VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
KOMPIUTERIŲ INŽINERIJOS KATEDRA

Arsenij Andrijanov

INFORMACIJOS SRAUTŲ KOMPIUTERIŲ SISTEMOSE MODELIAVIMAS IR TYRIMAS
SIMULATION AND INVESTIGATION OF THE INFORMATION FLOWS IN COMPUTER SYSTEMS

Magistro baigiamasis darbas

Elektronikos inžinerijos studijų kryptis
Kompiuterių inžinerijos studijų programa, valstybinis kodas 621H69001
Kompiuterių technologijų specializacija

Vilnius, 2013

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
KOMPIUTERIŲ INŽINERIJOS KATEDRA

TVIRTINU
Katedros vedėjas

_____
(*parašas*)
prof. dr. Algirdas Baškys
2013 m. _____ mėn. __ d.

Arsenij Andrijanov

INFORMACIJOS SRAUTŲ KOMPIUTERIŲ SISTEMOSE MODELIAVIMAS IR TYRIMAS
SIMULATION AND INVESTIGATION OF THE INFORMATION FLOWS IN COMPUTER SYSTEMS

Magistro baigiamasis darbas

Elektronikos inžinerijos studijų kryptis
Kompiuterių inžinerijos studijų programa, valstybinis kodas 621H69001
Kompiuterių technologijų specializacija

Vadovas	prof. habil. dr. Julius Skudutis	_____	_____
	(*mokslinis vardas ir laipsnis, vardas, pavardė*)	(*parašas*)	(*data*)
Lietuvių kalbos konsultantė	lektorė Vaida Buivydienė	_____	_____
	(*mokslinis vardas ir laipsnis, vardas, pavardė*)	(*parašas*)	(*data*)

Vilnius, 2013

Vilniaus Gedimino technikos universiteto egzaminų, sesijų ir baigiamųjų darbų rengimo bei gynimo organizavimo tvarkos aprašo 2011-2012 m. m.
1 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Arsenij Andrijanov, 20091257

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Elektronikos fakultetas

(Fakultetas)

Kompiuterių inžinerija, KTfm-11

(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO) SĄŽININGUMO DEKLARACIJA

2013 m. birželio 1 d.

Patvirtinu, kad mano baigiamasis darbas tema „Informacijos srautų kompiuterių sistemose modeliavimas ir tyrimas" patvirtintas 2011 m. lapkričio 22 d. dekano potvarkiu Nr. 260el, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Mano darbo vadovas prof. habil. dr. Julius Skudutis.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

	Arsenij Andrijanov
(Parašas)	(Vardas ir pavardė)

Vilnius, 2013

Vilniaus Gedimino technikos universitetas
Elektronikos fakultetas
Kompiuterių inžinerijos katedra

ISBN ISSN
Egz. sk. **1**
Data **2013-06-06**

Elektronikos studijų programos baigiamasis magistro darbas
Pavadinimas: **Informacijos srautų kompiuterių sistemose modeliavimas ir tyrimas**
Autorius: **Arsenij Andrijanov** Vadovas: **prof. habil. dr. Julius Skudutis**

Kalba
[X] lietuvių
[ ] užsienio

Anotacija

Baigiamojo darbo tikslas – sukurti masinio aptarnavimo sistemų modeliavimo programinę įrangą ir sumodeliuoti bei ištirti įvairias masinio aptarnavimo kompiuterines sistemas. Modeliavimo programinė įranga buvo parašyta *C#* programavimo kalba. Programos branduolys – *Delsi 2.0* masinio aptarnavimo sistemų modeliavimo biblioteka. Sukurta programa buvo sumodeliuota ir ištirta keletas masinio aptarnavimo sistemų: sistemos su apribotu paraiškų eilės ilgiu, sistemos su apribotu paraiškų laukimo eilėje laiku, mišrios sistemos, vieno ir daugelio aptarnavimo kanalų sistemos. Taip pat buvo sumodeliuotas vidutinio dydžio kompiuterių tinklo komutatoriaus darbas ir ištirtos jo pagrindinės charakteristikos. Nustatyta optimali komutatoriaus paketų aptarnavimo sparta ir optimalus paketų buferio dydis.

Darbo apimtis – 51 p. teksto be priedų, 14 iliustracijų, 6 lentelės, 16 bibliografinių šaltinių.

Reikšminiai žodžiai: masinio aptarnavimo sistema, MAS, aptarnavimo intensyvumas, atvykimų intensyvumas, buferio talpa, modeliavimas, C#, paprastasis srautas, eksponentinis pasiskirstymas, Delsi.

Vilniaus Gediminas technical university
Faculty of **Electronics**
Computer Engineering department

ISBN ISSN
Copies No. **1**
Date **2013-06-06**

Electronics Engineering study programme master thesis.

Title: **Simulation and investigation of the information flows in computer systems**

Author **Arsenij Andrijanov** Executive **prof. habil. dr. Julius Skudutis**

Thesis language

[X] Lithuanian
[ ] Foreign (English)

Annotation

Master thesis objective – to create universal modeling software for queueing systems and with its help to conduct simulation experiments, investigating several various queueing systems. The software was written with the help of *C#* programming language. The core of the created software is *Delsi. 2.0* simulation library. Using created software several different queueing systems were modeled and investigated: systems with the restricted queue length, systems with the restricted transaction waiting time in a queue, combined systems, semi- and multichannel systems. Also there was modeled the operation of a middle-sized computer network switch and its main characteristics were investigated. The optimal packet serving rate and packet buffer size of the switch were found.

Thesis consists of: 51 p. of text without extras, 14 illustrations, 6 tables, 16 bibliographical entries.

Keywords: queueing system, service rate, arrival rate, buffer size, modeling, C#, exponential distribution, Delsi.

TURINYS

Santrumpos ir žymenys

MAS	(liet. *Masinio aptarnavimo sistema*) – sistema, aptarnaujanti į ją patekusias paraiškas
JMT	(angl. *Java Modelling Tools*) – MAS modeliavimo įrankis
XML	(angl. *eXtensible Markup Language*) – „žymėjimo“ kalba dokumentams, kuriuose saugoma struktūrizuota tekstinė informacija.
FIFO	(angl. *First In First Out*) – vienas iš galimų eilių reguliavimo metodų, pagal kurį eilę pirmiausiai palieka tos paraiškos, kurios pateko į sistemą anksčiau.
λ	Paraiškų atvykimo srauto intensyvumas
μ	Paraiškų aptarnavimo srauto intensyvumas
ν	Paraiškų paliekančių eilę srauto intensyvumas
m_{atv}	Vidutinis paraiškų atvykimo į sistemą laikas
m_{apt}	Vidutinis paraiškų aptarnavimo laikas
α	MAS efektyvumas (paraiškų atvykimo ir aptarnavimo srautų intensyvumų santykis)
β	Paraiškų išėjimo iš eilės ir aptarnavimo srautų santykis
q	Santykinė MAS pralaida
Q	Absoliutinė MAS pralaida
p_{apt}	Tikimybė, kad paraiška bus aptarnauta
p_{atm}	Tikimybė, kad paraiška bus atmesta
$\overline{k}$	Vidutinis užimtų kanalų skaičius / Vidutinis paraiškų skaičius sistemoje (aptarnaujamų ir stovinčių eilėje)
$\overline{r}$	Vidutinis paraiškų, stovinčių eilėje, skaičius
$\overline{t}_{lauk}$	Vidutinis paraiškų laukimo laikas eilėje
m_{lauk}	Vidutinis laukiančių eilėje paraiškų skaičius
$\overline{t}_{MAS}$	Vidutinis paraiškos gyvavimo laikas
C#	„C“ šeimos programavimo kalba
Mbps	Megabitai per sekundę
kB	Kilobaitas

Įvadas

Kompiuterių sistemų informacijos srautus patogu modeliuoti, taikant Masinio aptarnavimo sistemų (MAS) teoriją [1]. MAS teorija tai tikimybių teorijos poskyris, kurio tyrimų tikslas – racionalus aptarnavimo sistemos ir aptarnavimo proceso struktūros parinkimas, priklausomai nuo sistemai keliamų reikalavimų, įeinančių/išeinančių srautų parametrų, eilių ilgių, laukimo laikų ir kitų charakteristikų. MAS teorijoje naudojami tikimybių teorijos ir matematiniai statistiniai metodai.

MAS teorija atsirado XX a. pradžioje, esant poreikiui tvarkyti ir reguliuoti telefoninių stočių darbą [2]. Pirmas MAS teorijos tikslas buvo iš anksto apskaičiuoti vartotojų aptarnavimo kokybę, priklausomai nuo naudojamų įrenginių kiekio. Šiais laikais, sparčiai vystantis skaitmeninėms kompiuterinėms technologijoms, MAS teorijos aktualumas vis ryškėja. Nors MAS teorija gali būti pritaikyta daugybėje įvairiausių sričių (pvz., klientų aptarnavimo kokybės tyrimai įmonėse, gamybos proceso optimizavimas pramonėje, žmonių eilių sumažinimo sprendimai įstaigose), abstrahavus, galima teigti kad MAS teorijos tyrimo objektas yra paraiška (klientas, detalė, transakcija) [2].

MAS teorijos elementai ir sąvokos tinkami, tiriant informacijos srautus kompiuterinėse sistemose. Tokiu būdu MAS teorijos požiūriu, paraiškos kompiuterinėse sistemose – tai informacijos paketai (vienetai, blokai). Būdami kompiuterių sistemoje, pvz., tinkle, informacijos paketai turi tam tikras savybes: paketų srauto atvykimo laiko į sistemą pasiskirstymo dėsnis, paketų aptarnavimo pasiskirstymo dėsnis, vidutiniai aptarnavimo ir atvykimo laikai, laukimo eilėje laikas ir t. t. Visi parametrai matematiškai aprašomi MAS teorijos, tai suteikia galimybę ją taikant, skaitinėmis išraiškomis įvertinti sistemos parametrus. Tokiu būdu, gavus griežtai ir tiksliai apibrėžtus kompiuterinės sistemos parametrus, galima imtis priemonių, siekiant patobulinti tiriamą sistemą arba pakeisti jos veikimą pagal iškeltus reikalavimus [2].

Taigi, informacijos srautų kompiuterių sistemose tyrimams puikiai tinka MAS teorija. Ją taikant, įmanoma apskaičiuoti aptarnavimo tikimybę, aptarnavimo sistemos (serverio) prastovos tikimybę, vidutinį eilės ilgį, vidutinį paraiškos gyvavimo sistemoje laiką ir t. t. Turint šiuos duomenis, galima priimti atitinkamus sprendimus dėl sistemos pokyčių, reikalingų užsibrėžtiems tikslams pasiekti.

Darbo tikslas – sumodeliuoti ir ištirti įvairias informacijos srautų kompiuterines sistemas, taikant MAS teoriją [3]. Norint modeliuoti ir tyrinėti informacijos srautus kompiuterių sistemose, reikalingas tam tikras įrankis, kitaip tariant, programinis paketas. Todėl, darbo tikslo siekiant, keliami tokie uždaviniai:

- Sukurti funkcionalų ir patogų įrankį, leidžiantį modeliuoti ir tyrinėti informacijos srautus kompiuterių sistemose, taikant MAS teoriją.
- Išnagrinėti gautus modeliavimo duomenis ir toliau juos panaudoti tiriamų sistemų optimizavimui pagal iškeltus reikalavimus.

Norint išsamiai ištirti ir aprašyti sistemos parametrus, reikia kad įrankis leistų kurti modeliavimui įvairiausias masinio aptarnavimo sistemas. Todėl reikia, kad naudojamas modeliavimo įrankis turėtų tokias savybes:

- Įvairūs paraiškų atvykimo, laukimo ir aptarnavimo srautų statistinio pasiskirstymo dėsniai.
- Galimybė keisti paraiškų atvykimo (λ), aptarnavimo (μ) ir laukimo srautų pasiskirstymo dėsnius.
- Galimybė rinktis įvairius aptarnaujančių serverių kanalų skaičius.
- Keičiama sistemos buferio talpa.
- Galimybė stebėti įvairius parametrus:
    - paraiškų aptarnavimo tikimybė.
    - paraiškų atmetimo tikimybė.
    - sistemos prastovos tikimybė.
    - paraiškų gyvavimo laikas.
    - vidutinis eilės ilgis.
    - vidutinis paraiškų laukimo eilėje laikas.
- Galimybė stebėti sistemos charakteristikų pokyčius, kintant šitiems parametrams:
    - sistemos efektyvumas (α).
    - vidutinis paraiškų laukimo eilėje laikas.
    - buferio talpa (maksimalus eilės ilgis).
    - aptarnaujančių kanalų skaičius.

1. Masinio aptarnavimo teorijos elementai

Masinio aptarnavimo teorija pagrįsta matematiniais principais. Norint giliau suprasti masinio aptarnavimo teorijos esmę, reikia išnagrinėti esminius jos elementus. Kituose poskyriuose pateikiami trumpi pagrindinių MAS sistemų matematiniai aprašymai.

1.1. Masinio aptarnavimo sistemų klasifikavimas

Į kiekvieną MAS ateina paraiškų *atvykimo srautas* aptarnavimui. MAS darbo rezultatas – aptarnautų paraiškų *aptarnavimo srautas* [4].

Įvykių srautu yra vadinama vienalyčių įvykių, atsitinkančių tam tikrais, bendrai sakant, atsitiktiniais laiko momentais, seka.

Jeigu MAS vienu metu gali aptarnauti keletą paraiškų, tokia MAS vadinama *daugiakanale*, priešingu atveju MAS vadinama *vienkanale.*

Tiek vienkanalės MAS, tiek ir daugiakanalės MAS skirstomos į *atmetimais paraiškas MAS* ir *laukiančias MAS.*

Esant *atmetančiai* MAS, į sistemą atėjusi paraiška tuo momentu, kai visi kanalai yra užimti, gauna aptarnavimo atsisakymą ir palieka MAS.

Esant laukiančiai MAS, į sistemą atėjusi paraiška tuo momentu, kai visi kanalai yra užimti, stoja į paraiškų, laukiančių aptarnavimo, eilę. Kai tik vienas iš aptarnavimo kanalų atsilaisvina, aptarnavimui priimama viena iš paraiškų, stovinčių eilėje.

MAS su paraiškų eile skirstomos pagal eilės *sudarymo (disciplinos) principą*.

Eilės sudarymo principu yra vadinama schema, pagal kurią iš eilės atrenkamos paraiškos aptarnavimui. Dažniausiai naudojama:

1. Atsitiktinė paraiškų atranka iš eilės;
2. Paraiškos atranka iš eilės pagal jos *prioritetą*.
3. Paraiškos atranka priklausomai nuo jos patekimo į eilę *eiliškumo*.

Esant trečiam atvejui, paraiškos gali būti aptarnaujamos ir pagal schemą „pirmas atėjo – pirmas aptarnaujamas“, ir pagal schemą „paskutinis atėjo – pirmas aptarnaujamas“.

MAS su *eile* taip pat skirstomos į MAS su *neapribotu laukimu* ir MAS su *apribotu laukimu*.

Esant MAS su *neapribotu laukimu* kiekviena paraiška, patekusi į MAS, anksčiau ar vėliau bus aptarnauta.

Esant MAS su *apribotu laukimu*, paraiškų buvimui eilėje egzistuoja įvairūs apribojimai. Šitie apribojimai gali būti susiję su *paraiškų eilės ilgiu, paraiškos laukimo eilėje laiku, bendro*

paraiškos buvimo MAS laiko ir t. t. Pavyzdžiui MAS su apribotu laukimo eilėje laiku, paraiška palieka MAS po to, kai baigiasi jos laukimo laikas eilėje.

1.2. Paprasčiausias srautas ir jo savybės

Įvykių srautas vadinamas *paprasčiausiu įvykių srautu*, jeigu jis pasižymi *stacionarumo, poveiksmio nebuvimo* ir *ordinarumo* savybėmis.

1. Įvykių srautas vadinamas *stacionariu*, jeigu laiko intervale *T* įvykių skaičius priklauso tik nuo intervalo ilgio ir nepriklauso nuo intervalo *T* padėties laiko *t* ašyje.
2. *Be poveiksmio* (be sąveikos) vadinamas toks įvykių srautas, jeigu bet kuriems nepersidengiantiems laiko intervalams įvykių skaičius intervaluose nepriklauso nuo į kitus intervalus patenkančių įvykių skaičiaus.
3. Įvykių srautas vadinamas *ordinariu*, jei įvykiai, sudarantys srautą, atsitinka po vieną, o ne poromis, po tris ir t. t.

   Pastabos:

- Srautas, kurio įvykiai atsitinka tam tikru fiksuotu dažnumu, nėra paprasčiausias įvykių srautas.
- Paprasčiausias srautas turi *pastovų intensyvumą* (srauto *intensyvumu* vadinamas vidutinis įvykių skaičius, atsitinkantis per laiko vienetą).
- Toliau nagrinėjant MAS, bus laikoma, kad visi įvykių srautai yra paprasčiausi.

Paprasčiausias įvykių srautas glaudžiai susijęs su *Puasono pasiskirstymu*. Todėl, būtų teisingi tokie teiginiai:

1. Tikimybė, kad per laiko tarpą lygų *T* atsitiks lygiai *k* įvykių, esant paprasčiausiam srautui su intensyvumu λ , išreiškiama Puasono formule:

$$P_k = \frac{(\lambda T)^k}{k!} e^{-\lambda T}, k = 0,1,... \qquad (1.2.1)$$

2. Laiko atkarpos tarp gretimų įvykių iš paprasčiausio srauto su intensyvumu λ ilgis yra atsitiktinis dydis, pasiskirstęs pagal *rodiklinį (eksponentinį)* dėsnį su parametru λ . Rodiklinio pasiskirstymo tankis išreiškiamas formule:

$$f(t) = \begin{cases} 0 & , \ -\infty < t < 0 \\ \lambda \cdot e^{-\lambda t}, & 0 \le t < +\infty \end{cases} \qquad (1.2.2)$$

1.3. MAS efektyvumo rodikliai

Iš pradžių išnagrinėsime atmetančią MAS.

Svarbiausi atmetančios MAS efektyvumo rodikliai yra šitie parametrai:

1. Absoliutinė sistemos pralaida,
2. Santykinė sistemos pralaida.

- Absoliutinė MAS pralaida – tai vidutinis paraiškų, kurias sistema gali aptarnauti per laiko vienetą, skaičius.
- Santykinė MAS pralaida – tai vidutinis sistemos aptarnaujamų atėjusių paraiškų skaičius. Kitaip tariant, tai vidutinio paraiškų, kurias gali aptarnauti sistema per laiko vienetą, skaičiaus ir vidutinio paraiškų, per tą laiką patekusių į sistemą, santykis.

Praktinėse užduotyse naudojami ir kiti MAS efektyvumo rodikliai, pavyzdžiui, vidutinis užimtų kanalų skaičius, vidutinis santykinis sistemos prastovos laikas, vidutinis santykinis atskiro kanalo prastovos laikas ir t. t.

Aptarsime laukiančią MAS.

Efektyvumo rodikliai, naudojami laukiančią MAS charakterizuoti:

1. Vidutinis paraiškų skaičius eilėje,
2. Vidutinis aptarnautų paraiškų skaičius,
3. Vidutinis paraiškos laukimo laikas eilėje,
4. Vidutinis vienos paraiškos aptarnavimo laikas.

Dėl to, kad laukiančioje MAS kiekviena paraiška galiausiai yra aptarnaujama, tokioms sistemoms absoliutinė pralaida sutampa su paraiškų atvykimo srauto intensyvumu.

