis. Taip pat sistemos su buferiu blokavimo tikimybė yra mažesnė.

3.2.3. SDM sistemų su trijų, keturių šerdžių skaidulomis Markovo proceso modelis

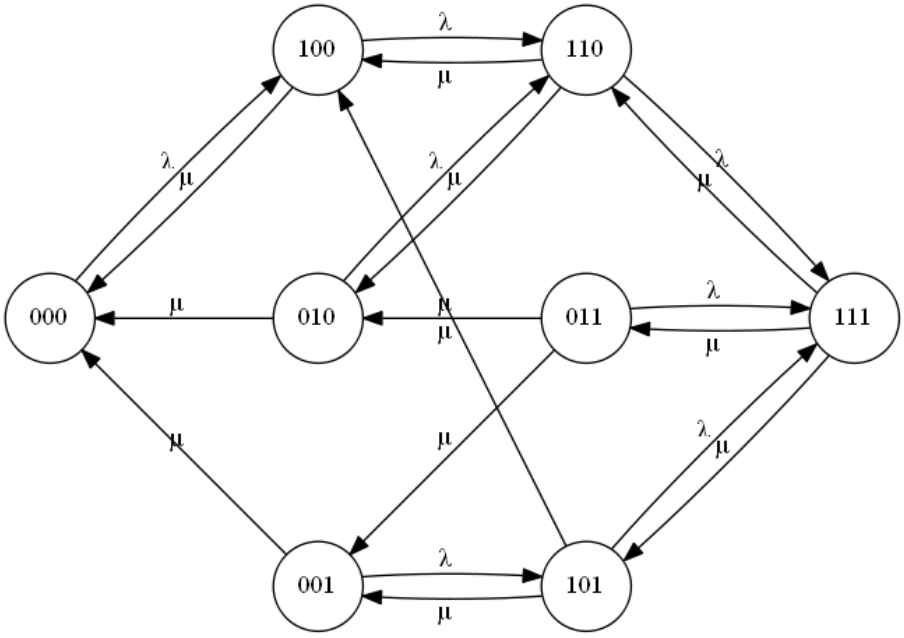
Taip pat, kaip ir buvo modeliuojamas Markovo procesas su dviem šerdim, bus modeliuojamas su trimis bei keturiomis šerdimis.

Skaidulos šerdims suteikti prioritetai (1 žymi reikšmingiausią prioritetą) pateikiami 28 paveiksle.



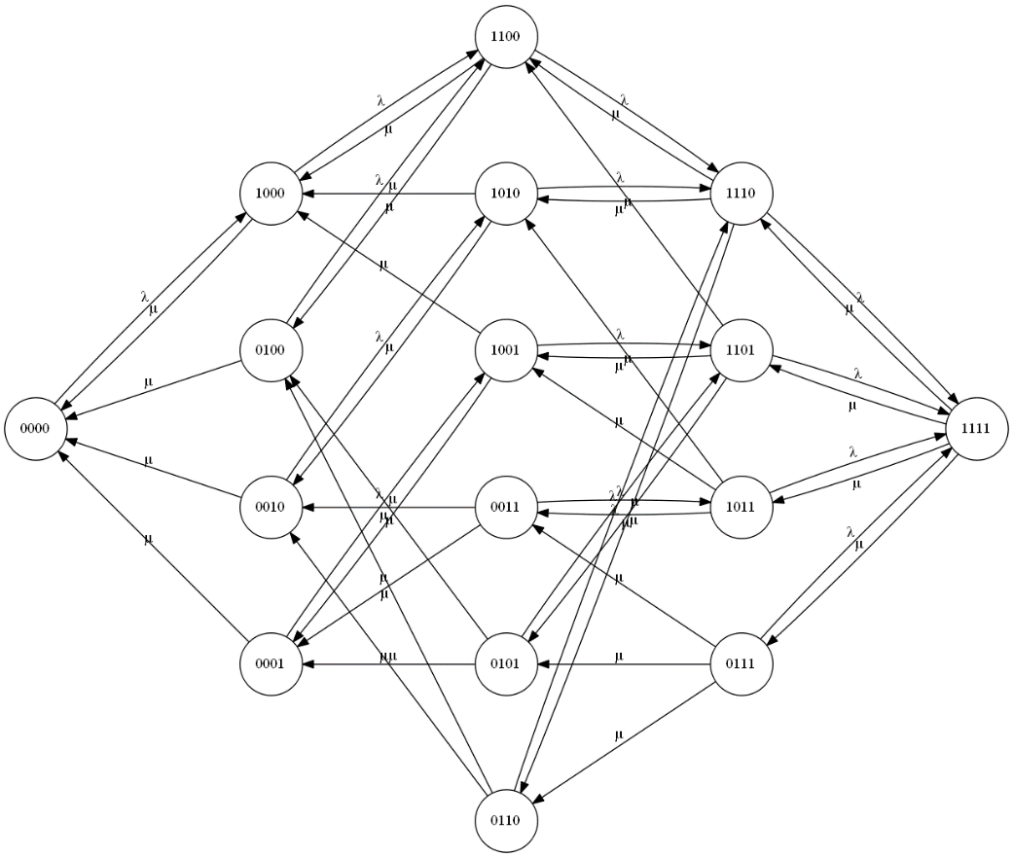
28 pav. Skaidulų šerdžių prioritetai

Naudojantis „python“ programavimo kalbos „graphviz“ programine įranga, buvo sumodeliuotos grandinės pagal Markovo procesą pateikiamos 29, 30 pav. Programos kodas kartu su gautomis lygtimis pateikiamas 2 darbo prieduose.



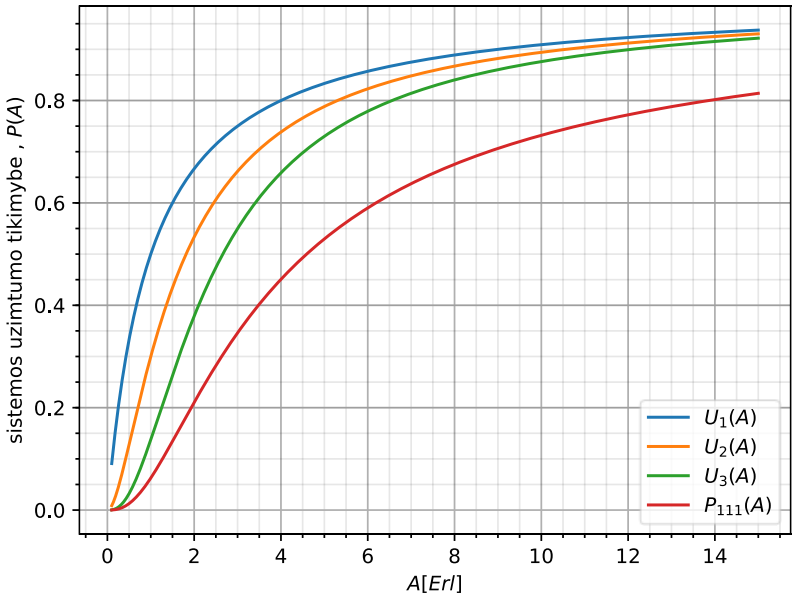
29 pav. Skaidulos šerdžių būsenų grafas, naudojant 3 šerdžių skaidulą be buferio

Lygčių sistemos, skaičiavimai pateikiami 2 darbo prieduose.

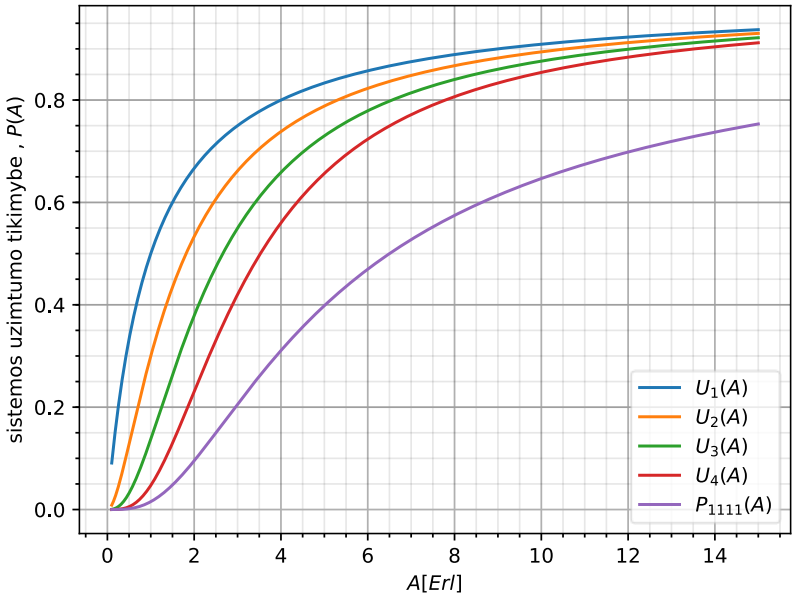


30 pav. Skaidulos šerdžių būsenų grafas, naudojant 4 šerdžių skaidulą be buferio

Sistemų apkrautumo grafikai, kuomet sistemos sukurtoji apkrova kinta nuo 0 iki 15 Erl pateikiami 31, 32 pav.



31 pav. Sistemos su trimis šerdimis apkrautumo grafikas



32 pav. Sistemos su keturiomis šerdimis apkrautumo grafikas

Grafikuose $P_{111}(A)$ ir $P_{1111}(A)$ tai blokavimo tikimybės, kurios efektyvumo įvertinimo metodikoje aprašytos kaip P_b. Galima pastebėti, jog blokavimo tikimybė $P_{1111}(A)$ lyginant su $P_{111}(A)$ sistemoje, turinčioje keturias šerdis, sumažėja apytiksliai 5%.

3.3. Imitacinio modelio kūrimas

Didinant šerdžių skaičių sudėtingėja Markovo procesų analitinis modelis. Tai galima pastebėti iš 24, 26, 29, 30 paveikslų. Kaip matome 30 paveiksle, 4 šerdžių skaidula turi net 16 būsenų grafų (2^4), kuriuos visus reikia aprašyti atskirai. Todėl norint sumodeliuoti 7 šerdžių skaidulą reikėtų aprašyti net 128 (2^7) būsenų grafus. Dėl šios priežasties kuriamas imitacinis modelis. Imitacinis modelis taip pat bus kuriamas „python“ programavimo kalba.

Keičiant šerdžių skaičių, buferio vietų skaičių, paraiškų aptarnavimo ir atėjimo intensyvumą (sistemos apkrautumą), skaidulos ilgį bus galima įvertinti daugiašerdžių skaidulų su optiniu pliūpsnių komutavimu efektyvumą.

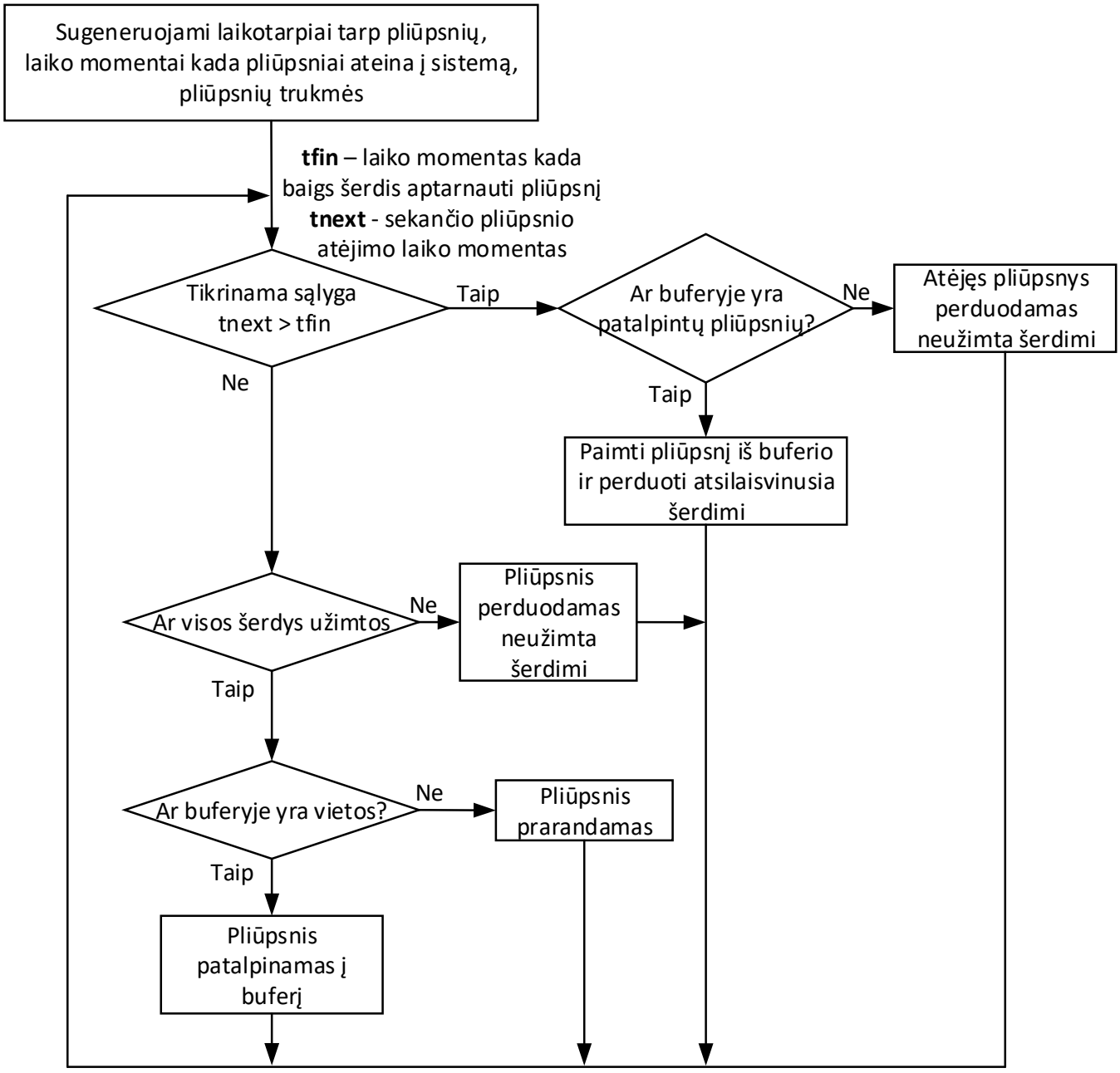
Kuriant imitacinį modelį pirmiausiai sudaromas algoritmas, kurio principu turi veikti visa daugiašerdžių skaidulų su optiniu pliūpsnių komutavimu sistema.

Pradžioje sugeneruojami pliūpsniai su visai jiems priklausančiais kintamaisiais (pliūpsnių skaičius, pliūpsnių trukmės, laikotarpiai tarp pliūpsnių, laiko momentai, kada pliūpsniai ateina į sistemą). Tuomet yra patikrinama kas įvyks greičiau, t.y. greičiau ateis sekantis pliūpsnis ar greičiau aptarnaus dabar atėjusį pliūpsnį nei ateis sekantis.

Tokiu atveju, jei sekančio pliūpsnio atėjimo laiko momentas yra didesnis už laiko momentą kada šerdis baigs aptarnauti dabartinį pliūpsnį, bus tikrinama ar buferyje yra patalpintų pliūpsnių – jei taip tai pliūpsnis iš buferio bus perduodamas atsilaisvinusia šerdimi, jei ne – atėjęs pliūpsnis bus perduodamas laisva šerdimi.

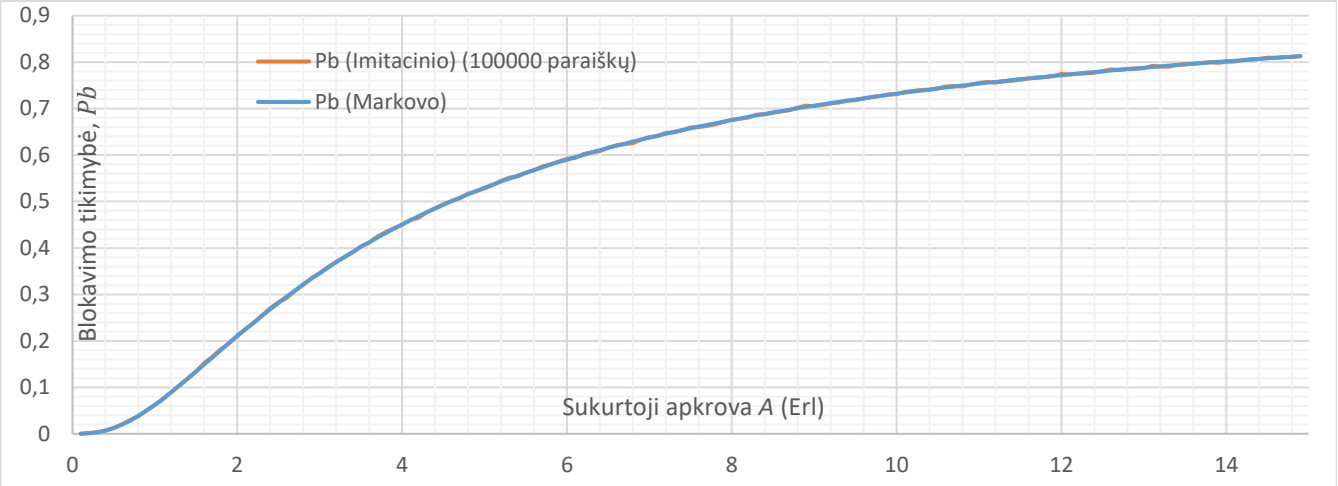
Jei sekančio pliūpsnio atėjimo laiko momentas yra mažesnis už laiko momentą kada šerdis baigs aptarnauti dabartinį pliūpsnį, bus tikrinama ar yra laisvų šerdžių. Jei yra – atėjęs pliūpsnis perduodamas jomis, jei nėra, tuomet tikrinama ar buferyje vietos ir jeigu nėra, tuomet pliūpsnis yra prarandamas.

Imitacinio modelio algoritmas pateikiamas 33 pav.



