

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Sąsūkos (konvoliucinis) kodas ir tiesioginis dekodavimas

Ataskaita

4 kurso 5 grupės studentas

Rokas Jaruševičius

Vilnius - 2018

TURINYS

1. PROGRAMOS SPECIFIKACIJA	2
2. VARTOTOJO SĄSAJA.....	3
3. KLAIDAS TAISANTIS KODAS	6
4. PROGRAMINIAI SPRENDIMAI	7
4.1. Renkamės greitį, ne atmintį	7
4.2. Bitų iškraipymas kanale	8
4.3. MemoryOperations pagalbinė klasė	8
4.4. Esminės ir pagrindinės klasės	8
5. ATLIKTI EKSPERIMENTAI	9
5.1. Neištaisytų klaidų kiekio priklausomybė nuo klaidos kanale tikimybės	9
5.2. Siunčiamo failo dydžio priklausomybė nuo siuntimo laiko	10
5.3. Operacinės sistemos (windows) geba atpažinti iškraipyto failo formatą.....	11
6. TYRIMŲ IŠVADOS.....	12
ŠALTINIAI	

1. PROGRAMOS SPECIFIKACIJA

Programa parašyta windows aplinkoje, naudojant .NET technologijas ir C# programavimo kalbą.

Programos paleidimas: *CodingTheory/Release/CodingTheory.exe*

Programos klasė: *CodingTheory/Program.cs*

Esminės klasės:

CodingTheory/Core/...

- *.../ConvolutionalEncoder.cs* - sasujokos (konvoliucinis) enkoderis
- *.../DirectDecoder.cs* - tiesioginis dekoderis
- *.../NoisyChannel.cs* - nepatikimas kanalas

Pagalbinės klasės:

CodingTheory/Utilities/...

- *.../ArrayModifier.cs* - masyvų modifikavimui
- *.../BitPrinter.cs* - bool reikšmių spausdinimui 1 ir 0
- *.../CollectionConverter.cs* - BitArray ir bool[] transformacijos
- *.../FileComparer.cs* - failų lyginimo operacijos
- *.../FileConverter.cs* - binary to file ir atvirkščiai
- *.../LogicGate.cs* - XOR ir MDE operacijos
- *.../MemoryOperations* - bitų pastūmimui atmintyje
- *.../RandomGenerator.cs* - atsitiktinio skaičiaus generatorius

Failai, su kuriais buvo atlikti eksperimentai*:

CodingTheory/Input/...

- *.../canada-usa border.png (53.8 KB)*
- *.../lorem.txt (3.31 KB)*
- *.../square.bmp (117 KB)*
- *.../spengla.png (571 KB)*

Iš kanalo gautas failas, kuris kanalu siųstas koduojant:

CodingTheory/Release/receivedFileWithEncoding.ext

Iš kanalo gautas failas, kuris kanalu siųstas nekoduojant:

CodingTheory/Release/receivedFileWithoutEncoding.ext

* Eksperimentai buvo atlikti unit-testų pagalba (nenaudojant vartotojo sąsajos), todėl naudoti ir .png failai, nors programa (vykdant per vartotojo sąsają) priima tik .txt bei .bmp failų formatus.

2. VARTOTOJO SĄSAJA

Paleidus programą, matome iššokusį pagrindinį langą (1 pav.) su pasirinkimais:

- 1 – esamos klaidos tikimybės peržiūra ar pakeitimas nauja
- 2 – dvejetainės žinutės siuntimas nepatikimu kanalu
- 3 – .txt arba .bmp failo siuntimas nepatikimu kanalu
- cd.. – grįžimas į ankstesnį langą

Į konsolę įrašome vieną iš pasirinkimų (pvz.: 1) ir spaudžiame klavišą *Enter*. Programa nuolatos prašys įvesti naują pasirinkimą, kol neatpažins vieno iš galimų išvardintų variantų. Klaidos tikimybės dešimtainė dalis atskiriama tašku, ne kableliu (pvz.: 0.005, bet ne 0,005).

```
#####
#-----#
#                               #
#   Convolutional encoder & Threshold decoder   #
#                               #
#-----#
#####

1   - Adjust error probability
2   - Send binary message
3   - Send .txt or .bmp
cd.. - Return

>>> _
```

