## 100071011 Computer Networks 2023-2024-2

# Project-1

# Reliable File Transfer using Go-Back-N protocol

# Specification

| | |
|---|---|
| 学号 (Student ID) | 1820243077 |
| 姓名 (Name) | Pimenov Gleb |
| 班号 (Class No.) | 1107 |
| 授课教师 (Instructor) | 杨松 |

**School of Computer**
**Beijing Institute of Technology**
**April 20, 2024**

## 1. Requirement Analysis

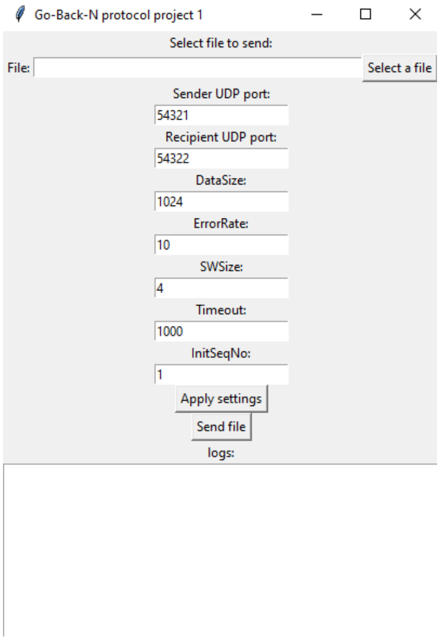
The project requires the development of a software client that can send files using the go back n protocol. The program must convert a file into packets for sending and build a file from the received packets. The program must use only UDP. The maximum packet size is 4KB. The GBN protocol implemented shall support full duplex. The program must be able to generate sending errors when sending. The program must send files of any size, for testing it is recommended to use files more than 3MB. The program should be configurable by the user. The program should keep logs.

GBN is reliable for file transfers because if the user is not sure whether the packet has been received, the algorithm repeats sending until it is sure of receipt. The protocol also makes efficient use of time.

The go back n protocol is a sliding window protocol used in network communication for reliable data transmission. It allows a sender to transmit multiple frames before receiving an acknowledgment from the receiver. If an acknowledgment is not received within a specified time frame, the sender retransmits all unacknowledged frames starting from the last acknowledged one.

## 2. Design

The program is an application with a graphical interface in which it is possible to set parameters and track program activity in the log (log is written both to the user and to the logsrun.log file).



Sender UDP port: 54321
Recipient UDP port: 54322
DataSize: 1024
ErrorRate: 10
SWSize: 4
Timeout: 1000
InitSeqNo: 1

Pic. 1. Interface application.

Send and receive file has one source code. Runs 2 iterations of the program, specifies the port of the sender and recipient.

Inside the program contains 2 classes Sender and Receiver. These classes are responsible for sending and receiving data via UDP. Sender class is used to send a file using GBN protocol. For this purpose it has a method send_data(self,data). As input it receives an array of packets into which the file was converted for sending (the file is divided into packets when the Send file button is pressed). Also Sender class has auxiliary methods responsible for timer management.

The Receiver class is executed in parallel from the moment the configuration settings are applied. The class waits for receiving packets and determines their type. If the packet is a part of a file, then the packet will be processed as a part of the file, namely: the data (number, crc, data) will be extracted, the crc32 hash sum for the data will be calculated, and a data integrity response will be sent. If the packet is a response from the recipient, the need to return to the N.

The file is divided into packets with the specified size in the configuration, but cannot be larger than 4KB. A packet consists of: packet number, crc, data. Only the data size can be changed. The size of number and crc is strictly defined: 4 bytes integer.

After pressing the Send file button, this happens:

1. Packets to be sent are generated. The number starting from the number specified in the configuration is set. The crc part of data is calculated.
2. The send_data procedure is started in parallel mode.
3. the range of working indexes is defined (from InitSeqNo - to data_chunks_count + InitSeqNo).
4. Packet sending cycle starts.
    a. Working with sending window.
    b. Checking the return to N.
    c. Sending a packet.
    d. Starting the timer.
    e. Checking the end of the timer (runs in parallel).