Charakterizuojant MAS su apribotu laukimo eilėje laiku, naudojami ir atmetančios MAS efektyvumo rodikliai, ir laukiančios MAS efektyvumo rodikliai.

Nagrinėjant daugiakanales sistemas, be aukščiau išvardintų efektyvumo rodiklių, papildomai naudojami parametrai, aprašantys atskirai kiekvieną iš kanalų.

1.4. Vienkanalės atmetančios MAS efektyvumo rodiklių skaičiavimas

3.4.1 lentelė. Naudojamų terminų ir žymėjimų sąrašas

№	Terminas	Žymėjimas
1	Paraiškų atvykimo srauto intensyvumas	λ
2	Paraiškų aptarnavimo srauto intensyvumas	μ
3	Sistemos efektyvumo koeficientas	α
4	Vidutinis vienos paraiškos aptarnavimo laikas	m_{apt}
5	Santykinė MAS pralaida	q
6	Absoliutinė MAS pralaida	Q

7	Tikimybė, kad paraiška bus aptarnauta	p_{apt}
8	Tikimybė, kad paraiška bus atmesta	p_{atm}

Parametrai λ ir μ žinomi. Reikia surasti m_{apt}, α, q, Q, p_{apt}, p_{atm}.

Skaičiavimo formulės

MAS teorijoje įrodoma, kad vienkanalės atmetančios MAS efektyvumo rodikliai suskaičiuojami pagal 1.4.1 - 1.4.6 formules:

$$m_{apt} = \frac{1}{\mu}, \quad (1.4.1)$$

$$\alpha = \frac{\lambda}{\mu}, \quad (1.4.2)$$

$$q = \frac{1}{\alpha + 1}, \quad (1.4.3)$$

$$Q = \lambda q, \quad (1.4.4)$$

$$p_{apt} = q, \quad (1.4.5)$$

$$p_{atm} = 1 - p_{apt}. \quad (1.4.6)$$

1.5. Daugiakanalės atmetančios MAS efektyvumo rodiklių skaičiavimas

1.5.1 lentelė. Naudojamų terminų ir žymėjimų sąrašas

№	Terminas	Žymėjimas
1	Aptarnavimo kanalų skaičius	*n (n>1)*
2	Paraiškų atvykimo srauto intensyvumas	λ
3	Paraiškų aptarnavimo srauto intensyvumas	μ
4	Sistemos efektyvumo koeficientas	α
5	Tikimybė, kad užimta 0, 1, ..., n kanalų, atitinkamai	$p_0, p_1, \ldots, p_n$
6	Santykinė MAS pralaida	q
7	Absoliutinė MAS pralaida	Q
8	Tikimybė, kad paraiška bus aptarnauta	p_{apt}
9	Tikimybė, kad paraiška bus atmesta	p_{atm}
10	Vidutinis užimtų kanalų skaičius	$\overline{k}$

Parametrai n, λ ir μ žinomi. Reikia surasti α, $p_0, p_1, ..., p_n$, p_{atm}, q, Q, p_{apt}, $\overline{k}$.

Skaičiavimo formulės

Sistemos efektyvumo koeficientas skaičiuojamas pagal formule

$$\alpha = \frac{\lambda}{n \cdot \mu}. \tag{1.5.1}$$

Tikimybės $p_0, p_1, ..., p_n$ skaičiuojamos pagal Erlango formules:

$$\begin{cases} p_0 = \left(\sum_{k=0}^{k=n} \frac{\alpha^k}{k!} \right)^{-1}, \\ p_k = \frac{\alpha^k}{k!} \cdot p_0, \quad k = 1, 2, ..., n. \end{cases} \tag{1.5.2}$$

Paraiška gaus atsisakymą aptarnauti tik tuo atveju, kai visi aptarnavimo kanalai bus užimti, todėl paraiškos atmetimo tikimybė bus skaičiuojama pagal šitą formulę:

$$p_{atm} = p_n = \frac{\alpha^n}{n!} \cdot p_0. \tag{1.5.3}$$

Likę parametrai apskaičiuojami pagal šitas formules:

$$q = 1 - p_{atm} = 1 - \frac{\alpha^n}{n!} \cdot p_0, \tag{1.5.4}$$

$$Q = \lambda q = \lambda \left(1 - \frac{\alpha^n}{n!} \cdot p_0 \right), \tag{1.5.5}$$

$$p_{apt} = 1 - p_{atm} = q, \tag{1.5.6}$$

$$\overline{k} = \frac{Q}{\mu} = \alpha \cdot \left(1 - \frac{\alpha^n}{n!} \cdot p_0 \right). \tag{1.5.7}$$

1.6. Vienkanalės MAS su apribota eile efektyvumo rodiklių skaičiavimas

1.6.1 lentelė. Naudojamų terminų ir žymėjimų sąrašas

№	Terminas	Žymėjimas
1	Eilės ilgis	*m (m>1)*
2	Paraiškų atvykimo srauto intensyvumas	λ

3	Paraiškų aptarnavimo srauto intensyvumas	μ
4	Sistemos efektyvumo koeficientas	α
5	Tikimybė, kad MAS laisva ir gali aptarnauti paraišką	p_0
6	Tikimybė, kad MAS užimta, o eilėje paraiškų nėra	p_1
7	Tikimybė, kad MAS užimta, o eilėje yra 1,2,...,m paraiškų atitinkamai	$p_2,...,p_{m+1}$
8	Santykinė MAS pralaida	q
9	Absoliutinė MAS pralaida	Q
10	Tikimybė, kad paraiška bus aptarnauta	p_{apt}
11	Tikimybė, kad paraiška bus atmesta	p_{atm}
12	Vidutinis paraiškų, stovinčių eilėje, skaičius	$\overline{r}$
13	Vidutinis paraiškų skaičius sistemoje (aptarnaujamų ir stovinčių eilėje)	$\overline{k}$
14	Vidutinis laukimo laikas eilėje	$\overline{t}_{lauk}$
15	Vidutinis paraiškos buvimo MAS laikas	$\overline{t}_{MAS}$

Parametrai m, λ ir μ žinomi. Reikia surasti α, p_0, $p_2,...,p_{m+1}$, q, Q, p_{apt}, p_{atm}, $\overline{r}$, $\overline{k}$, $\overline{t}_{lauk}$, $\overline{t}_{MAS}$.

Skaičiavimo formulės

Sistemos efektyvumas skaičiuojamas pagal tą pačią formulę, kaip ir prieš tai esančiuose skyriuose:

$$\alpha = \frac{\lambda}{\mu}. \tag{1.6.1}$$

Tikimybės $p_0, p_1,..., p_{m+1}$ skaičiuojamos pagal formules:

$$\begin{cases} p_0 = \begin{cases} \dfrac{1-\alpha}{1-\alpha^{m+2}}, & \alpha \neq 1, \\ \dfrac{1}{m+2}, & \alpha = 1, \end{cases} \\ p_k = \alpha^k \cdot p_0, \quad k = 1,2,...,m+1. \end{cases} \tag{1.6.2}$$

Dėl to, kad paraiška gauna atsisakymą aptarnauti, jeigu MAS užimta, o eilėje yra m paraiškų, gauname, kad:

$$p_{atm} = p_{m+1}, \tag{1.6.3}$$

Toliau gauname:

$$q = p_{apt} = 1 - p_{atm} = 1 - p_{m+1}, \tag{1.6.4}$$

$$Q = \lambda q\ . \tag{1.6.5}$$

Likusiu parametrų formulės:

$$\overline{r} = \begin{cases} \dfrac{\alpha^2\left[1-\alpha^m\left(m+1-m\alpha\right)\right]}{\left(1-\alpha^{m+2}\right)\left(1-\alpha\right)}, & \alpha \neq 1, \\ \dfrac{m\left(m+1\right)}{2\left(m+2\right)}, & \alpha = 1. \end{cases}, \tag{1.6.6}$$

$$\overline{k} = \overline{r} + 1 - p_0, \tag{1.6.7}$$

$$\overline{t}_{lauk} = \frac{\overline{r}}{\lambda}, \tag{1.6.8}$$

$$\overline{t}_{MAS} = \frac{\overline{r}}{\lambda} + \frac{q}{\mu}\ . \tag{1.6.9}$$

1.7. Vienkanalės MAS su begaline eile efektyvumo rodiklių skaičiavimas

1.7.1 lentelė. Naudojamų terminų ir žymėjimų sąrašas

№	Terminas	Žymėjimas
1	Eilės ilgis	∞
2	Paraiškų atvykimo srauto intensyvumas	λ
3	Paraiškų aptarnavimo srauto intensyvumas	μ
4	Sistemos efektyvumo koeficientas	α
5	Tikimybė, kad MAS laisva ir gali aptarnauti paraišką	p_0
6	Tikimybė, kad MAS užimta, o eilėje paraiškų nėra	p_1
7	Tikimybė, kad MAS užimta, o eilėje yra 1,2,...,m paraiškų atitinkamai	$p_2, \ldots, p_{m+1}$
8	Santykinė MAS pralaida	q
9	Absoliutinė MAS pralaida	Q
10	Tikimybė, kad paraiška bus aptarnauta	p_{apt}

11	Tikimybė, kad paraiška bus atmesta	p_{atm}
12	Vidutinis paraiškų, stovinčių eilėje, skaičius	$\overline{r}$
13	Vidutinis paraiškų skaičius sistemoje (aptarnaujamų ir stovinčių eilėje)	$\overline{k}$
14	Vidutinis laukimo laikas eilėje	$\overline{t}_{lauk}$
15	Vidutinis paraiškos buvimo MAS laikas	$\overline{t}_{MAS}$

Parametrai λ ir μ žinomi. Reikia surasti α, p_0, $p_2,...,p_{m+1}$, q, Q, p_{apt}, p_{atm}, $\overline{r}$, $\overline{k}$, $\overline{t}_{lauk}$, $\overline{t}_{MAS}$.

Skaičiavimo formulės

Sistemos efektyvumas skaičiuojamas pagal tą pačią formulę, kaip ir prieš tai esančiuose skyriuose:

$$\alpha = \frac{\lambda}{\mu} < 1. \tag{1.7.1}$$

Jeigu formulėse (3.6.2) - (3.6.9) suskaičiuoti ribą, esant $m \to \infty$, gausime tokias formules:

$$\begin{cases} p_0 = 1-\alpha \\ p_k = \alpha^k \cdot p_0, \quad k = 1,2,... \end{cases} \tag{1.7.2}$$

Esant begalinei eilei, galų gale kiekviena paraiška bus aptarnauta. Todėl:

$$p_{atm} = 0, \tag{1.7.3}$$

$$q = 1 = p_{apt}, \tag{1.7.4}$$

$$A = \lambda. \tag{1.7.5}$$

Likę parametrai suskaičiuojami pagal formules:

$$\overline{r} = \frac{\alpha^2}{1-\alpha}, \tag{1.7.6}$$

$$\overline{k} = \frac{\alpha}{1-\alpha}, \tag{1.7.7}$$

$$\overline{t}_{lauk} = \frac{\alpha^2}{\lambda(1-\alpha)}, \tag{1.7.8}$$

$$\overline{t}_{MAS} = \frac{1}{\mu(1-\alpha)}. \tag{1.7.9}$$

2. MAS modeliavimo įrankių parinkimas ir pagrindimas

Prieš parenkant konkretų modeliavimo įrankį darbo tikslams įgyvendinti, verta apžvelgti keletą jų, siekiant nuspręsti kuris yra tinkamiausias. MAS modeliavimo įrankių rinkoje egzistuoja mokami ir nemokami sprendimai. Keletas iš jų ir jų charakteristikos trumpai aptariami šiame skyriuje.

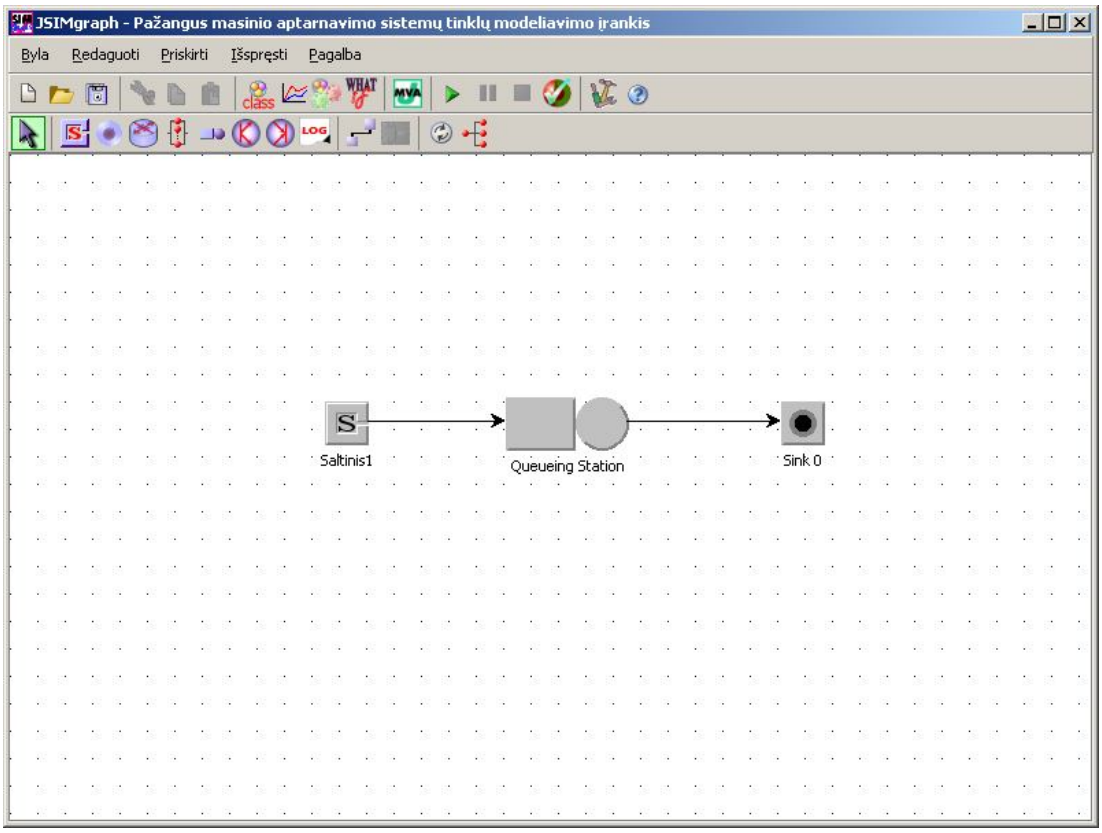
2.1. Modeliavimo įrankis *JAVA Modelling Tools (JMT)*

JMT modeliavimo įrankis [5] – tai atviro kodo sprendimas, sudarytas iš šešių įrankių, skirtų kompiuterių ir komunikacijų sistemų įvairių charakteristikų (naudingumo, talpos planavimo, apkrovos ir t. t.) įvertinimui. Šiame įrankyje naudojami keli realaus laiko modeliavimo algoritmai masinio aptarnavimo sistemų tinklus modeliuoti, taikant asimptotinę ir simuliacinę analizę. Tiriamas modelis gali būti sukurtas, naudojantis specialiu vedliu arba grafine sąsaja. Gamintojai teigia, jog dėl to kad programoje yra panaudotas XML duomenų sluoksnis [6], tai duoda optimaliai išnaudoti kompiuterio skaičiavimo resursus, darant simuliacijas. Toliau trumpai aptarsime kiekvieno iš šešių JMT programos įrankių funkcijas.

JSIMwiz: vedlio tipo sąsajos įrankis, skirtas JSIM stimuliatoriui, kuris analizuoja MAS modelius. Reikalingi sistemos parametrai nustatomi atitinkamuose vedlio languose. JSIM simuliatoriaus branduolys leidžia pasirinkti keletą galimų tikimybinių pasiskirstymų atvykimo ir aptarnavimo srautams nustatyti. JSIMwiz palaiko nuo būsenos nepriklausančias paraiškų maršrutizavimo strategijas (pvz. Markovo, ar cikliško), tuo pačiu palaikant ir nuo būsenos priklausančias maršrutizavimo strategijas (pvz. nukreipti paraiškas į mažiausiai apkrautą serverį, arba į serverį su trumpiausiu atsako laiku ar eilės ilgiu). Modeliavimo branduolys palaiko keletą ypatingų funkcijų, tokių kaip: sritys su apribota talpa, serverių lygiagretus sujungimas, prioritetų priskirimas paraiškoms. JSIM automatiškai suskaičiuoja simuliacijos laiką, realiuoju laiku braižo grafikus iš gautų verčių, įvertinus galimas paklaidas. Taip pat palaikoma galimybė vykdyti keletą simuliacijų iš eilės, kiekvieną kartą keičiant norimus parametrus, kad stebėti atitinkamus pokyčius sistemoje.

JSIMgraph: grafinė, vartotojams intuityviai suprantama sąsaja, skirta tam pačiam JSIM branduoliui, kaip ir JSIMwiz. Šis įrankis pasižymi tokiu pat funkcionalumu, kaip ir JSIMwiz. Toks būdas leidžia lengvai aprašyti modeliuojamo tinklo struktūrą, taip pat palengvina

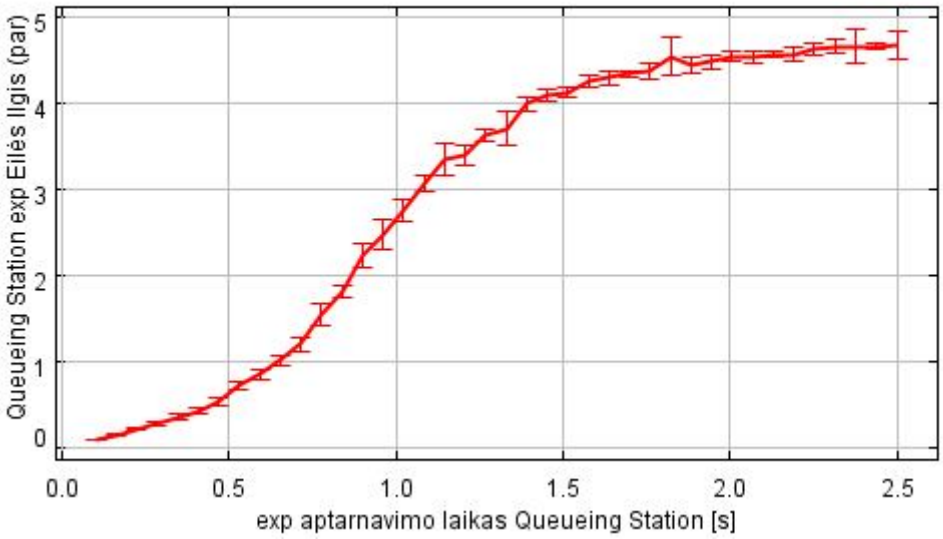
modeliuojamos sistemos įvesties ir vykdymo parametrų apibrėžimą (2.1.1 pav.). Sukurto tinklo topologija gali būti eksportuojama į išorinį grafinį formatą paveikslo pavidalu.



2.1.1 pav. JSIMgraph įrankio darbinis langas

JMVA: naudojamas vienos arba keleto klasių MAS tinklų analizėms, taikant atvirą, uždarą arba mišriąją apkrovą. Naudojamas specialus stabilizuotas Vidutinės Vertės Analizės algoritmas. MAS tinklo struktūra nustatoma, naudojantis grafiniu vedliu. Taip pat, kaip ir JSIM atveju, yra galimybė vykdyti simuliacijų serijas, keičiant sistemos parametrus kiekvienai simuliacijai. Todėl gautus duomenis taip pat galima atvaizduoti grafiškai.

