Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

ОТЧЁТ

По лабораторной работе № 6

На тему

**МЕТОДЫ ВЗАИМОДЕЙСТВИЯ В ЗВЕНЕ СЕТИ**

**ПЕРЕДАЧИ ДАННЫХ**

Выполнил: Проверил:

Студент группы 050503 Преподаватель

Сикорин А. С. Марцинкевич В. А.

Минск 2022

1. **Цель работы:**

Изучить методы доступа к каналам СрПД в компьютерных сетях. Разработать программу, реализующую процедуру передачи данных по алгоритму транспортного уровня TCP.

1. **Задание к лабораторной работе:**

Написать программу передачи данных через COM-порты упрощенным алгоритмом транспортного уровня TCP в соответствии с требованиями.

**3. Теоретическая часть**

При планировании полномасштабной сети возникают задачи оптимизации взаимодействия между станциями. Одним из методов контроля доставки пакетов является оконный метод. Суть в том, что передатчик отправляет сразу пакет данных, а не отдельные сообщения, и ожидает получить в ответ сообщения от приемника об корректном принятии данных также пакетом. Отправка данных пакетами существенно экономит время цикла отправка-ответ. Статическое окно закладывает неизменяемое количество пакетов для единоразовой передачи. Минус в том, что условия в СрПД могут изменяться, а окно будет таким же. Динамическое же окно способно увеличиваться или уменьшаться по запросу одной из сторон, например, клиента или сервера. Например, уменьшение окна обычно запрашивает клиент. Классической реализацией оконно метода является протокол TCP. Сообщение разбивается на сегменты, каждый байт которого нумеруется последовательным номером начиная с некоторого случайного числа. Станция, работающая по TCP протоколу, имеет буферы для подлежащих отправке сообщений и их принятия, размеры которых зачастую разные. Буфер приема поддерживает механизм упорядочивания пакетов, основываясь на их номере.