1 pav. Pagrindinis programos langas

Pasirinkę dvejetainės žinutės siuntimą nepatikimu kanalu (įvedę 2), matome ekrane išvestą tarpinį rezultatą (2 pav., „*Tarpinių rezultatų blokas*“), kurį sudaro:

- Užkoduotų bit'ų seka
- Persiųstų bit'ų seka
- Klaidų, padarytų siunčiant nepatikimu kanalu, kiekis

```
Enter new binary message:
>>> 11110000

Encoded bits:      111110100100000000001000000000
Transmitted bits:  111110100100000000001000000000
Errors occurred:   0

Enter D to decode message, enter U to update first, then decode:
>>> U
Enter new updated message:
>>> 11111010010000000000100001100

Input message:      11110000
Received message: 11110000
Error probability 0
Errors left unfixed: 0
File transmission time: 15460ms
```

Tarpinių rezultatų blokas

Galutinių rezultatų blokas

2 pav. Dvejetainės žinutės siuntimas nepatikimu kanalu

Toliau matome (2 pav.) pasirinkimą tarp įvesties „D“ – dekoduoti iškart ir „U“ – pakoreguoti seką ir tik tuomet dekoduoti. Įvedus pasirinkimą „U“, vartotojas paprašomas įvesti naują, pakoreguotą bit'ų seką, kuri toliau bus dekoduojama.

Galiausiai, į ekraną išvedamas galutinis rezultatas (2 pav., „Galutinių rezultatų blokas“), kurį sudaro:

- Įvestis
- Išvestis
- Klaidos tikimybė
- Neištaisytų klaidų kiekis
- Siuntimo laikas*

Pasirinkus 3 variantą (.txt ar .bmp failo siuntimą kanalu) iš pagrindinio lango (1 pav.), vartotojas paprašomas įvesti pilną kelią iki failo, kurį norima siųsti kanalu (pvz.: „C:\Users\ManoKompiuteris\Desktop\failas.bmp“ be kabučių). Įvedus teisingą kelią, į konsolę išvedami siuntimo kanalu rodmenys ir rezultatai:

- Klaidos tikimybė
- Neištaisytų klaidų kiekis
- Siuntimo laikas
- Failo dydis (baitais)

* 0 ms reiškia, kad siuntimas įvyko taip greitai, kad neverta tikslios ms dalies minėti vartotojui.

Po šios išvesties, programa pasiūlo 2 pasirinkimus (3 pav.):

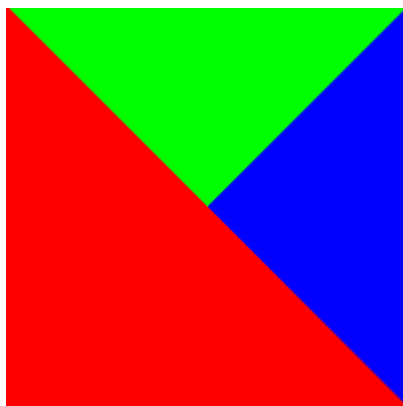
```
Type 0 to open 2 files received from channel or type cd.. to return
File 1: /receivedFileWithoutEncoding.bmp
File 2: /receivedFileWithEncoding.bmp
>>>
```

3 pav. Užklausa, matoma ekrane, persiuntus failą nepatikimu kanalu

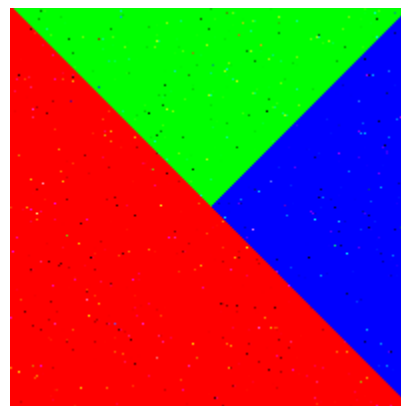
- Įvesti „0“ – kas reiškia – atidaryti du failus (failą, gautą siunčiant užkoduotą bit'ų seką ir failą, gautą siunčiant bit'us be kodavimo)
- Įvesti „cd..“ – kas grąžintų į pradinį pasirinkimų meniu

Pasirinkus variantą „0“, kiekvienam iš atidarytų failų, galimi šie variantai:

- Atvaizduojamas identiškas failas duotajam (4 pav.)
- Atvaizduojamas iškraipytas failas (5 pav.)
- Atvaizduojamas operacinės sistemos (windows) pranešimas, kad failo formatas neatpažintas* (6 pav.)