## 3. Development and Implementation

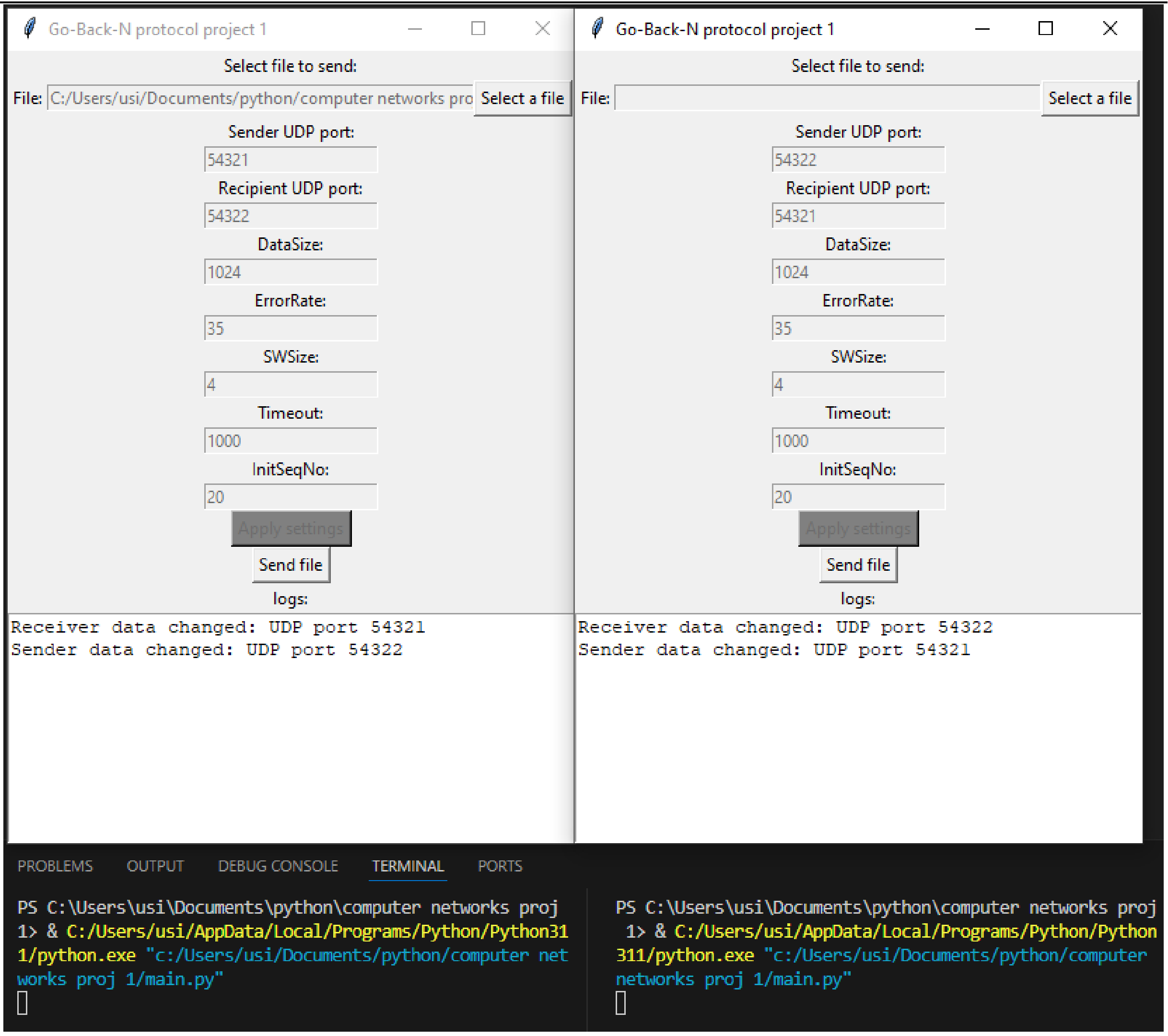
The python programming language is chosen for development because the team has more experience with this language. Version python 3.11.7. OS windows 10/11.

The send and receive procedures are executed in parallel using the threading library. The struct library is used to represent a file as binary packets. The socket library is used to send packets via UDP. Checking for crc is done using zlib. The program interface is made using tkinter.

## 4. System Deployment, Startup, and Use

The project can be run in 2 ways: through the python interpreter by running main.py or by running the exe file of the program. The user runs the program 2 times to get the sender and receiver interface.