JMCH: įrankis specialiai sukurtas vieno kanalo Markovo grandinėms modeliuoti. Jis naudojamas sistemoms su apribota (M/M/1/k) arba begaline (M/M/1) eile. Yra galimybė dinamiškai keisti vidutinius paraiškų atvykimo arba aptarnavimo laikus, simuliacijos metu.



2.1.2 pav. JSIM įrankio nubraižyto rezultatų grafiko pavyzdys

JABA: įrankis skirtas lėčiausius mazgus keleto klasių sistemose identifikuoti. Tai reiškia, kad įrankis leidžia surasti tas sistemos vietas, kur sistemos produktyvumas nukenčia daugiausiai. Naudojamas išgaubto paviršiaus (angl. *convex hull*) algoritmas. Galima naudoti iki trijų skirtingų klasių paraiškas. Tokiu būdu tampa įmanoma ieškoti sistemos potencialiai silpnas vietas, priklausomai nuo skirtingų paraiškų klasių mišinių, vykdant simuliaciją. Įrankis palaiko modelius su tūkstančiais paraiškų eilėje. Galima studijuoti optimizavimo galimybes (sistemos pralaidos padidinimas, atsako laiko sumažinimas, optimalios apkrovos nustatymas), identifikuojant perkrautus sistemos mazgus, esant tam tikriems paraiškų klasių mišiniams.

JWAT: įrankis skirtas apkrovų charakteristikoms nustatyti. Duomenų įvesties formatas gali būti nustatytas pagal norą, tačiau yra palaikomi kai kurie standartiniai formatai (pvz. „Apache HTTP“, „IIS log files“). Įvesties duomenys gali būti aprašyti, naudojant statistinius įrankius (vidurkis, koreliacija, histogramos, sklaidos), taip pat duomenys gali būti daugiamačių masyvu pavidalu. Specialūs algoritmai leidžia identifikuoti panašių charakteristikų vartotojų grupes. Šis įrankis gali būti naudojamas apkrovos parametrizavimui.

2.1.1 pav. pavaizduotas JSIMgraph įrankio darbinis langas. Šiame pavyzdyje yra modeliuojama vieno kanalo MAS. Eilės ilgis gali būti apribotas arba begalinis. Šie nustatymai parenkami per modelio objektų parametrų redagavimo pasirinkimą. „Šaltinis“ modelyje – tai paraiškų generavimo objektas. „Queueing station“ – aptarnavimo stotis su atitinkamu eilės ilgiu. „Sink0“ – paraiškų terminavimo objektas. Kiekviena sugeneruota paraiška turi užbaigti savo

kelią vienoje iš terminavimo stočių. Panašiu būdu galima kurti ir analizuoti įvairesnes ir sudėtingesnes MAS.

2.1.2 pav. pavaizduotas vienos iš JSIM branduolio padarytų simuliacijų rezultatas. Šiuo atveju buvo tiriama vidutinio paraiškų eilės ilgio priklausomybė nuo sistemos efektyvumo koeficiento. Vienkanalės sistemos efektyvumo koeficientas [1] – tai paraiškų atvykimo ir aptarnavimo srautų intensyvumų santykis, išreiškiamas procentais arba santykiniais vienetais:

$$\alpha = \frac{\lambda}{\mu} = \frac{m_\mu}{m_\lambda} \qquad (2.1.1)$$

Iš gauto grafiko galima matyti, kad didėjant sistemos efektyvumui – didėja ir vidutinis paraiškų skaičius eilėje. Panašiu būdu galima tirti ir kitų charakteristikų priklausomybes nuo efektyvumo koeficiento.

JMT modeliavimo įrankis yra gana galingas instrumentas, tačiau jis turi savo apribojimų ir silpnųjų pusių.

1. Įrankis sukurtas JAVA programavimo platformoje, kuri yra žymiai lėtesnė, paliginus su kitomis programavimo kalbomis (Basic, C).
2. Galima tirti tik tas charakteristikas, kurias leidžia programos nustatymai. Norint tirti kažką kitą (pvz. naują tikimybinį pasiskirstymo dėsnį, arba kokį nors subtilų parametrą, tokį kaip sistemą palikusių dėl laukimo laiko eilėje apribojimo paraiškų intensyvumą) reikia ieškoti kito programinio sprendimo, arba prašyti gamintojo modifikuoti programą.
3. Įrankis yra universalus, todėl užima daug vietos atmintyje. Jeigu jis naudojamas tik kokiam nors vienam konkrečiam tikslui – visos kitos programos galimybės nereikalingos, kas daro programos naudojimą mažiau efektyviu, nes negalima išskirti tik tam tikrą dalį iš programos.

2.2. Modeliavimo įrankis *SimEvents*

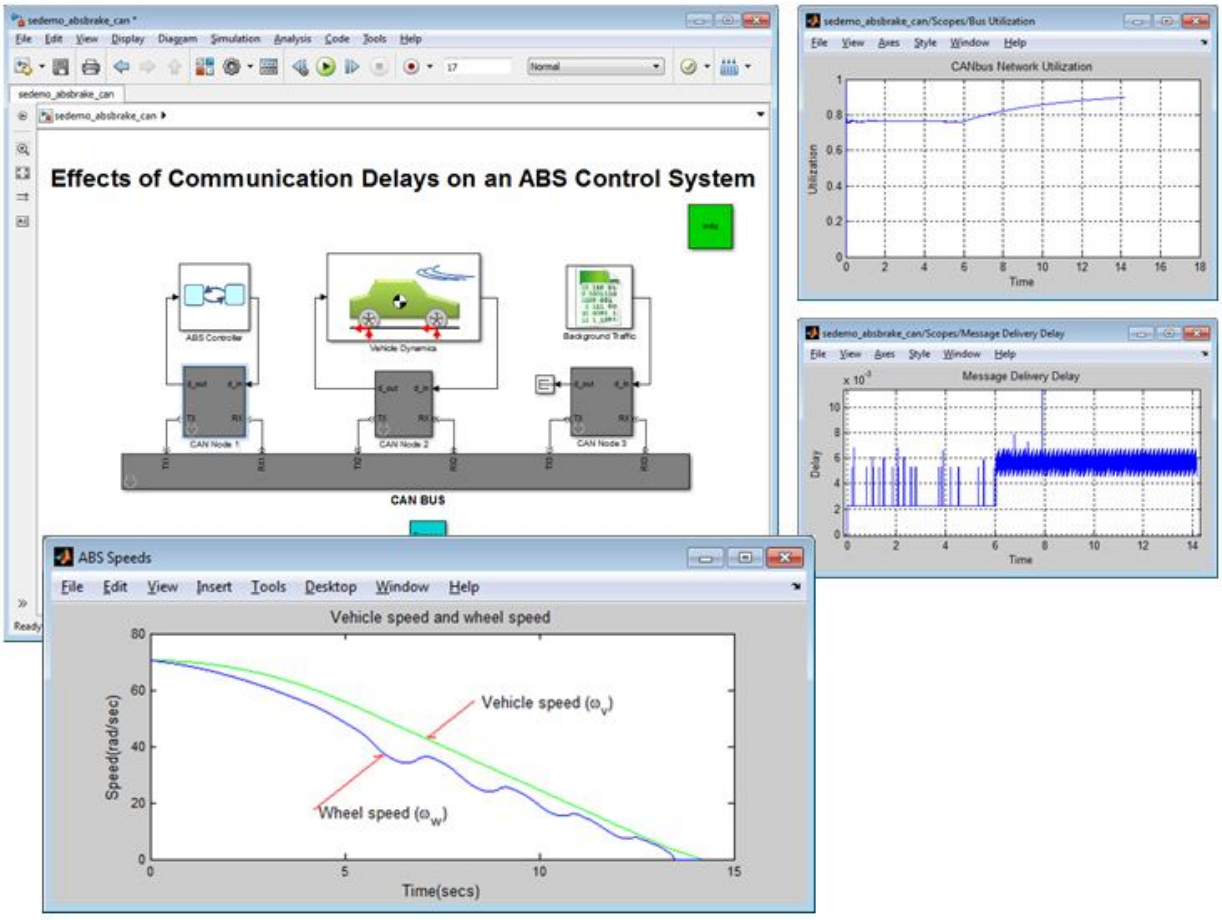
SimEvents [7] – tai „MathWorks“ kompanijos sukurtas diskrečiųjų įvykių simuliatorius, veikiantis *MATLAB* programos platformoje, naudojant *Simulink* paketą [8]. *SimEvents* yra mokamas įrankis, tačiau yra galimybė parsisiųsti ir išbandyti bandomąją versiją, veikiančią 30 dienų. Bandomąją versiją gali išbandyti tik organizacijos, studentams jinai nėra prieinama.

Pagrindinės *SimEvents* programinio įrankio charakteristikos:

- Diskrečiųjų įvykių modeliavimo branduolys, naudojamas sudėtingų sistemų simuliacijoms *Simulink* programiniame pakete.
- Bibliotekos su aprašytais objektais (kuriuos galima naudoti MAS modeliavimui), tokiais kaip eilės, aptarnavimo stotys, paraiškų generatoriai, maršrutizavimo mazgai ir kiti.
- Galimybė individualiai nustatyti duomenų parametrus, kad būtų lengva ir patogų atvaizduoti užduočių rezultatus, paketus ar jų dalis.
- Vidinis statistinis modulis, leidžiantis lengvai sužinoti tokius sistemos parametrus, kaip vidutinis eilės ilgis, pralaida ir kitus.
- Speciali biblioteka, aprašanti specialiuosius modelio mazgus, tokius kaip komunikacijos kanalai, perdavimo sąsajos ir konvejerinės struktūros.
- Modeliavimo animacija ir vizualizacija.

Subjektai ir įvykiai. Naudojant *SimEvents* įrankį, galima sukurti subjektus, kurie atitiks mus dominančius subjektus realybėje, pvz. paketai kompiuteriniuose telekomunikacijų tinkluose arba lėktuvai oro uoste. Subjektų generavimas, judėjimas tarp blokų ir apdorojimas vykdomas per įvykius, tokius kaip paketų atvykimas sistemoje, arba lėktuvų išvykimas oro uoste. Šitie įvykiai modifikuoja sistemos būseną, kas savo ruožtu pakeičia sistemos elgseną. Modeliavimo subjektams galima priskirti tam tikrus parametrus, tokius kaip paskirties punkto adresas, apdorojimo laikas, serverio užlaikymas. *SimEvents* kiekvieną subjektą (paraiška) apdoroja individualiai, kas duoda galimybę kurti atskirus skirtingų paraiškų maršrutus tarp modelio blokų ir vykdyti atitinkamus skaičiavimus.

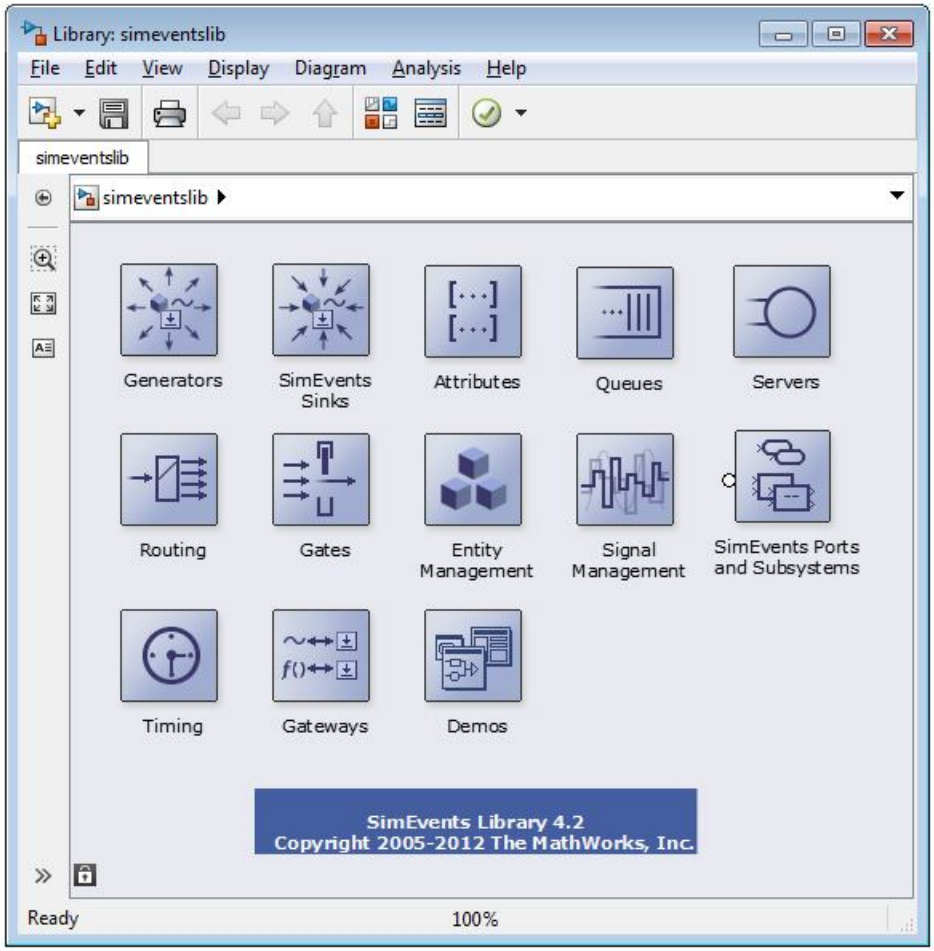
***SimEvents* įrankio integravimas į *Simulink* ir *Stateflow* programas.** *SimEvents* duoda galimybę įvykių modeliavimo simuliaciją integruoti į *Simulink* programos laikinius modelius, naudojant blokus iš šliuzų (angl. *gateway*) bibliotekos. „Laikas-įvykis“ signalų blokas paverčia laikinius signalus į įvykių signalus; ir atvirkščiai, „Įvykis-laikas“ signalų blokas paverčia įvykius atgal į laikinius signalus. Naudojant tokį dvikryptį konvertavimą, galima modeliuoti sudėtingas sistemas, kurios laikiniai komponentai galėtų komunikuoti tarpusavyje. 2.2.1 pav. parodytas *SimEvents* programoje sukurto modelio pavyzdys. Pavyzdyje esanti sistema modeliuoja automobilio stabdžių antiblokavimo sistemą (ABS – angl. *Anti-lock braking system*), siekiant sužinoti, kaip CAN duomenų sąsajos informacijos srautų intensyvumas paveikia automobilio charakteristikas (automobilio, ratų greičius).



2.2.1 pav. *SimEvents* įrankio veikimo pavyzdys

Galima tiesiogiai apdoroti įvykių signalus, naudojant *Simulink* ir *StateFlow* bibliotekų blokus, įskaitant matematines operacijas, *MATLAB* funkcijas, *StateFlow* grafikus ir loginius operatorius. Taip pat bet kuriuos *Simulink* skaičiavimus, kurie turi būti daromi įvykių signalų pagrindu, galima inkapsuliuoti tolimesniems veiksmams, naudojant „Atomic Subsystem“ bloką.

Objektai ir jų bibliotekos. *SimEvents* įrankis pateikia galimybę kurti, apdoroti, saugoti ir kilnoti subjektus sistemoje. Galima modeliuoti ir paprastas, ir sudėtingas serverių-eilių tinklo sistemas. Sujungiant blokus (objektus) tarpusavyje, galima nustatyti tam tikrus kelius, kuriais subjektai keliaus, reaguojant į atitinkamus įvykius. Šitie keliai gali turėti užlaikymus ir specialias maršrutizavimo taisykles. Dauguma parametrų *SimEvents* programoje gali būti statistiškai apibrėžti, modeliuojant tikimybinius pokyčius sistemoje. Taipogi įmanoma kurti savo blokus, naudojantis *Simulink* galimybėmis, tokiomis kaip posistemių maskavimas, bibliotekų kūrimas, specialiųjų SimEvents įrankio prievadų prijungimas prie *Simulink* programos posistemių, kas leidžia kurti ir vartoti įvykius.



2.2.2 pav. SimEvents modeliavimo įrankio objektų biblioteka

2.2.2 pav. pavaizduota SimEvents programos objektų biblioteka. Keleto iš jų trumpi aprašymai:

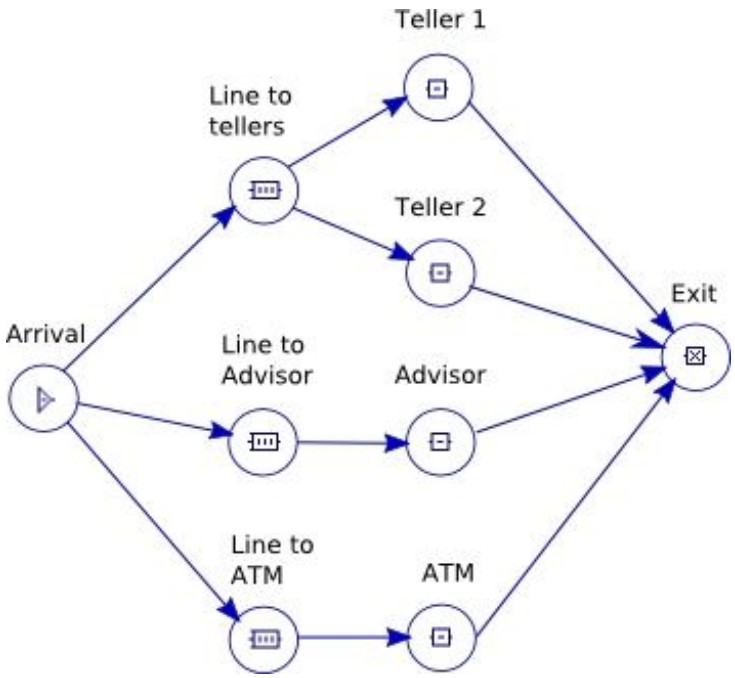
- Generators – signalų (subjektų, paraiškų) generatorių blokas.
- Entity Management – paraiškų valdymo blokas, leidžiantis, esant reikalui, sugrupuoti arba išgrupuoti atitinkamų paraiškų grupes.
- Attributes – paraiškų charakteristikų blokas, per kurį paraiškoms priskiriamos tam tikros savybės.
- Queues – eilių blokas. Valdo paraiškų buvimo eilėje ypatumus (pvz. priskiriant FIFO taisyklę).
- Servers – paraiškų apdorojimo stočių blokas.

Apibendrinant, reikia pasakyti, kad SimEvents modeliavimo įrankis yra labai efektyvus ir galingas. Jis naudoja stiprias ir praktiškai įrodžiusias savo efektyvumą bei pranašumą programines platformas, tokias kaip *MATLAB* ir *Simulink*. Tačiau pagrindinis trūkumas yra programos kaina. Pilno funkcionalumo versija per daug kainuoja, kad pirkti vienam fiziniam asmeniui, o bandomoji versija prieinama tik organizacijų darbuotojams, nes gamintojai nenori kad šis įrankis būtų naudojamas mokymo tikslams. SimEvents labiau orientuotas komerciniams tikslams.

2.3. Programinė modeliavimo biblioteka *Delsi 2.0*

Delsi 2.0 įrankis [9] – tai diskrečiųjų įvykių simuliacijos sistema, sukurta sudėtingos struktūros MAS modeliams simuliuoti. Įrankis realizuotas, kaip komponentų rinkinys *Microsoft .NET 4.0 Framework* platformoje [10].