4 pav. Siunčiamas failas arba iš kanalo gautas identiškas failas duotajam



5 pav. Iškraipytas failas su 0.005 klaidos tikimybe, padarius 2255 klaidų 120054 baitų faile

```
Error probability 0.05
Errors left unfixed: 128731
File transmission time: 775ms
File size: 120054 bytes
```

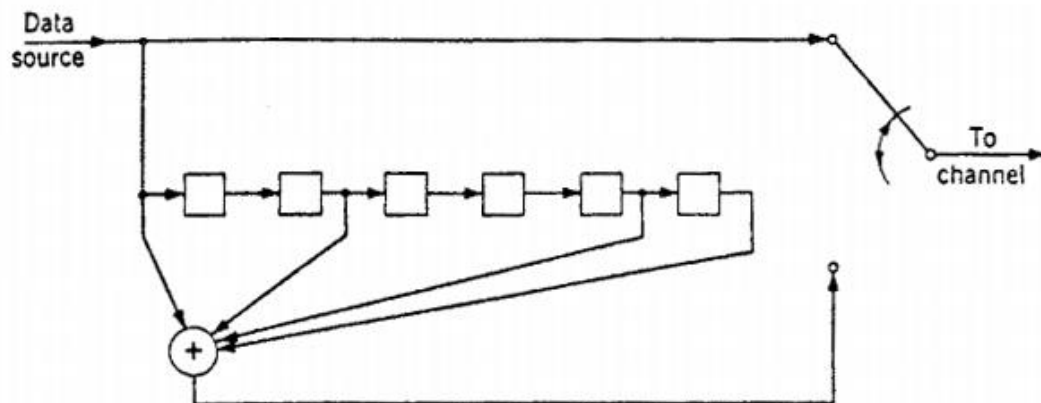
It looks like we don't support this file format.

6 pav. Operacinės sistemos (windows) negebėjimas atpažinti iškraipyto failo formato.

* Gilesnė analizė pateikta 5.3 skyrelyje.

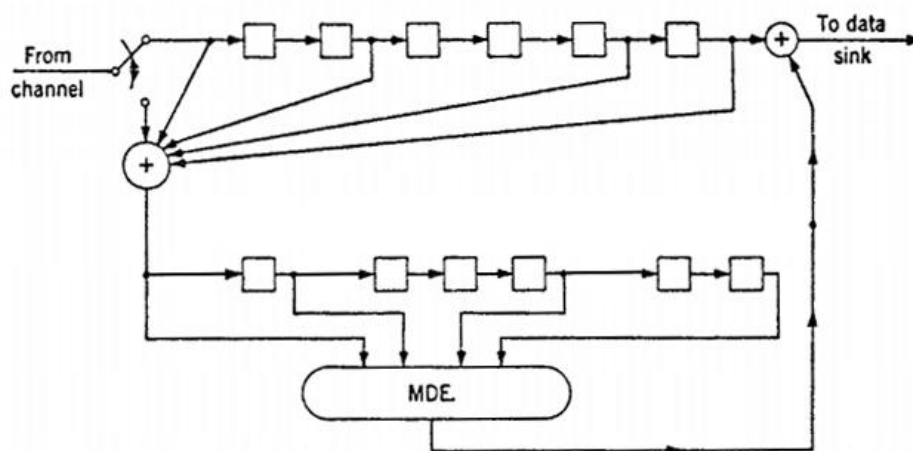
3. KLAIDAS TAISANTIS KODAS

Septintame paveikslėlyje (7 pav.) pavaizduota sąsūkos (konvoliucinio) kodo schema. Įeitis 1 bitas – išeitis 2 bitų pora. Pirmas bitas toks pat kaip įjėš bitas, antras (sindrominis) bitas gaunamas atlikus XOR¹ operaciją. Visiems duomenų bitams patekus į kodavimo schemą, papildomai į atmintį įstumiami dar 6-i 0-niai bitai.



7 pav. Sąsūkos (konvoliucinis) kodas

Aštuntame paveikslėlyje (8 pav.) matome (7 pav.) pavaizduoto kodo dekodavimo schemą. Įeitis – 2 bitų pora, išeitis – 1 bitas. Pirmasis poros bitas keliauja per schemos viršų, antrasis nueina į apačią. Po XOR¹ ir MDE² loginių operacijų, gaunamas sindrominis bitas, su kuriuo atliekama dar viena XOR¹ operacija ir gaunamas 1 bitas iš 2 bitų poros. Išėję pirmi 6 bitai neturi reikšmės. Tai ne duomenų bitai, o dekoderio būsenos bitai, tad į juos nekreipiame dėmesio.