The user enters parameters into the configuration, selects the file to be sent, and applies the settings. An example can be seen in pic. 2.



Go-Back-N protocol project 1

Select file to send:

File: C:/Users/usi/Documents/python/computer networks pro

Sender UDP port: 54321

Recipient UDP port: 54322

DataSize: 1024

ErrorRate: 35

SWSize: 4

Timeout: 1000

InitSeqNo: 20

logs:

```
Receiver data changed: UDP port 54321
Sender data changed: UDP port 54322
```

Go-Back-N protocol project 1

Select file to send:

File:

Sender UDP port: 54322

Recipient UDP port: 54321

DataSize: 1024

ErrorRate: 35

SWSize: 4

Timeout: 1000

InitSeqNo: 20

logs:

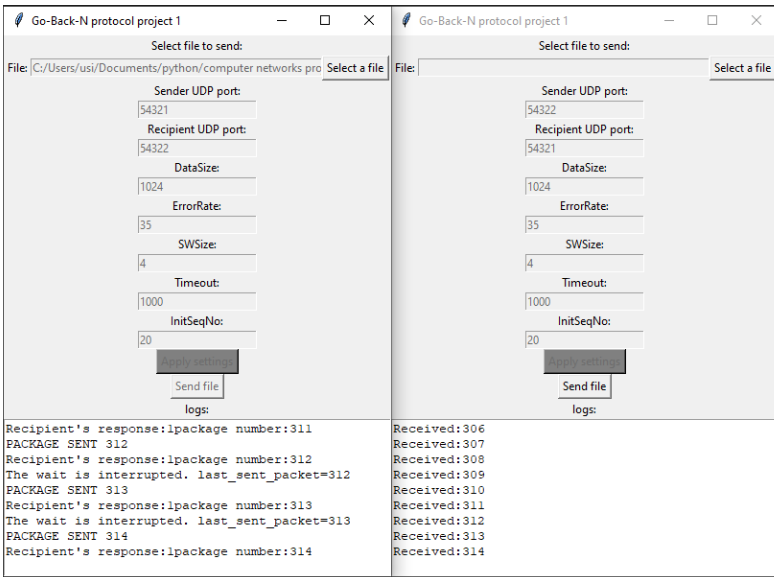
```
Receiver data changed: UDP port 54322
Sender data changed: UDP port 54321
```

```
PS C:\Users\usi\Documents\python\computer networks proj
1> & C:/Users/usi/AppData/Local/Programs/Python/Python31
1/python.exe "c:/Users/usi/Documents/python/computer net
works proj 1/main.py"
```

```
PS C:\Users\usi\Documents\python\computer networks proj
 1> & C:/Users/usi/AppData/Local/Programs/Python/Python
311/python.exe "c:/Users/usi/Documents/python/computer
networks proj 1/main.py"
```

Pic. 2. Example of usage.

The sender then clicks the Send file button. The program starts sending the file (pic. 3).



Go-Back-N protocol project 1

Select file to send:

File: C:/Users/usi/Documents/python/computer networks pro

Sender UDP port: 54321

Recipient UDP port: 54322

DataSize: 1024

ErrorRate: 35

SWSize: 4

Timeout: 1000

InitSeqNo: 20

logs:

```
Recipient's response:1package number:311
PACKAGE SENT 312
Recipient's response:1package number:312
The wait is interrupted. last_sent_packet=312
PACKAGE SENT 313
Recipient's response:1package number:313
The wait is interrupted. last_sent_packet=313
PACKAGE SENT 314
Recipient's response:1package number:314
```

Go-Back-N protocol project 1

Select file to send:

File:

Sender UDP port: 54322

Recipient UDP port: 54321

DataSize: 1024

ErrorRate: 35

SWSize: 4

Timeout: 1000

InitSeqNo: 20

logs:

```
Received:306
Received:307
Received:308
Received:309
Received:310
Received:311
Received:312
Received:313
Received:314
```