Pagrindinė *Delsi* modeliavimo idėja – tai, kad masinio aptarnavimo sistema gali būti atvaizduota, kaip schema su mazgais, kiekvienas iš kurių atitinka tam tikrą modelio komponentą (tokį kaip generatorių, eilę, aptarnavimo stotį ir t. t.). Rodyklės, jungiančios modelio mazgus, parodo apdorojamų elementų srautus sistemoje. Šitie elementai vadinami transakcijomis. 2.3.1 paveiksle parodyta *Delsi* įrankiu sukurta MAS modelio schema.



2.3.1 pav. *Delsi* modelio schema

Kad suprastum pateiktą schemą, įsivaizduokime banko padalinį. Į padalinį nuolat atvyksta klientai. Kai kurie iš jų nori būti aptarnauti konsultanto, kai kurie finansinio patarėjo,

kiti ateina pasinaudoti bankomato paslaugomis. Tam, kad gauti norimą paslaugą, klientai gali laukti atitinkamose eilėse. Po aptarnavimo, klientai išeina iš banko padalinio. Turint tokį pavyzdį, galima grafiškai sukurti sistemos modelį, kuris turėtų mazgus, atitinkančius aptarnavimo elementus (blokus) [11]:

- Generatorius „Arrival“.
- Eilės „Line to tellers“, „Line to Advisor“, „Line to ATM“.
- Serveriai „Teller1“, „Teller2“, „Advisor“, „ATM“.
- Išėjimas „Exit“.

Šitame pavyzdyje aptarnaujami elementai (transakcijos) bus klientai. Transakcijos generuojamos generatoriuje, po to eina iš vieno bloko į kitą ir galiausiai šalinamos išėjime.

Naudojantis *Delsi,* galima kontroliuoti objektų (blokų) veikimą ir transakcijų srautus, naudojant specializuotus algoritmus. Realizacijos požiūriu, *Delsi 2.0* – tai simuliacijos komponentų ir objektų biblioteka, naudojama *Microsoft .NET Framework 4.0* aplinkoje.

Simuliacijos metu transakcijos keliauja nuo vienų blokų į kitus. Specializuoti simuliavimo algoritmai turi būti realizuoti kaip įvykių stebėtojai, tam tikriems įvykiams registruoti. Įvykiai gali būti tokie: įėjimas į bloką, maršrutizavimas, išėjimas iš bloko, aptarnavimo pertrauktis ir kiti. Naudojant metodus, savybes ir įvykius komponentuose, galima kontroliuoti modeliavimo procesą ir gauti reikiamus statistinius duomenis.

Delsi duoda vartotojams galimybę naudoti pilną *Microsoft .NET Framework* platformos galingumą ir jos kūrimo aplinką, tokią kaip *Microsoft Visual Studio 2010* [12]. Tai įgalina vartotojus kurti platų ruožą įvairiausių simuliacijų modelių. Kartu su *Microsoft Visual Studio 2010 įrankis Delsi 2.0* formuoja prieinamą platformą plačiam ruožui galingų ir efektyvių modeliavimo programinių įrangų kurti.

Delsi įrankis gali būti pritaikytas įvairiausioms simuliacijoms vykdyti, jas taikant skirtingose srityse:

- Telekomunikacijos.
- Gamyba.
- Kompiuterių tinklai.
- Telefoninės stotys.
- Logistika.
- Sveikatos priežiūra.
- Verslo procesai.

Delsi įrankio privalumai:

1. *Delsi* duoda galimybę naudotis nebrangia, tačiau labai galinga modeliavimo platforma. Minimalūs reikalavimai – paprastas personalinis kompiuteris ir *Visual Studio 2010 Express Edition* programa. Demonstracinės versijos veikimas neapribotas laike, o apribojimai labai lojalūs (galima kurti savo modelius, net nepastebint apribojimų).
2. Visa sudėtinga modelio logika, kaip reakcija į simuliacijos metu vykstančius įvykius, gali būti parašyta bendra programavimo kalba, tokia kaip C# [13].
3. Simuliacijos algoritmai sukurti tokiu būdu, kad simuliacijos produktyvumas nepriklausytų nuo transakcijų skaičiaus sistemoje. *Delsi* gali palaikyti modelius, kuriuose yra milijonai transakcijų vienu metu.
4. Priešingai kai kurioms kitoms modeliavimo programoms, laikas *Delsi* įrankyje yra *double* tipo, todėl laiko intervalai tarp simuliacijos įvykių atitinka tikimybinius pasiskirstymus su dideliu tikslumu.
5. Simuliacijos metu *Delsi* bibliotekos blokai renka gausius statistinius duomenis.
6. Transakcijos gali turėti prioritetus, būti pertraukiamos arba nepertraukiamos.
7. *Delsi* turi atsitiktinių skaičių generatorius su įvairiais tikimybiniais pasiskirstymais: Beta, Gamma, Erlang, Uniform, Exponential, Lognormal, Normal, Triangular, Weibull.
8. Kuriant simuliacijos programinę įrangą su *Delsi* įrankio pagalba, galima naudoti šimtus trečios šalies komponentus, parašytus .NET 4.0 platformai.
9. Naudojant *Delsi*, simuliacijos programinis kodas gali būti integruotas į skirtingas platformas: Windows desktop, Web, Web Services, Console, API.

3. MAS modeliavimo programinė įranga

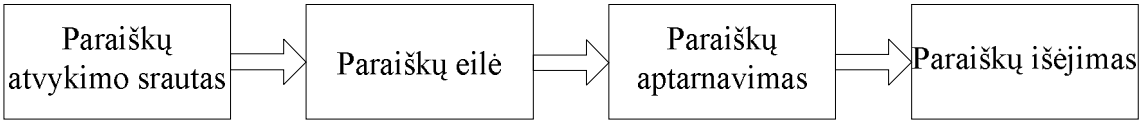
Išnagrinėjus šiuo metu rinkoje egzistuojančius programinius sprendimus, buvo nuspręsta darbo tikslams naudoti *Delsi 2.0* modeliavimo įrankį. Toks pasirinkimas nulemtas keleto faktų. Visų prima reikia pasakyti, kad *Delsi* yra labai galingas ir efektyvus įrankis, skirtas plačiam ruožui įvairių modeliavimo uždavinių spręsti. Kita vertus, *Delsi* bandomosios versijos apribojimai nėra labai griežti, ir net bandomosios versijos pilnai pakaktų esamo darbo tikslams įgyvendinti. Naudojant *Delsi*, simuliacijų modelius patogu kurti populiaria programavimo kalba C#, naudojant programavimo platformą *Microsoft Visual Studio 2010 Express Edition*. Naudojant šią platformą, galima kurti ir lanksčiai modifikuoti modeliavimo programas, skirtas įvairiausiems tikslams. Programos, sukurtos šioje platformoje, veikia operacinės sistemos branduolyje, kas duoda pranašumą skaičiavimų spartos atžvilgiu prieš programas, veikiančias virtualiuose branduoliuose (pvz. Java).

3.1. Reikalavimai kuriamai programinei įrangai

Vienas iš esamo darbo tikslų – sukurti patogią ir efektyvią programinę įranga, kuria būtų galima vykdyti skirtingų charakteristikų masinio aptarnavimo sistemų modeliavimus. Tai yra labai svarbu, nes turint patogią ir efektyvią programinę įrangą, galima nesikoncentruoti ties rezultatų gavimo procesu, o labiau gilintis į gautų rezultatų esmę, juos analizuoti ir daryti atitinkamas išvadas.

Kaip jau buvo minėta anksčiau, esamo darbo tikslams įgyvendinti buvo parinktas *Delsi 2.0* modeliavimo įrankis. Tačiau pati *Delsi* programa tai nėra jau paruoštas sprendimas konkretiems uždaviniams spręsti – tai yra tik tam tikrų modeliavimo objektų ir komponentų rinkinys (biblioteka), kurį kiekvienas vartotojas gali naudoti savaip, pagal turimus uždavinius. Galima sakyti, kad *Delsi* yra branduolys, programos šerdis, o pačią programą vartotojas turi sukurti pats, atsižvelgiant į savo tikslus ir esamus uždavinius. Būtent toks požiūris daro *Delsi* įrankį labai lankstų, kurį galima taikyti pagal įvairius konkrečius tikslus ir poreikius.

Esamo darbo tikslams pasiekti buvo nuspręsta sukurti programinę įranga, kurios simuliacijų branduolys turėtų galimybę modeliuoti masinio aptarnavimo sistemas pagal 3.1 pav. pateiktą schemą:



3.1 pav. Supaprastintą kuriamos modeliavimo įrangos branduolio veikimo schema

Kiekvienas iš 3.1 pav. nupaišytų blokų privalo turėti tam tikras savybes ir charakteristikas.

Paraiškų atvykimo srautas. Kuriamoje programoje turi būti galimybė naudoti skirtingus atvykimo srauto tikimybinio pasiskirstymo dėsnius. Kiekvienam iš galimų pasiskirstymų turi egzistuoti galimybė keisti jo parametrus pagal savo nuožiūrą (vidurkis, vidutinis kvadratinis nuokrypis ir t.t). Jeigu simuliacijos metu atvykimo srauto parametrai turi keistis – reikia kad programa leistų užduoti parametrų verčių keitimo ribas.

Paraiškų eilė. Paraiškų eilė, arba buferis, gali būti begalinė, baigtinė, arba jos visiškai gali nebūti (atmetanti MAS, be eilės). Laikas, kurį paraiškos gali laukti eilėje, irgi turi būti užduodamas skirtingais tikimybiniais pasiskirstymais, kaip ir atvykimo srautas. Paraiškos gali laukti eilėje be galo ilgai, arba palikti eilę po tam tikro laiko, užduoto atitinkamu tikimybiniu pasiskirstymu. Taip pat turi būti galimybė nustatyti nulinį laukimo laiką, t.y. kad paraiškos, esant užimtai aptarnavimo stočiai, būtų iš karto atmetamos. Atmestos ir po laukimo laiko pabaigos eilę palikusios paraiškos turi būti atskirai registruojamos tolimesniam duomenų apdorojimui.

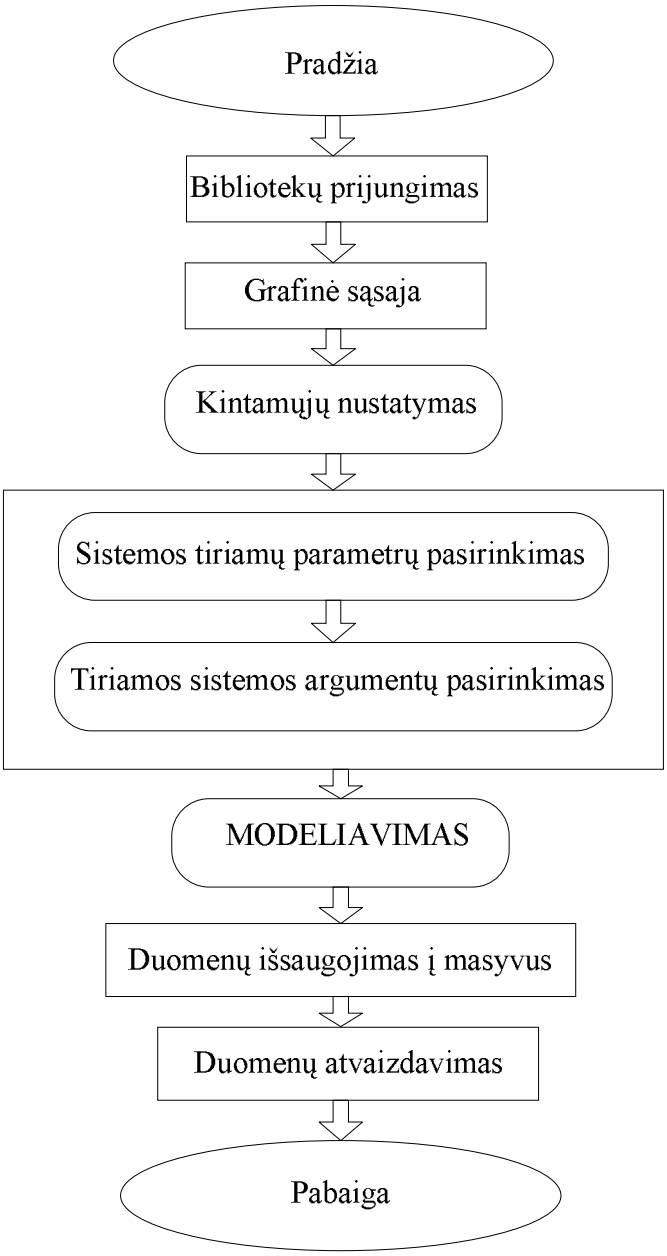
Paraiškų aptarnavimas. Paraiškų aptarnavimo stotis (serveris) gali būti vieno, arba daugelio kanalų. Laikas, kurį aptarnavimo stotis panaudoja paraiškų aptarnavimui, turi būti užduodamas skirtingais pasiskirstymo dėsniais (kaip atvykimo srauto, ir laukimo eilėje). Jeigu visi kanalai yra užimti, aptarnavimo stotis turi atmetinėti paraiškas, kurios palieka eilę. Aptarnautos paraiškos po aptarnavimo stoties nukreipiamos į išėjimą.

Paraiškų išėjimas. Visos paraiškos (atmestos, palikusios eilę, aptarnautos) galų gale eina į išėjimą. Ten jos turi būti registruojamos (išsaugojami duomenis apie jas) ir naikinamos.

Simuliacijos ilgis gali būti įvairus. Programoje turi būti galimybė nustatyti norimą simuliacijos ilgį. Akivaizdu, kad kuo ilgesnė simuliacija, tuo mažesnės galimos gautų rezultatų paklaidos, kai skaičiuojamos tam tikrų dydžių vidutinės vertės. Simuliacijų rezultatų duomenys turi būti pateikiami lentelės pavidalų, iš kur juos būtų galima eksportuoti į kitas platformas tolimesniam apdorojimui. Taip pat pageidautina, kad pačioje simuliacijų programoje būtų galimybė braižyti gautų rezultatų grafikus, kad rezultatai būtų aiškiau suprantami iš karto juos gavus.

3.2. MAS modeliavimo programinės įrangos struktūra

Atsižvelgiant į aukščiau iškeltus reikalavimus modeliavimo programai ir į darbo tikslus (ž. 1 skyrių), esamo darbo tikslams pasiekti buvo sukurta modeliavimo programa. Programos šerdis – *Delsi 2.0* simuliacijų objektų biblioteka. Kitaip tariant, būtent *Delsi* branduolys atlieka visas funkcijas, susijusias su statistinių duomenų apie paraiškas rinkimu, apdorojimu ir išsaugojimu. Visa kita programos logika (duomenų atvaizdavimas, parametrų nustatymas, grafikų braižymas, apipavidalinimas) parašyta C# programavimo kalba, naudojant *Microsoft Visual Studio 2010* programavimo platformą ir bendras funkcijų bibliotekas. Sukurtos modeliavimo programos struktūrinė schema parodyta 3.2 paveiksle.



3.2 pav. Modeliavimo programinės įrangos struktūrinė schema

Trumpai aptarsime kiekvieną programos struktūrinės schemos elementą.

Bibliotekų prijungimas. Kaip jau buvo minėta, *Microsoft Visual Studio 2010* tai galinga platforma, naudojant kurią galima kurti įvairiausias programines įrangas. Todėl egzistuoja labai daug bibliotekų, ir vidinių, ir trečiųjų šalių, kurias galima integruoti į kuriamos programos kodą, tam kad būtų galima naudotis tam tikromis funkcijomis, kurios gali būti naudingos. Šitoje programoje įtrauktos keletas vidinių sisteminių bibliotekų (pvz. grafikų braižymo biblioteka) ir dvi išorinės – *Delsi* modeliavimo įrankio biblioteka „Delsi.dll“ ir atsitiktinių skaičių generavimo biblioteka „Troschuetz.Random.dll“ [14].

Grafinė sąsaja. Vienas iš *Visual Studio 2010* platformos privalumų ir išskirtinių bruožų – galimybė patogiai kurti grafinę programos sąsaja. Programos langai, mygtukai, teksto laukai ir kiti objektai išdėliojami, naudojant specialų konstruktorių. Programos kode kiekvienam grafiniam objektui galima priskirti atitinkamus parametrus, pririšti juos prie tam tikrų kintamųjų ir patogiai naudoti bet kurioje programos kodo vietoje.

Kintamųjų nustatymas. Akivaizdu, kad kiekvienos programos pagrindiniai objektai yra kintamieji. Visu pirma aprašomi kintamieji, į kuriuos įnešami į programos langus įvesti duomenys. Tarpiniai kintamieji reikalingi skaičiavimams atlikti ir duomenims apdoroti. Galiausiai, per kintamuosius aprašomi masyvai, į kuriuos keliami modeliavimo metų gauti rezultatai tolimesniam apdorojimui arba atvaizdavimui.

Sistemos tiriamų parametrų pasirinkimas. Modeliavimo programa buvo parašyta tokiu būdu, kad būtų galima tyrinėti įvairius modeliuojamos masinio aptarnavimo sistemos parametrus. Esamoje programoje jų yra septyni:

- Vidutinis paraiškų laukimo laikas eilėje.
- Paraiškų aptarnavimo tikimybė (arba aptarnautų paraiškų procentas).
- Paraiškų atmetimo tikimybė (arba atmestų paraiškų procentas).
- Eilės palikimo tikimybė (arba eilę palikusių paraiškų procentas po laukimo laiko pabaigos).
- Paraiškų gyvavimo sistemoje laikas.
- Vidutinis eilės ilgis.
- Sistemos prastovos tikimybė.

Tiriamos sistemos argumentų pasirinkimas. Argumentas – tai yra tam tikras kintamasis, kuriam kintant simuliacijos metu, stebimi tiriamų parametrų pasikeitimai. Kitaip tariant, sistemos argumentai – tai grafiko X ašyje esantys duomenys, o sistemos parametrai – tai Y ašyje esantys duomenys. Parašytoje modeliavimo programoje yra 5 galimi argumentai:

- Sistemos efektyvumas (α), kintantis, keičiant paraiškų atvykimo srautą.
- Sistemos efektyvumas (α), kintantis, keičiant paraiškų aptarnavimo srautą.
- Paraiškų laukimo laikas eilėje.
- Aptarnavimo stoties kanalų skaičius.
- Maksimalus galimas eilės ilgis.

Modeliavimas. Tai yra pagrindinė programos dalis. Pats modelis realizuotas pagal 3.1 pav. pateiktą schema. Modelis veikia absoliučiai taip pat, kaip aprašyta 3.1 skyriuje apie modeliavimo įrangai keliamus reikalavimus. Reikia paminėti, kad atvykimo, laukimo ir aptarnavimo srautams yra galimybė pasirinkti įvairius pasiskirstymo dėsnius, kurių šioje programoje įdėta 11. Aišku esant reikalui, galima įdėti ir daugiau galimų pasiskirstymų, kaip ir kitų modeliavimo parametrų (tai nesunkiai leidžia padaryti *Delsi* bibliotekos naudojimo patogumas ir lankstumas). Kai kurie pasiskirstymai paimti iš *Delsi* bibliotekos, kiti paimti iš išorinės Troschuetz [15] bibliotekos. Pasiskirstymai yra tokie:

1. Eksponentinis.
2. Reguliarus.
3. Normalusis.
4. Beta.
5. Gamma.
6. Lognormalusis.
7. Trikampinis.
8. Tolydusis.
9. Diskretusis.
10. Puasono.
11. Binominis.

