Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Основы компьютерных сетей

ОТЧЕТ

по лабораторной работе №4

на тему

СЛУЧАЙНЫЕ МЕТОДЫ ДОСТУПА К МОНОКАНАЛУ

Студент Т.Ю. Петрович

Преподаватель В.А. Марцинкевич

МИНСК 2024

**1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

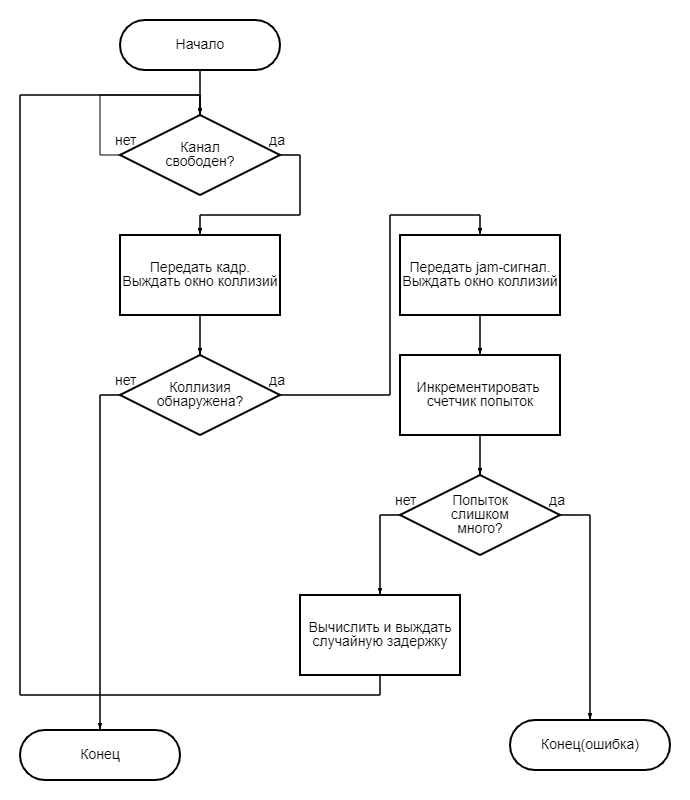


Рисунок 1.1 – Упрощенный алгоритм CSMA/CD на стороне передатчика

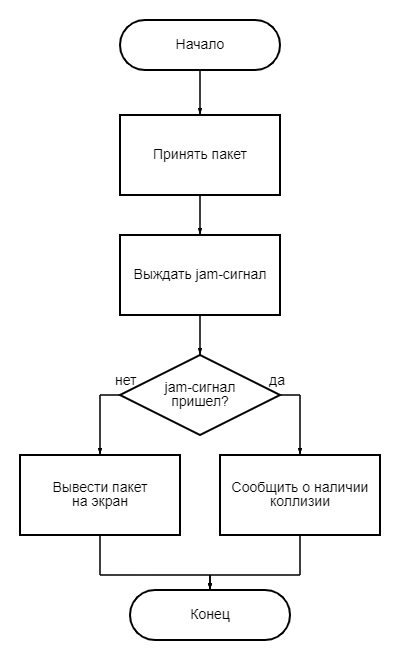


Рисунок 1.2 – Упрощенный алгоритм CSMA/CD на стороне приемника

**2 ПРАКТИЧЕСКАЯ ЧАСТЬ**

Содержимое файла main.py:

import serial

import threading

import time

from tkinter import Tk, Label, Button, Entry, StringVar, Text, Scrollbar, END, messagebox, Frame, ttk

from packet\_handler import string\_by\_packages, unpack\_string\_packages, corrupt\_bit\_with\_probability

import random

packet\_size = 26

channel\_busy\_probability = 0.5

collision\_probability = 0.6

max\_retries = 10

slot\_time = 0.5

jam\_signal = b"12345678900987654321=+98674"

def is\_channel\_busy():

    return random.random() < channel\_busy\_probability

def is\_collision():

    return random.random() < collision\_probability

def send\_jam\_signal(transmitter):

    global jam\_signal

    transmitter.write(jam\_signal)

    text\_area.insert(END, "Передатчик: отправлен jam-сигнал\n")

def csma\_cd\_send(data, ser1):

    attempt = 0

    while attempt < max\_retries:

        if is\_channel\_busy():