8 pav. Tiesioginis dekodavimas

¹ XOR – jei įeinančių 1-etinių bitų kiekis nelyginis, grąžinamas 1-etinis bitas.

² MDE – įeitis – 4 bitai. Jei 2 ir daugiau bitų yra 1-etiniai bitai, grąžinamas 1-etinis bitas.

4. PROGRAMINIAI SPRENDIMAI

4.1. Renkamės greitį, ne atmintį

Darbai su bitais pasirinktas bool tipo masyvas – atsisakyta naudoti sisteminę BitArray kolekciją. Priežastis – greitesnis masyvo elementų pasiekimas, nors atminties atžvilgiu – poreikis didesnis. BitArray išsaugo 8 elementus 1 baite. Tuo tarpu bool masyvas išsaugo tik 1 elementą 1 baite. Tačiau, atlikus eksperimentą su 2 mln. BitArray ir bool array elementų, pakartojant po 3 kartus ir išvedant visų elementų pasiekimo greičio vidurkį (9 pav.), matome rezultatus (10 pav.), kad bool array efektyvesnis vidutiniškai 3,5 karto už BitArray. Tad renkamės greitį, ne atmintį.

```
const int limit = 2000000;
const int iterations = 3;

var bitArray = new BitArray(limit);
var boolArray = new bool[limit];

long bitArrayAccessTime = 0;
long boolArrayAccessTime = 0;

for (int j = 0; j < iterations; j++)
{
    watch = Stopwatch.StartNew();
    for (int i = 0; i < limit; i++)
        bitArray[i] = !bitArray[i];

    watch.Stop();
    bitArrayAccessTime += watch.ElapsedMilliseconds;
}

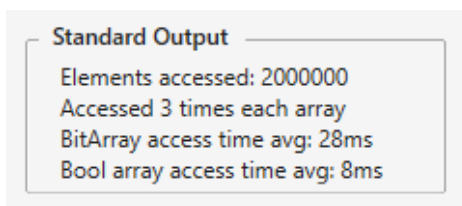
for (int j = 0; j < iterations; j++)
{
    watch = Stopwatch.StartNew();
    for (int i = 0; i < limit; i++)
        boolArray[i] = !boolArray[i];

    watch.Stop();
    boolArrayAccessTime += watch.ElapsedMilliseconds;
}

Console.WriteLine("Elements accessed: " + limit);
Console.WriteLine("Accessed " + iterations + " times each array");

Console.WriteLine("BitArray access time avg: " + (bitArrayAccessTime/iterations) + "ms");
Console.WriteLine("Bool array access time avg: " + (boolArrayAccessTime / iterations) + "ms");
```

9 pav. Kodas, skirtas BitArray ir bool array elementų pasiekimo greičiams testuoti



Standard Output

```
Elements accessed: 2000000
Accessed 3 times each array
BitArray access time avg: 28ms
Bool array access time avg: 8ms
```

10 pav. BitArray ir bool array elementų pasiekimo greičiai

4.2. Bitų iškraipymas kanale

Algoritmas paprastas. Turime klaidos tikimybę p , priklausančią intervalui $[0; 1]$. Generuojame atsitiktinį realųjį skaičių a , kuris priklauso intervalui $(0; 1)$. Jeigu $a < p$, bitą esantį kanale – iškraipome. Kadangi dirbame tik su dvejomis reikšmėmis (0 ir 1), tai iškreipiant 0, jis virsta 1 ir atvirkščiai.

4.3. MemoryOperations pagalbinė klasė

Kadangi tiek sąsūkos (konvoliucinis) enkoderis, tiek tiesioginis dekoderis dirba su 6 blokų atmintimi (konstanta), nuspręsta atsikartojantį bit'ų įstūmimo į atmintį metodą iškelti į naują pagalbinę klasę, kurią naudos tiek enkoderis, tiek dekoderis.