Programos modeliavimo branduolio veikimo kodo pavyzdys su komentarais pateiktas A priede, kur trumpai aprašomi programoje naudojami *Delsi* bibliotekos objektai ir jų funkcijos. B priede parodyta, kaip tam tikram objektui (pvz. paraiškų atvykimo srauto generatoriui) gali būti parinkti įvairūs statistinio pasiskirstymo dėsniai.

Duomenų išsaugojimas į masyvus. Kaip jau buvo minėta aukščiau, visi simuliacijos metu gauti rezultatai sukeliami į masyvus. Simuliacijai pasibaigus, naudojantis šiais masyvais,

galima atvaizduoti gautus duomenis arba tekstiniu pavidalu skaičių lentelėje, arba pateikti juos grafiškai.

Duomenų atvaizdavimas. Akivaizdu, kad patogiausias būdas suprasti modeliavimo metu gautus duomenis – pažiūrėti į jų grafikus. Naudojantis specialia biblioteka, programoje buvo sukurtas grafikų braižymo modulis. Jis veikia tokiu būdu, kad galima braižyti daugybės simuliacijų grafikus viename lange, kad būtų galima stebėti atitinkamų pasikeitimų dinamiką. Yra galimybė pasirinkti, kokius parametrus braižyti. Galima atvaizduoti arba vieną parametrą, arba keletą jų vienu metu, priklausomai nuo vartotojo noro ir poreikių. Paprastumo dėlei, grafikų braižymo grafinė sąsaja turi tik du mygtukus, vienas iš kurių prideda naują grafiką o kitas – ištrina paskutinį. Tokiu būdu patogu valdyti grafikų atvaizdavimą. Reikia paminėti, kad ašių vardai, matavimo vienetai ir kintamųjų legendos braižomos automatiškai. Grafikų braižymo programinio kodo dalies pavyzdys parodytas C priede.

Pilnas parašytas modeliavimo programinės įrangos programos kodas užima 67 A4 formato puslapius. Esminės kodo dalys (modeliavimo branduolys, tikimybinių pasiskirstymų pasirinkimas, grafikų braižymas) aprašytos priede. Likusios kodo dalys aprašo programą bendrosiomis programavimo funkcijomis (daugybės ciklų panaudojimas, sąlyginės išraiškos, masyvai, objektai, klasės), kurių neverta atskirai nagrinėti, nes egzistuoja daugybė žinynų ir vadovėlių, kur lengvai galima surasti kiekvienos funkcijos reikšmę ir aprašymą. Taip pat didelė kodo dalis aprašo grafinę sąsaja, per kurią vartotojas įveda pradinius simuliacijų duomenis ir gauna modeliavimo rezultatus. Programos kintamieji, nustatantys simuliacijos ilgį ir esminius tikimybinio pasiskirstymo parametrus, yra „double“ tipo. Tai reiškia, kad teoriškai jų vertes galima pasirinkti iš labai plataus ruožo (nuo $\pm 5.0 \times 10^{-324}$ iki 1.7×10^{308}). Tačiau praktiškai reikia atsižvelgti į naudojamo kompiuterio resursus, nes parinkus per ilgą simuliacijos laiką, modeliavimas gali būti vykdomas labai ilgai. Todėl verta palaipsniui didinti modeliavimo laiką ir žiūrėti kaip nuo tai priklauso simuliacijos trukmė. Kaip parodė tyrimų rezultatai, praktiškai užtenka, kad vidutinio laiko tarp dviejų atvykstančių paraiškų santykis (arba vidutinio aptarnavimo/laukimo eilėje laiko) su modeliavimo laiku būtų lygus 1:5000. Toliau mažinant šį santykį, gautų rezultatų paklaidų mažėjimas praktiškai nepastebimas.

Programos naudojimosi instrukcija pateikta CD plokštelėje.

4. Įvairių MAS modeliavimas ir tyrimų rezultatai

Naudojantis darbo metu sukurta MAS modeliavimo programine įranga, buvo atlikti keletas tyrimų. Sumodeliuotos skirtingos masinio aptarnavimo sistemos, atitinkančios tam tikras kompiuterines sistemas su informacijos srautais. Atlikti tyrimai parodo modeliavimo programinės įrangos pajėgumus ir galimybes. Pagal simuliacijų metu gautus duomenis, galima nustatyti tam tikras tendencijas, priklausomybes ir numatyti galimus modeliuojamų sistemų optimizavimo būdus.

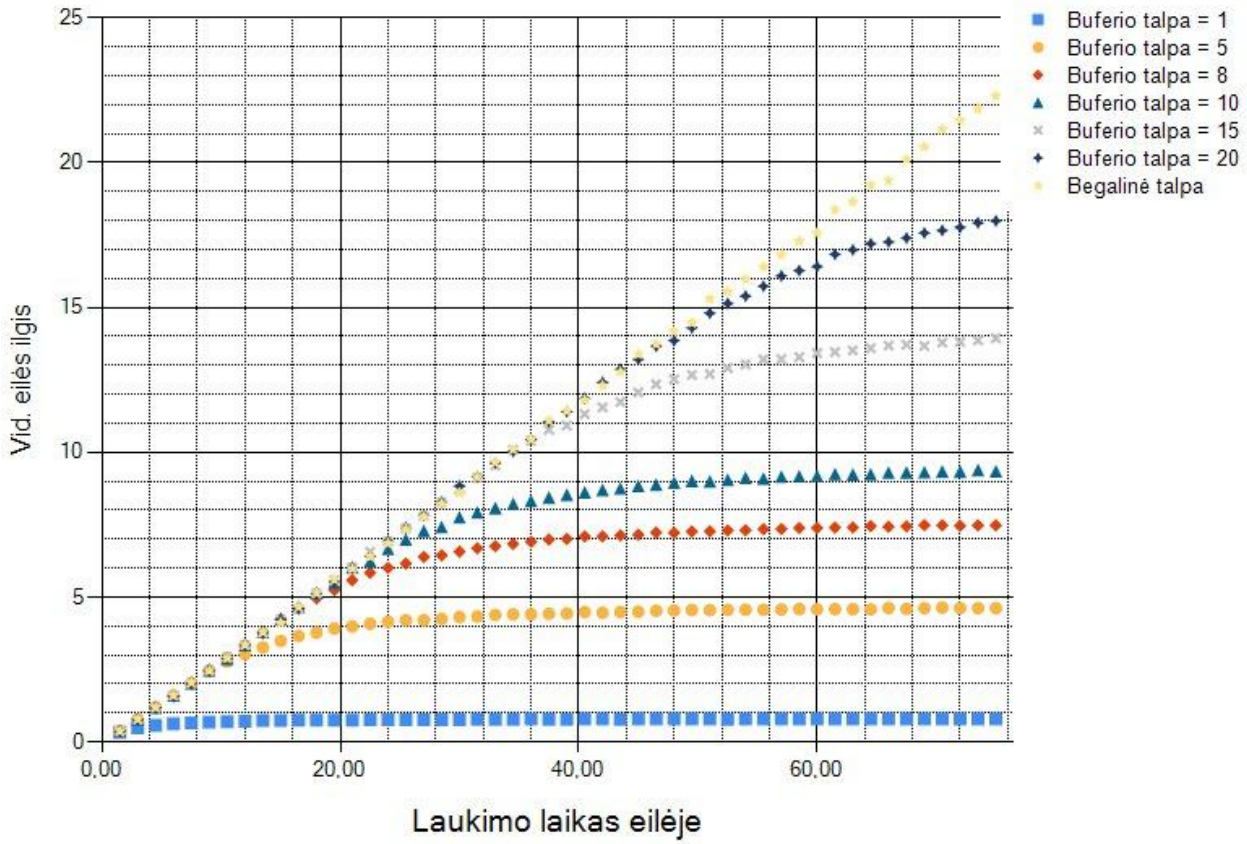
4.1. Vidutinio eilės ilgio priklausomybės nuo laukimo laiko eilėje ir buferio talpos tyrimas

Šio tyrimo metu buvo sumodeliuota masinio aptarnavimo sistema su apribotų eilės ilgiu ir laukimo eilėje laiku. Sistemos parametrai yra tokie:

- Aptarnaujančios stoties kanalų skaičius: 1.
- Atvykimo srauto charakteristikos: Normalusis pasiskirstymas (vidutinis laikas tarp paraiškų atvykimo momentų m_{atv}=3; vidutinis kvadratinis nuokrypis σ = 1).
- Aptarnavimo laiko charakteristikos: Eksponentinis pasiskirstymas su vidutiniu paraiškų aptarnavimo laiku m_{apt} = 10.
- Vidutinis paraiškų laukimo eilėje laikas $\overline{t}_{lauk}$ palaipsniui kinta nuo tolygiai pasiskirsčiusio tarp 0 ir 3 iki tolygiai pasiskirsčiusio tarp 0 ir 150 (t.y. laukimo laikas eilėje simuliacijos metu didėja).

Visi sistemos laikiniai parametrai nurodyti santykiniais vienetais. Simuliacijos ilgis lygus 50000 (santykinių laiko vienetų). Simuliacijos rezultatai rodomi grafike, sudarytame iš 50 taškų.

Šios simuliacijos tikslas – pažiūrėti, kaip kinta vidutinis paraiškų eilės ilgis, esant skirtingoms buferio talpoms. Modelio simuliacija padaryta septynis kartus, esant įvairioms buferio talpoms atitinkamai: 1, 5, 8, 10, 15, 20, begalinė talpa. Simuliacijos rezultatai parodyti 4.1.1 pav.



4.1.1 pav. Vidutinio paraiškų eilės ilgio priklausomybė nuo laukimo laiko eilėje, esant skirtingoms buferio talpoms

Gauti duomenys leidžia padaryti iš karto keletą išvadų.

1. Didėjant buferio talpai, vidutinis paraiškų eilės ilgis didėja. Tai yra akivaizdu, nes kuo daugiau yra vietų eilėje, tuo daugiau tų vietų bus užimta.
2. Didėjant paraiškų laukimo eilėje laikui, vidutinis paraiškų eilės ilgis didėja iki tam tikros laukimo laiko vertės, po kurios pradeda įsisotinti iki tam tikros pastovios eilės ilgio vertės (maksimalios buferio talpos).
3. Skirtumas tarp vidutinio paraiškų eilės ilgio, esant tam tikrai buferio talpai, ir vidutinio paraiškų eilės ilgio, esant begalinei buferio talpai, proporcingas atmestų paraiškų skaičiui dėl apriboto eilės ilgio.

Iš gautų rezultatų matosi, kad kiekvienai buferio talpos vertei egzistuoja tam tikras kritinis vidutinis paraiškų laukimo laikas eilėje, po kurio vidutinis paraiškų eilės ilgis pradeda įsisotinti ir daugiau nebedidėti. Kuo didesnė yra buferio talpa – tuo šis kritinis laikas yra didesnis. Norint optimizuoti masinio aptarnavimo sistemos su panašiomis charakteristikomis darbą, reikia nustatyti, kokiose ribose kinta vidutinis paraiškų laukimo eilėje laikas, ir priklausomai nuo tai, pasirinkti buferio talpą. Gauti modeliavimo duomenys rodo, kad esant tam

tikriems paraiškų laukimo eilėje laikams, nėra tikslinga didinti buferio talpą, nes vidutinis eilės ilgis nuo to nesikeičia. Kuo mažesnis yra laukimo laikas eilėje – tuo mažiau atmestų paraiškų skaičius priklauso nuo buferio talpos.

4.2. Serverio prastovos tikimybės priklausomybės nuo paraiškų laukimo laiko eilėje, esant skirtingiems serverio kanalų skaičiams, tyrimas

Šitos simuliacijos metu buvo tiriama sistema, kurios charakteristikos yra tokios:

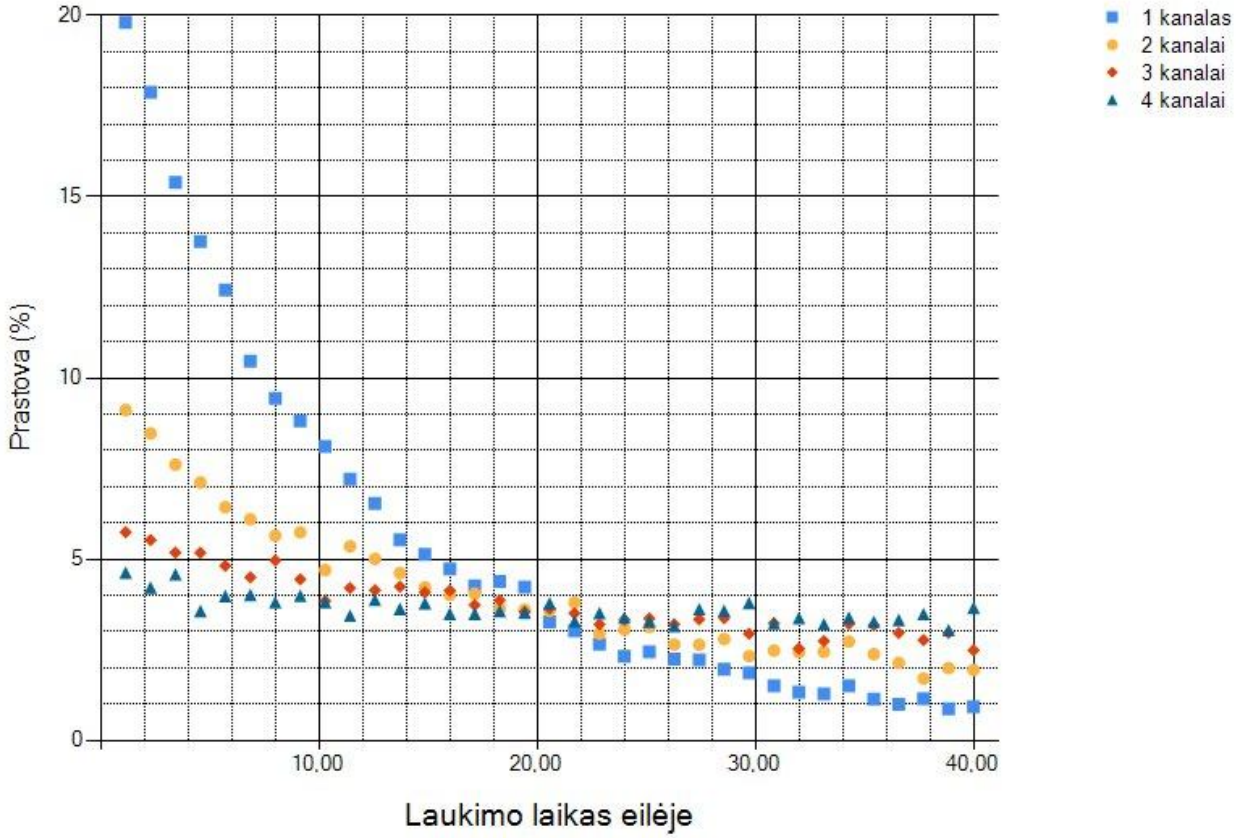
- Buferio talpa: begalinė.
- Atvykimo srauto charakteristikos: Eksponentinis pasiskirstymas su vidutiniu laiku tarp paraiškų atvykimo momentų m_{atv}=6.
- Aptarnavimo laiko charakteristikos: Eksponentinis pasiskirstymas su vidutiniu paraiškų aptarnavimo laiku m_{apt} = 20.
- Paraiškų laukimo eilėje laikas pasiskirstęs eksponentiškai. Laukimo laiko vidurkis $\overline{t}_{lauk}$ kinta nuo 0 iki 50.

Simuliacijos ilgis 150000. Grafiko taškų skaičius 35. Simuliacija buvo daryta 4 kartus, kiekvieną kartą keičiant aptarnaujančios stoties (serverio) skaičių. Kanalų skaičius simuliacijoje kinta nuo 1 iki 4. Tyrimo rezultatai pateikti 4.2.1 pav.

Iš gautų grafikų matosi, kad didėjant paraiškų laukimo eilėje laikui, prastovos tikimybė mažėja. Tai yra logiška, nes kuo ilgiau paraiška gali laukti eilėje, tuo didesnė tikimybė, kad jinai bus aptarnauta. Kita vertus, kuo didesnis yra serverio kanalų skaičius – tuo mažiau prastova priklauso nuo paraiškų laukimo laiko eilėje. Taip yra todėl, kad kuo daugiau kanalų – tuo mažiau paraiškoms reikia laukti eilėje, ir jos greičiau patenka į serverį. Kitaip tariant, mažėja vidutinis paraiškų buvimo laikas eilėje, ir serverio užimtumas yra daugmaž pastovus, didėjant galimam paraiškų laukimo eilėje laikui.

Tačiau įdomiausias šio tyrimo rezultatas yra tai, kad kai paraiškų laukimo laikas eilėje yra lygus paraiškų aptarnavimo laikui ($\overline{t}_{lauk}$ = m_{apt}) – serverio prastova nepriklauso nuo kanalų skaičiaus. $\overline{t}_{lauk}$ = m_{apt} yra tas laikas, ties kuriuo susikerta visos priklausomybės. Sprendžiant iš sistemų optimizavimo požiūrio, galima pasakyti kad jeigu norima, jog MAS serverio prastovos tikimybė būtų nepriklausoma nuo kanalų skaičiaus – reikia taip organizuoti sistemą, kad vidutinis paraiškų aptarnavimo laikas būtų lygus paraiškų laukimo eilėje laikui. Jeigu $\overline{t}_{lauk}$ < m_{apt}

– norint sumažinti prastovos tikimybę, reikia didinti kanalų skaičių; o kai $\overline{t}_{lauk} > m_{apt}$ – kanalų skaičių reikėtų mažinti.