33 pav. Imitacinio modelio algoritmas

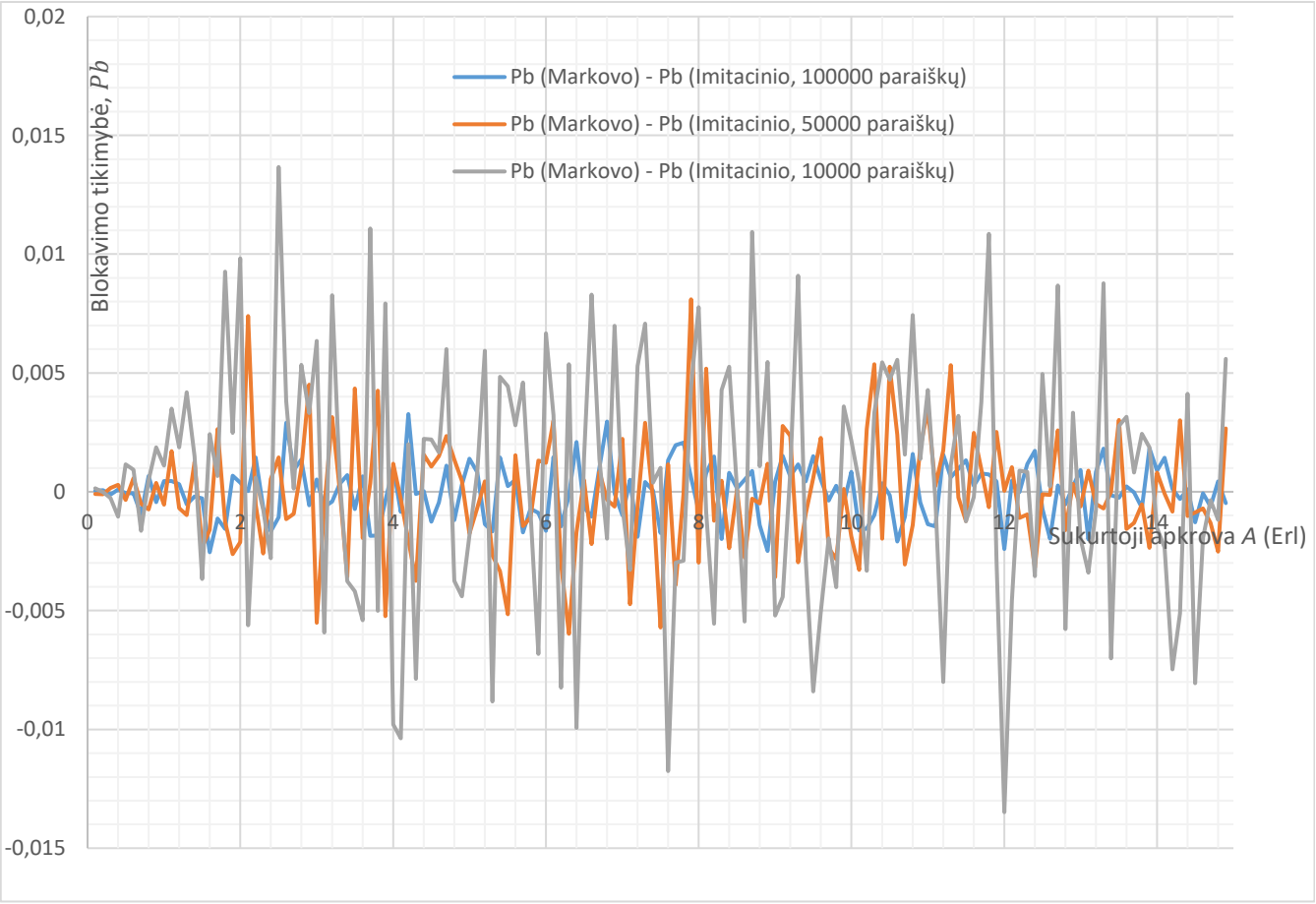
Sukūrus imitacinį modelį, galime patikrinti, ar rezultatai sutampa su prieš tai gautais iš matematinio Markovo proceso. Palyginsime 3 šerdžių sistemos be buferio blokavimo tikimybę gautą Markovo pricesu ir imitaciniu modeliu. Grafikų palyginimas pateikiamas 34 pav.



34 pav. 3 šerdžių sistemos blokavimo tikimybė gauta modeliuojant imitaciniu modeliu ir Markovo procesu

Lyginant imitacinį su matematiniu modeliu rezultatai sutampa, tačiau imitacinio modelio rezultatuose galime pastebėti tam tikrus reikšmių išsibarstymus, nes imitacinis modelis generuoja baigtines atsitiktinių reikšmių imtis.

35 pav. Grafike pateikiami analitinio ir imitacinio modelio skirtumai ($P_b(Markovo) - P_b(Imitacinio)$), kuomet imitacinis modelis sudarytas su 100000, 50000, 10000 paraiškų.



35 pav. Gauti analitinio ir imitacinio modelio 3 šerdžių skaidulos blokavimo tikimybės skirtumai

Iš 35 pav. matome jog imitacinio modelio reikšmių išsibarstymus galima sumažinti didinant modeliavimo trukmę, nes jie asimptotiškai artėja prie reikšmių, kurios gaunamos nusistovėjusio proceso atveju.

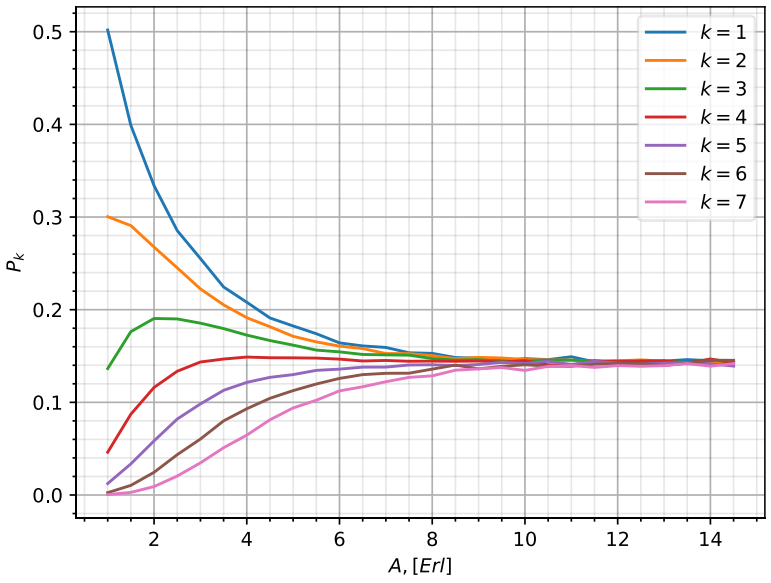
Imitacinio modelio programos kodas pateikiamas 3 darbo prieduose.

3.3.1. Maksimalaus skaidulos ilgio radimas

Šerdimis perduodamų signalų persidengimas daugiašerdėje skaiduloje (XT) turi įtakos maksimaliam skaidulos ilgiui. Pavyzdžiui, iš [23] šaltinio randame, kad 100Gb/s DP-QPSK moduliacijai kritinė persidengimo vertė $XT_{kr} = -18$ (dB).

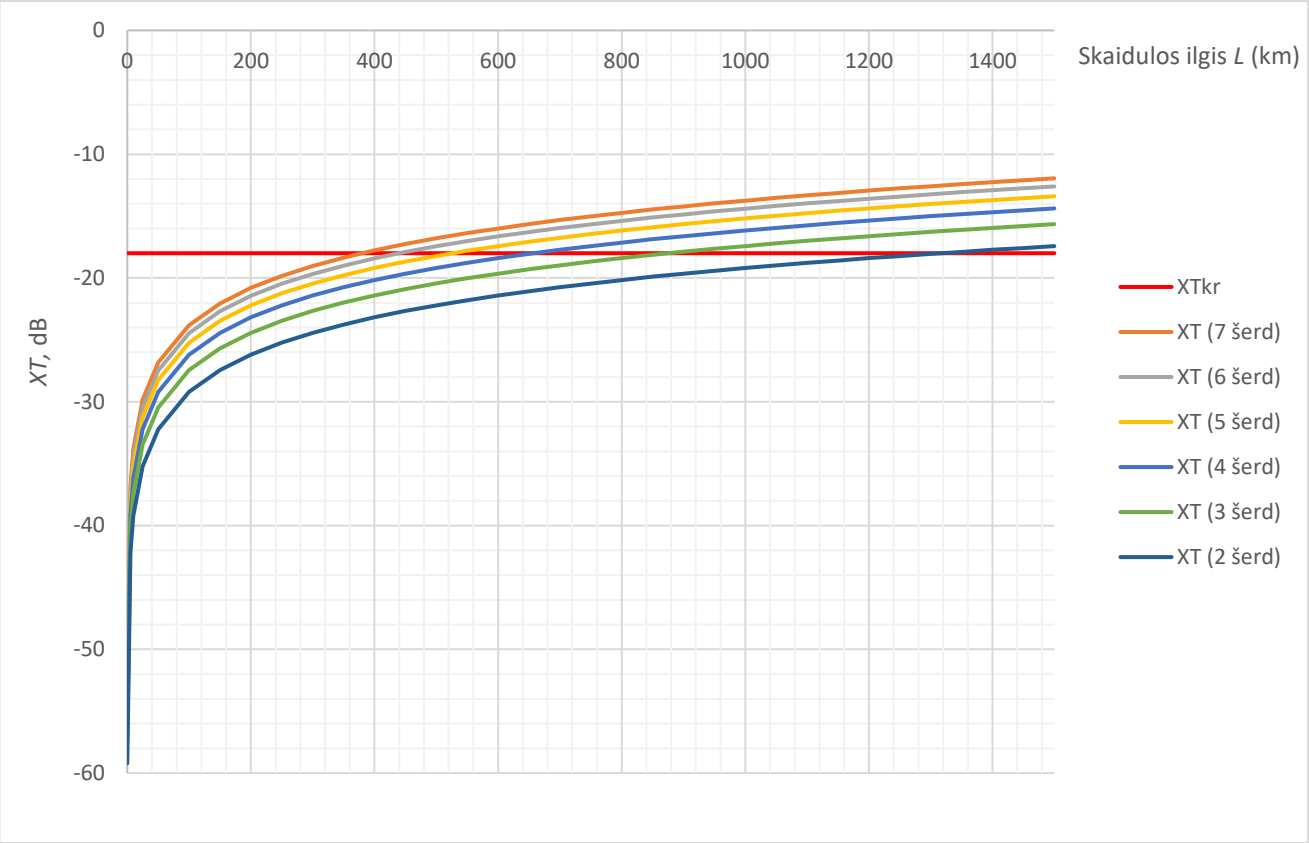
Žinant maksimalią leistiną persidengimo daugiašerdėje skaiduloje vertę, galima nustatyti didžiausią galimą skaidulos ilgį, kuri tenkina sąlyga: $XT < XT_{kr}$.

Didinant sistemos apkrautumui kartu didėja ir persidengimas. Nagrinėjant persidengimą tarp šerdžių, 7 šerdžių skaiduloje laikysime, kad sistema bus apkrauta $A = 10$ (Erl) apkrova. 36 paveiksle pateikiamas pliūpsnių pasiskirstymo tarp 7 šerdžių grafikas. Iš šio grafiko matyti, kad 10 Erlangų vertė pasirenkama, nes prie šios apkrovos pliūpsnių pasiskirstymas tarp kanalų tampa vienodas, visos šerdys užimtos. Todėl, didėjant apkrovai ir persidengimo vertės nebedidės. Ši apkrova bus paliekama ir modeliuojant mažiau šerdžių turinčių skaidulų didžiausiam įmanomam ilgiui nustatyti.



36 pav. Pliūpsnių pasiskirstymo tarp 7 šerdžių grafikas

Didžiausi galimi daugiašerdžių sistemų skaidulų ilgiai, tenkinantys sąlygą $XT < XT_{kr}$ matomi 37 paveiksle. Taip pat šie ilgiai pateikiami 3 lentelėje.



37 pav. Didžiausi galimi daugiašerdžių skaidulų ilgiai

3 lentelė. Didžiausi daugiašerdžių skaidulų ilgiai, tenkinantys $XT < XT_{kr}$ sąlygą

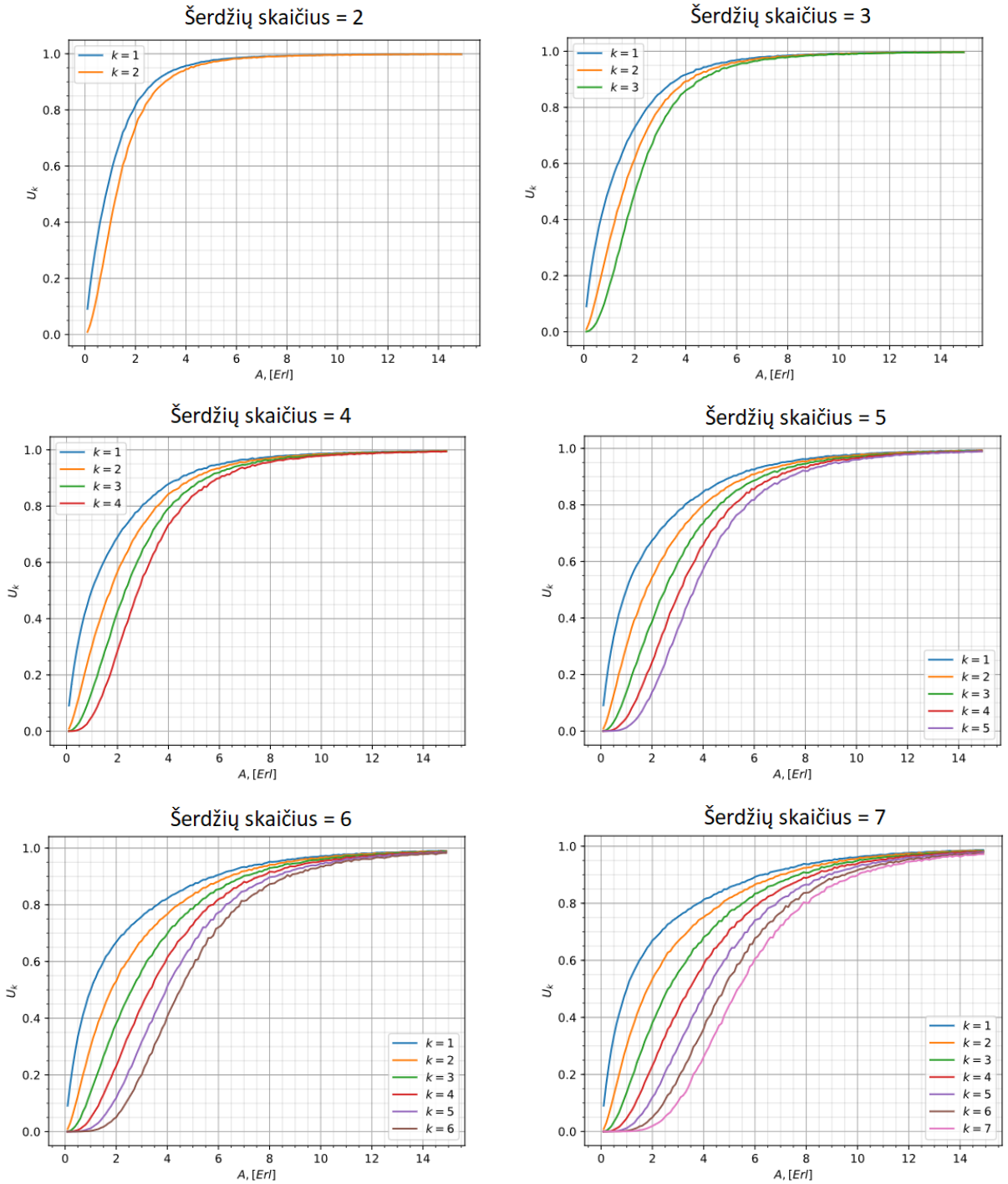
Skaidulos tipas	**Didžiausias galimas skaidulos ilgis**
2 šerdžių skaidula	1320 km
3 šerdžių skaidula	870 km
4 šerdžių skaidula	650 km
5 šerdžių skaidula	530 km
6 šerdžių skaidula	440 km
7 šerdžių skaidula	380 km

Detalūs persidengimų skaičiavimų rezultatai pateikiami 3 darbo prieduose.

3.3.2. Šerdžių užimtumas

Šerdis laikoma užimta, kuomet ja yra aptarnaujamas optinis pliūpsnis. Modeliuojant pirmiausiai bus atsižvelgiama į šerdžių skaičiaus skaiduloje įtaką šerdžių užimtumui, kuomet buferio talpa nekeičiama.

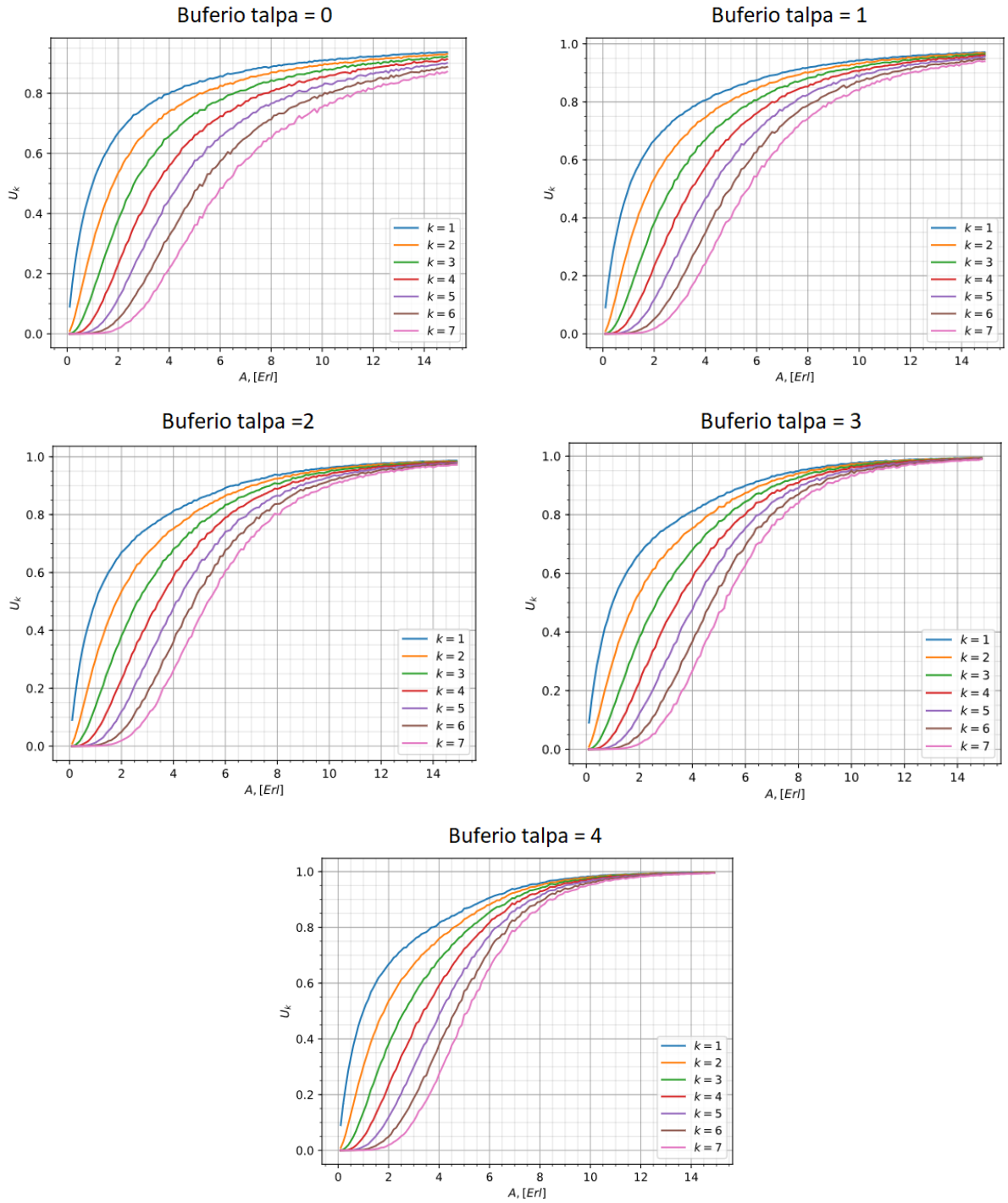
38 paveiksle pavaizduotas šerdžių užimtumas 2-7 šerdžių skaidulose, kuomet buferio talpa – 2. Čia k – šerdies numeris skaiduloje



38 pav. Šerdžių užimtumas 2-7 šerdžių skaidulose (buferio talpa – 2)

Iš grafikų galime matyti, kad kuo daugiau šerdžių skaidulose, tuo mažiau, prie tų pačių apkrovų, užimamos paskutinės pagal prioritetą šerdys.

