Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

ОТЧЁТ

По лабораторной работе № 4

На тему

**СЛУЧАЙНЫЕ МЕТОДЫ ДОСТУПА К МОНОКАНАЛУ**

Выполнил: Проверил:

Студент группы 050503 Преподаватель

Деруго Д. В. Марцинкевич В. А.

Минск 2022

1. **Цель работы:**

Изучить методы доступа к каналам СрПД в компьютерных сетях. Разработать программу, реализующую процедуру случайного доступа к моноканалу по алгоритму CSMA/CD.

1. **Задание к лабораторной работе:**

Написать программу передачи данных через COM-порты упрощенным алгоритмом CSMA/CD в соответствии с требованиями.

**3. Теоретическая часть**

Зачастую в компьютерных сетях станции разделяют единый ресурс — общий канал передачи данных, моноканал. При «столкновении» данных из разных передатчиков возникают коллизии в связи с физической реализацией сети. Стоит задача организовать работу станций таким образом, чтобы станции не мешали друг другу, редко попадая в трудноразрешимые ситуации, или же ослабить контроль за порядком передачи данных, что часто ведет к конфликтам, которые, однако, легко разрешаются. Случайные методы доступа к моноканалу делают ставку на второй вариант, лучше всего такие методы работают с шинной топологией сети. Благодаря такой структуре станция-передатчик получит информацию о состоянии шины сразу же после собственной передачи, и если другая станция тоже передавала в этот момент данные, то после сравнения только что переданных данных и полученных с шины обе станции поймут, что мешают друг другу. Тогда станции используют генератор случайных чисел, в результате работы которого выбирается случайная задержка во времени до следующей попытки передачи, причем с количеством попыток интервал времени увеличивается до определенного предела. Таким образом, когда-нибудь станции получат такие значения задержек, чтобы друг другу не помешать. Вместе с этим станции-передатчики посылают jam-сигнал, оповещающий станции-приемники о коллизии и сигнализирующий приемникам о начале отсчета случайной задержки.

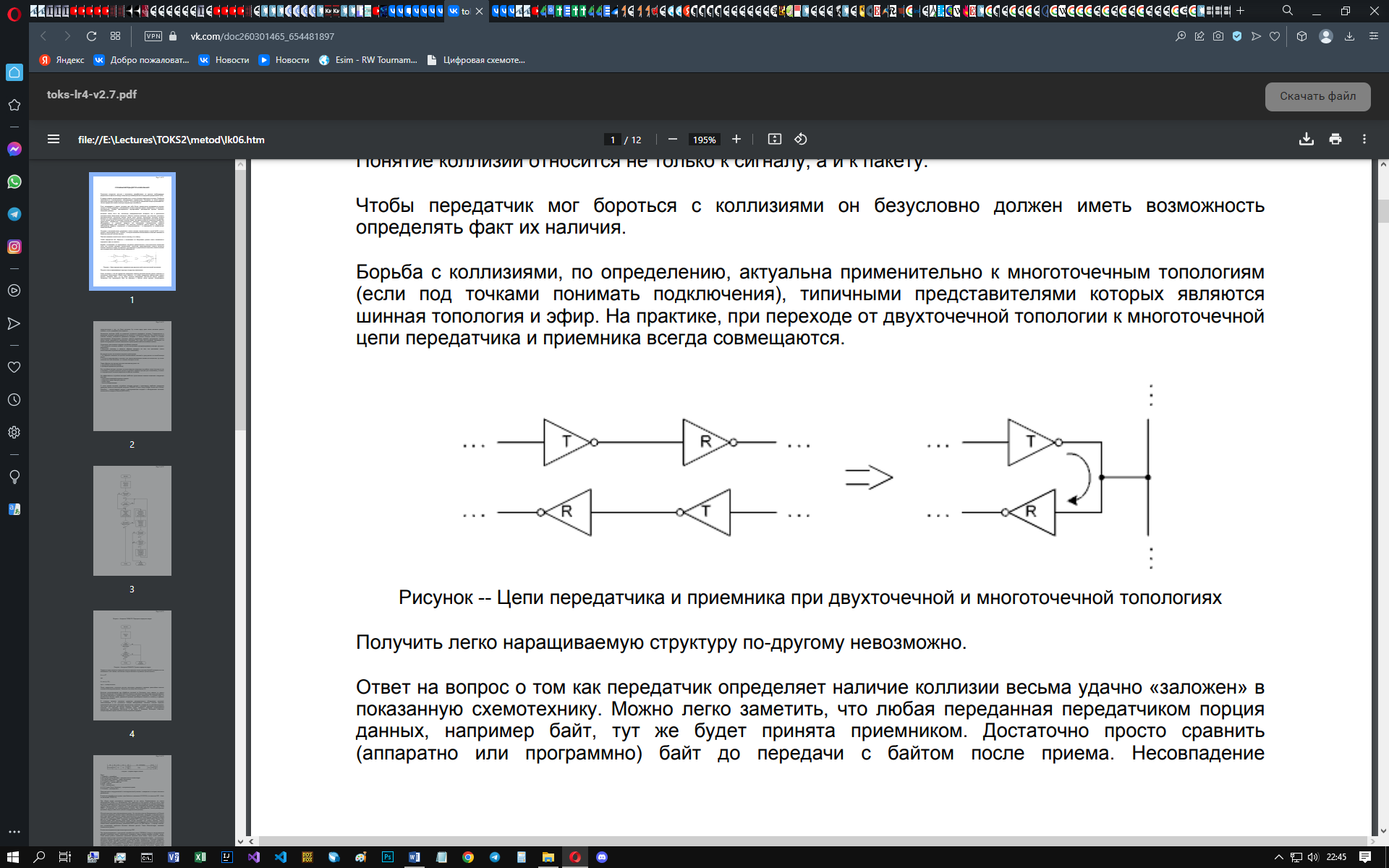


Рис. 1 – Двухточечная (слева) и шинная (справа) топологии

Программа случайным образом генерирует сообщение, кодирует его и отправляет в канал, при этом с шансом в 50% вносит ошибки в кадр. При обнаружении ошибки посылается jam-сигнал и выбирается случайная задержка, инкрементируется число попыток передачи.