4.2.1 pav. Serverio prastovos tikimybės priklausomybė nuo paraiškų laukimo laiko eilėje, esant skirtingiems kanalų skaičiams

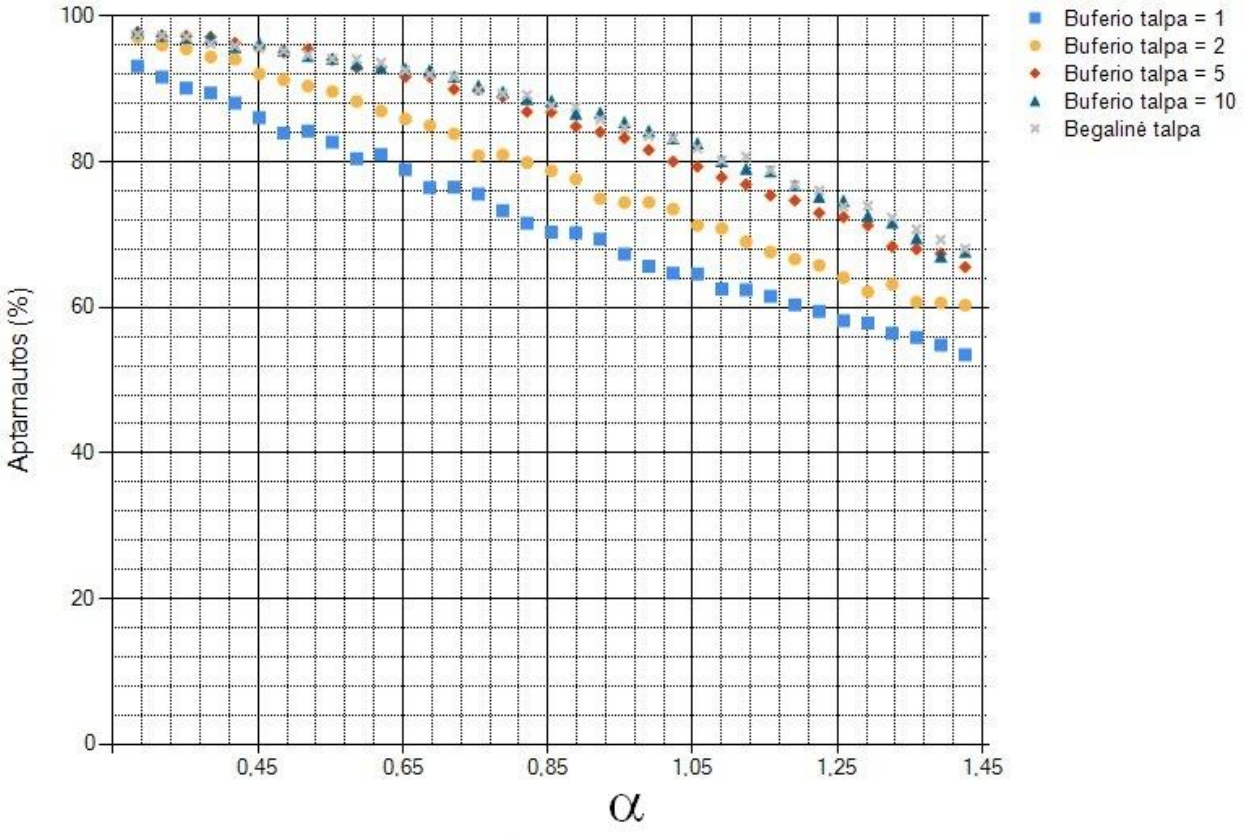
4.3. Paraiškų aptarnavimo tikimybės priklausomybės nuo sistemos efektyvumo koeficiento, esant skirtingoms buferio talpoms, tyrimas

Šito tyrimo tikslas – išsiaiškinti, kaip kinta simuliacijos metu aptarnautų paraiškų procentas (kitaip tariant, kokia yra aptarnavimo tikimybė), didėjant sistemos efektyvumui α. MAS buvo modeliuojama 5 kartus su skirtingomis buferio talpomis (1, 2, 5, 10, begalinė talpa). Modeliuojamos sistemos charakteristikos yra tokios:

- Aptarnaujančios stoties kanalų skaičius: 1.
- Atvykimo srauto charakteristikos: Eksponentinis pasiskirstymas (vidutinis laikas tarp paraiškų atvykimo momentų m_{atv} kinta nuo 35 iki 7).
- Aptarnavimo laiko charakteristikos: Eksponentinis pasiskirstymas su vidutiniu paraiškų aptarnavimo laiku $m_{apt} = 10$.

- Vidutinio paraiškų laukimo eilėje laiko charakteristikos: Eksponentinis pasiskirstymas su vidutiniu paraiškų laukimo eilėje laiku $\overline{t}_{lauk} = 150$.

Simuliacijos ilgis 150000. Grafiko taškų skaičius 35. Eksperimento rezultatai pateikti 4.3.1 pav.



4.3.1 pav. Aptarnavimo tikimybės priklausomybė nuo sistemos efektyvumo koeficiento, esant skirtingoms buferio talpoms

Matome, kad didėjant sistemos efektyvumui, aptarnavimo tikimybė mažėja. Taip yra todėl, kad didėjant sistemos efektyvumui, paraiškų atvykimo srauto intensyvumas šiuo atvejų didėja, kol vidutinis paraiškų aptarnavimo laikas lieka pastovus. Dėl to didesnė dalis paraiškų yra atmetama. Taip pat iš gautų duomenų matyt, kad didinant buferio talpą virš 10, aptarnavimo tikimybė praktiškai visiškai nesikeičia. Tai reiškia kad turint tokių parametrų kompiuterinę sistemą, yra visiškai beprasmiška didinti buferio talpą virš 10, norint pasiekti didesnę paraiškų aptarnavimo tikimybę. Šiuo atveju reikėtų ieškoti kitų problemos sprendimo kelių (pvz. didinti aptarnavimo stoties kanalų skaičių).

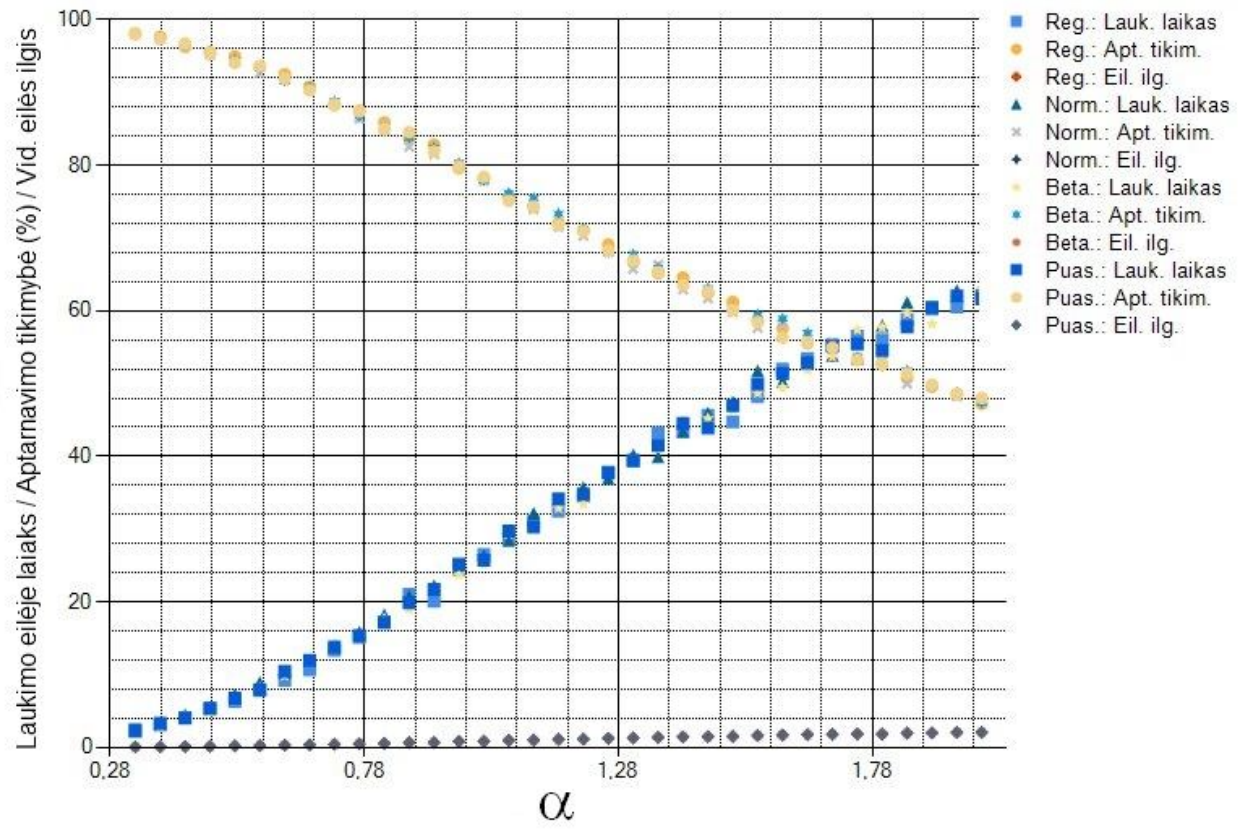
4.4. Įvairių masinio aptarnavimo sistemos parametrų priklausomybių nuo skirtingų aptarnavimo srauto pasiskirstymo dėsnių tyrimas

Šito tyrimo tikslas yra pažiūrėti, kaip priklauso vidutinis paraiškų eilės ilgis, aptarnavimo tikimybė ir vidutinis paraiškų laukimo laikas eilėje nuo skirtingų aptarnavimo srauto pasiskirstymo dėsnių. Tai reiškia, kad tyrimo metu sistemos parametrai išlieka pastovūs, ir kinta tik vidutinio aptarnavimo laiko pasiskirstymo dėsnis. Modeliuojamos sistemos parametrai yra tokie:

- Aptarnaujančios stoties kanalų skaičius: 1.
- Buferio talpa: begalinė.
- Atvykimo srauto charakteristikos: Eksponentinis pasiskirstymas su vidutiniu laiku tarp paraiškų atvykimo momentų $m_{atv} = 30$.
- Aptarnavimo laiko charakteristikos: Vidutinis paraiškų aptarnavimo laikas m_{apt} kinta nuo 20 iki 60.
- Vidutinio paraiškų laukimo eilėje laiko charakteristikos: Eksponentinis pasiskirstymas su vidutiniu paraiškų laukimo eilėje laiku $\overline{t}_{lauk} = 120$.

Simuliacijų ilgis 200000. Grafiko taškų skaičius 35. Simuliacija buvo daryta 4 kartus, kiekvieną kartą keičiant aptarnavimo laiko pasiskirstymą (reguliarusis, normalusis, Beta ir Puasono pasiskirstymai). Eksperimento rezultatai pateikti 4.4.1 pav. Grafiko legendos santrumpų paaiškinimai:

- Reg. – reguliarusis aptarnavimo srautas.
- Norm. – normalusis aptarnavimo srautas.
- Beta – Beta pasiskirstymo srautas.
- Puas. – Puasono pasiskirstymo srautas.
- Lauk. laikas – vidutinis paraiškų laukimo laikas eilėje.
- Apt. tikim. – paraiškų aptarnavimo tikimybė.
- Eil. ilg. – vidutinis paraiškų eilės ilgis.



4.4.1 pav. Paraiškų laukimo laiko eilėje, aptarnavimo tikimybės ir paraiškų vidutinio eilės ilgio priklausomybės nuo sistemos efektyvumo koeficiento, esant įvairiems aptarnavimo srauto pasiskirstymo dėsniams

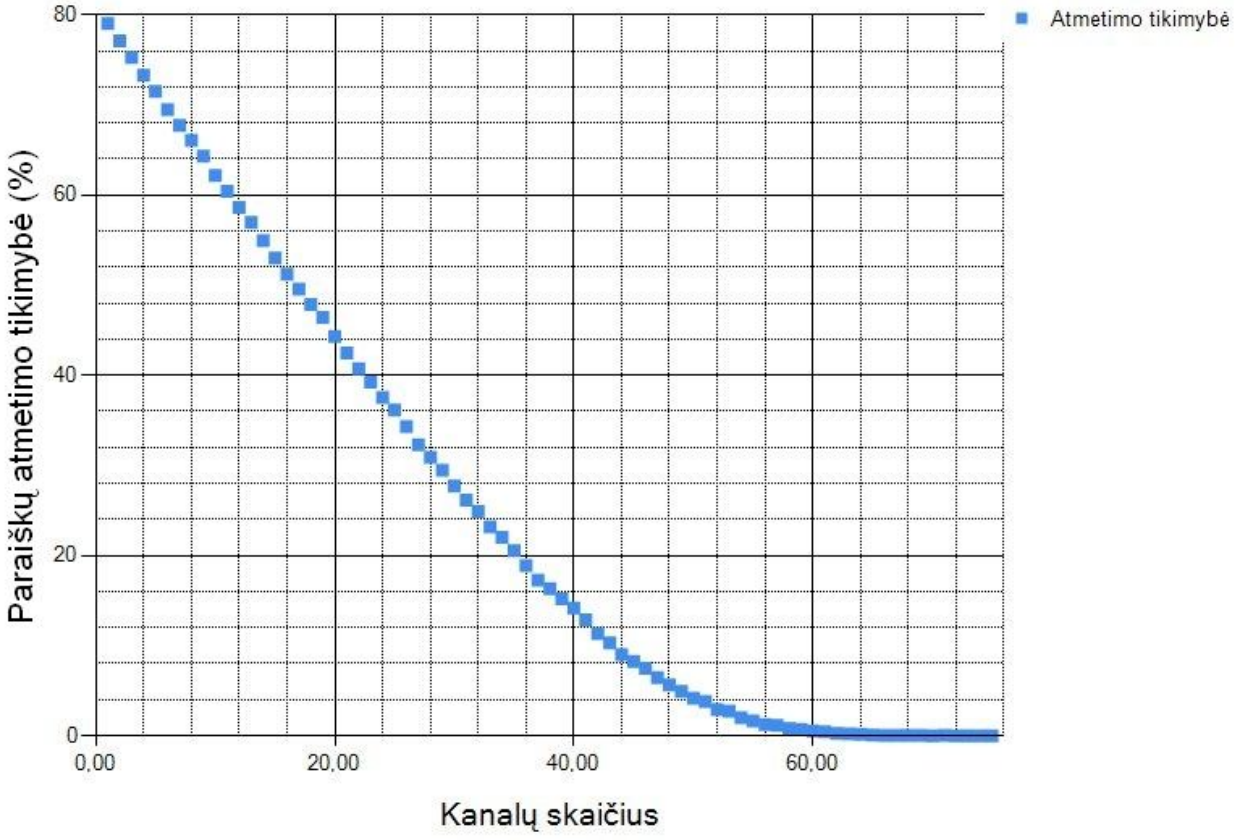
Aiškiai matyt, kad esant skirtingiems vidutinio aptarnavimo laiko pasiskirstymo dėsniams, tiriamų sistemos parametrų vertės nesikeičia. Skirtingų pasiskirstymų grafikų taškai susilieja į vieną tašką taip, kad juos net sunku vizualiai atskirti. Taigi, galima daryti išvadą, kad norint optimizuoti masinio aptarnavimo kompiuterinę sistemą, galima pasirinkti tokį vidutinio paraiškų aptarnavimo laiko pasiskirstymo dėsnį, kokį naudoti bus patogiau, nes sistemos darbo rezultatai nuo to nesikeičia.

4.5. Atmestų paraiškų procento priklausomybės nuo aptarnaujančios stoties kanalų skaičiaus tyrimas

Tyrimo tikslas – nustatyti, kaip kinta atmestų paraiškų procentas (atmetimo tikimybė), didėjant aptarnaujančio serverio kanalų skaičiui, kai kiti sistemos parametrai nekinta. Atmestos – tai tokios paraiškos, kurios, ateinant, nukreipiamos į išėjimą iš sistemos dėl to, kad buferis yra užpildytas. Modeliuojamos sistemos charakteristikos:

- Buferio talpa: 5.
- Atvykimo srauto charakteristikos: Eksponentinis pasiskirstymas su vidutiniu laiku tarp paraiškų atvykimo momentų $m_{atv} = 1$.
- Aptarnavimo laiko charakteristikos: Eksponentinis pasiskirstymas su vidutiniu paraiškų aptarnavimo laikų $m_{apt} = 50$.
- Vidutinio paraiškų laukimo eilėje laiko charakteristikos: Eksponentinis pasiskirstymas su vidutiniu paraiškų laukimo eilėje laiku $\overline{t}_{lauk} = 25$.

Simuliacijos ilgis 50000. Grafiko taškų skaičius 75, nes kanalų skaičius kinta nuo 1 iki 75. Eksperimento rezultatų grafikas pateiktas 4.5.1 pav.



4.5.1 pav. Paraiškų atmetimo tikimybės priklausomybė nuo aptarnaujančios stoties kanalų skaičiaus

Iš eksperimento metu gautų rezultatų galima daryti išvadą, kad atmestų paraiškų kiekis tiriamoje sistemoje mažėja, didėjant aptarnaujančios stoties kanalų skaičiui. Paraiškų atvykimo srautas yra toks, kad per vieną laiko vienetą į sistemą ateina viena paraiška. Vidutiniškai viena paraiška aptarnaujama per 50 laiko vienetų. Tai reiškia, kad pradinis sistemos efektyvumo koeficientas (kai kanalų skaičius lygus vienam) yra $\alpha = m_{apt} / m_{atv} = 50$. Esant tokiam atvejui, paraiškos ateina į sistemą 50 kartų intensyviau, negu jos yra apdorojamos serverio. Akivaizdu,

kad serveris visada bus perkrautas ir bus labai daug dėl atmėtimo neaptarnautų paraiškų. Didėjant kanalų skaičiui, koeficientas α mažėja, dėl to tiesiškai mažėja atmestų paraiškų skaičius. Galiausiai, kai sistemos efektyvumo koeficientas tampa lygus $\alpha = 1$ (t.y. serverio kanalų skaičius tampa lygus vidutiniam paraiškų aptarnavimo laikui 50), atmetimo tikimybės priklausomybės nuo kanalų skaičiaus kreivė pradeda įsisotinti ties nulinę vertę. Toliau didinant serverio kanalų skaičių, paraiškos nebeatmetamos visiškai, nes serverio pajėgumų užtenka visoms paraiškoms aptarnauti. Tačiau tai nereiškia, kad aptarnautos yra visos paraiškos, atėjusios į sistemą. Kai kurios paraiškos palieka sistemą, kai baigiasi jų laukimo eilėje laikas.

Galima padaryti išvada, kad kai sistemos efektyvumo koeficiento vertė yra lygi arba mažesnė už vienetą $\alpha < 1$, nėra tikslinga didinti aptarnaujančios stoties kanalų skaičių, norint sumažinti paraiškų atmetimo tikimybę (nes ji ir taip yra artima nuliui).

5. Kompiuterių tinklo komutatoriaus darbo modeliavimas ir jo parametrų tyrimas

Aukščiau pateiktuose pavyzdžiuose buvo tiriamos abstrakčios masinio aptarnavimo sistemos. Pradiniai duomenys buvo parinkti tokiu būdu, kad tyrimu rezultatai gautųsi kuo akivaizdesni ir aiškesni. Pagrindinis pateiktų tyrimų tikslas – parodyti sukurtos MAS modeliavimo programos galimybes ir galimas taikymo sritis. Norint tirti modelį, panašų į realias masinio aptarnavimo sistemas, reikia imti duomenis iš realiai egzistuojančių kompiuterinių sistemų.

Šitame skyriuje bus modeliuojamas vidutinio dydžio tinklo (iki 100 kompiuterių) komutatoriaus darbas. Pradiniai modelio duomenys parinkti, remiantis realiai egzistuojančių sistemų parametrais. Šie duomenys pateikti 5.1 lentelėje:

5.1 lentelė. Pradiniai modelio parametrai

Paketų atvykimo sparta (paketai/sek)	3330
Vidutinis vieno paketo dydis (kB)	0.85
Paketų buferio dydis (kB)	512

Pirmieji du parametrai (paketų atvykimo sparta ir vidutinis vieno paketo dydis) buvo parinkti, naudojantis duomenimis apie VGTU elektronikos fakulteto tinklo drabą didžiausios apkrovos laiku (10:00 - 16:30 val). Paketų buferio dydis parinktas, naudojantis HP V1910-48G komutatoriaus charakteristikomis [16].

Vienas iš svarbiausių MAS parametrų – aptarnautų paraiškų procentas (aptarnavimo tikimybė). Todėl visų pirmą reikia parinkti pakankamą paraiškų aptarnavimo spartą, kad atmestų dėl komutatoriaus buferio perpildymo paraiškų procentas patektų į leistinas ribas. Tarkime, kad mūsų nagrinėjamai sistemai leidžiamas atmestų paraiškų procentas yra 10 %. Tai reiškia, kad aptarnavimo sparta turi būti tokia, kad nemažiau 90 % į sistemą patekusių paraiškų turi būti aptarnauta.