Taip pat norint nustatyti buferio įtaką šerdžių užimtumui bus keičiama buferio talpa 7 šerdžių skaiduloje nuo 0 iki 5. Grafikai vaizduojami 39 paveiksle.



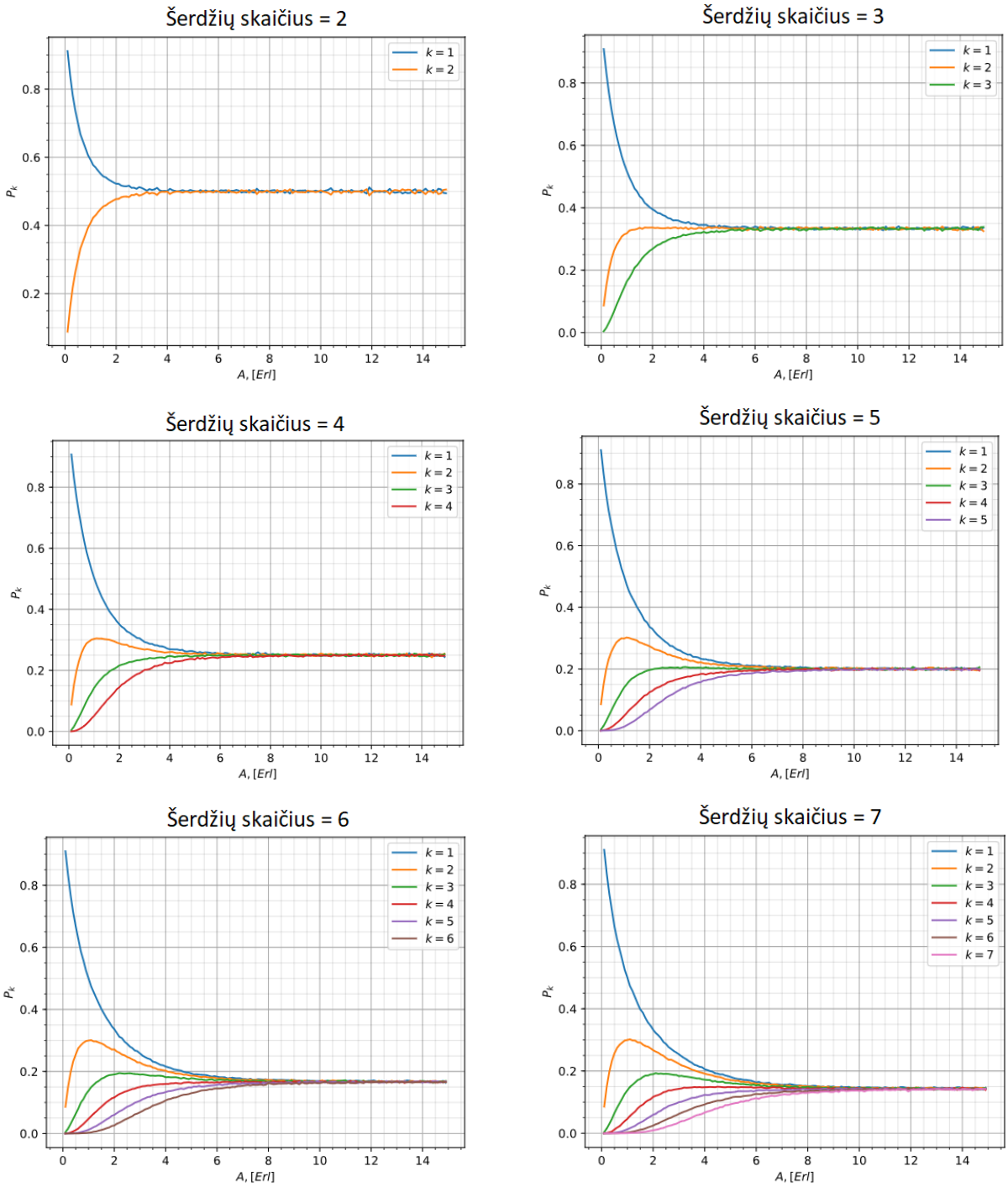
39 pav. Šerdžių užimtumas 7 šerdžių skaiduloje, keičiant buferio talpą nuo 0 iki 4

Iš grafikų aiškiai matosi, jog didinant buferio talpą labiau išnaudojamos šerdys. Taip yra dėl to, jog iš karto perduodamas sekantis duomenų pliūpsnis iš buferio kuomet šerdis baigia aptarnauti atėjusį duomenų pliūpsnį, o naujai į sistemą atėję duomenų pliūpsniai yra talpinami buferyje, jei visos šerdys užimtos.

3.3.3. Pliūpsnių pasiskirstymas tarp šerdžių

Pliūpsnių pasiskirstymas tarp šerdžių parodo, kuria skaidula bus perduodami į sistemą atėję duomenų pliūpsniai.

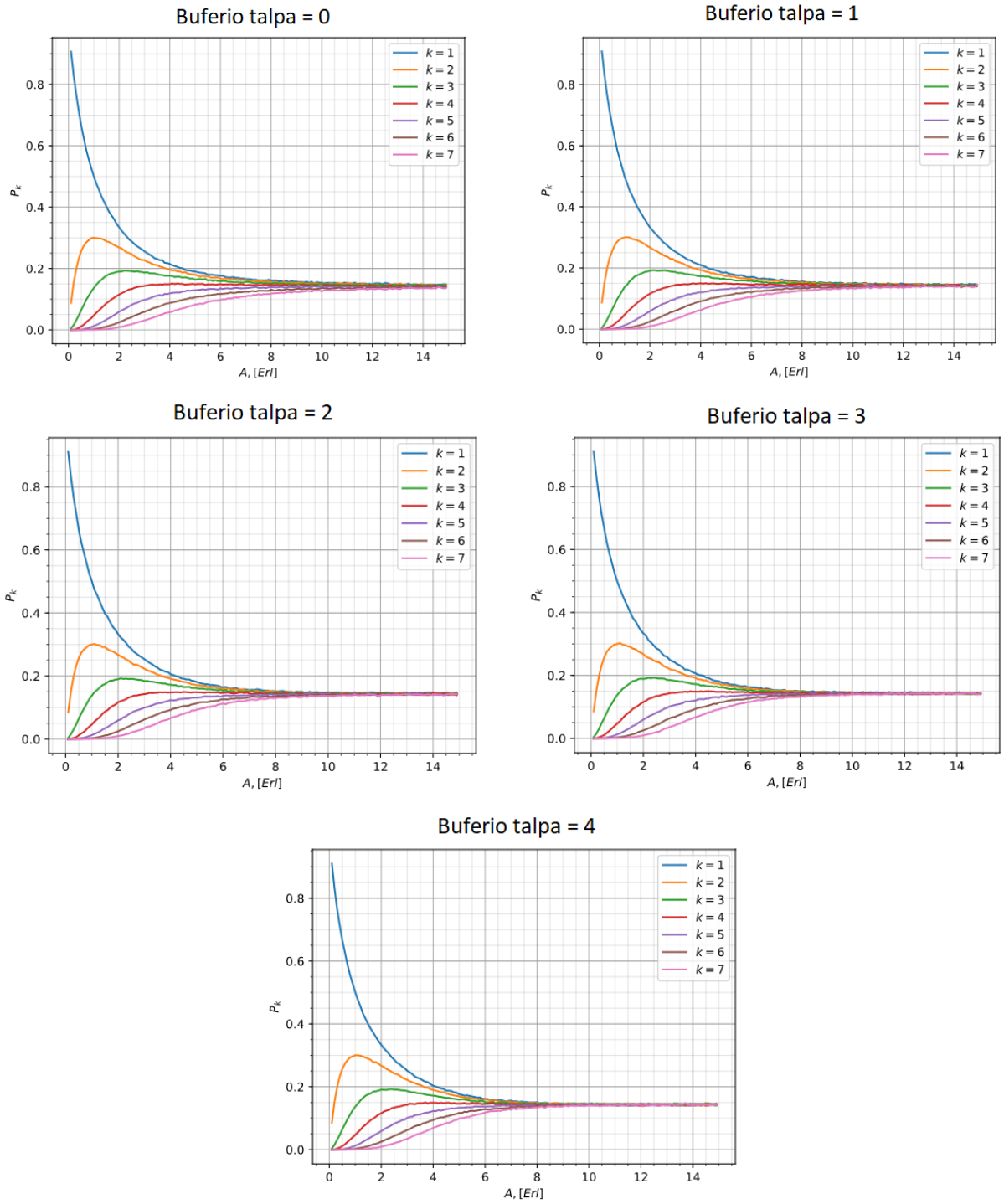
40 paveiksle pavaizduotas pliūpsnių pasiskirstymas tarp šerdžių 2-7 šerdžių skaidulose, kuomet buferio talpa – 2. Čia *k* – šerdies numeris skaiduloje.



40 pav. Pliūpsnių pasiskirstymas tarp šerdžių 2-7 šerdžių skaidulose (buferio talpa – 2)

Buvo gauta, kad kuo mažiau šerdžių skaidulose, tuo prie mažesnių apkrovų pliūpsniai pasiskirsto vienodai tarp šerdžių.

Nustatant buferio įtaką pliūpsnių pasiskirstymui tarp šerdžių, bus keičiama buferio talpa 7 šerdžių skaiduloje nuo 0 iki 5. Grafikai vaizduojami 41 paveiksle.



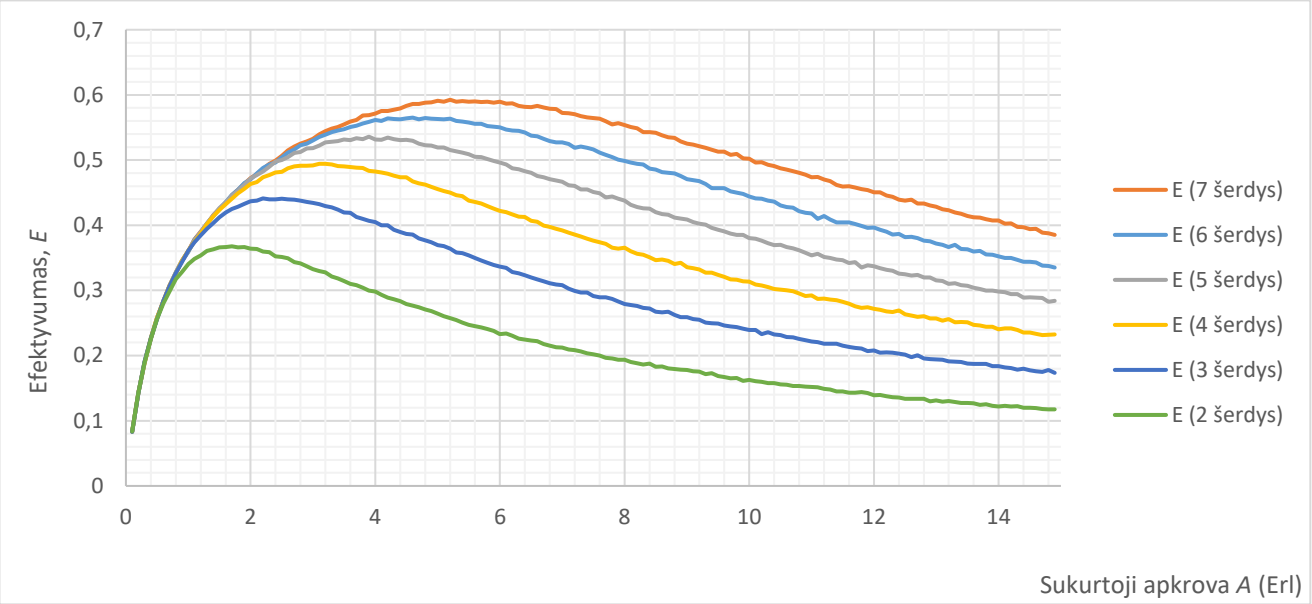
41 pav. Pliūpsnių pasiskirstymas tarp 7 šerdžių, keičiant buferio talpą nuo 0 iki 4

Kuo didesnė buferio talpa, tuo labiau pliūpsniai pasiskirsto tarp šerdžių.

3.3.4. Galutinis efektyvumo įvertinimas

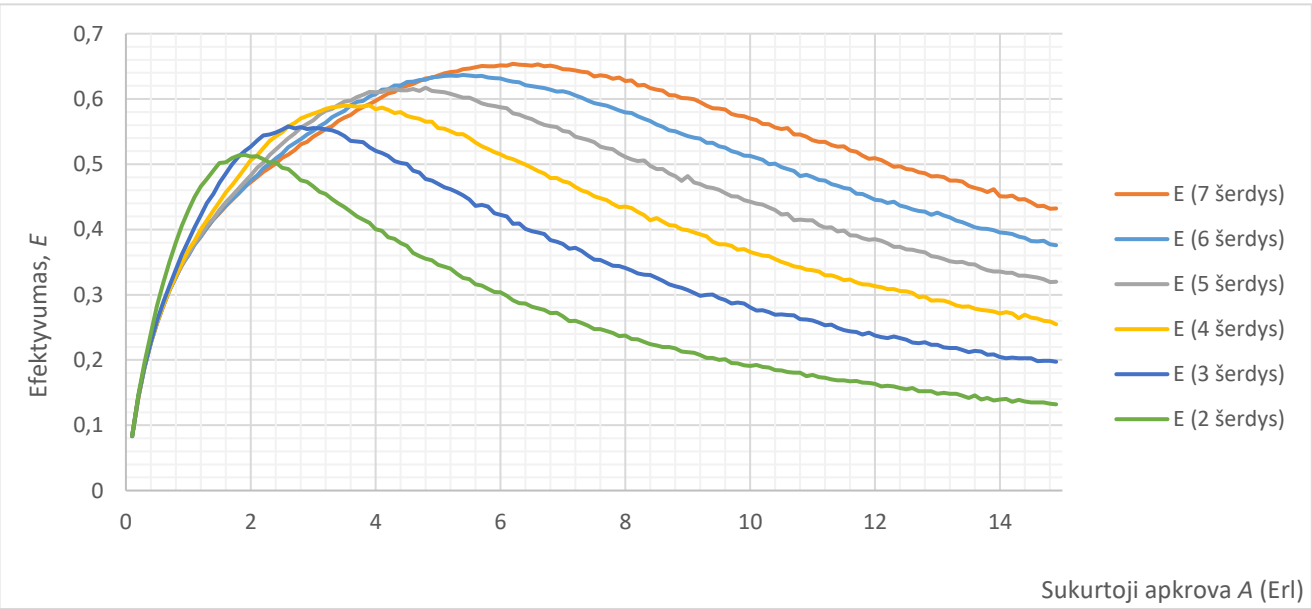
Efektyvumas bus įvertinamas naudojant prieš tai aprašytą metodiką ir (8) formulę.

Vertinant efektyvumą bus pasirenkamas 100km skaidulos ilgis, tenkinantis $XT < XT_{kr}$ sąlygą. Bus lyginami SDM sistemų su 2-7 šerdžių skaidulomis efektyvumai, prie tos pačios buferio talpos. 42 paveiksle pavaizduota SDM sistemų turinčių skaidulas su 2-7 šerdimis efektyvumo įvertinimas (buferio talpa – 0).



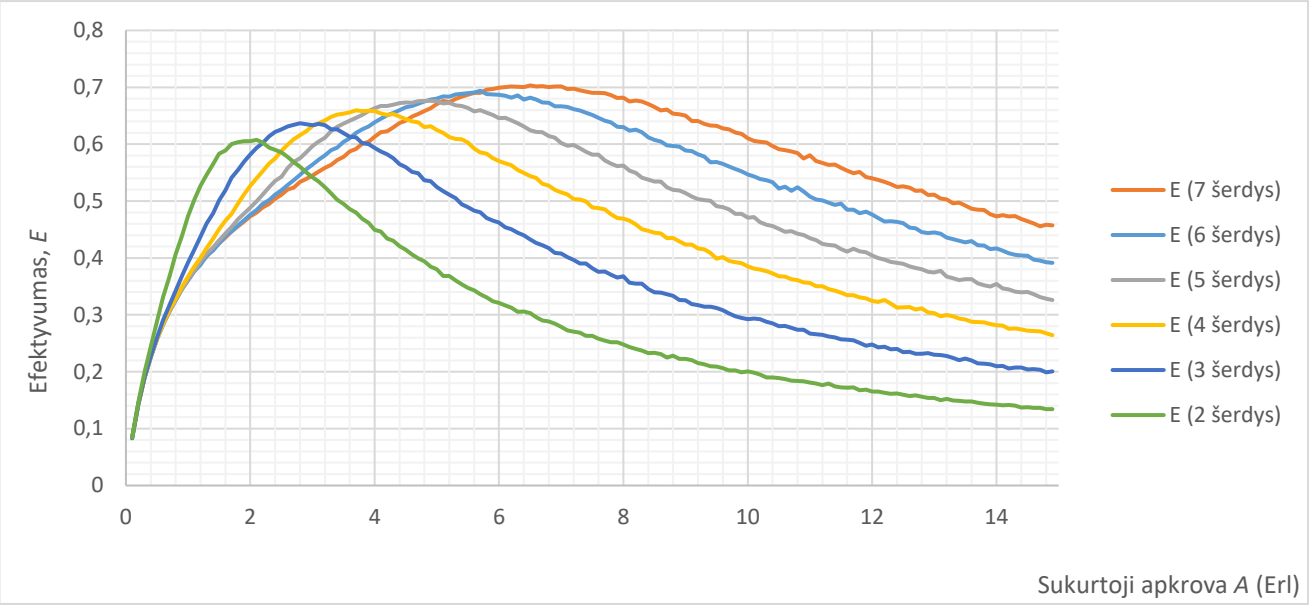
42 pav. SDM sistemų, turinčių 2-7 šerdžių skaidulas, efektyvumo įvertinimas (buferio talpa – 0)

43 paveiksle paveiksle pavaizduota SDM sistemų turinčių skaidulas su 2-7 šerdimis efektyvumo įvertinimas (buferio talpa – 1).