1. **Практическая часть**

Writer-подпрограмма:

import serial

import time

import random

clear\_console = "\033[2J\033[1;1H"

RED\_COLOR = "\x1b[0;31m"

WHITE\_COLOR = "\x1b[0m"

GREEN\_COLOR = "\x1b[0;32m"

def encode\_hamming(hamming\_encoded, r):

n = len(hamming\_encoded)

for i in range(r):

count = 0

for j in range(2 \*\* i - 1, n, (2 \*\* i) \* 2):

# Check all bits in

for k in range(j, j + 2 \*\* i):

if len(hamming\_encoded) > k and hamming\_encoded[k] == '1':

count += 1

hamming\_encoded = hamming\_encoded[:2 \*\* i - 1] + str(count % 2) + hamming\_encoded[2 \*\* i:]

return hamming\_encoded

def detect\_errors(hamming\_encoded):

c = (len(hamming\_encoded) - 1).bit\_length()

n = len(hamming\_encoded)

wrong\_bits = []

for i in range(c):

count = 0

for j in range(2 \*\* i - 1, n, (2 \*\* i) \* 2):

# Check all bits in

for k in range(j, j + 2 \*\* i):

if len(hamming\_encoded) > k != 2 \*\* i - 1 and hamming\_encoded[k] == '1':

count += 1

if hamming\_encoded[2 \*\* i - 1] != str(count % 2):

wrong\_bits.append(2 \*\* i)

if len(wrong\_bits) != 0:

wrong\_bit = sum(wrong\_bits)

hamming\_encoded = hamming\_encoded[:wrong\_bit - 1] + str(

int(not int(hamming\_encoded[wrong\_bit - 1]))) + hamming\_encoded[wrong\_bit:]

return hamming\_encoded

def control\_bits\_count(m):

for i in range(m):

if 2 \*\* i >= m + i + 1:

return i

return 0

def find\_bits\_position(data, r):

if r == 0:

return data

j, k, m, res = (0, 0, len(data), '')

for i in range(1, m + r + 1):

if i == 2 \*\* j:

res = res + 'X'

j += 1

else:

res += data[k]

k += 1

return res

def decode\_hamming(hamming\_encoded):

c = (len(hamming\_encoded) - 1).bit\_length()

for i in range(c, -1, -1):

if len(hamming\_encoded) > 2 \*\* i:

hamming\_encoded = hamming\_encoded[:2 \*\* i - 1] + hamming\_encoded[2 \*\* i:]

return hamming\_encoded

def encode\_message(data):

c = control\_bits\_count(len(data))

data\_found\_pos = find\_bits\_position(data, c)

data\_encoded = encode\_hamming(data\_found\_pos, c)

return data\_encoded

def randomize\_data(data\_length):

data = ""

for i in range(data\_length):

data += str(random.randint(0, 1))

return data

def writer():

ser = serial.Serial('/dev/tnt0', timeout=None)

data\_length = 8

jam\_signal = "111111111111"

print(clear\_console, end="")

while True:

choice = input(WHITE\_COLOR + "Enter the choice: \n1 - generate data \nq - quit \n>")

if choice == "1":

print(clear\_console, end="")

randomized\_data = randomize\_data(data\_length)

encoded\_data = encode\_message(randomized\_data)

print(f"Generated data: {randomized\_data}")

print(f"Encoded data: {encoded\_data}")

attempts\_amount = 0

while attempts\_amount < 10:

if random.randint(0, 1) == 1:

final\_data = randomize\_data(data\_length + control\_bits\_count(len(randomized\_data)))

print(RED\_COLOR + f"Wrong data: {final\_data}")

else:

final\_data = encoded\_data

print(WHITE\_COLOR + f"Final data: {final\_data}")

ser.write(final\_data.encode("windows-1251"))

time.sleep(0.1)

if final\_data != encoded\_data:

print(GREEN\_COLOR + "Collision detected, sending jam-signal")

attempts\_amount += 1

# jam-signal

ser.write(jam\_signal.encode("windows-1251"))

wait = random.randint(1, 2\*\*attempts\_amount)

print(f"Wait for {wait} seconds")

time.sleep(wait)

else:

break

if choice == "q":

ser.close()

break

if \_\_name\_\_ == '\_\_main\_\_':

writer()

Reader-подпрограмма:

import serial

clear\_console = "\033[2J\033[1;1H"

RED\_COLOR = "\x1b[0;31m"

WHITE\_COLOR = "\x1b[0m"

GREEN\_COLOR = "\x1b[0;32m"

def reader():

ser = serial.Serial('/dev/tnt1', timeout=None)

correct\_messages = 0

while True:

data = ""

i = 0

while ser.inWaiting():

data = data + ser.read().decode("windows-1251")

i += 1

if i == 12:

if data == "111111111111":

print(RED\_COLOR + f"Jam-signal received")

correct\_messages -= 1

else:

print(WHITE\_COLOR + f"Message: {data}, correct = {correct\_messages + 1}")

correct\_messages += 1

if \_\_name\_\_ == '\_\_main\_\_':

reader()