            R = random.randint(0, min(2 \*\* attempt - 1, 1023))

            delay = R \* slot\_time

            text\_area.insert(END, f"Передатчик: ожидание {delay:.2f} секунд перед повторной попыткой...\n")

            time.sleep(delay)

            attempt += 1

            continue

        ser1.write(data)

        if is\_collision():

            text\_area.insert(END, "Передатчик: коллизия обнаружена!\n")

            send\_jam\_signal(ser1)

        break

    if attempt == max\_retries:

        text\_area.insert(END, "Передатчик: не удалось отправить данные после максимального количества попыток\n")

def send\_data():

    message = string\_by\_packages(entry\_message.get(), ser1.port[3])

    if message:

        try:

            sent\_message = []

            for char in message:

                csma\_cd\_send(char, ser1)

                sent\_message.append(char)

                time.sleep(0.1)

            update\_state(len(sent\_message))

            entry\_message.set("")

        except serial.SerialException as e:

            messagebox.showerror("Ошибка отправки", f"Ошибка при отправке данных: {e}")

def highlight\_debug\_info():

    start = "1.0"

    while True:

        start = text\_area.search(r'\\\\\#w', start, stopindex=END, regexp=True)

        if not start:

            break

        end = f"{start}+4c"

        text\_area.tag\_add("highlight", start, end)

        start = end

def read\_data():

    global packet\_size

    received\_data = b''

    while True:

        try:

            if ser2.in\_waiting > 0:

                received\_data += ser2.read\_all()

            elif len(received\_data) >= packet\_size:

                print("vsfsag bfd")

                print(received\_data[-27:0])

                if received\_data[-27:] == jam\_signal:

                    text\_area.insert(END, "Пришел jam-signal\n")

                    received\_data = b''

                    continue

                data\_string = unpack\_string\_packages(received\_data)

                received\_data = corrupt\_bit\_with\_probability(received\_data)

                data\_string = data\_string.decode('utf-8')

                text\_area.insert(END, f"Received (decoded): {received\_data}\n")

                text\_output.insert(END, f"{data\_string}")

                received\_data = b''

                text\_output.see(END)

                text\_area.see(END)

            time.sleep(0.01)

        except serial.SerialException as e:

            messagebox.showerror("Ошибка приёма", f"Ошибка при чтении данных: {e}")

            break

def update\_state(bytes\_sent):

    total\_bytes = 0

    total\_bytes += bytes\_sent

    state\_label.config(text=f"Скорость: {ser1.baudrate} бод, Передано байт: {total\_bytes}")

def update\_ports\_direction(\*args):

    selected\_direction = direction\_var.get()

    if selected\_direction == "1 -> 2":

        send\_var.set("COM1")

        receive\_var.set("COM2")

    elif selected\_direction == "5 <- 6":

        send\_var.set("COM6")

        receive\_var.set("COM5")

def start\_program():

    global ser1, ser2

    send\_port = send\_var.get() if send\_var.get() else "COM1"

    receive\_port = receive\_var.get() if receive\_var.get() else "COM2"

    baudrate = int(baudrate\_var.get())

    try:

        ser1 = serial.Serial(send\_port, baudrate=baudrate, timeout=1)

        ser2 = serial.Serial(receive\_port, baudrate=baudrate, timeout=1)

        threading.Thread(target=read\_data, daemon=True).start()

        update\_state(0)

    except serial.SerialException as e:

        messagebox.showerror("Ошибка порта", f"Не удалось открыть порт: {e}")

root = Tk()

root.title("COM-порты: Передача и приём данных")

root.geometry("600x750")

root.configure(bg="#1E1E1E")

root.resizable(False, False)

frame\_top = Frame(root, bg="#3C3F41", relief="solid", bd=2)

frame\_top.pack(pady=20, padx=10)

frame\_middle = Frame(root, bg="#3C3F41", relief="solid", bd=2)

frame\_middle.pack(pady=20, padx=10)

frame\_bottom = Frame(root, bg="#3C3F41", relief="solid", bd=2)

frame\_bottom.pack(pady=20, padx=10)

direction\_var = StringVar(root)

direction\_var.set("1 -> 2")

direction\_var.trace("w", update\_ports\_direction)