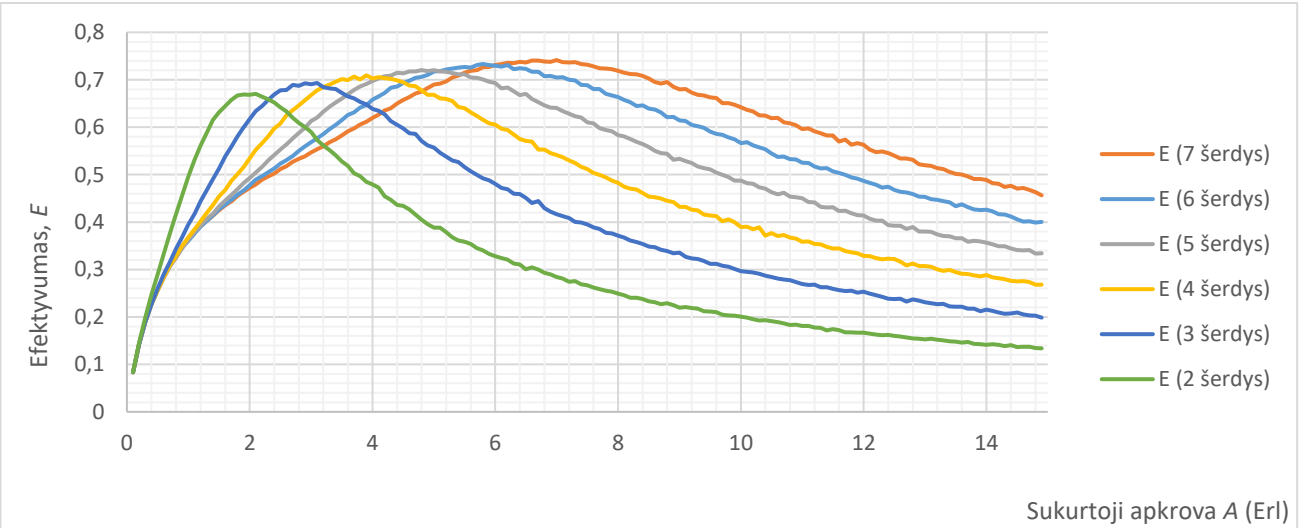
43 pav. SDM sistemų, turinčių 2-7 šerdžių skaidulas, efektyvumo įvertinimas (buferio talpa – 1)

44 paveiksle pavaizduota SDM sistemų turinčių skaidulas su 2-7 šerdimis efektyvumo įvertinimas (buferio talpa – 2).



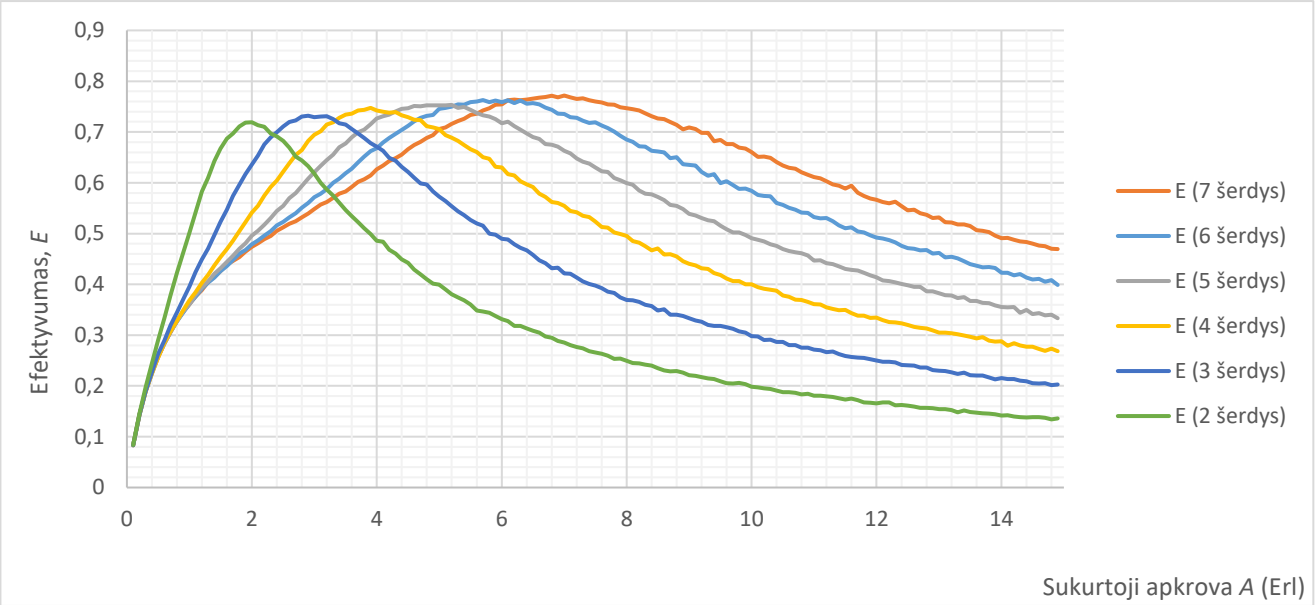
44 pav. SDM sistemų, turinčių 2-7 šerdžių skaidulas, efektyvumo įvertinimas (buferio talpa – 2)

45 paveiksle pavaizduota SDM sistemų turinčių skaidulas su 2-7 šerdimis efektyvumo įvertinimas. (buferio talpa – 3).



45 pav. SDM sistemų, turinčių 2-7 šerdžių skaidulas, efektyvumo įvertinimas (buferio talpa – 3)

46 paveiksle pavaizduota SDM sistemų turinčių skaidulas su 2-7 šerdimis efektyvumo įvertinimas (buferio talpa – 4).



46 pav. SDM sistemų, turinčių 2-7 šerdžių skaidulas, efektyvumo įvertinimas (buferio talpa – 4)

Taigi, įvertinus efektyvumą galima pastebėti, jog prie mažesnių apkrovų duomenų pliūpsnius yra efektyviau perduoti SDM sistemomis turinčiomis skaidulas su mažiau šerdžių. Tačiau prie didesnių apkrovų – duomenis efektyviau yra perduoti SDM sistemomis, turinčiomis skaidulas su daugiau šerdžių. Taip pat, didinant buferio talpą, didėja efektyvumas. Taip yra dėl to, jog didinant buferio talpą šerdys tampa tolygiau užimtos (39 pav.) ir pliūpsniai tolygiau pasiskirsto tarp šerdžių (41 pav.).

Išvados

1. Darbo metu buvo išanalizuoti erdvinio atskyrimo sutankinimo (SDM) sistemų ypatumai. Tai technologija, kai naudojamas fizinis duomenų perdavimo atskyrimas norint vienu metu perduoti skirtingus duomenų srautus. Buvo atlikta literatūros analizė, kurioje nagrinėjami SDM klausimai. Išsiaiškinta, jog egzistuoja keli skaidulų tipai, kurios naudojamos SDM – tai UMCF, CMCF, FMF. Daugiašerdėse skaidulose vyrauja įvairūs šerdžių išsidėstymo principai.
2. SDM sistemose yra daug komponentų kaip ir SMF sistemose, įskaitant stiprintuvus ir bangų ilgių nukreipimo ir perjungimo elementus. Kai kurie komponentai, pavyzdžiui laisvosios erdvės prietaisai, dažnai gali būti pritaikyti iš SMF sistemų su minimaliais pakeitimais.
3. Tyrinėjant mokslinę literatūrą buvo pastebėta, jog nėra atliktų tyrimų, kuriuose analizuojami optinių pliūpsnių perdavimo (OBS) per daugiašerdes skaidulas atvejai. OBS technologijos pagrindinis uždavinys - pliūpsnių surinkimo (buferizavimo ir nukreipimo) procesas, kuomet kraštiniuose tinklo mazguose į pliūpsnius (duomenų blokus), surenkami ateinantys paketai iš įvairių interneto šaltinių (maršrutizatorių).
4. Pasitelkiant Markovo grandines buvo sukurti 2, 3 ir 4 šerdžių optinio perdavimo sistemų modeliai. Tačiau buvo pastebėta jog šis modeliavimo principas yra per daug sudėtingas modeliuoti skaidulas, kurios turi daugiau šerdžių (7 šerdžių skaidulos modeliavimui reikėtų atskirai apsirašyti 128 būsenų grafus), todėl buvo sukurtas imitacinis modelis. Imitacinis modelis buvo patikrintas lyginant gautą blokavimo tikimybę su prieš tai sukurto Markovo proceso modelio gauta blokavimo tikimybe. Abiejuose modeliuose rezultatai sutapo, dėl to galima teigti, jog modelis sukurtas sėkmingai.
5. Įvertinus efektyvumą, pagal darbe apsirašytą metodiką, buvo gauta, jog prie mažesnių apkrovų efektyviau duomenų pliūpsnius perduoti mažiau šerdžių turinčiomis skaidulomis. Prie didesnių apkrovų – skaidulomis, turinčiomis daugiau šerdžių. Nustatyti maksimalūs daugiašerdžių skaidulų ilgiai, prie kurių nėra viršijamos kritinės persidengimo ribos. 2 šerdžių skaidulai – 1320 km, 3 šerdžių – 870 km, 4 šerdžių – 650 km, 5 šerdžių – 530 km, 6 šerdžių – 440km, 7 šerdžių – 380km. Viršijus šiuos atstumus efektyvumas tampa lygus 0, nes viršijama leistina persidengimo tarp šerdžių vertė (XT_{kr}). Taip pat nustatyta, jog didinant buferio talpą pliūpsniai tolygiau pasiskirsto tarp šerdžių, jos tampa tolygiau užimtos, dėl to didėja efektyvumas.

Informacijos šaltinių sąrašas

1. NORDRUM, Amy. *Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated.* IEEE SPECTRUM [interaktyvus]. 2016. [žiūrėta 2018-02-11]. Prieiga per: https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated
2. NAKAJIMA, K., T. MATSUI, K. SAITO, T. SAKAMOTO, N. ARAKI. *Multi-Core Fiber Technology: Next Generation Optical Communication Strategy.* IEEE Xplore [interaktyvus], 2016, 1-7 [žiūrėta 2018-02-11]. Prieiga per: doi: 10.1109/ITU-WT.2016.7805716
3. *What is SDM (Space Division Multiplexing.* IGI Global [interaktyvus]. [žiūrėta 2018-02-11]. Prieiga per: https://www.igi-global.com/dictionary/sdm-space-division-multiplexing/34744.
4. Chen, H. and A.M.J. Koonen. *Spatial Division Multiplexing. In: Fibre Optic Communication*. Springer, Cham, 2017, pp. 1-48.
5. MIZUNO, T., Y. MIYAMOTO. *High-capacity dense space division multiplexing transmission*. Optical Fiber Technology [interaktyvus]. 2017, 108-117 [žiūrėta 2018-03-05]. ISSN 1068-5200. Prieiga per: https://doi.org/10.1016/j.yofte.2016.09.015
6. STOCKTON, Dave. *Multicore networks – the solution to future fiber bandwidth needs?* PPC [interaktyvus]. [žiūrėta 2018-03-05]. Prieiga per: http://www.ppc-online.com/blog/multicore-networks-the-solution-to-future-fiber-bandwidth-needs
7. RYF, R. et al. *32-bit/s/Hz spectral efficiency WDM transmission over 177-km few-mode fiber*. 2013 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC). [interaktyvus]. 2013, 1-3 [žiūrėta 2018-03-05]. Prieiga per: https://ieeexplore.ieee.org/document/6533221/
8. Zhou, X. And C. Xie. *Enabling Technologies for High Spectral-efficiency Coherent Optical Communication Networks*, Wiley, 2016, pp. 547-593.
9. Kobayashi, T. et al. *2 × 344 Tb/s propagation-direction interleaved transmission over 1500-km MCF enhanced by multicarrier full electric-field digital back-propagation* [interaktyvus]. 2013, 1-3 [žiūrėta 2018-04-10]. Prieiga per: doi: 10.1049/cp.2013.169
10. Takara, H. Et al. *1000-km 7-core fiber transmission of 10 x 96-Gb/s PDM-16QAM using Raman amplification with 6.5 W per fiber.* Opt, Express 20 [interaktyvus]. 2012, 10100-10105 [žiūrėta 2018-04-10]. Prieiga per: https://doi.org/10.1364/OE.20.010100
11. HAYASHI, T., Y. TAMURA, T. HASEGAWA, T, NAKASHINI, T. TARU. *Coupled Multi-Core Optical Fiber Suitable for Long-Haul Transmission.* SEI Technical Review [interaktyvus]. 2017, 19-23 [žiūrėta 2018-04-10]. Prieiga per: https://www.researchgate.net/publication/321152024_Coupled_multi-core_optical_fiber_suitable_for_long-haul_transmission
12. PERRIN, S. *Next-Generation ROADM Architectures & Benefits.* Fujitsu [interaktyvus]. 2015, 1-10 [žiūrėta 2018-10-21]. Prieiga per: https://www.fujitsu.com/us/Images/Fujitsu-NG-ROADM.pdf
13. MAROM, D. M. et al. *Wavelength-selective 1 × K switches using free-space optics and MEMS micromirrors: theory, design, and implementation*. OSA publishing [interaktyvus]. 2015, 1620- [žiūrėta 2018-12-02]. Prieiga per: https://www.osapublishing.org/jlt/abstract.cfm?uri=jlt-23-4-1620

14. SIHUA, L., W. ZHUJUN, X. JING, Z. SHAOLONG, W. YAMING. *Wavelength-Selective Switch Based on a Polarization-Independent Transmission Grating and a High Fill-Factor Micromirror Array*. IEEE Photonics Technology Letters [interaktyvus]. 2011, 1249-1251. Prieiga per: doi: 10.1109/LPT.2011.2159107
15. COLLINGS, N., T. DAVEY, J. CHRISTMAS, D. CHU, B. CROSSLAND. *The Applications and Technology of Phase-Only Liquid Crystal on Silicon Devices*. Journal of Display Technology [interaktyvus]. 2011, 112-119 [žiūrėta 2018-12-02]. Prieiga per: doi: 10.1109/JDT.2010.2049337
16. CHEN, H. S., A. M. J. KOONEN. *LP01 and LP11 mode division multiplexing link with mode crossbar switch*. Electronics Letters vol. 48, no. 19[interaktyvus]. 2012 1222-1223 [žiūrėta 2018-12-02]. Prieiga per: doi: 10.1049/el.2012.1674
17. FEUER, N. D. et al., *ROADM system for space division multiplexing with spacial superchannels*. 2013 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC) [interaktyvus]. 2013, 1-3 [žiūrėta 2018-12-02]. Prieiga per: https://ieeexplore.ieee.org/document/6533237/
18. NELSON, L. E. et al. *Spatial Superchannel Routing in a Two-Span ROADM system for Space Division Multiplexing*. Journal of Lightwave Technology, vol. 32 no. 4 [interaktyvus]. 2014, 783-789 [žiūrėta 2018-12-02]. Prieiga per: doi: 10.1109/JLT.2013.2283912
19. WATANABE K., T. SAITO, K. IMAMURA, Y. NAKAYAMA, M. SHIINO. *Study of fusion splice for single-mode multicore fiber*. 17th Microopics Conference (MOC) [interaktyvus]. 2011, 1-2 [žiūrėta 2018-12-02]. Prieiga per: https://ieeexplore.ieee.org/document/6110364/
20. KUSCHNEROV, M., F. N. HAUSKE, K. PIYAWANNO, B. SPINNLER, A. NAPOLI, B. LANKL. *Adaptive chromatic dispersion equalization for non-dispersion managed coherent systems*. 2009 Conference on Optical Fiber Communication – includes post deadline papers [interaktyvus]. 2009, 1-3 [žiūrėta 2018-12-02]. Prieiga per: doi: 10.1364/OFC.2009.OMT1
21. SHOHEI, F., H. YUSUKE, T. HIDEKI, M. KOSO. *On-Demand Spectrum and Core Allocation for Reducing Crosstalk in Multicore Fibers in Elastic Optical Networks*. Journal of Optical Communications and Networking [interaktyvus]. 2014, 1059-1071 [žiūrėta 2018-12-02]. Prieiga per: doi: 10.1364/JOCN.6.001059
22. ANDANI, A., D. DEWIANI, K. RIKLAN. *Crosstalk reduction for network multicore fiber with management core and spectrum method* [interaktyvus]. 2017, 139-146 [žiūrėta 2018-12-02]. Prieiga per: https://www.researchgate.net/publication/316491172_Crosstalk_reduction_for_network_multicore_fiber_with_management_core_and_spectrum_method
23. YUAN, H., M. FURDEK, A. MUHAMMAD, A. SALJOGHEI, L. WOSINSKA, G. ZERVAS. *Space-division multiplexing in data center networks: on multi-core fiber solutions and crosstalk-suppressed resource allocation*. IEEE/OSA Journal of Optical Communications and Networking [interaktyvus]. 2018 272-288 [žiūrėta 2018-12-02]. Prieiga per: doi: 10.1364/JOCN.10.000272
24. MORITA, I., K. IGARASHI, T. TSURITANI. *1 Exabit/s·km transmission with multi-core fiber and spectral efficient modulation format*. 2014 OptoElectronics and Communication Conference and Australian Conference on Optical Fibre Technology [interaktyvus]. 2014, 316-318 [žiūrėta 2018-12-02]. Prieiga per: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6888092&isnumber=6887957