Pagalbinė klasė randama: *CodingTheory/Utilities/MemoryOperations.cs*

4.4. Esminės ir pagrindinės klasės

Pagrindinės klasės – objektinės klasės, kadangi saugo savyje tam tikras būsenas. Randamos: *CodingTheory/Core/...* Jas sudaro: enkoderis, dekoderis ir nepatikimas kanalas.

Pagalbinės klasės – procedūrinės klasės, t.y. kalbant C# kalba, *static* klasės. Norint vykdyti šių klasių metodus, objektų kurti nereikia. Skirtos failų konvertavimui, darbui su masyvų elementais, atsitiktinio skaičiaus generavimui ir kt. Randamos: *CodingTheory/Utilities/...*

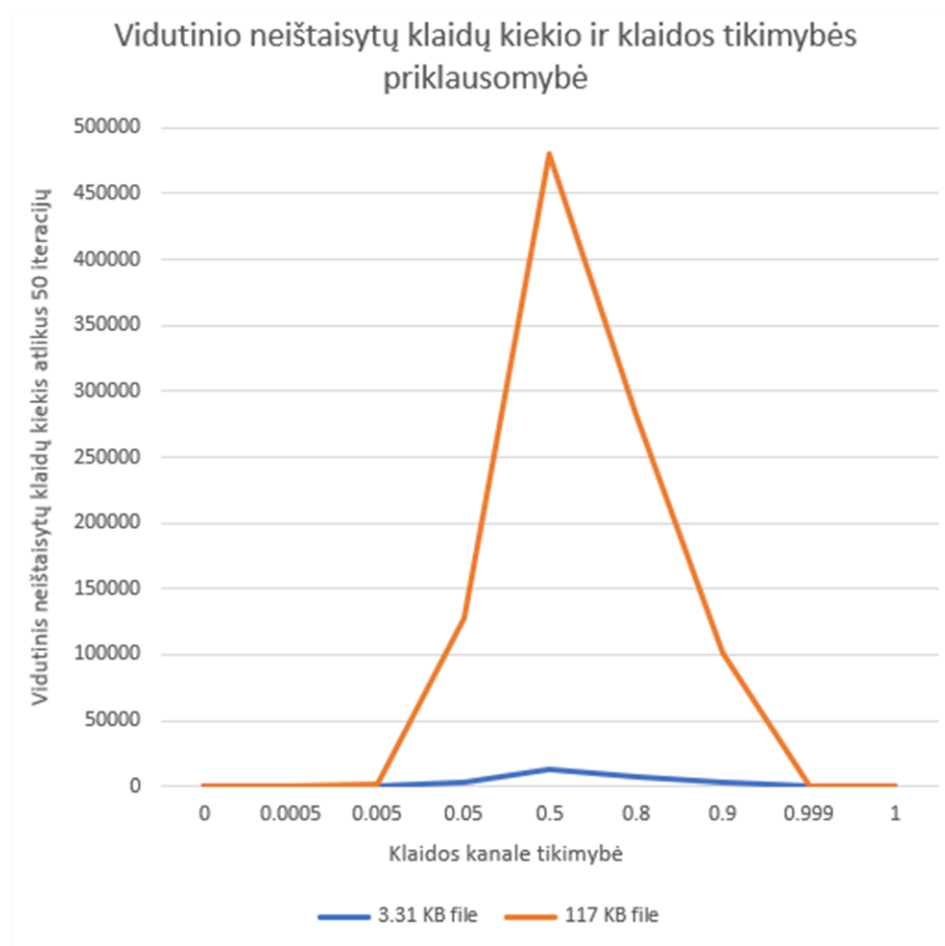
5. ATLIKTI EKSPERIMENTAI

5.1. Neištaisyčių klaidų kiekio priklausomybė nuo klaidos kanale tikimybės

Buvo atliktas eksperimentas su dviejų dydžių failais (3.31 KB ir 117 KB), siekiant išsiaiškinti, kaip kinta neištaisyčių klaidų kiekis, klaidos tikimybe varijuojant intervale (0; 1). Su kiekviena įvestimi (klaidos tikimybe, failas) buvo atlikta po 50 iteracijų ir paimtas neištaisyčių klaidų vidurkis. Rezultatai matomi 1 lentelėje (1 lentelė). Grafiškai duomenys atvaizduoti 11 paveikslėlyje (11 pav.).