Label(frame\_top, text="Выберите направление передачи данных:", bg="#3C3F41", fg="white", font=("Arial", 12)).pack()

direction\_menu = ttk.OptionMenu(frame\_top, direction\_var, "1 -> 2", "1 -> 2", "5 <- 6")

direction\_menu.pack(pady=5)

send\_var = StringVar(root, value="COM1")

receive\_var = StringVar(root, value="COM2")

Label(frame\_top, text="Порт для отправки данных:", bg="#3C3F41", fg="white", font=("Arial", 12)).pack()

send\_label = Label(frame\_top, textvariable=send\_var, bg="#3C3F41", fg="white", font=("Arial", 10, "bold"))

send\_label.pack()

Label(frame\_top, text="Порт для получения данных:", bg="#3C3F41", fg="white", font=("Arial", 12)).pack()

receive\_label = Label(frame\_top, textvariable=receive\_var, bg="#3C3F41", fg="white", font=("Arial", 10, "bold"))

receive\_label.pack()

Label(frame\_top, text="Выберите скорость передачи данных (бод):", bg="#3C3F41", fg="white", font=("Arial", 12)).pack(pady=5)

baudrate\_var = StringVar(root)

baudrate\_var.set("9600")

baudrate\_menu = ttk.OptionMenu(frame\_top, baudrate\_var, "9600", "9600", "19200", "38400", "57600", "115200")

baudrate\_menu.pack()

start\_button = Button(frame\_top, text="Запустить", command=start\_program, bg="#76C7C0", fg="black", font=("Arial", 10, "bold"), padx=10, pady=5)

start\_button.pack(pady=10)

Label(frame\_middle, text="Введите сообщение для отправки:", bg="#3C3F41", fg="white", font=("Arial", 12)).pack()

entry\_message = StringVar()

message\_entry = Entry(frame\_middle, textvariable=entry\_message, width=50, font=("Arial", 10))

message\_entry.pack(pady=5)

send\_button = Button(frame\_middle, text="Отправить", command=send\_data, bg="#FFD700", fg="black", font=("Arial", 10, "bold"), padx=10, pady=5)

send\_button.pack(pady=10)

Label(frame\_bottom, text="Принятые данные:", bg="#3C3F41", fg="white", font=("Arial", 12)).pack()

text\_output = Text(frame\_bottom, height=5, width=60, bg="#282A36", fg="white", insertbackground='white', font=("Courier", 10))

text\_output.pack(pady=5)

state\_label = Label(frame\_bottom, text="Скорость: неизвестно, Передано байт: 0", bg="#3C3F41", fg="white", font=("Arial", 10))

state\_label.pack()

Label(frame\_bottom, text="Отладочная информация:", bg="#3C3F41", fg="white", font=("Arial", 12)).pack(pady=5)

text\_area = Text(frame\_bottom, height=10, width=60, bg="#282A36", fg="white", insertbackground='white', font=("Courier", 10))

scrollbar = Scrollbar(frame\_bottom)

scrollbar.pack(side="right", fill="y")

text\_area.pack(side="left", fill="both")

text\_area.config(yscrollcommand=scrollbar.set)

scrollbar.config(command=text\_area.yview)

text\_area.tag\_config("highlight", background="yellow", foreground="red")

root.mainloop()

if 'ser1' in globals() and ser1.is\_open:

    ser1.close()

if 'ser2' in globals() and ser2.is\_open:

    ser2.close()