25. MUHAMMAD, A., G. ZERVAS, D. SIMEONIDOU, R. FORCHHEIMER. *Routing, spectrum and core allocation in flexgrid SDM networks with multi-core fibers*. 2014 International Conference on Optical Network Design and Modeling [interaktyvus]. 2014, 192-197 [žiūrėta 2018-12-02]. Prieiga per: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6855762&isnumber=6855750
26. YE, F., K. SAITOH, H. TAKARA, R. ASIF, T. MORIOKA. *High-count multi-core fibers for space-division multiplexing with propagation-direction interleaving*. 2015 Optical Fiber Communications Conference and Exhibition (OFC) [interaktyvus]. 2015, 1-3 [žiūrėta 2019-02-17]. Prieiga per: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7121750&isnumber=7121453
27. HUI, Y., F. MARIJA, M. AJMAL, S. ARSALAN, W. LENA, Z. GEORGIOS. *Space-Division Multiplexing in Data Center Networks: On Multi-Core Fiber Solutions and Crosstalk-Suppressed Resource Allocation*. Journal of Optical Communications and Networking [interaktyvus]. 2018 [žiūrėta 2019-03-15]. Prieiga per: doi: 10.1364/JOCN.10.000272
28. ACQUAAH, Pascal E.K. *An Emission and Discard Priority Scheme for Optical Burst Switched Networks: master's thesis: Electrical Engineering* [interaktyvus], University of Cape Town. Cape Town, 2007. Prieiga per: https://open.uct.ac.za/bitstream/handle/11427/5231/thesis_ebe_2007_acquaah_pek.pdf?sequence=1&isAllowed=y
29. DOLZER, K. *Mechanisms for Quality of Service Differentiation in Optical Burst Switched Networks: doctoral dissertation: Fakultät Informatik, Elektrotechnik und Informationstechnik* [interaktyvus], Universität Stuttgart. Stuttgart, 2004. Prieiga per: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.455.9174&rep=rep1&type=pdf
30. ESA, H. *Resource Allocation and Performance Analysis Problems in Optical Networks* [interaktyvus]. 2019 [žiūrėta 2018-05-15]. Prieiga per: https://www.researchgate.net/figure/Optical-burst-switching-OBS-network-Edge-nodes-aggregate-several-packets-going-to_fig13_27516111
31. KLINKOWSKI, M. Offset Time-Emulated Architecture for Optical Burst Switching - Modelling and Performance Evaluation: doctoral dissertation: Arquitectura i Tecnologia de Computadors
[interaktyvus], Universitat Politecnica De Catalunya, 2007. Prieiga per: https://tdx.cat/bitstream/handle/10803/6000/01_miroslawKlinkowski.pdf?sequence=1

Priedai

1 priedas. Dviejų šerdžių skaidulos Markovo modelio sudarymas

Python kodas be buferio:

```
%matplotlib
from graphviz import Digraph #importuojama diagramų paišymo biblioteka
from graphviz import *
from IPython.display import Image, Math #importuojamos bibliotekos, kurios
#parodys diagramos paveiksliuką ir matematines formules LaTex formatu
from sympy import * # importuojama simbolinių skaičiavimų biblioteka
#init_printing() # įjungiamas sympy vaizdavimas LaTex formatu
from matplotlib.pyplot import *
from numpy import *
init_printing() # įjungiamas sympy vaizdavimas LaTex formatu

dot = Digraph(engine='fdp') # sukuriamas grafo objektas
dot.attr('node', shape='circle')
dot.graph_attr['rankdir'] = 'LR'
dot.attr(overlap='true', splines='true')
dot.node('00',pos='0,1!')
dot.node('10',pos='2,2!')
dot.node('01',pos='2,0!')
dot.node('11',pos='4,1!')
dot.edge('00', '10', label="λ")
dot.edge('10', '00', label="μ")
dot.edge('10', '11', label="λ")
dot.edge('01', '00', label="μ")
dot.edge('01', '11', label="λ")
dot.edge('11', '01', label="μ")
dot.edge('11', '10', label="μ")
dot.format = 'png'
dot.render('graph')
img = Image("graph.png") # išsaugomas sugeneruoto grafo paveikslėlis
img # parodomas grafo paveikslėlis

# Lygčių sistemos aprašymas LaTex formatu
Math(r'''''
\begin{cases}
-{\lambda}P_{00}+{\mu}P_{10}+{\mu}P_{01}=0 \\
{\lambda}P_{00}-({\lambda+\mu})P_{10}+{\mu}P_{11}=0 \\
-({\lambda+\mu})P_{01}+{\mu}P_{11}=0 \\
-{2\mu}P_{11}+{\lambda}P_{10}+{\lambda}P_{01}=0 \\
P_{00}+P_{10}+P_{01}+P_{11}=1
\end{cases}
''')

l,m = symbols('{\lambda}, {\mu}') # aprašomi instensyvumų kintamieji
P00,P10,P01,P11 = symbols("P00,P10,P01,P11") # aprašomi būsenų kintamieji

e1 = Eq(-l*P00+m*P10+m*P01,0) # 1-a lygčių sistemos lygtis
e1

e2 = Eq(l*P00-(l+m)*P10+m*P11,0) # 2-a lygčių sistemos lygtis
e2

e3 = Eq(-(l+m)*P01+m*P11,0) # 3-ia lygčių sistemos lygtis
e3

e4 = Eq(-(2*m)*P11+l*P10+l*P01,0) # 4-a lygčių sistemos lygtis
e4

e5 = Eq(P00+P10+P01+P11,1) # 5-a lygčių sistemos lygtis
e5

Rez=solve([e1,e2,e3,e4,e5],[P00,P10,P01,P11]) # lygčių sistemos sprendimas
```

```
Rez

l=linspace(0.1,15,10000)
m = 1
A = l/m

P00 = 2*m**2/(l**2 + 2*m*(l + m))
P01 = l**2*m/((l + m)*(l**2 + 2*m*(l + m)))
P10 = l*m*(l + 2*m)/((l + m)*(l**2 + 2*m*(l + m)))
P11 = l**2/(l**2 + 2*m*(l + m))

U1 = P10 + P11
U2 = P01 + P11

plot(A,U1, label='$U_{1}(A)$')
plot(A,U2, label='$U_{2}(A)$')
plot(A,P11, label='$P_{11}(A)$')
legend()
grid()
xlabel('$A[Erl]$')
ylabel('sistemos uzimtumo tikimybe , $P(A)$')
legend()
grid(b=True, which='minor', color='k', linestyle='-',alpha=0.1)
minorticks_on()
```

Python kodas su buferiu:

```
%matplotlib
from graphviz import Digraph #importuojama diagramų paišymo biblioteka
from graphviz import *
from IPython.display import Image, Math #importuojamos bibliotekos, kurios
#parodys diagramos paveiksliuką ir matematines formules LaTex formatu
from sympy import * # importuojama simbolinių skaičiavimų biblioteka
#init_printing() # įjungiamas sympy vaizdavimas LaTex formatu
from matplotlib.pyplot import *
from numpy import *
init_printing() # įjungiamas sympy vaizdavimas LaTex formatu

dot = Digraph(engine='fdp') # sukuriamas grafo objektas
dot.attr('node', shape='circle')
dot.graph_attr['rankdir'] = 'LR'
dot.attr(overlap='true', splines='true')
dot.node('000',pos='0,1!')
dot.node('100',pos='2,2!')
dot.node('010',pos='2,0!')
dot.node('110',pos='4,1!')
dot.node('111',pos='6,1!')
dot.node('112',pos='8,1!')
dot.edge('000', '100', label="λ")
dot.edge('100', '000', label="μ")
dot.edge('100', '110', label="λ")
dot.edge('010', '000', label="μ")
dot.edge('010', '110', label="λ")
dot.edge('110', '111', label="λ")
dot.edge('110', '010', label="μ")
dot.edge('110', '100', label="μ")
dot.edge('111', '112', label="λ")
dot.edge('111', '110', label="μ")
dot.edge('112', '111', label="μ")
dot.format = 'png'
dot.render('graph')
img = Image("graph.png") # išsaugomas sugeneruoto grafo paveikslėlis
img # parodomas grafo paveikslėlis

# Lygčių sistemos aprašymas LaTex formatu
Math(r'''''
\begin{cases}
```

```
-{\lambda}P_{000}+{\mu}P_{100}+{\mu}P_{010}=0 \\
{\lambda}P_{000}-({\lambda+\mu})P_{100}+{\mu}P_{110}=0 \\
-({\lambda+\mu})P_{010}+{\mu}P_{110}=0 \\
-({\lambda+2\mu})P_{110}+{\lambda}P_{100}+{\lambda}P_{010}+{\mu}P_{111}=0 \\
-({\lambda+\mu})P_{111}+{\lambda}P_{110}+{\mu}P_{112}=0 \\
-{\mu}P_{112}+{\lambda}P_{111}=0 \\
P_{000}+P_{100}+P_{010}+P_{110}+P_{111}+P_{112}=1
\end{cases}
''')

l,m = symbols('{\lambda}, {\mu}') # aprašomi instensyvumų kintamieji
P000,P100,P010,P110,P111,P112 = symbols("P00,P10,P01,P11,P111,P112") # aprašomi būsenų kintamieji

e1 = Eq(-l*P000+m*P100+m*P010,0) # 1-a lygčių sistemos lygtis
e1

e2 = Eq(l*P000-(l+m)*P100+m*P110,0) # 2-a lygčių sistemos lygtis
e2

e3 = Eq(-(l+m)*P010+m*P110,0) # 3-ia lygčių sistemos lygtis
e3

e4 = Eq(-(l+2*m)*P110+l*P100+l*P010+m*P111,0) # 4-a lygčių sistemos lygtis
e4

e5 = Eq(-(l+m)*P111+l*P110+m*P112,0) # 5-a lygčių sistemos lygtis
e5

e6 = Eq(-m*P112+l*P111,0) # 6-a lygčių sistemos lygtis
e6

e7 = Eq(P000+P100+P010+P110+P111+P112,1) # 7-a lygčių sistemos lygtis
e7

Rez=solve([e1,e2,e3,e4,e5,e6,e7],[P000,P100,P010,P110,P111,P112]) # lygčių sistemos sprendimas
Rez

l=linspace(0.1,15,10000)
m = 1
A = l/m

P000= 2*m**4/(l**4 + m*(l**3 + m*(l**2 + 2*m*(l + m))))
P010= l**2*m**3/((l + m)*(l**4 + m*(l**3 + m*(l**2 + 2*m*(l + m)))))
P100= l*m**3*(l + 2*m)/((l + m)*(l**4 + m*(l**3 + m*(l**2 + 2*m*(l + m)))))
P110= l**2*m**2/(l**4 + m*(l**3 + m*(l**2 + 2*m*(l + m))))
P111= l**3*m/(l**4 + m*(l**3 + m*(l**2 + 2*m*(l + m))))
P112= l**4/(l**4 + m*(l**3 + m*(l**2 + 2*m*(l + m))))


U1 = P100 + P110 + P111 + P112
U2 = P010 + P110 + P111 + P112


plot(A,U1, label='$U_{1}(A)$')
plot(A,U2, label='$U_{2}(A)$')
plot(A,P112, label='$P_{112}(A)$')
legend()
grid()
xlabel('$A[Erl]$')
ylabel('sistemos uzimtumo tikimybe , $P(A)$')
legend()
grid(b=True, which='minor', color='k', linestyle='-',alpha=0.1)
minorticks_on()
```

Gautos tikimybių lygtys:

$$\left\{ P_{00}: \frac{2\mu^4}{\lambda^4+\mu\left(\lambda^3+\mu\left(\lambda^2+2\mu(\lambda+\mu)\right)\right)},\quad P_{01}: \frac{\lambda^2\mu^3}{(\lambda+\mu)\left(\lambda^4+\mu\left(\lambda^3+\mu\left(\lambda^2+2\mu(\lambda+\mu)\right)\right)\right)},\right.$$

$$P_{10}: \frac{\lambda\mu^3(\lambda+2\mu)}{(\lambda+\mu)\left(\lambda^4+\mu\left(\lambda^3+\mu\left(\lambda^2+2\mu(\lambda+\mu)\right)\right)\right)},\quad P_{11}: \frac{\lambda^2\mu^2}{\lambda^4+\mu\left(\lambda^3+\mu\left(\lambda^2+2\mu(\lambda+\mu)\right)\right)},$$

$$\left. P_{111}: \frac{\lambda^3\mu}{\lambda^4+\mu\left(\lambda^3+\mu\left(\lambda^2+2\mu(\lambda+\mu)\right)\right)},\quad P_{112}: \frac{\lambda^4}{\lambda^4+\mu\left(\lambda^3+\mu\left(\lambda^2+2\mu(\lambda+\mu)\right)\right)} \right\}$$

2 priedas. Trijų, keturių šerdžių skaidulos Markovo modelio sudarymas

Trijų šerdžių Python kodas:

```
%matplotlib
from graphviz import Digraph #importuojama diagramų paišymo biblioteka
from graphviz import *
from IPython.display import Image, Math #importuojamos bibliotekos, kurios
#parodys diagramos paveiksliuką ir matematines formules LaTex formatu
from sympy import * # importuojama simbolinių skaičiavimų biblioteka
#init_printing() # įjungiamas sympy vaizdavimas LaTex formatu
from matplotlib.pyplot import *
from numpy import *

dot = Digraph(engine='fdp') # sukuriamas grafo objektas
dot.attr('node', shape='circle')
dot.graph_attr['rankdir'] = 'LR'
dot.attr(overlap='true', splines='true')
dot.node('000',pos='0,0!')
dot.node('100',pos='2,2!')
dot.node('010',pos='2,0!')
dot.node('001',pos='2,-2!')
dot.node('110',pos='4,2!')
dot.node('011',pos='4,0!')
dot.node('101',pos='4,-2!')
dot.node('111',pos='6,0!')
dot.edge('000', '100', label="λ")
dot.edge('100', '000', label="μ")
dot.edge('100', '110', label="λ")
dot.edge('010', '000', label="μ")
dot.edge('010', '110', label="λ")
dot.edge('001', '000', label="μ")
dot.edge('001', '101', label="λ")
dot.edge('110', '111', label="λ")
dot.edge('110', '100', label="μ")
dot.edge('110', '010', label="μ")
dot.edge('011', '111', label="λ")
dot.edge('011', '001', label="μ")
dot.edge('011', '010', label="μ")
dot.edge('101', '111', label="λ")
dot.edge('101', '001', label="μ")
dot.edge('101', '100', label="μ")
dot.edge('111', '110', label="μ")
dot.edge('111', '011', label="μ")
dot.edge('111', '101', label="μ")
dot.format = 'png'
dot.render('graph')
img = Image("graph.png") # išsaugomas sugeneruoto grafo paveikslėlis
img # parodomas grafo paveikslėlis

# Lygčių sistemos aprašymas LaTex formatu
Math(r'''''
\begin{cases}
```

```
-{\lambda}P_{000}+{\mu}P_{100}+{\mu}P_{010}+{\mu}P_{001}=0 \\
{\lambda}P_{000}-({\lambda+\mu})P_{100}+{\mu}P_{110}+{\mu}P_{101}=0 \\
-({\lambda+\mu})P_{010}+{\mu}P_{110}+{\mu}P_{011}=0 \\
-({\lambda+\mu})P_{001}+{\mu}P_{011}+{\mu}P_{101}=0 \\
{\lambda}P_{100}+{\lambda}P_{010}-({\lambda+2\mu})P_{110}+{\mu}P_{111}=0 \\
-({\lambda+2\mu})P_{011}+{\mu}P_{111}=0 \\
{\lambda}P_{001}-({\lambda+2\mu})P_{101}+{\mu}P_{111}=0 \\
P_{000}+P_{100}+P_{010}+P_{001}+P_{110}+P_{011}+P_{101}+P_{111}=1
\end{cases}
''')

l,m = symbols('{\lambda}, {\mu}') # aprašomi instensyvumų kintamieji
P000,P100,P010,P001,P110,P011,P101,P111 = symbols("P000,P100,P010,P001,P110,P011,P101,P111") # aprašomi būsenų kintamieji

e1 = Eq(-l*P000+m*P100+m*P010+m*P001,0) # 1-a lygčių sistemos lygtis
e1

e2 = Eq(l*P000-(l+m)*P100+m*P110+m*P101,0) # 2-a lygčių sistemos lygtis
e2

e3 = Eq(-(l+m)*P010+m*P110+m*P011,0) # 3-ia lygčių sistemos lygtis
e3

e4 = Eq(-(l+m)*P001+m*P011+m*P101,0) # 4-a lygčių sistemos lygtis
e4

e5 = Eq(l*P100+l*P010-(l+2*m)*P110+m*P111,0) # 5-a lygčių sistemos lygtis
e5

e6 = Eq(-(l+2*m)*P011+m*P111,0) # 6-a lygčių sistemos lygtis
e6

e7 = Eq(l*P001-(l+2*m)*P101+m*P111,0) # 7-a lygčių sistemos lygtis
e7

e8 = Eq(P000+P100+P010+P001+P110+P011+P101+P111,1) # 8-a lygčių sistemos lygtis
e8

Rez=solve([e1,e2,e3,e4,e5,e6,e7,e8],[P000,P100,P010,P001,P110,P011,P101,P111]) # lygčių sistemos sprendimas
Rez


l=linspace(0.1,15,10000)
m = 1
A = l/m

P000= 6*m**3/(l**3 + 3*l**2*m + 6*l*m**2 + 6*m**3)
P001= 2*l**3*m**2/(l**5 + 5*l**4*m + 14*l**3*m**2 + 24*l**2*m**3 + 24*l*m**4 + 12*m**5)
P010= 2*l**2*m**2*(l**3 + 4*l**2*m + 8*l*m**2 + 6*m**3)/(l**7 + 8*l**6*m + 31*l**5*m**2 + 76*l**4*m**3 + 124*l**3*m**4 + 132*l**2*m**5 + 84*l*m**6 + 24*m**7)
P011= l**3*m/(l**4 + 5*l**3*m + 12*l**2*m**2 + 18*l*m**3 + 12*m**4)
P100= 2*l*m**2*(l**2 + 6*l*m + 6*m**2)/(l**5 + 6*l**4*m + 17*l**3*m**2 + 30*l**2*m**3 + 30*l*m**4 + 12*m**5)
P101= l**3*m*(l**2 + 4*l*m + 2*m**2)/(l**6 + 7*l**5*m + 24*l**4*m**2 + 52*l**3*m**3 + 72*l**2*m**4 + 60*l*m**5 + 24*m**6)
P110= l**2*m*(l**2 + 4*l*m + 6*m**2)/(l**5 + 5*l**4*m + 14*l**3*m**2 + 24*l**2*m**3 + 24*l*m**4 + 12*m**5)
P111= l**3/(l**3 + 3*l**2*m + 6*l*m**2 + 6*m**3)

U1 = P100 + P101 + P110 + P111
U2 = P010 + P011 + P110 + P111
U3 = P001 + P011 + P101 + P111


plot(A,U1, label='$U_{1}(A)$')
plot(A,U2, label='$U_{2}(A)$')
plot(A,U3, label='$U_{3}(A)$')
```

```
plot(A,P111, label='$P_{111}(A)$')
legend()
grid()
xlabel('$A[Erl]$')
ylabel('sistemos uzimtumo tikimybe , $P(A)$')
legend()
grid(b=True, which='minor', color='k', linestyle='-',alpha=0.1)
minorticks_on()


plot(A,P111, label='$P_{111}(A)$')
grid()
xlabel('$A, [Erl]$')
ylabel('$P_b(A)$')
grid(b=True, which='minor', color='k', linestyle='-',alpha=0.1)
minorticks_on()
```