Optimaliam paketų aptarnavimo laikui surasti, naudojantis sukurta MAS modeliavimo programa, buvo atliktas tyrimas, kurio nustatymai pateikti 5.2 lentelėje:

5.2 lentelė. MAS modeliavimo programos nustatymai pirmam tyrimui

Taškų skaičius	50
Simuliacijos ilgis	120
Pasirinkite X ašį	Alfa (aptarnavimas)
Pasirinkite Y ašį	Aptarnautų paraiškų procentas
Buferio talpa	602
Kanalų skaičius	1
Atvykimo pasiskirstymo dėsnis	Eksponentinis; $1/\lambda = 0.0003$
Laukimo pasiskirstymo dėsnis	Reguliarusis; $1/\nu = 1$
Aptarnavimo pasiskirstymo dėsnis (pradinis)	Eksponentinis; Pradinis: $\mu = 0.00025$ Galutinis: $\mu = 0.0004$

Atvykimo srauto intensyvumas yra pakankamai didelis ir lygus vidutiniškai 3330 paketų per sekundę (žr. 5.1 lent.). Todėl simuliacijos ilgis neturi būti labai didelis, ir dviejų minučių tinklo darbo simuliacijos pilnai užtektų. Dėl to simuliacijos ilgis šiam tyrimui lygus 120 santykinių laiko vienetų (nagrinėjamu atveju sekundžių).

Mus domina aptarnautų paraiškų procento priklausomybė nuo paketų aptarnavimo intensyvumo. Todėl funkcijos argumentu pasirenkame efektyvumo koeficientą, kuris lygus:

$$\alpha = \frac{\lambda}{\mu}, \tag{5.1}$$

λ šiuo atveju yra pastovus viso modeliavimo metu ir lygus 3330 ($m_{atv} = 1/\lambda = 0.0003$ sek). μ kinta palaipsniui eksperimento metu, todėl stebint sistemos parametrų priklausomybę nuo α mes faktiškai stebime parametrų priklausomybę nuo μ, nes

$$\mu = \frac{\lambda}{\alpha}, \text{ kur } \lambda = \text{const}=3330. \tag{5.2}$$

Modeliavimo metu m_{apt} kinta nuo 0.00025 iki 0.0004. Paketų atvykimo ir aptarnavimo laikų pasiskirstymo dėsnis modelyje parinktas eksponentinis, nes iš aukščiau pateiktų eksperimentų (žr. 4.4 sk.) matyt, kad sistemos parametrai mažai priklauso nuo tikimybinio pasiskirstymo dėsnio, o eksponentinis pasiskirstymo dėsnis yra patogiausias skaičiavimams.

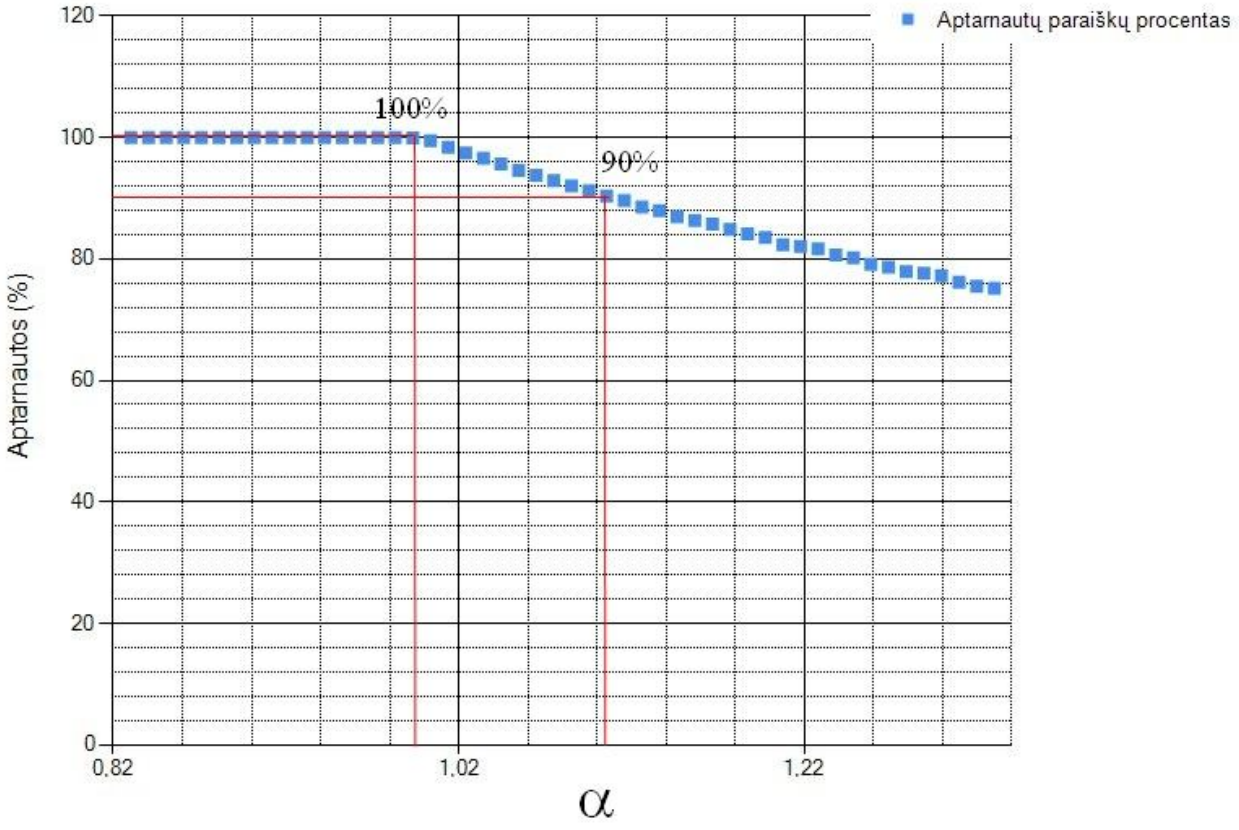
Modeliuojamo tipo tinkluose paketai nepalieka buferio dėl laukimo laiko pabaigos. Jeigu paketas pateko į eilę, jis lauks tol, kol bus aptarnautas. Todėl paketų laukimo laikas eilėje parinktas pakankamai didelis ($\overline{t}_{lauk}$ = 1 sek), palyginus su paketų atvykimo ir aptarnavimo laikais, tam kad nei vienas paketas nepaliktų eilės dėl laukimo laiko pabaigos (kaip nustatyti laukimo laiką tokiu būdu, kad paraiškos nebūtų atmetamos dėl laukimo eilėje laiko pabaigos, parašyta programos naudojimosi instrukcijoje, esančioje CD plokštelėje).

Paketų buferio talpa suskaičiuojama pagal formulę:

$$\text{Buferio talpa} = \frac{\text{Paketų buferio dydis } (\text{kB})}{\text{Vidutinis vieno paketo dydis } (\text{kB})} = \frac{512}{0.85} \approx 602. \qquad (5.3)$$

Vienas komutatorius – tai viena aptarnavimo stotis, todėl kanalų skaičius lygus 1.

Modeliavimo metu gauti rezultatai grafiškai atvaizduoti 5.1 pav.:



5.1 pav. Paketų aptarnavimo tikimybės priklausomybė nuo sistemos efektyvumo koeficiento

Gauti duomenys pateikti 5.3 lentelėje:

5.3 lentelė. Tiriamos MAS pirmo modeliavimo rezultatai

α	m_{apt} (sek)	Aptarnautos (%)	α	m_{apt} (sek)	Aptarnautos (%)
0,83	0,000250	100,00	1,09	0,000326	92,01
0,84	0,000253	100,00	1,10	0,000330	91,19
0,85	0,000256	100,00	1,11	0,000333	90,33
0,86	0,000259	100,00	1,12	0,000336	89,60
0,87	0,000262	100,00	1,13	0,000339	88,54
0,88	0,000265	100,00	1,14	0,000342	87,92
0,90	0,000269	100,00	1,15	0,000345	86,91
0,91	0,000272	100,00	1,16	0,000348	86,36
0,92	0,000275	100,00	1,17	0,000351	85,76
0,93	0,000278	100,00	1,18	0,000354	84,87
0,94	0,000281	100,00	1,19	0,000357	84,15
0,95	0,000284	100,00	1,20	0,000360	83,54
0,96	0,000287	100,00	1,21	0,000363	82,29
0,97	0,000290	100,00	1,22	0,000366	82,06
0,98	0,000293	100,00	1,23	0,000369	81,65
0,99	0,000296	100,00	1,24	0,000372	80,62
1,00	0,000299	99,90	1,25	0,000376	80,19
1,01	0,000302	99,46	1,26	0,000379	79,11
1,02	0,000305	98,36	1,27	0,000382	78,63
1,03	0,000308	97,46	1,28	0,000385	77,91
1,04	0,000311	96,57	1,29	0,000388	77,65
1,05	0,000314	95,61	1,30	0,000391	77,25
1,06	0,000317	94,59	1,31	0,000394	76,18
1,07	0,000320	93,73	1,32	0,000397	75,53
1,08	0,000323	92,88	1,33	0,000400	75,17

Iš modeliavimo metu gautų duomenų aiškiai matosi, kad komutatoriaus vidutinis vieno paketo aptarnavimo laiko m_{apt} ruožas, kuriam esant bus aptarnauta 90 % - 100 % į sistemą patekusių paketų, bus:

0.000333 - 0.000296 sek.

Nėra prasmės mažinti vieno paketo aptarnavimo laiką mažiau kaip 0,000296 sek., nes skaičiavimo galios padidinimas nuolat reikalauja naujų resursų/kaštų, didės prastova, o rezultatas

vis tiek bus toks pats: 100 % aptarnautų paraiškų. Tuo pačiu negalima didinti vieno paketo aptarnavimo laiką daugiau kaip 0.000333, nes tuo atveju aptarnavimo tikimybė bus mažesnė už 90 %, o nagrinėjamos sistemos atveju tai netinka, nes leidžiamas atmetamų paketų procentas yra lygus tik 10 %. Todėl šiame tyrime modeliuojamai sistemai optimaliausiu variantu būtų pasirinkti gauto aptarnavimo laiko m_{apt} ruožo vidurkį, kuris lygus

$$m_{apt} = 0.000315 \text{ sek.}$$

Tokiu būdu, aptarnavimo tikimybė bus apytiksliai lygi 95 % (5.1 pav., 5.3 lent.). Spartą, kuria komutatorius turėtų apdoroti paketus galima paskaičiuoti pagal formulę:

$$\text{Komutatoriaus sparta} = \frac{\text{Vidutinis vieno paketo dydis } (\text{kB})}{m_{apt}\ (\text{sek})} = \frac{0.85}{0.000315} \approx \qquad (7.4)$$

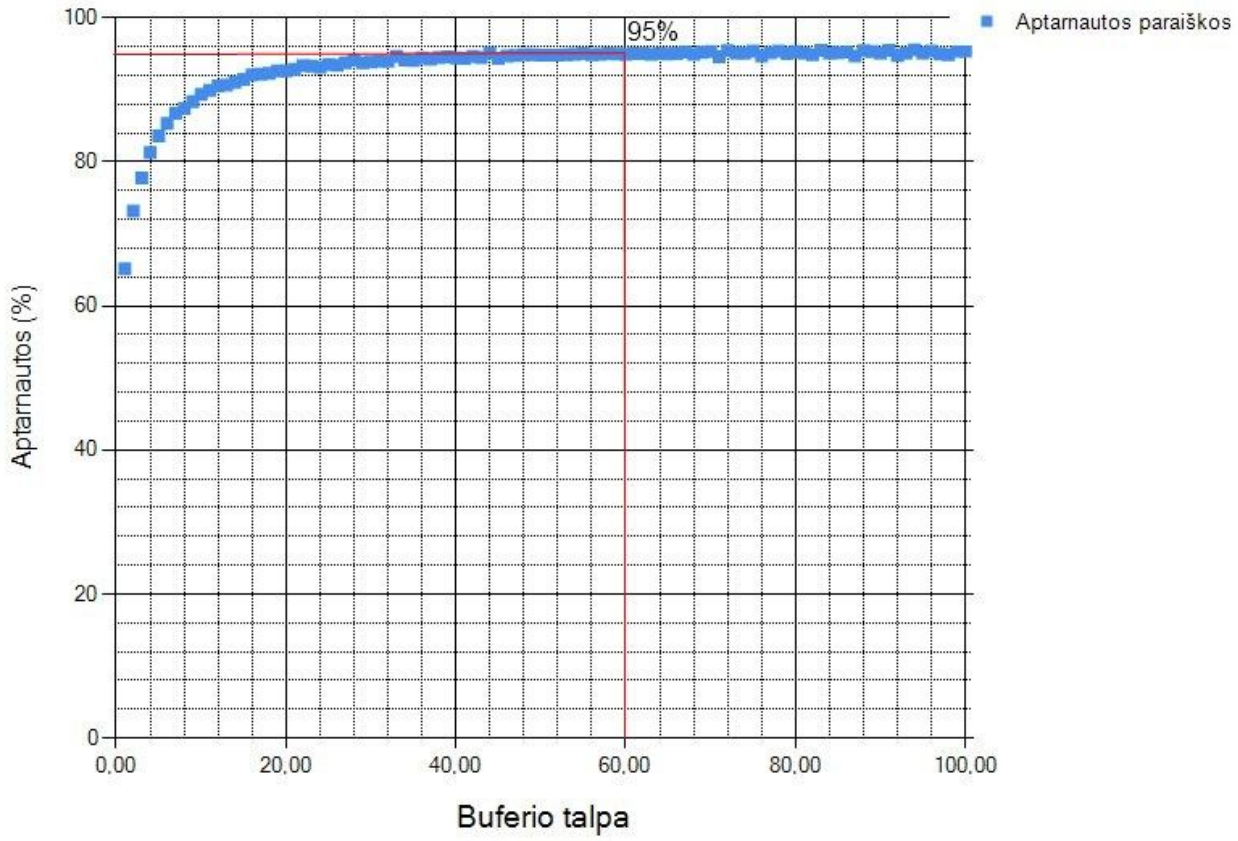
≈ 2.7 MB/sek = 21.6 Mbps.

Aukščiau buvo minėta, kad nagrinėjamame modelyje buvo naudojama sistemos buferio talpa, lygi 602 paketams. Tačiau tai dar nereiškia, kad toks talpos pasirinkimas yra optimalus. Tam kad surasti optimaliausią buferio talpą nagrinėjamai sistemai, buvo atliktą dar vieną simuliaciją, kurios metu buvo modeliuojama aptarnavimo tikimybės priklausomybė nuo buferio talpos. Modeliavimo programos nustatymai, optimaliai buferio talpai surasti, parodyti 5.4 lentelėje:

5.4 lentelė. MAS modeliavimo programos nustatymai antram tyrimui

Simuliacijos ilgis	120
Pasirinkite X ašį	Buferio talpa
Pasirinkite Y ašį	Aptarnautų paraiškų procentas
Buferio talpa	Nuo: 1 Iki: 100
Kanalų skaičius	1
Atvykimo pasiskirstymo dėsnis	Eksponentinis; $1/\lambda = 0.0003$
Laukimo pasiskirstymo dėsnis	Reguliarusis; $1/\nu = 1$
Aptarnavimo pasiskirstymo dėsnis (pradinis)	Eksponentinis: $\mu = 0.000315$

Žinoma, kad komutatoriaus aptarnavimo tikimybė bus lygi 95 % tuo atveju, kai vidutinis vieno paketo aptarnavimo laikas bus lygus $m_{apt} = 0.000315$ sek. Antros simuliacijos tikslas – surasti minimalią buferio talpą, kuriai esant aptarnavimo tikimybė vis dar būtų lygi 95 %. Gautos priklausomybės grafikas parodytas 5.2 pav.:



5.2 pav. Paketų aptarnavimo tikimybės priklausomybė nuo sistemos buferio talpos

Iš gauto grafiko matosi, kad aptarnavimo tikimybės priklausomybės nuo sistemos buferio talpos kreivė įsisotina ties 95 % verte, kaip ir buvo tikėtasi. Kreivė įsisotina visiškai, kai buferio talpa lygi 60 paketų. Tai reiškia, kad nėra tikslinga naudoti 602 pakėtų buferio talpą; nagrinėjamai sistemai buferio talpą galima sumažinti net 10 kartų, ir nuo to sistemos charakteristikos nepablogėtų. Komutatoriaus atminties talpos skaičiavimo išraišką:

$$60\text{paketų} \cdot 0.85\text{kB/paketas} = 51\text{kB}.$$

Galima padaryti išvadą, kad modeliuojamo kompiuterių tinklo optimaliam darbui užtikrinti užtektų 51kB komutatoriaus paketų buferio dydžio.

Remiantis gautais rezultatais, buvo atliktas dar vienas modeliavimas, siekiant nustatyti kitus sistemos parametrus: vidutinį paketų laukimo laiką eilėje, vidutinį paketų eilės ilgį ir sistemos prastovos tikimybę. Modeliavimo programos nustatymai duotiems parametrams rasti pateikti 5.5 lentelėje:

5.5 lentelė. MAS modeliavimo programos nustatymai trečiam tyrimui

Taškų skaičius	1
Simuliacijos ilgis	120
Pasirinkite X ašį	Alfa (aptarnavimas)
Pasirinkite Y ašį	Vid. laukimo laikas eilėje Vid. eilės ilgis Prastovos tikimybė
Buferio talpa	60
Kanalų skaičius	1
Atvykimo pasiskirstymo dėsnis	Eksponentinis; $1/\lambda = 0.0003$
Laukimo pasiskirstymo dėsnis	Reguliarusis; $1/\nu = 1$
Aptarnavimo pasiskirstymo dėsnis (pradinis)	Eksponentinis; Pradinis: $\mu = 0.000315$ Galutinis: $\mu = 0.000315$

Taškų skaičius lygus 1, nes visi kiti parametrai mums yra žinomi. Buferio talpa lygi 60, kaip buvo nustatyta iš prieš tai esančio tyrimo duomenų. Aptarnavimo laikas $m_{apt} = 0.000315$, ir pradinis ir galutinis, nes faktiškai jis yra tik vienas, todėl kad modeliavimo metu programa vykdys visus skaičiavimus tik vieną kartą.

Tyrimo rezultatai, kartu su prieš tai gautais duomenimis, pateikti 5.6 lentelėje:

5.6 lentelė. Modeliavimo rezultatai

Vidutinis vieno paketo aptarnavimo laikas	0.000315 sek
Komutatoriaus buferio talpa	60
Komutatoriaus sparta	21.6 Mbps
Komutatoriaus atminties talpa	51 kB
Vidutinis paketų laukimo laikas eilėje	0.014 sek
Vidutinis paketų eilės ilgis	42.9
Sistemos prastovos tikimybė	0.286 %

Atlikto modeliavimo metu buvo nagrinėjama supaprastinta kompiuterių tinklo komutatoriaus veikimo schema. Buvo padaryta prielaida, kad paketų atvykimo tikimybinio pasiskirstymo dėsnis yra eksponentinis. Realybėje sunku nustatyti koks yra tinklo paketų atvykimo intensyvumo tikimybinio pasiskirstymo dėsnis, nes kompiuterių tinklo apkrova priklauso nuo daugybės faktorių: paros meto, savaitės dienos, mėnesio, tinklo anomalijų, vartotojų ir t. t. Dėl to realių komutatorių paketų buferio talpos ir paketų apdorojimo spartos

būna žymiai didesnės. Tai reikalinga tam, kad užtikrinti kuo stabilesnį viso tinklo darbą, net esant pernelyg didelei tinklo apkrovai.