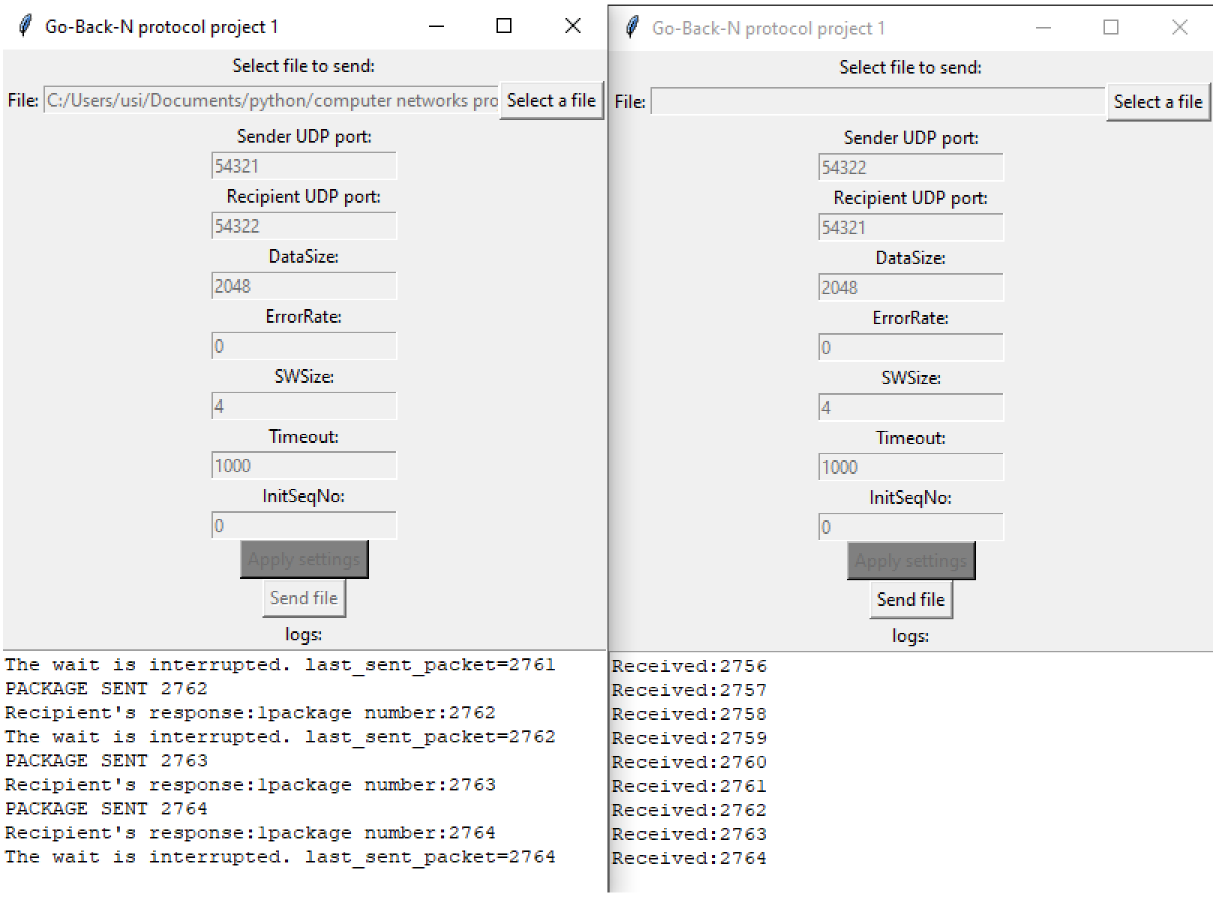
Pic.3. Sending file.

When the sending is finished, the received file will be written next to the recipient's executable file with the ending "_received" (if file in the receiver not specified – you will get just _received file and you need to add .jpg/.png/.jpeg etc to the end)

## 5. System Test

Possible cases:

- sending without errors/ with errors.
- window size 1 / larger 1.
- packet fails to timeout.