Trijų šerdžių gautos lygtys:

$$\begin{cases} -\lambda P_{000} + \mu P_{100} + \mu P_{010} + \mu P_{001} = 0 \\ \lambda P_{000} - (\lambda+\mu) P_{100} + \mu P_{110} + \mu P_{101} = 0 \\ -(\lambda+\mu) P_{010} + \mu P_{110} + \mu P_{011} = 0 \\ -(\lambda+\mu) P_{001} + \mu P_{011} + \mu P_{101} = 0 \\ \lambda P_{100} + \lambda P_{010} - (\lambda+2\mu) P_{110} + \mu P_{111} = 0 \\ -(\lambda+2\mu) P_{011} + \mu P_{111} = 0 \\ \lambda P_{001} - (\lambda+2\mu) P_{101} + \mu P_{111} = 0 \\ P_{000} + P_{100} + P_{010} + P_{001} + P_{110} + P_{011} + P_{101} + P_{111} = 1 \end{cases}$$

$$\left\{ P_{000} : \frac{6\mu^3}{\lambda^3 + 3\lambda^2\mu + 6\lambda\mu^2 + 6\mu^3}, \quad P_{001} : \frac{2\lambda^3\mu^2}{\lambda^5 + 5\lambda^4\mu + 14\lambda^3\mu^2 + 24\lambda^2\mu^3 + 24\lambda\mu^4 + 12\mu^5}, \right.$$

$$P_{010} : \frac{2\lambda^2\mu^2(\lambda^3 + 4\lambda^2\mu + 8\lambda\mu^2 + 6\mu^3)}{\lambda^7 + 8\lambda^6\mu + 31\lambda^5\mu^2 + 76\lambda^4\mu^3 + 124\lambda^3\mu^4 + 132\lambda^2\mu^5 + 84\lambda\mu^6 + 24\mu^7}, \quad P_{011} : \frac{\lambda^3\mu}{\lambda^4 + 5\lambda^3\mu + 12\lambda^2\mu^2 + 18\lambda\mu^3 + 12\mu^4},$$

$$P_{100} : \frac{2\lambda\mu^2(\lambda^2 + 6\lambda\mu + 6\mu^2)}{\lambda^5 + 6\lambda^4\mu + 17\lambda^3\mu^2 + 30\lambda^2\mu^3 + 30\lambda\mu^4 + 12\mu^5}, \quad P_{101} : \frac{\lambda^3\mu(\lambda^2 + 4\lambda\mu + 2\mu^2)}{\lambda^6 + 7\lambda^5\mu + 24\lambda^4\mu^2 + 52\lambda^3\mu^3 + 72\lambda^2\mu^4 + 60\lambda\mu^5 + 24\mu^6},$$

$$\left. P_{110} : \frac{\lambda^2\mu(\lambda^2 + 4\lambda\mu + 6\mu^2)}{\lambda^5 + 5\lambda^4\mu + 14\lambda^3\mu^2 + 24\lambda^2\mu^3 + 24\lambda\mu^4 + 12\mu^5}, \quad P_{111} : \frac{\lambda^3}{\lambda^3 + 3\lambda^2\mu + 6\lambda\mu^2 + 6\mu^3} \right\}$$

Trijų šerdžių skaičiavimai kai A kinta nuo 0,1 iki 2 Erl:

λ	μ	A	U1	U2	U3	P000	P001	P010	P011	P100	P101	P110	P111
0,1	1	0,1	9,09E-02	8,64E-03	4,37E-04	0,9048	0,0001	0,004	7,18E-05	0,0863	7,83E-05	0,0044	0,0002
0,2	1	0,2	0,166667	3,01E-02	3,06E-03	0,8188	0,0009	0,0132	0,000496229	0,1497	0,00057758	0,0153	0,0011
0,3	1	0,3	0,230769	5,92E-02	9,04E-03	0,74102	0,0025	0,0243	0,001449812	0,1955	0,00177319	0,0301	0,0033
0,4	1	0,4	0,285714	9,27E-02	1,88E-02	0,6708	0,0048	0,0356	0,002981515	0,2279	0,00378733	0,0469	0,0072
0,5	1	0,5	0,333333	1,28E-01	3,21E-02	0,6076	0,0078	0,0462	0,005063291	0,2498	0,00662123	0,0643	0,0127
0,6	1	0,6	0,375	1,64E-01	4,88E-02	0,5507	0,0111	0,0556	0,007624534	0,2637	0,0101946	0,0813	0,0198
0,7	1	0,7	0,411765	2,00E-01	6,82E-02	0,4995	0,0147	0,0635	0,010574964	0,2714	0,01438086	0,0974	0,0286
0,8	1	0,8	0,444444	2,35E-01	8,98E-02	0,4534	0,0183	0,07	0,013819312	0,2745	0,01903415	0,1122	0,0387
0,9	1	0,9	0,473684	2,68E-01	1,13E-01	0,4121	0,0217	0,0752	0,017266248	0,274	0,02400795	0,1256	0,0501
1	1	1	0,5	3,00E-01	1,38E-01	0,375	0,025	0,0792	0,020833333	0,2708	0,02916667	0,1375	0,0625
1,1	1	1,1	0,52381	3,30E-01	1,63E-01	0,3417	0,028	0,0821	0,024449339	0,2658	0,03439177	0,1479	0,0758
1,2	1	1,2	0,545455	3,59E-01	1,88E-01	0,3117	0,0307	0,084	0,028054863	0,2593	0,039584259	0,1568	0,0898
1,3	1	1,3	0,565217	3,85E-01	2,14E-01	0,2848	0,0332	0,0852	0,031601917	0,2519	0,04466471	0,1644	0,1043
1,4	1	1,4	0,583333	4,11E-01	2,39E-01	0,2606	0,0353	0,0858	0,035052937	0,2438	0,04957191	0,1708	0,1192
1,5	1	1,5	0,6	4,34E-01	2,64E-01	0,2388	0,0371	0,0858	0,038379531	0,2354	0,05426072	0,176	0,1343
1,6	1	1,6	0,615385	4,57E-01	2,88E-01	0,2192	0,0386	0,0853	0,04156114	0,2268	0,05869975	0,1803	0,1496
1,7	1	1,7	0,62963	4,78E-01	3,12E-01	0,2015	0,0398	0,0845	0,04458375	0,2181	0,062869	0,1837	0,165
1,8	1	1,8	0,642857	4,97E-01	3,35E-01	0,1855	0,0408	0,0835	0,047438701	0,2096	0,06675763	0,1862	0,1803
1,9	1	1,9	0,655172	5,16E-01	3,58E-01	0,171	0,0415	0,0822	0,050121632	0,2012	0,07036204	0,1882	0,1955
2	1	2	0,666667	5,33E-01	3,79E-01	0,1579	0,0421	0,0807	0,052631579	0,193	0,07368421	0,1895	0,2105

Keturių šerdžių Python kodas:

```
%matplotlib
from graphviz import Digraph #importuojama diagramų paišymo biblioteka
from graphviz import *
from IPython.display import Image, Math #importuojamos bibliotekos, kurios
#parodys diagramos paveiksliuką ir matematines formules LaTex formatu
from sympy import * # importuojama simbolinių skaičiavimų biblioteka
init_printing() # įjungiamas sympy vaizdavimas LaTex formatu
from matplotlib.pyplot import *
from numpy import *

dot = Digraph(engine='fdp') # sukuriamas grafo objektas
dot.attr('node', shape='circle')
dot.graph_attr['rankdir'] = 'LR'
dot.attr(overlap='true', splines='true')
dot.node('0000',pos='0,1!')
dot.node('1000',pos='3,4!')
dot.node('0100',pos='3,2!')
dot.node('0010',pos='3,0!')
dot.node('0001',pos='3,-2!')
dot.node('1100',pos='6,6!')
dot.node('1010',pos='6,4!')
dot.node('1001',pos='6,2!')
dot.node('0011',pos='6,0!')
dot.node('0101',pos='6,-2!')
dot.node('0110',pos='6,-4!')
dot.node('1110',pos='9,4!')
dot.node('1101',pos='9,2!')
dot.node('1011',pos='9,0!')
dot.node('0111',pos='9,-2!')
dot.node('1111',pos='12,1!')
dot.edge('0000', '1000', label="λ")
dot.edge('1000', '0000', label="μ")
dot.edge('1000', '1100', label="λ")
dot.edge('0100', '0000', label="μ")
dot.edge('0100', '1100', label="λ")
dot.edge('0010', '0000', label="μ")
dot.edge('0010', '1010', label="λ")
dot.edge('0001', '0000', label="μ")
dot.edge('0001', '1001', label="λ")
dot.edge('1100', '1110', label="λ")
dot.edge('1100', '1000', label="μ")
dot.edge('1100', '0100', label="μ")
dot.edge('1010', '1110', label="λ")
dot.edge('1010', '0010', label="μ")
dot.edge('1010', '1000', label="μ")
dot.edge('1001', '1101', label="λ")
dot.edge('1001', '0001', label="μ")
dot.edge('1001', '1000', label="μ")
dot.edge('0011', '1011', label="λ")
dot.edge('0011', '0010', label="μ")
dot.edge('0011', '0001', label="μ")
dot.edge('0101', '1101', label="λ")
dot.edge('0101', '0001', label="μ")
dot.edge('0101', '0100', label="μ")
dot.edge('0110', '1110', label="λ")
dot.edge('0110', '0010', label="μ")
dot.edge('0110', '0100', label="μ")
dot.edge('1110', '1111', label="λ")
dot.edge('1110', '1100', label="μ")
dot.edge('1110', '1010', label="μ")
dot.edge('1110', '0110', label="μ")
dot.edge('1101', '1111', label="λ")
dot.edge('1101', '1100', label="μ")
dot.edge('1101', '1001', label="μ")
dot.edge('1101', '0101', label="μ")
dot.edge('1011', '1111', label="λ")
```

```
dot.edge('1011', '1010', label="μ")
dot.edge('1011', '1001', label="μ")
dot.edge('1011', '0011', label="μ")
dot.edge('0111', '1111', label="λ")
dot.edge('0111', '0011', label="μ")
dot.edge('0111', '0101', label="μ")
dot.edge('0111', '0110', label="μ")
dot.edge('1111', '1110', label="μ")
dot.edge('1111', '1101', label="μ")
dot.edge('1111', '1011', label="μ")
dot.edge('1111', '0111', label="μ")
dot.format = 'png'
dot.render('graph')
img = Image("graph.png") # išsaugomas sugeneruoto grafo paveikslėlis
img # parodomas grafo paveikslėlis

# Lygčių sistemos aprašymas LaTex formatu
Math(r'''''
\begin{cases}
-{\lambda}P_{0000}+{\mu}P_{1000}+{\mu}P_{0100}+{\mu}P_{0010}+{\mu}P_{0001}=0 \\
{\lambda}P_{0000}-({\lambda+\mu})P_{1000}+{\mu}P_{1100}+{\mu}P_{1010}+{\mu}P_{1001}=0 \\
-({\lambda+\mu})P_{0100}+{\mu}P_{1100}+{\mu}P_{0101}+{\mu}P_{0110}=0 \\
-({\lambda+\mu})P_{0010}+{\mu}P_{1010}+{\mu}P_{0011}+{\mu}P_{0110}=0 \\
-({\lambda+\mu})P_{0001}+{\mu}P_{1001}+{\mu}P_{0011}+{\mu}P_{0101}=0 \\
{\lambda}P_{1000}+{\lambda}P_{0100}-
({\lambda+2\mu})P_{1100}+{\mu}P_{1110}+{\mu}P_{1101}=0 \\
{\lambda}P_{0010}-({\lambda+2\mu})P_{1010}+{\mu}P_{1110}+{\mu}P_{1011}=0 \\
{\lambda}P_{0001}-({\lambda+2\mu})P_{1001}+{\mu}P_{1101}+{\mu}P_{1011}=0 \\
-({\lambda+2\mu})P_{0011}+{\mu}P_{1011}+{\mu}P_{0111}=0 \\
-({\lambda+2\mu})P_{0101}+{\mu}P_{1101}+{\mu}P_{0111}=0 \\
-({\lambda+2\mu})P_{0110}+{\mu}P_{1110}+{\mu}P_{0111}=0 \\
{\lambda}P_{1100}+{\lambda}P_{1010}+{\lambda}P_{0110}-
({\lambda+3\mu})P_{1110}+{\mu}P_{1111}=0 \\
{\lambda}P_{1001}+{\lambda}P_{0101}-({\lambda+3\mu})P_{1101}+{\mu}P_{1111}=0 \\
{\lambda}P_{0011}-({\lambda+3\mu})P_{1011}+{\mu}P_{1111}=0 \\
-({\lambda+3\mu})P_{0111}+{\mu}P_{1111}=0 \\
P_{0000}+P_{1000}+P_{0100}+P_{0010}+P_{0001}+P_{1100}+P_{1010}+P_{1001}+P_{0011}+P_{0101
}+P_{0110}+P_{1110}+P_{1101}+P_{1011}+P_{0111}+P_{1111}=1
\end{cases}
''')

l,m = symbols('{\lambda}, {\mu}') # aprašomi instensyvumų kintamieji
P0000,P1000,P0100,P0010,P0001,P1100,P1010,P1001,P0011,P0101,P0110,P1110,P1101,P1011,P011
1,P1111 = symbols("P0000,P1000,P0100,P0010,P0001,P1100,P1010,P1001,P0011,P0101,P0110,P1110
,P1101,P1011,P0111,P1111") # aprašomi būsenų kintamieji

e1 = Eq(-l*P0000+m*P1000+m*P0100+m*P0010+m*P0001,0) # 1-a lygčių sistemos lygtis
e1

e2 = Eq(l*P0000-(l+m)*P1000+m*P1100+m*P1010+m*P1001,0) # 2-a lygčių sistemos lygtis
e2

e3 = Eq(-(l+m)*P0100+m*P1100+m*P0101+m*P0110,0) # 3-ia lygčių sistemos lygtis
e3

e4 = Eq(-(l+m)*P0010+m*P1010+m*P0011+m*P0110,0) # 4-a lygčių sistemos lygtis
e4

e5 = Eq(-(l+m)*P0001+m*P1001+m*P0011+m*P0101,0) # 5-a lygčių sistemos lygtis
e5

e6 = Eq(l*P1000+l*P0100-(l+2*m)*P1100+m*P1110+m*P1101,0) # 6-a lygčių sistemos lygtis
e6

e7 = Eq(l*P0010-(l+2*m)*P1010+m*P1110+m*P1011,0) # 7-a lygčių sistemos lygtis
e7

e8 = Eq(l*P0001-(l+2*m)*P1001+m*P1101+m*P1011,0) # 8-a lygčių sistemos lygtis
e8
```

```
131.
132. e9 = Eq(-(l+2*m)*P0011+m*P1011+m*P0111,0) # 9-a lygčių sistemos lygtis
133. e9
134.
135. e10 = Eq(-(l+2*m)*P0101+m*P1101+m*P0111,0) # 10-a lygčių sistemos lygtis
136. e10
137.
138. e11 = Eq(-(l+2*m)*P0110+m*P1110+m*P0111,0) # 11-a lygčių sistemos lygtis
139. e11
140.
141. e12 = Eq(l*P1100+l*P1010+l*P0110-(l+3*m)*P1110+m*P1111,0) # 12-
    a lygčių sistemos lygtis
142. e12
143.
144. e13 = Eq(l*P1001+l*P0101-(l+3*m)*P1101+m*P1111,0) # 13-a lygčių sistemos lygtis
145. e13
146.
147. e14 = Eq(l*P0011-(l+3*m)*P1011+m*P1111,0) # 14-a lygčių sistemos lygtis
148. e14
149.
150. e15 = Eq(-(l+3*m)*P0111+m*P1111,0) # 15-a lygčių sistemos lygtis
151. e15
152.
153. e16 = Eq(P0000+P1000+P0100+P0010+P0001+P1100+P1010+P1001+P0011+P0101+P0110+P1110+P1101+P
    1011+P0111+P1111,1) # 16-a lygčių sistemos lygtis
154. e16
155.
156. Rez=solve([e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12,e13,e14,e15,e16],[P0000,P1000,P0100,P0
    010,P0001,P1100,P1010,P1001,P0011,P0101,P0110,P1110,P1101,P1011,P0111,P1111]) # lygčių sis
    temos sprendimas
157. Rez
158.
159.
160. l=linspace(0.1,15,10000)
161. m = 1
162. A = l/m
163.
164. P0000 = 24* m**4/(l**4 + 4*l**3* m + 12*l**2* m**2 + 24*l* m**3 + 24* m**4)
165. P0001 = 6*l**4* m**3/(l**7 + 7*l**6* m + 30*l**5* m**2 + 90*l**4* m**3 + 192*l**3* m**4
    + 288*l**2* m**5 + 288*l* m**6 + 144* m**7)
166. P0010 = l**3* m**3*(6*l**5 + 40*l**4* m + 144*l**3* m**2 + 336*l**2* m**3 + 456*l* m**4
    + 288* m**5)/(l**11 + 13*l**10* m + 88*l**9* m**2 + 402*l**8* m**3 + 1364*l**7* m**4 + 356
    4*l**6* m**5 + 7248*l**5* m**6 + 11400*l**4* m**7 + 13536*l**3* m**8 + 11520*l**2* m**9 +
    6336*l* m**10 + 1728* m**11)
167. P0011 = 2*l**4* m**2/(l**6 + 8*l**5* m + 34*l**4* m**2 + 96*l**3* m**3 + 192*l**2* m**4
    + 240*l* m**5 + 144* m**6)
168. P0100 = l**2* m**3*(6*l**5 + 56*l**4* m + 260*l**3* m**2 + 624*l**2* m**3 + 792*l* m**4
    + 432* m**5)/(l**10 + 14*l**9* m + 95*l**8* m**2 + 418*l**7* m**3 + 1328*l**6* m**4 + 3164
    *l**5* m**5 + 5628*l**4* m**6 + 7248*l**3* m**7 + 6384*l**2* m**8 + 3456*l* m**9 + 864* m*
    *10)
169. P0101 = 2*l**4* m**2*(l**5 + 9*l**4* m + 36*l**3* m**2 + 72*l**2* m**3 + 72*l* m**4 + 36
    * m**5)/(l**11 + 16*l**10* m + 125*l**9* m**2 + 638*l**8* m**3 + 2376*l**7* m**4 + 6768*l*
    *6* m**5 + 14964*l**5* m**6 + 25560*l**4* m**7 + 32976*l**3* m**8 + 30528*l**2* m**9 + 181
    44*l* m**10 + 5184* m**11)
170. P0110 = 2*l**3* m**2*(l**4 + 6*l**3* m + 21*l**2* m**2 + 42*l* m**3 + 36* m**4)/(l**9 +
    12*l**8* m + 71*l**7* m**2 + 282*l**6* m**3 + 822*l**5* m**4 + 1788*l**4* m**5 + 2880*l**3
    * m**6 + 3312*l**2* m**7 + 2448*l* m**8 + 864* m**9)
171. P0111 = l**4* m/(l**5 + 7*l**4* m + 24*l**3* m**2 + 60*l**2* m**3 + 96*l* m**4 + 72* m**
    5)
172. P1000 = 6*l* m**3*(l**2 + 10*l* m + 12* m**2)/(l**6 + 8*l**5* m + 31*l**4* m**2 + 84*l**
    3* m**3 + 156*l**2* m**4 + 168*l* m**5 + 72* m**6)
173. P1001 = 2*l**4* m**2*(l**3 + 9*l**2* m + 18*l* m**2 + 6* m**3)/(l**9 + 12*l**8* m + 71*l
    **7* m**2 + 282*l**6* m**3 + 822*l**5* m**4 + 1788*l**4* m**5 + 2880*l**3* m**6 + 3312*l**
    2* m**7 + 2448*l* m**8 + 864* m**9)
174. P1010 = 2*l**3* m**2*(l**8 + 16*l**7* m + 107*l**6* m**2 + 430*l**5* m**3 + 1164*l**4* m
    **4 + 2136*l**3* m**5 + 2508*l**2* m**6 + 1656*l* m**7 + 432* m**8)/(l**13 + 18*l**12* m +
     159*l**11* m**2 + 920*l**10* m**3 + 3902*l**9* m**4 + 12796*l**8* m**5 + 33252*l**7* m**6
     + 69024*l**6* m**7 + 114024*l**5* m**8 + 147600*l**4* m**9 + 145152*l**3* m**10 + 102528*
    l**2* m**11 + 46656*l* m**12 + 10368* m**13)
```

```
P1011 = l**4* m*(l**2 + 6*l* m + 6* m**2)/(l**7 + 11*l**6* m + 58*l**5* m**2 + 198*l**4* m**3 + 480*l**3* m**4 + 816*l**2* m**5 + 864*l* m**6 + 432* m**7)
P1100 = 2*l**2* m**2*(l**4 + 12*l**3* m + 54*l**2* m**2 + 96*l* m**3 + 72* m**4)/(l**8 + 10*l**7* m + 52*l**6* m**2 + 180*l**5* m**3 + 452*l**4* m**4 + 816*l**3* m**5 + 1008*l**2* m**6 + 768*l* m**7 + 288* m**8)
P1101 = l**4* m*(l**4 + 8*l**3* m + 24*l**2* m**2 + 24*l* m**3 + 12* m**4)/(l**9 + 11*l**8* m + 64*l**7* m**2 + 252*l**6* m**3 + 732*l**5* m**4 + 1596*l**4* m**5 + 2592*l**3* m**6 + 3024*l**2* m**7 + 2304*l* m**8 + 864* m**9)
P1110 = l**3* m*(l**3 + 6*l**2* m + 18*l* m**2 + 24* m**3)/(l**7 + 7*l**6* m + 30*l**5* m**2 + 90*l**4* m**3 + 192*l**3* m**4 + 288*l**2* m**5 + 288*l* m**6 + 144* m**7)
P1111 = l**4/(l**4 + 4*l**3* m + 12*l**2* m**2 + 24*l* m**3 + 24* m**4)

U1 = P1000 + P1001 + P1010 + P1011 + P1100 + P1101 + P1110 + P1111
U2 = P0100 + P0101 + P0110 + P0111 + P1100 + P1101 + P1110 + P1111
U3 = P0010 + P0011 + P0110 + P0111 + P1010 + P1011 + P1110 + P1111
U4 = P0001 + P0011 + P0101 + P0111 + P1001 + P1011 + P1101 + P1111

plot(A,U1, label='$U_{1}(A)$')
plot(A,U2, label='$U_{2}(A)$')
plot(A,U3, label='$U_{3}(A)$')
plot(A,U4, label='$U_{4}(A)$')
plot(A,P1111, label='$P_{1111}(A)$')
legend()
grid()
xlabel('$A[Erl]$')
ylabel('sistemos uzimtumo tikimybe , $P(A)$')
legend()
grid(b=True, which='minor', color='k', linestyle='-',alpha=0.1)
minorticks_on()
```