Apibendrinimas. Išvados

1. Darbo metu padaryta trumpa šiuo metu rinkoje egzistuojančių MAS modeliavimo programinių sprendimų apžvalga. Buvo apžvelgtos programos: *Java Modelling Tools*, *SimEvents* ir *Delsi 2.0*.
2. Esamo darbo tikslams įgyvendinti buvo sukurta specializuota MAS modeliavimo programą, naudojantis *Delsi 2.0* modeliavimo biblioteka. Programa parašyta C# programavimo kalba, naudojantis *Microsoft Visual Studio 2010* programavimo platforma. Parašyta programa leidžia modeliuoti ir tirti įvairių konfigūracijų masinio aptarnavimo sistemas. Tokiu būdu galima modeliuoti informacijos srautus kompiuterių sistemose ir tirti jų parametrus ir savybes, priklausomai nuo sistemų ir srautų charakteristikų. Esminės programos kodo dalys su komentarais pateiktos esamo darbo prieduose.
3. Naudojantis parašyta MAS modeliavimo programa, buvo sukurti keletas paprastų sistemų modelių ir ištirtos jų charakteristikos. Gauti simuliacijų rezultatai pateikti grafiškai. Iš gautų duomenų aiškiai stebimos tam tikros tendencijos ir priklausomybės, kurios leidžia suprasti kokius veiksmus reikia atlikti, norint optimizuoti sistemos darbą arba tam tikru būdu ji pakeisti, nedarant neigiamos įtakos sistemos charakteristikoms.
4. Naudojant sukurtą MAS modeliavimo programą, buvo atlikta keletas tyrimų, sumodeliuota vidutinio dydžio kompiuterių tinklo komutatoriaus veikla. Buvo padaryta prielaida, kad paketų atvykimo į tinklą intensyvumo statistinio pasiskirstymo dėsnis yra eksponentinis. Paketų atvykimo intensyvumas lygus vidutiniškai 3330 paketų per sekundę. Komutatorius turi veikti tokiu būdu, kad paketų aptarnavimo tikimybė būtų nemažesnė, kaip 90 %. Modeliavimo rezultatai parodė, kad tokiu atveju optimaliausia komutatoriaus paketų apdorojimo sparta būtų lygi 21.6 Mbps, o paketų buferio talpa 51 kB. Taip pat modeliavimo metu buvo gautos ir kitos nagrinėjamos MAS charakteristikos: vidutinis paketų laukimo laikas eilėje lygus 0.014 sek, vidutinis paketų eilės ilgis lygus 42.9, ir sistemos prastovos tikimybė lygi 0.286 %.

Literatūra

1. Sakalauskas L. 2000. *Masinio aptarnavimo sistemos transporte*: vadodėlis. Vilnius: Technika. 156 p.

2. Queueing theory [interaktyvus]. 2013. Wikipedia [žiūrėta 2013 m. gegužės 17 d.]. Prieiga per internetą: <https://en.wikipedia.org/wiki/Queueing_theory>

3. Adan I., Resing J. 2002. *Queueing Theory* [Masinio aptarnavimo teorija]: vadovėlis. The Netherlands: Eindhoven University of Technology. 180 p.

4. Самаров К. Л. 2009. *Элементы теории массового обслуживания* [Masinio aptarnavimo teorijos elementai]: vadovėlis. Prieiga per internetą: <http://www.resolventa.ru/data/metodstud/servtheory.pdf>

5. Java Modelling Tools – Users manual 2010. Politecnico di Milano [žiūrėta 2013 m. sausio 18 d.]. Prieiga per internetą: <http://jmt.sourceforge.net/Papers/JMT_users_Manual.pdf>

6. XML. 2013. Wikipedia [žiūrėta 2013 m. gegužės 17 d.]. Prieiga per internetą: <http://en.wikipedia.org/wiki/XML>

7. SimEvents Discrete-Event Simulation [interaktyvus]. 2013. The MathWorks, Inc [žiūrėta 2013 m. sausio 18 d.]. Prieiga per internetą: <http://www.mathworks.se/products/simevents/description2.html>

8. Simulink. Simulation and Model-Based Design. 2013. The MathWorks, Inc. [žiūrėta 2013 m. gegužės 17 d.]. Prieiga per internetą: <http://www.mathworks.se/products/simulink>

9. DELSI 2.0 API Reference [interaktyvus]. 2011. Holushko Software [žiūrėta 2013 m. sausio 18 d.]. Prieiga per internetą: <http://www.holushko.com/download/API_Reference.pdf>

10. .NET Framework 4.5. 2013. Microsoft [žiūrėta 2013 m. gegužės 17 d.]. Prieiga per internetą: <http://msdn.microsoft.com/en-us/library/w0x726c2.aspx>

11. DELSI 2.0 Tutorial [interaktyvus]. 2011. Holushko Software [žiūrėta 2013 m. sausio 18 d.]. Prieiga per internetą: <http://www.holushko.com/download/Tutorial.pdf>

12. Visual Studio 2010 Express Products. 2013. Microsoft [žiūrėta 2013 m. gegužės 17 d.]. Prieiga per internetą: <http://www.microsoft.com/visualstudio/eng/products/visual-studio-2010-express>

13. C# Language Specification [interaktyvus]. 2013. Microsoft [žiūrėta 2013 m. gegužės 17 d.]. Prieiga per internetą: <http://www.holushko.com/download/Tutorial.pdf>

14. .NET random number generators and distributions by Stefan Troschuetz [interaktyvus]. 2007. CodeProject, 1999-2013 [žiūrėta 2013 m. sausio 19 d.]. Prieiga per internetą: <http://www.codeproject.com/Articles/15102/NET-random-number-generators-and-distributions>

15. Random Class Library [interaktyvus]. 2007. Stefan Troschuetz [žiūrėta 2013 m. sausio 22 d.]. Prieiga per internetą: <http://www.codeproject.com/KB/recipes/Random/Random_doc.zip>

16. Коммутатор HP V1910-48G (JE009A) [interaktyvus]. 2013. «Системы ПК» [žiūrėta 2013 m. gegužės 17 d.]. Prieiga per internetą: <http://www.systemspc.ru/catalog/fullvend_20/sect_516/prod_1781/>

PRIEDAI

A priedas

Modeliavimo programinės įrangos branduolio kodas, parašytas C# programavimo kalba, naudojant *Delsi. 2.0* simuliacijų biblioteką.

```
//Inicializuojami visi programos komponentai

public Form1()
        {
            InitializeComponent();
        }
        //inicializuojami Delsi bibliotekos objektai, naudojami programoje

        private void Form1_Load(object sender, EventArgs e)
        {
            tModel1.Add(IncomingCalls); //Paraiškų generatorius
            tModel1.Add(CallingQueue);  //Laukimo eilė
            tModel1.Add(Reps);          //Aptarnavimo stotis
            tModel1.Add(EndOfService);  //Išėjimas aptarnautoms paraiškoms
            tModel1.Add(BusySignal);    //Išėjimas atmestoms paraiškoms
            tModel1.Add(HangedUp);      //Išėjimas eilę palikusioms paraiškoms
        }

        //Aprašomi veiksmai, atliekami programos, kai paspaudžiamas
        //simuliacijos paleidimo mygtukas

        private void button1_Click(object sender, EventArgs e)
        {
               //Šita komanda paleidžia simuliaciją (skliaustuose nurodytas simuliacijos
               //ilgis santykiniais vienetais)

                tModel1.Simulate(480.0);

             //Žemiau pateikti vieni iš galimų duomenų, kuriuos galima gauti po
             //simuliacijos vykdymo (funkcijos tModel1.Simulate įvykdymo)

             //Aptarnautų paraiškų skaičius
             EndOfService.Entries()

             //Atmestų paraiškų skaičius
             BusySignal.Entries()

             //Paraiškų, palikusių eilę dėl laukimo laiko pabaigos, skaičius
             HangedUp.Entries()

             //Vidutinis paraiškų eilės ilgis
             CallingQueue.AverageCount()

             //Vidutinis paraiškų laukimo laikas eilėje
             CallingQueue.AverageTime()

             //Aptarnavimo stoties užimtumas (procentais)
             Reps.Usage()

               //Šitoje programos kodo vietoje aprašoma visa logika ir veiksmai, daromi
               //su duomenimis, gautais simuliacijos metu (rezultatų skaičiavimas,
               //įnešimas į masyvus, tiriamų sistemos parametrų pasirinkimas, argumentų
               //pasirinkimas
```

```
    //Reset() funkcija perkrauna simuliaciją. Prieš darant naują simuliaciją,
    //reikia naudotis simuliacijos perkrovimo funkcija, kad nunulinti visus iki
    //šiol gautų rezultatų duomenis.

    tModel1.Reset();
}

//Žemiau pateikti objektai aprašo veiksmus, kuriuos atlieka su paraiškomis
//pagrindinė simuliacijos funkcija tModel1.Simulate, per nurodytą simuliacijos
//laiko trukmę

//Paraiškų generavimas (čia laikas tarp gretimų paraiškų atvykimo momentų
//generuojamas pagal eksponentinį pasiskirstymo dėsnį)

private void IncomingCalls_OnExit(TGenerator sender, Transaction transaction)
{
    sender.SetTime(tMultiRand1.Exponential(2.0));
}

//šita funkcija apdoroja paraiškas, joms išeinant iš generatoriaus

private void IncomingCalls_OnRouting(TGenerator sender, Transaction transaction)
{
    //Jeigu laukimo eilė neužpildyta - paraiška siunčiama į eilę

    if (CallingQueue.isReadyToGet(transaction))
    {
        sender.Send(CallingQueue);
    }

    //Jeigu laukimo eilė užpildyta - paraiška siunčiama į atitinkamą išėjimą
    else
    {
        sender.Send(BusySignal);
    }
}

//Laukimo laikas eilėje apibūdinamas tikimybinių pasiskirstymu (šiame pavyzdyje
//normaliuoju)

private void CallingQueue_OnEnter(TQueue sender, Transaction transaction)
{
    sender.SetTime(tMultiRand1.Normal(25.0, 6.0));
}

//Kai laukimo laikas eilėje baigėsi, o aptarnavimo stotis dar neatsilaisvino -
//paraiška siunčiama į atitinkamą išėjimą.

private void CallingQueue_AfterTimeEnded(TQueue sender, Transaction transaction)
{
    sender.Send(HangedUp);
}

//Kai aptarnavimo stotis atsilaisvina - paraiška siunčiama į ją

private void CallingQueue_OnRouting(TQueue sender, Transaction transaction)
{
    sender.Send(Reps);
}

//Būdama serveryje, paraiška aptarnaujama tam tikrą laiką (šiame pavyzdyje
//aptarnavimo laikas pasiskirstęs pagal normalųjį pasiskirstymo dėsnį)

private void Reps_OnEnter(TDelay sender, Transaction transaction)
{
```

```
    sender.SetTime(tMultiRand1.Normal(10.0, 3.0));
}

//Po aptarnavimo paraiška siunčiama į atitinkamą išėjimą

private void Reps_OnRouting(TDelay sender, Transaction transaction)
{
    sender.Send(EndOfService);
}
```

B priedas

Modeliavimo programinės įrangos kodo dalis, parašyta C# kalba, parodanti kaip realizuojamas skirtingų pasiskirstymo dėsnių pasirinkimas paraiškų atvykimo srautui, naudojant *Delsi 2.0* modeliavimų biblioteką ir Troschuetz atsitiktinių skaičių generavimo biblioteką.

```
//Programos grafinėje sąsajoje yra realizuotas įvairių paraiškų atvykimo srauto
//pasiskirstymo dėsnių sąrašas. Šis sąrašas pavadintas _atvyk_pasisk. Žemiau patekta kodo
//dalis parodo, kaip programoje realizuojamas skirtingų pasiskirstymų pasirinkimas,
//priklausomai nuo to, kokį objektą iš esamo sąrašo pasirinko vartotojas (visi objektai
//sunumeruoti nuo 0 (reguliarusis pasiskirstymas) iki 10 (binominis pasiskirstymas), iš
//viso programoje yra 11 galimų pasiskirstymo dėsnių)

private void IncomingCalls_OnExit(TGenerator sender, Transaction transaction)
{
            //Reguliarusis pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 0)
            {
                sender.SetTime(atv1);
            }

            //Eksponentinis pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 1)
            {
                sender.SetTime(tMultiRand1.Exponential(atv1));
            }

            //Normalusis pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 2)
            {
                sender.SetTime(tMultiRand1.Normal(atv1, atv2));
            }

            //Beta pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 3)
            {
                sender.SetTime(tMultiRand1.Beta(atv1, atv2, atv3, atv4));
            }

            //Gamma pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 4)
            {
                sender.SetTime(tMultiRand1.Gamma(atv1, atv2));
            }

            //Lognormalusis pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 5)
            {
                sender.SetTime(tMultiRand1.Lognormal(atv1, atv2));
            }

            //Trikampinis pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 6)
            {
                sender.SetTime(tMultiRand1.Triangular(atv3, atv4, atv1
```

```
            }
            //Tolydusis pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 7)
            {
                sender.SetTime(tMultiRand1.Uniform(atv3, atv4
            }

            //Puasono pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 8)
            {
                n1.Lambda = atv1;
                sender.SetTime(n1.NextDouble());
            }

            //Diskretusis pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 9)
            {
                int mmm = Convert.ToInt32(atv3);
                int mmmm = Convert.ToInt32(atv4);
                n4.Beta = mmmm;
                n4.Alpha = mmm;
                sender.SetTime(n4.NextDouble());
            }

            //Binominis pasiskirstymas

            if (_atvyk_pasisk.SelectedIndex == 10)
            {
                int mmm = Convert.ToInt32(atv1);
                n7.Beta = mmm;
                n7.Alpha = atv2;
                sender.SetTime(n7.NextDouble());
            }
}
```

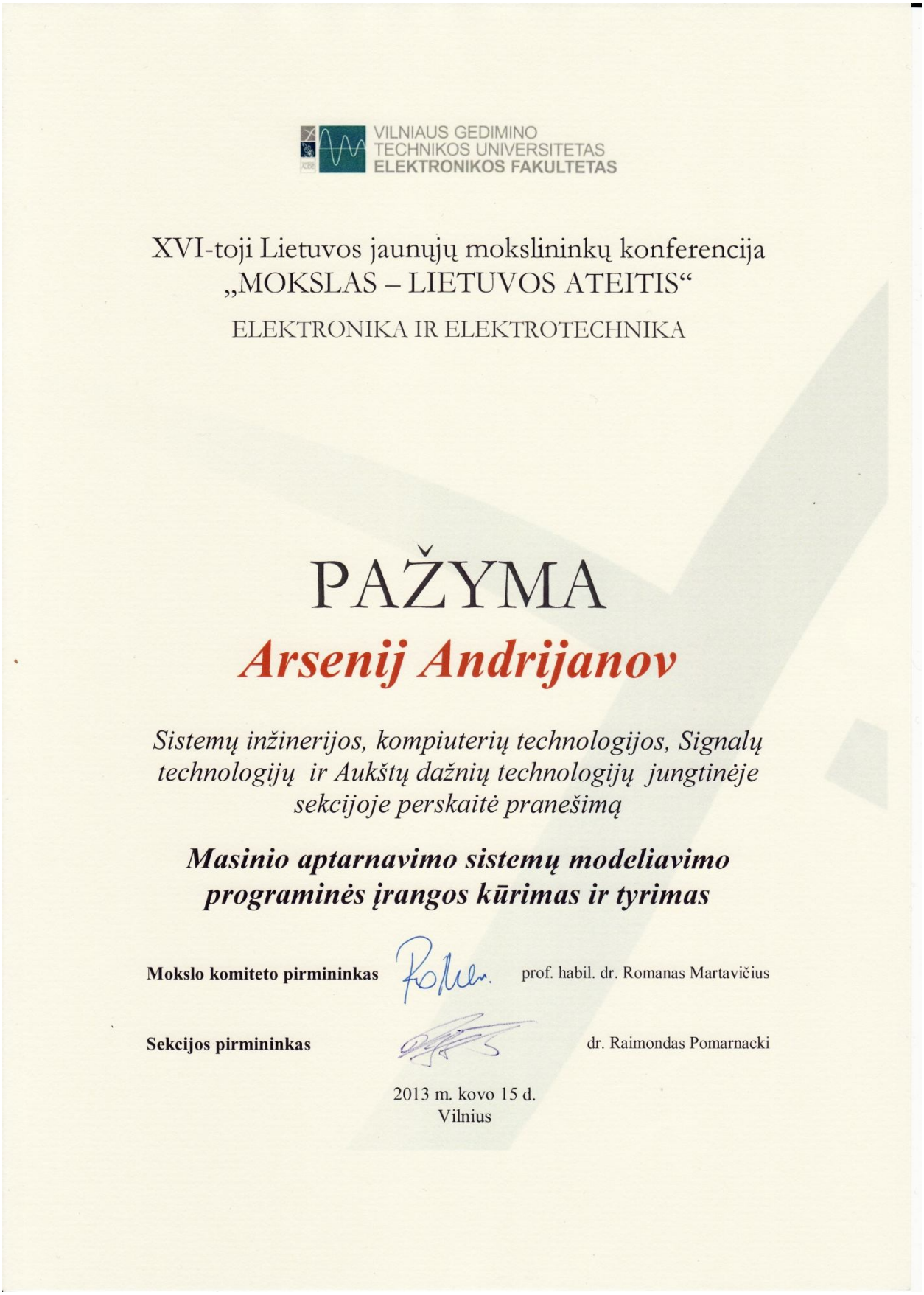
C priedas

Modeliavimo programos kodo pavyzdys, parašytas C# kalba, parodantis kaip realizuojamas grafikų braižymas, naudojantis simuliacijos metu gautais duomenimis.

```
//Naujo grafiko kūrimas

chart1.Series.Add(a);

//Nustatoma sritis, kur bus braižomas naujas grafikas

chart1.Series[a].ChartArea = "ChartArea1";

//Įjungiama grafiko legenda

chart1.Series[a].IsVisibleInLegend = true;

//Nustatomas grafiko tipas (taškai, linija, „spline“ ir t. t.)

chart1.Series[a].ChartType = SeriesChartType.Point;

//Sukuriama nauja legenda, kur bus duomenys

chart1.Legends.Add(new Legend(a));

//Legendos teksto šrifto nustatymai

chart1.Legends[a].Font = new Font("Arial", 10);

//X ašies pavadinimas

chart1.ChartAreas["ChartArea1"].AxisX.Title = "Alfa (atvykimas)";

//Y ašies pavadinimas

chart1.ChartAreas["ChartArea1"].AxisY.Title = str1;

//Parašomas legendos tekstas

chart1.Series[a].LegendText = "Laukimas" + b;

//Grafiko taškų didumas (arba linijos storis)

chart1.Series[a].MarkerSize = 8;

//Žemiau esantis ciklas realizuoja paties grafiko braižymą. Priklausomai nuo turimų
//duomenų kiekio „n“, vienas po kito braižomi grafiko taškai. Šiame pavyzdyje braižoma
//vidutinio paraiškų laukimo laiko eilėje priklausomybė nuo sistemos efektyvumo
//koeficiento α

for (int i = 0; i < n; i++)
{
chart1.Series[a].Points.AddXY(_alf[i], _laukimas[i]);
}
```

D priedas

Pranešimo 16-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga



VILNIAUS GEDIMINO
TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS

XVI-toji Lietuvos jaunųjų mokslininkų konferencija
„MOKSLAS – LIETUVOS ATEITIS“

ELEKTRONIKA IR ELEKTROTECHNIKA

PAŽYMA

Arsenij Andrijanov

Sistemų inžinerijos, kompiuterių technologijos, Signalų technologijų ir Aukštų dažnių technologijų jungtinėje sekcijoje perskaitė pranešimą

Masinio aptarnavimo sistemų modeliavimo programinės įrangos kūrimas ir tyrimas

Mokslo komiteto pirmininkas prof. habil. dr. Romanas Martavičius

Sekcijos pirmininkas dr. Raimondas Pomarnacki

2013 m. kovo 15 d.
Vilnius