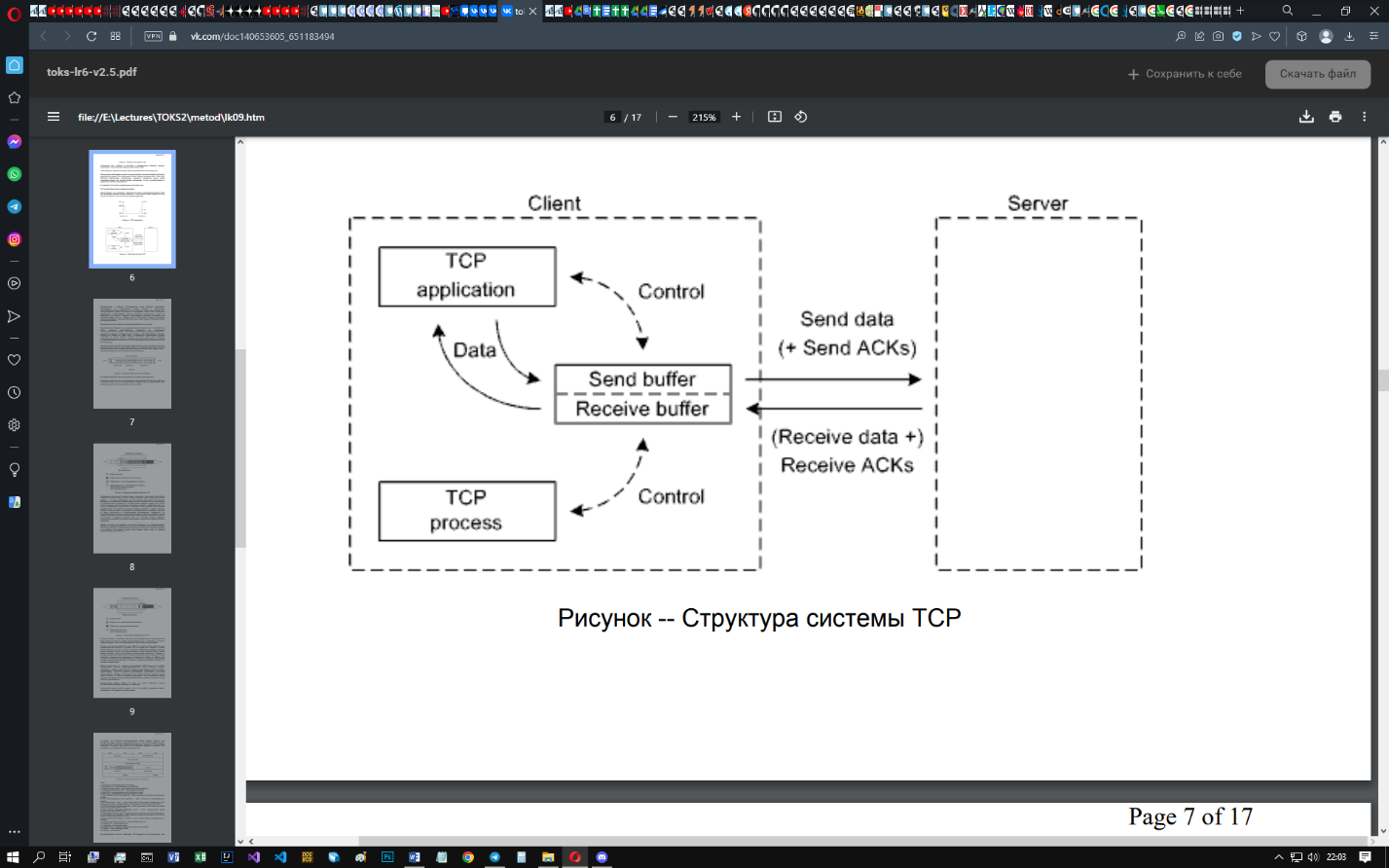


Рис. 1 — структура системы TCP

Программа эмулирует передачу данных по упрощенному протоколу TCP. Сообщение сервера делится на сегменты, на выбор пользователя можно изменить размер окна.

1. **Практическая часть**

#!/usr/bin/env python3

import textwrap

import time

from dataclasses import dataclass

SEGMENT\_SIZE = 3

@dataclass

class Packet:

sequence\_number: int = 0

acknowledgment\_number: int = 1

is\_acknowledgment: bool = False

is\_synchronize: bool = True

is\_finish: bool = False

window\_size: int = SEGMENT\_SIZE

message: str = "None"

def \_\_repr\_\_(self):

return f"Packet(" \

f"SN={self.sequence\_number} " \

f"AN={self.acknowledgment\_number} " \

f"ACK={self.is\_acknowledgment} " \

f"SYN={self.is\_synchronize} " \

f"FIN={self.is\_finish} " \

f"W={self.window\_size} " \

f"Message={self.message})"

class Client:

@classmethod

def send\_packet(cls, packet: Packet):

print(f"Client send {packet}")

Server.packet = packet

@classmethod

def send\_acknowledgement(cls, packet: Packet) -> Packet:

ack\_packet = Packet(packet.acknowledgment\_number,

packet.acknowledgment\_number + SEGMENT\_SIZE,

True,

False,

packet.is\_finish,

packet.window\_size,

packet.message)

print(f"Client send {ack\_packet}")

return ack\_packet

class Server:

packet: Packet

@classmethod

def send\_acknowledgement(cls) -> Packet:

ack\_packet = Packet(Server.packet.acknowledgment\_number,

Server.packet.acknowledgment\_number + SEGMENT\_SIZE,

True,

Server.packet.is\_synchronize,

Server.packet.is\_finish,

Server.packet.window\_size,

Server.packet.message)

print(f"Server send {ack\_packet}")

return ack\_packet

def main():

first\_packet = Packet()

Client.send\_packet(first\_packet)

ack\_packet = Server.send\_acknowledgement()

last\_packet = Client.send\_acknowledgement(ack\_packet)

while True:

option = input("Change window size (y/n or q to quit):")

if option == "q":

break

elif option == "y":

window\_size = int(input("Enter new window size:"))

packet = Packet(last\_packet.acknowledgment\_number,

last\_packet.acknowledgment\_number + SEGMENT\_SIZE,

False,

False,

False,

window\_size)

Client.send\_packet(packet)

last\_packet = Server.send\_acknowledgement()

message = input("Enter message:")

segments = textwrap.wrap(message, 3)

i = 0

for seg in segments:

packet = Packet(last\_packet.acknowledgment\_number,

last\_packet.acknowledgment\_number + SEGMENT\_SIZE,

False,

False,

seg == segments[-1],

last\_packet.window\_size,

seg)

Client.send\_packet(packet)

last\_packet = Server.send\_acknowledgement()

i += 1

if i == last\_packet.window\_size or seg == segments[-1]:

print(f"Send {last\_packet.window\_size} packets in window")

time.sleep(1)

i = 0

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

main()