Keturių šerdžių gautos lygtys:

$$
\begin{cases}
-\lambda P_{0000} + \mu P_{1000} + \mu P_{0100} + \mu P_{0010} + \mu P_{0001} = 0 \\
\lambda P_{0000} - (\lambda + \mu) P_{1000} + \mu P_{1100} + \mu P_{1010} + \mu P_{1001} = 0 \\
-(\lambda + \mu) P_{0100} + \mu P_{1100} + \mu P_{0101} + \mu P_{0110} = 0 \\
-(\lambda + \mu) P_{0010} + \mu P_{1010} + \mu P_{0011} + \mu P_{0110} = 0 \\
-(\lambda + \mu) P_{0001} + \mu P_{1001} + \mu P_{0011} + \mu P_{0101} = 0 \\
\lambda P_{1000} + \lambda P_{0100} - (\lambda + 2\mu) P_{1100} + \mu P_{1110} + \mu P_{1101} = 0 \\
\lambda P_{0010} - (\lambda + 2\mu) P_{1010} + \mu P_{1110} + \mu P_{1011} = 0 \\
\lambda P_{0001} - (\lambda + 2\mu) P_{1001} + \mu P_{1101} + \mu P_{1011} = 0 \\
-(\lambda + 2\mu) P_{0011} + \mu P_{1011} + \mu P_{0111} = 0 \\
-(\lambda + 2\mu) P_{0101} + \mu P_{1101} + \mu P_{0111} = 0 \\
-(\lambda + 2\mu) P_{0110} + \mu P_{1110} + \mu P_{0111} = 0 \\
\lambda P_{1100} + \lambda P_{1010} + \lambda P_{0110} - (\lambda + 3\mu) P_{1110} + \mu P_{1111} = 0 \\
\lambda P_{1001} + \lambda P_{0101} - (\lambda + 3\mu) P_{1101} + \mu P_{1111} = 0 \\
\lambda P_{0011} - (\lambda + 3\mu) P_{1011} + \mu P_{1111} = 0 \\
-(\lambda + 3\mu) P_{0111} + \mu P_{1111} = 0 \\
P_{0000} + P_{1000} + P_{0100} + P_{0010} + P_{0001} + P_{1100} + P_{1010} + P_{1001} + P_{0011} + P_{0101} + P_{0110} + P_{1110} + P_{1101} + P_{1011} + P_{0111} + P_{1111} = 1
\end{cases}
$$

$$\begin{cases} P_{0000}: \dfrac{24\mu^4}{\lambda^4+4\lambda^3\mu+12\lambda^2\mu^2+24\lambda\mu^3+24\mu^4}, \quad P_{0001}: \dfrac{6\lambda^4\mu^3}{\lambda^7+7\lambda^6\mu+30\lambda^5\mu^2+90\lambda^4\mu^3+192\lambda^3\mu^4+288\lambda^2\mu^5+288\lambda\mu^6+144\mu^7}, \\ P_{0010}: \dfrac{\lambda^3\mu^3\left(6\lambda^5+40\lambda^4\mu+144\lambda^3\mu^2+336\lambda^2\mu^3+456\lambda\mu^4+288\mu^5\right)}{\lambda^{11}+13\lambda^{10}\mu+88\lambda^9\mu^2+402\lambda^8\mu^3+1364\lambda^7\mu^4+3564\lambda^6\mu^5+7248\lambda^5\mu^6+11400\lambda^4\mu^7+13536\lambda^3\mu^8+11520\lambda^2\mu^9+6336\lambda\mu^{10}+1728\mu^{11}}, \\ P_{0011}: \dfrac{2\lambda^4\mu^2}{\lambda^6+8\lambda^5\mu+34\lambda^4\mu^2+96\lambda^3\mu^3+192\lambda^2\mu^4+240\lambda\mu^5+144\mu^6}, \\ P_{0100}: \dfrac{\lambda^2\mu^3\left(6\lambda^5+56\lambda^4\mu+260\lambda^3\mu^2+624\lambda^2\mu^3+792\lambda\mu^4+432\mu^5\right)}{\lambda^{10}+14\lambda^9\mu+95\lambda^8\mu^2+418\lambda^7\mu^3+1328\lambda^6\mu^4+3164\lambda^5\mu^5+5628\lambda^4\mu^6+7248\lambda^3\mu^7+6384\lambda^2\mu^8+3456\lambda\mu^9+864\mu^{10}}, \\ P_{0101}: \dfrac{2\lambda^4\mu^2\left(\lambda^5+9\lambda^4\mu+36\lambda^3\mu^2+72\lambda^2\mu^3+72\lambda\mu^4+36\mu^5\right)}{\lambda^{11}+16\lambda^{10}\mu+125\lambda^9\mu^2+638\lambda^8\mu^3+2376\lambda^7\mu^4+6768\lambda^6\mu^5+14964\lambda^5\mu^6+25560\lambda^4\mu^7+32976\lambda^3\mu^8+30528\lambda^2\mu^9+18144\lambda\mu^{10}+5184\mu^{11}}, \\ P_{0110}: \dfrac{2\lambda^3\mu^2\left(\lambda^4+6\lambda^3\mu+21\lambda^2\mu^2+42\lambda\mu^3+36\mu^4\right)}{\lambda^9+12\lambda^8\mu+71\lambda^7\mu^2+282\lambda^6\mu^3+822\lambda^5\mu^4+1788\lambda^4\mu^5+2880\lambda^3\mu^6+3312\lambda^2\mu^7+2448\lambda\mu^8+864\mu^9}, \\ P_{0111}: \dfrac{\lambda^4\mu}{\lambda^5+7\lambda^4\mu+24\lambda^3\mu^2+60\lambda^2\mu^3+96\lambda\mu^4+72\mu^5}, \quad P_{1000}: \dfrac{6\lambda\mu^3\left(\lambda^2+10\lambda\mu+12\mu^2\right)}{\lambda^6+8\lambda^5\mu+31\lambda^4\mu^2+84\lambda^3\mu^3+156\lambda^2\mu^4+168\lambda\mu^5+72\mu^6}, \\ P_{1001}: \dfrac{2\lambda^4\mu^2\left(\lambda^3+9\lambda^2\mu+18\lambda\mu^2+6\mu^3\right)}{\lambda^9+12\lambda^8\mu+71\lambda^7\mu^2+282\lambda^6\mu^3+822\lambda^5\mu^4+1788\lambda^4\mu^5+2880\lambda^3\mu^6+3312\lambda^2\mu^7+2448\lambda\mu^8+864\mu^9}, \\ P_{1010}: \dfrac{2\lambda^3\mu^2\left(\lambda^8+16\lambda^7\mu+107\lambda^6\mu^2+430\lambda^5\mu^3+1164\lambda^4\mu^4+2136\lambda^3\mu^5+2508\lambda^2\mu^6+1656\lambda\mu^7+432\mu^8\right)}{\lambda^{13}+18\lambda^{12}\mu+159\lambda^{11}\mu^2+920\lambda^{10}\mu^3+3902\lambda^9\mu^4+12796\lambda^8\mu^5+33252\lambda^7\mu^6+69024\lambda^6\mu^7+114024\lambda^5\mu^8+147600\lambda^4\mu^9+145152\lambda^3\mu^{10}}, \\ +102528\lambda^2\mu^{11}+46656\lambda\mu^{12}+10368\mu^{13} \\ P_{1011}: \dfrac{\lambda^4\mu\left(\lambda^2+6\lambda\mu+6\mu^2\right)}{\lambda^7+11\lambda^6\mu+58\lambda^5\mu^2+198\lambda^4\mu^3+480\lambda^3\mu^4+816\lambda^2\mu^5+864\lambda\mu^6+432\mu^7}, \\ P_{1100}: \dfrac{2\lambda^2\mu^2\left(\lambda^4+12\lambda^3\mu+54\lambda^2\mu^2+96\lambda\mu^3+72\mu^4\right)}{\lambda^8+10\lambda^7\mu+52\lambda^6\mu^2+180\lambda^5\mu^3+452\lambda^4\mu^4+816\lambda^3\mu^5+1008\lambda^2\mu^6+768\lambda\mu^7+288\mu^8}, \\ P_{1101}: \dfrac{\lambda^4\mu\left(\lambda^4+8\lambda^3\mu+24\lambda^2\mu^2+24\lambda\mu^3+12\mu^4\right)}{\lambda^9+11\lambda^8\mu+64\lambda^7\mu^2+252\lambda^6\mu^3+732\lambda^5\mu^4+1596\lambda^4\mu^5+2592\lambda^3\mu^6+3024\lambda^2\mu^7+2304\lambda\mu^8+864\mu^9}, \\ P_{1110}: \dfrac{\lambda^3\mu\left(\lambda^3+6\lambda^2\mu+18\lambda\mu^2+24\mu^3\right)}{\lambda^7+7\lambda^6\mu+30\lambda^5\mu^2+90\lambda^4\mu^3+192\lambda^3\mu^4+288\lambda^2\mu^5+288\lambda\mu^6+144\mu^7}, \quad P_{1111}: \dfrac{\lambda^4}{\lambda^4+4\lambda^3\mu+12\lambda^2\mu^2+24\lambda\mu^3+24\mu^4} \end{cases}$$

Keturių šerdžių skaičiavimai kai A kinta nuo 0,1 iki 2 Erl:

λ	μ	A	U1	U2	U3	U4	P0000	P0001	P0010	P0011	P0100	P0101	P0110	P0111	P1000	P1001	P1010	P1011	P1100	P1101	P1110	P1111
0,1	1	0,1	9,09E-02	0,008638	0,000437	1,47E-05	0,904837	3,41E-06	0,000135	1,18E-06	0,004041	1,20E-06	7,06E-05	1,22E-06	0,086305	1,38E-06	7,71E-05	1,25E-06	0,004373	1,30E-06	0,000147	3,77E-06
0,2	1	0,2	0,166667	0,030055	0,00306	0,000207	0,818733	4,47E-05	0,000879	1,60E-05	0,013148	1,66E-05	0,000479	1,71E-05	0,149675	2,11E-05	0,00056	1,81E-05	0,015282	1,94E-05	0,001037	5,46E-05
0,3	1	0,3	0,230769	0,059194	0,009037	0,000925	0,74083	0,000185	0,002411	6,86E-05	0,024214	7,27E-05	0,001374	7,58E-05	0,195439	9,96E-05	0,001691	8,20E-05	0,030031	9,14E-05	0,003085	0,00025
0,4	1	0,4	0,285714	0,092664	0,018759	0,002576	0,670361	0,00048	0,004651	0,000184	0,035429	0,000199	0,002771	0,00021	0,227585	0,000288	0,003555	0,000232	0,046631	0,000268	0,006441	0,000715
0,5	1	0,5	0,333333	0,128205	0,032132	0,005539	0,606635	0,00096	0,007407	0,000383	0,045797	0,000421	0,004612	0,000451	0,249154	0,000635	0,006115	0,000506	0,063663	0,000602	0,011078	0,00158
0,6	1	0,6	0,375	0,164326	0,04878	0,010115	0,549028	0,001633	0,01046	0,000677	0,05482	0,000758	0,006801	0,000824	0,262504	0,001178	0,009258	0,000936	0,080154	0,001146	0,016859	0,002965
0,7	1	0,7	0,411765	0,20006	0,068188	0,016506	0,496976	0,002483	0,01361	0,00107	0,062305	0,001216	0,009231	0,001344	0,269485	0,001935	0,012835	0,001546	0,095472	0,00194	0,023581	0,004972
0,8	1	0,8	0,444444	0,234801	0,089799	0,024812	0,449964	0,003482	0,016691	0,001561	0,068241	0,001797	0,011798	0,002021	0,271557	0,00291	0,016685	0,002349	0,109237	0,003012	0,031015	0,007679
0,9	1	0,9	0,473684	0,268181	0,11307	0,035038	0,407525	0,004591	0,019583	0,00214	0,072716	0,002494	0,01441	0,002857	0,269882	0,004089	0,020657	0,003351	0,121257	0,004376	0,038931	0,011141
1	1	1	5,00E-01	0,3	0,1375	0,047115	0,369231	0,005769	0,022203	0,002797	0,075874	0,003293	0,016987	0,003846	0,265385	0,005449	0,024621	0,004545	0,131469	0,006031	0,047115	0,015385
1,1	1	1,1	0,52381	0,330165	0,162654	0,060913	0,33469	0,006976	0,024502	0,003517	0,077879	0,004176	0,019469	0,00498	0,258802	0,006956	0,028468	0,005924	0,1399	0,007967	0,055375	0,020417
1,2	1	1,2	0,545455	0,358655	0,18816	0,076259	0,303545	0,008175	0,02646	0,004285	0,078899	0,005126	0,021811	0,006244	0,250721	0,008574	0,032116	0,007469	0,146641	0,010159	0,063549	0,026226
1,3	1	1,3	0,565217	0,385498	0,213712	0,092956	0,275469	0,009336	0,028073	0,005086	0,079094	0,006122	0,023978	0,007624	0,241606	0,010266	0,035503	0,009161	0,151816	0,012578	0,071504	0,032782
1,4	1	1,4	0,583333	0,41075	0,239065	0,110792	0,250163	0,010435	0,029354	0,005906	0,078611	0,007144	0,025952	0,009101	0,231827	0,011994	0,038592	0,01098	0,155571	0,01519	0,079137	0,040043
1,5	1	1,5	0,6	0,434483	0,264025	0,129556	0,227353	0,011453	0,030325	0,006731	0,077583	0,008176	0,027722	0,010657	0,22167	0,013725	0,04136	0,012901	0,158059	0,017957	0,086371	0,047957
1,6	1	1,6	0,615385	0,45678	0,288443	0,149043	0,206794	0,012376	0,031013	0,007549	0,076123	0,009199	0,029285	0,012276	0,211359	0,015429	0,043798	0,014902	0,159434	0,020842	0,093152	0,056469
1,7	1	1,7	0,62963	0,477729	0,31221	0,169057	0,188259	0,013198	0,031446	0,008351	0,07433	0,010202	0,030644	0,013939	0,201065	0,017083	0,045909	0,01696	0,159845	0,023808	0,099445	0,065515
1,8	1	1,8	0,642857	0,497414	0,335248	0,189421	0,171544	0,013916	0,031656	0,009128	0,072288	0,011172	0,031807	0,015632	0,19092	0,018664	0,047703	0,019055	0,159428	0,02682	0,105234	0,075033
1,9	1	1,9	0,655172	0,515922	0,357504	0,209974	0,156466	0,014528	0,031673	0,009874	0,070067	0,012099	0,032783	0,017339	0,181018	0,020158	0,049194	0,021168	0,158313	0,029847	0,110513	0,084962
2	1	2	0,666667	0,533333	0,378947	0,230576	0,142857	0,015038	0,031523	0,010582	0,067725	0,012977	0,033584	0,019048	0,171429	0,021554	0,050404	0,02328	0,156614	0,03286	0,115288	0,095238