1 lentelė. Vidutinio neištaisyčių klaidų kiekio ir klaidos tikimybės priklausomybė

Klaidos kanale tikimybė		0	0.0005	0.005	0.05	0.5	0.8	0.9	0.999	1
Vidutinis neištaisyčių klaidų kiekis atlikus 50 iteracijų	3.31 KB file	0	0	64	3607	13586	8010	2868	0	0
	117 KB file	0	23	2160	128086	480162	282811	101830	0	0



11 pav. Vidutinio neištaisyty klaidų kiekio ir klaidos tikimybės priklausomybės grafinis atvaizdavimas.

5.2. Siunčiamo failo dydžio priklausomybė nuo siuntimo laiko

Rasta failo siuntimo kanalu trukmės priklausomybė nuo siunčiamo failo dydžio. Failo siuntimą kanalu sudaro *užkodavimas*, *perdavimas nepatikimu kanalu* su tikimybe p padaryti klaidą kiekviename bite ir išėjusios bitų sekos iš kanalo *dekodavimas*. Įvestis – 4 skirtingų dydžių failai (3.31 KB, 53.8 KB, 117 KB, 571 KB). Buvo atlikta po 50 iteracijų kiekvienam failo dydžiui ir paimtas siuntimo laiko vidurkis. Rezultatai milisekundėmis matomi 12 paveikslėlyje (12 pav.).



12 pav. Siunčiamo failo dydžio priklausomybės nuo siuntimo kanalu laiko vidurkio grafikas.

5.3. Operacinės sistemos (windows) geba atpažinti iškraipyto failo formatą

Ankščiau naudotų failų formatai: 3.31 KB – tekstinis .txt, 117 KB – paveikslėlis .bmp. Buvo atliktas eksperimentas, siekiant suprasti, kiek vidutiniškai reikia padaryti klaidų siunčiamajame faile, kad operacinė sistema (windows) nebeatpažintų gauto failo formato. Vidutinių neištaisytų klaidų kiekių duomenys paimti iš 1 lentelės (1 lentelė). Eksperimento rezultatai pateikti 2 lentelėje (2 lentelė).

2 lentelė. Vidutiniškai neištaisytų klaidų kiekis ir failų (.txt, .bmp) atpažįstamumas operacinės sistemos (windows).

Vidutinis neištaisytų klaidų kiekis atlikus 50 iteracijų	<i>3.31 KB file (.txt)</i>	0	0	64	3607	13586	8010	2868	0	0
	<i>117 KB file (.bmp)</i>	0	23	2160	128086	480162	282811	101830	0	0
Ar operacinė sistema (windows) atpažįsta iškraipyto failo formatą?	<i>3.31 KB file (.txt)</i>	T								
	<i>117 KB file (.bmp)</i>	T	T	T	N	N	N	N	T	T

6. TYRIMŲ IŠVADOS

- Nesvarbu kiek bus padaryta klaidų siunčiant tekstinį (.txt) failą nepatikimu kanalu – jį operacinė sistema (windows) visada atpažins kaip tekstinį failą, tik pateiks iškraipytą turinį.
- Paveikslėlio formato (.bmp) operacinė sistema (windows) nebeatpažįsta, padarius klaidų kiekį N , esantį intervale $(2160; 101830)$. Tiksliai nustatyti nepavyko. Didžiausias pasiektas tikslumas, remiantis tyrimais, rodė, kad iki 2160 klaidų, failas atpažįstamas kaip .bmp, o nuo 101830 klaidų kiekio operacinė sistema (windows) nurodo, kad failo formatas nepalaikomas.
- Iš nepatikimo kanalo gavus pilnai iškraipytą bitų seką (klaidos tikimybė 1, arba arti 1), tiesioginio dekodavimo algoritmas pilnai ištaiso visas klaidas.
- Neištaisytų klaidų kiekis didėja klaidos tikimybės intervale $(0; 0,5)$, o mažėti pradeda intervale $(0,5; 1)$.
- Siunčiamo failo dydžio ir siuntimo kanalu laiko vidurkio santykis svyruoja intervale $[7,8; 9,17]$. Apytiksliai 1 KB persiunčiamas per $\sim 8,6$ ms. Didinant imtį, rezultatas galimai kistų (taptų tikslesnis).

ŠALTINIAI

- <https://klevas.mif.vu.lt/~skersys/doc/ktkt/literatura23.pdf>
- <https://docs.microsoft.com/en-us/dotnet/api/system.collections.bitarray?redirectedfrom=MSDN&view=netframework-4.7.2>