3 priedas. Imitacinio modelio sudarymas

Imitacinio modelio Python kodas:

```
%pylab
from __future__ import division
import random

def ExpGen(N, intens):
    x = []
    for i in xrange(N):
        x.append(random.expovariate(intens))
    return x

def ConstGen(N, intens):
    x = []
    for i in xrange(N):
        x.append(1/intens)
    return x

def tpGen(dt):
    tp = []
    t = 0
    for dti in dt:
        t = t + dti
        tp.append(t)
    return tp

def Zeros(k):
    x = []
    for i in xrange(k):
        x.append(0)
    return x

def NegOnes(k):
    x = []
    for i in xrange(k):
        x.append(-1)
    return x
def H(n):
    if n<0:
        return 0
    else:
        return 1

def Hl(XTkr,XTl):
    h = []
    for i in range(len(XTl)):
        h.append(H(XTkr-XTl[i]))
    return array(h)

class Paraiska:
    def __init__(self,i,tp,tu):
        self.i = i
        self.tp = tp # laiko momentas kada par. atejo
        self.tu = tu # par. trukme
        self.tbuf = 0 # par. buvimo laikas buferyje
        self.tsis = -1 # par. buvimo laikas sistemoje (buferyje + kanale)
        self.tapr = -1 # par. apt. kanale pradzios laiko momentas
        self.tpab = -1 # par. apt. kanale pabaigos laiko momentas
        self.k = -1 # kanalas kuriuo par. buvo apt.
    def info(self):
        #self.tbuf = self.tapr - self.tp
        #self.tsis = self.tpab - self.tp
        #if self.k == -1:
        #    self.tbuf = -1
        #    self.tsis = -1
```

```
64.         #    self.tapr = -1
65.         #    self.tpab = -1
66.         return "i = %d,\ttp = %.3f,\ttu = %.3f,\ttbuf = %.3f,\ttsis = %.3f,\ttapr = %.3f,\
    ttpab = %.3f,\tk = %d"%(self.i,self.tp,self.tu,self.tbuf,self.tsis,self.tapr,self.tpab,sel
    f.k)
67.
68. class Paraiskos:
69.     def __init__(self):
70.         self.paraiskos = []
71.         self.N = 0
72.         self.tnext = None
73.
74.     def fill(self,tp,tu):
75.         if len(tp) == len(tu):
76.             for i in xrange(len(tp)):
77.                 self.add(Paraiska(i,tp[i],tu[i]))
78.
79.     def add(self,paraiska):
80.         self.paraiskos.append(paraiska)
81.         self.get_N()
82.
83.     def take(self):
84.         p = None
85.         if self.get_N() > 0:
86.             p = self.paraiskos.pop(0) #paima pirma paraiska ir pasalina is self.paraiskos
    listo
87.         return p
88.
89.     def get_N(self):
90.         self.N = len(self.paraiskos)
91.         return self.N
92.
93.     def get_tnext(self):
94.         if self.get_N() > 0:
95.             self.tnext = self.paraiskos[0].tp
96.         else:
97.             self.tnext = None
98.         return self.tnext
99.
100.     def info(self):
101.         if self.get_N() > 0:
102.             for i,p in enumerate(self.paraiskos):
103.                 #print "i = %d,\ttp = %.3f,\ttu = %.3f,\ttbuf = %.3f,\ttsis = %.3f,\ttap
    r = %.3f,\ttpab = %.3f,\tk = %d"%(p.i,p.tp,p.tu,p.tbuf,p.tsis,p.tapr,p.tpab,p.k)
104.                 print i,':',p.info()
105.         else:
106.             print None
107.
108. class Kanalai:
109.     def __init__(self, Nk):
110.         self.Nk = Nk
111.         self.kanalai = [] # kanaluose bus talpinamos paraiskos
112.         self.ktfin = [] # laiko momentai iki kuriu kanalai bus uzimti
113.         self.kanbusena = [] # kanaluose bus talpinamos paraiskos
114.         for i in xrange(Nk):
115.             self.kanalai.append(None)
116.             self.ktfin.append(None)
117.             self.kanbusena.append(None)
118.         self.tfin = None # artimiausias laiko momentas, kuriuo baigs aptarnauti par.
119.         self.Nfin = None # kanalo nr. kuriame bus tfin
120.         self.Nu_t = 0 # uzimtu kanalu skaicius laiko momentui t
121.         self.Nk_t = 0 # pirmas surastas laisvas kanalas laiko momentui t
122.         self.InParSr = Paraiskos() # statistikai - paraiskos, kurios buvo aptarnautos
123.
124.
125.     def get_tfin(self):
126.         # suranda artimiausia laiko momenta ir kanala, kuriame par. bus baigta apt. anks
    ciausiai
127.         # naudojama - tam, kad surasti laiko momenta, kada paimti is buferio kita par.
```

```
128.         self.tfin = None
129.         self.Nfin = None
130.         for i in xrange(self.Nk):
131.             if self.ktfin[i] != None:
132.                 if self.tfin == None or self.ktfin[i] < self.tfin:
133.                     self.tfin = self.ktfin[i]
134.                     self.Nfin = i
135.
136.         #--------atlaisvina kanala
137.         if self.Nfin != None and self.kanalai[self.Nfin] != None:
138.             #print 'kanalo info:',self.Nfin, self.kanalai[self.Nfin]
139.             #self.kanalai[self.Nfin].tpab = self.tfin
140.             #self.kanalai[self.Nfin].k = self.Nfin
141.             self.kanalai[self.Nfin] = None
142.         #--------
143.
144.         return self.tfin#, self.Nfin
145.
146.     def get_Nut(self,t):
147.         # suranda kiek laiko momentu t yra uzimtu kanalų ir laisvo kanalo numeri, kuri g
    alima butu uzimti
148.         self.Nu_t = 0 # uzimtu kanalu sk laiko momentu t
149.         self.Nk_t = None # kanalo nr. kuri galima uzimti laiko momentu t
150.         #kanbus = None
151.         for i in xrange(self.Nk):
152.             if self.ktfin[i] > t:
153.                 self.Nu_t = self.Nu_t + 1
154.                 self.kanbusena[i] = 1
155.             else:
156.                 self.kanbusena[i] = 0
157.                 #if self.Nk_t == None:
158.                 #    self.Nk_t = i # kai naudojamas kanalu uzemimo budas: uzimk nuo prad
    zios pirma surasta laisva kanala
159.                 #else:
160.                    # self.Nk_t = i
161.         #print self.kanbusena
162.         if self.Nu_t < self.Nk:
163.             for i in xrange(self.Nk):
164.                 if self.ktfin[i] <= t:
165.                     self.Nk_t = i
166.                     break
167.         #kanbus = self.kanbusena
168.         return self.Nu_t, self.Nk_t, list(self.kanbusena)
169.
170.
171.
172.     def add(self, t, par):
173.         # ideda paraiska i laisva kanala
174.         self.Nu_t, self.Nk_t, kanbus = self.get_Nut(t)
175.
176.         #if self.Nk_t != None:
177.         par.tapr = t
178.         par.tpab = t+par.tu
179.         par.tbuf = t-par.tp
180.         par.tsis = par.tpab - par.tp
181.         par.k = self.Nk_t
182.         self.kanalai[self.Nk_t] = par
183.         self.ktfin[self.Nk_t] = t+par.tu
184.         self.InParSr.add(par)
185.
186.         #else:
187.         #    print 'Nera laisvu kanalu'
188.
189.         #print 'Kanalu uzimtumas:' ,self.Nu_t, self.Nk_t,self.kanalai,self.ktfin
190.
191. class Buferis:
192.     def __init__(self, Nb):
193.         self.Nb = Nb # vietu sk buferyje
194.         self.Nbu = 0 # uzimtu vietu sk
```

```
195.         self.vietos = [] # cia bus talpinamos paraiskos
196.         self.BufNetilpo = Paraiskos() # statistikai -
    paraiskos, kurios netilpo buferyje
197.         self.BufBuvo = Paraiskos() # statistikai - paraiskos, kurios buvo buferyje
198.
199.     def add(self, par):
200.         self.Nbu = self.getNbu()
201.         if self.Nbu < self.Nb:
202.             self.vietos.append(par)
203.             self.BufBuvo.add(par)
204.         if self.Nbu == self.Nb:
205.             #print 'Buferis pilnas'
206.             self.BufNetilpo.add(par)
207.
208.     def getNbu(self):
209.         self.Nbu = len(self.vietos)
210.         return self.Nbu
211.
212.     def take(self):
213.         par = None
214.         if self.getNbu() > 0:
215.             par = self.vietos.pop(0)
216.         return par
217.
218. def Pkf(k,paraiskos): #tikimybes kad uzimtas k-asis kanalas
219.     n = 0
220.     for p in paraiskos:
221.         if p.k == k:
222.             n = n + 1
223.     return n/len(paraiskos)
224.
225. def Ukf(k,paraiskos): #kanalu apkrautumas
226.     tsum = 0
227.     Tpab = paraiskos[-1].tpab
228.     for p in paraiskos:
229.         if p.k == k:
230.             tsum = tsum + p.tpab-p.tapr
231.     return tsum/Tpab
232.
233. def XT(n,h,L):
234.     if n <= 1:
235.         xt = 0
236.     else:
237.         xt = (n-n*exp(-(n+1)*2*h*L))/(1+n*exp(-(n+1)*2*h*L))
238.     return xt
239.
240. def Sistema(N,l,m,Nk,Nb,h,L):
241.     dt = ExpGen(N,l) # laikotarpiai tarp paraisku
242.     #dt = ConstGen(N,l)
243.     tp = tpGen(dt) # paraiskos atejimo laiko momentai laiko momentai kada paraiskos/pliu
    psniai ateina i sistema
244.
245.     tu = ExpGen(N,m) # obs PLIUPSNIU/PARAISKU trukmes
246.     #tu = ConstGen(N,m) # paraiskos apt. trukme
247.
248.
249.     InParSr = Paraiskos() # Ateinanciu par. srautas
250.     InParSr.fill(tp,tu)
251.
252.     OrigPar = Paraiskos()
253.     OrigPar.fill(tp,tu)
254.
255.     Kan = Kanalai(Nk)
256.     #print Kan.kanalai
257.
258.     Buf = Buferis(Nb)
259.     #print Buf.vietos
260.
261.
```

```
    i = 0
    #par = InParSr.take()
    Stat = []
    while i < N:
        #print 'i-----------',i
        tnext = InParSr.get_tnext()
        #print 'tnext =',tnext
        tfin = Kan.get_tfin()
        #print 'tfin =',tfin

        if tnext < tfin or tfin==None or Buf.getNbu() == 0:
        #if tnext < tfin or tfin==None:
            t = tnext
            i = i + 1
            par = InParSr.take()

            Nut,Nkt,kanbus=Kan.get_Nut(t)
            if Nut == Nk:
                Buf.add(par)
            else:
                Kan.add(t,par)

            #stat #tik statistikai - veikimo algoritmui tai gali buti uzkomentuota
            Nut,Nkt,kanbus=Kan.get_Nut(t)
            #print kanbus
            Stat.append([t,Nut,Buf.getNbu(),kanbus])

        if tnext > tfin and tfin!=None:
            t = tfin

            if Buf.getNbu() > 0:
                par = Buf.take()
                Kan.add(t,par)

    Pblok =  1-Kan.InParSr.N/N

    apt_paraiskos = Kan.InParSr.paraiskos

    Uk = []
    for k in range(Nk):
        Uk.append(Ukf(k,apt_paraiskos))

    Pk = []
    for k in range(Nk):
        Pk.append(Pkf(k,apt_paraiskos))
    t = []
    Nk_t = []
    Buf_t = []
    kanbus_t = []
    XT_t = []
    for s in Stat:
        t.append(s[0])
        Nk_t.append(s[1])
        Buf_t.append(s[2])
        kanbus_t.append(s[3])
        XT_t.append(XT(s[1],h,L))

    XTvid = mean(XT_t)
        #print s[1],XT(int(s[1]),h,L)
    return Pblok,Uk,Pk,t,Nk_t,Buf_t,kanbus_t,XT_t,XTvid

N = 200000 # paraisku sk.
l = arange(0.1,15,0.1)
m = 1 # paraisku apt. intensyvumas
Nk = 3 # kanalu sk. (serdziu sk)
Nb = 0 # buferio talpa
L = 5 #km - skaidulos ilgis
h = 3e-6# koef ivertinantis XT
XTkr = -18# dB
```

```
331.
332. Pblok,Uk,Pk,t,Nk_t,Buf_t,kanbus_t,XT_t,XTvid = Sistema(N,l,m,Nk,Nb,L,h)
333.
334. Pblok_l = []
335. Uk_l = []
336. Pk_l = []
337. XT_t_l = []
338. Nk_t_l = []
339. Buf_t_l = []
340. t_l = []
341. XTvid_l = []
342. for li in l:
343.     print li
344.     Pblok,Uk,Pk,t,Nk_t,Buf_t,kanbus_t,XT_t,XTvid = Sistema(N,li,m,Nk,Nb,L,h)
345.     Pblok_l.append(Pblok)
346.     Uk_l.append(Uk)
347.     Pk_l.append(Pk)
348.     XT_t_l.append(XT_t)
349.     Nk_t_l.append(Nk_t)
350.     Buf_t_l.append(Buf_t)
351.     t_l.append(t)
352.     XTvid_l.append(XTvid)
```

Persidengimų skaičiavimų rezultatai:

XTkr	L	XT (7 šerd)	XT (6 šerd)	XT (5 šerd)	XT (4 šerd)	XT (3 šerd)	XT (2 šerd)
-18	0	-53,8256	-54,4643	-55,2386	-56,201	-57,448	-59,2083
-18	5	-36,8367	-37,4736	-38,2489	-39,2113	-40,458	-42,2186
-18	10	-33,83	-34,4622	-35,2381	-36,2002	-37,4477	-39,2082
-18	25	-29,8447	-30,4822	-31,2585	-32,2207	-33,4679	-35,2286
-18	50	-26,8329	-27,4713	-28,2464	-29,2091	-30,4571	-32,218
-18	100	-23,8188	-24,4571	-25,2334	-26,1973	-27,4451	-29,207
-18	150	-22,0531	-22,6933	-23,4701	-24,4341	-25,6831	-27,4455
-18	200	-20,7998	-21,4409	-22,2187	-23,1832	-24,4326	-26,1954
-18	250	-19,8285	-20,4664	-21,246	-22,2117	-23,462	-25,2257
-18	300	-19,0307	-19,6723	-20,4525	-21,4178	-22,6688	-24,4332
-18	350	-18,3574	-19,0005	-19,7803	-20,7469	-21,9982	-23,763
-18	400	-17,7737	-18,4159	-19,1965	-20,1647	-21,417	-23,1825
-18	450	-17,2612	-17,9005	-18,683	-19,6513	-20,9041	-22,6704
-18	500	-16,7985	-17,4417	-18,223	-19,1915	-20,4452	-22,2122
-18	550	-16,3816	-17,0252	-17,8065	-18,7757	-20,03	-21,7975
-18	600	-15,9985	-16,6432	-17,4258	-18,3959	-19,6509	-21,419
-18	650	-15,65	-16,2905	-17,0759	-18,0465	-19,3017	-21,0708
-18	700	-15,3208	-15,9676	-16,7518	-17,7226	-18,979	-20,7483
-18	750	-15,0192	-15,664	-16,4489	-17,4211	-18,6779	-20,4479
-18	800	-14,7339	-15,3791	-16,1656	-17,1389	-18,3963	-20,1671
-18	850	-14,4665	-15,1134	-15,9008	-16,8733	-18,1318	-19,9031
-18	900	-14,2157	-14,8619	-15,6494	-16,6239	-17,882	-19,6543
-18	950	-13,9755	-14,6226	-15,4128	-16,3867	-17,6462	-19,4187
-18	1000	-13,7497	-14,3991	-15,1866	-16,1618	-17,422	-19,1953
-18	1050	-13,5335	-14,1835	-14,9725	-15,9481	-17,2088	-18,9829
-18	1100	-13,3263	-13,9794	-14,768	-15,7441	-17,0055	-18,7802
-18	1150	-13,132	-13,7814	-14,572	-15,5493	-16,8111	-18,5864
-18	1200	-12,9416	-13,593	-14,3848	-15,3626	-16,6251	-18,401
-18	1250	-12,7625	-13,4128	-14,2056	-15,1834	-16,4464	-18,2231
-18	1300	-12,5849	-13,2387	-14,0318	-15,0106	-16,2749	-18,0523
-18	1350	-12,4163	-13,0718	-13,8648	-14,845	-16,1096	-17,8876
-18	1400	-12,2559	-12,91	-13,7052	-14,6851	-15,9506	-17,729
-18	1450	-12,1028	-12,7538	-13,5491	-14,531	-15,7968	-17,576
-18	1500	-11,9498	-12,6053	-13,3999	-14,3818	-15,6482	-17,4281