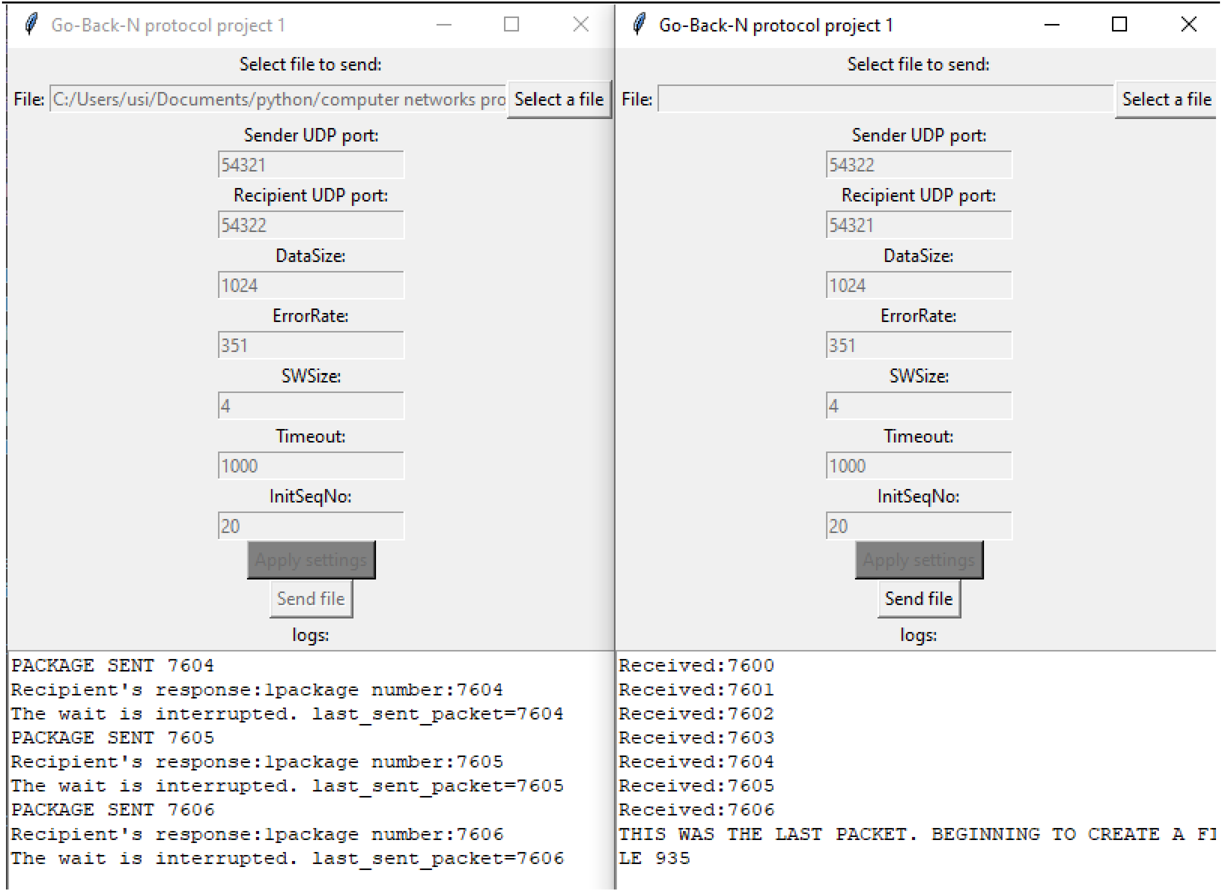
File: C:/Users/usi/Documents/python/computer networks pro
Sender UDP port: 54321
Recipient UDP port: 54322
DataSize: 2048
ErrorRate: 0
SWSize: 4
Timeout: 1000
InitSeqNo: 0

```
The wait is interrupted. last_sent_packet=2761
PACKAGE SENT 2762
Recipient's response:1package number:2762
The wait is interrupted. last_sent_packet=2762
PACKAGE SENT 2763
Recipient's response:1package number:2763
PACKAGE SENT 2764
Recipient's response:1package number:2764
The wait is interrupted. last_sent_packet=2764
```

File:
Sender UDP port: 54322
Recipient UDP port: 54321
DataSize: 2048
ErrorRate: 0
SWSize: 4
Timeout: 1000
InitSeqNo: 0

```
Received:2756
Received:2757
Received:2758
Received:2759
Received:2760
Received:2761
Received:2762
Received:2763
Received:2764
```

Pic.4. Without errors, ErrorRate 0.



File: C:/Users/usi/Documents/python/computer networks pro

Sender UDP port: 54321

Recipient UDP port: 54322

DataSize: 1024

ErrorRate: 351

SWSize: 4

Timeout: 1000

InitSeqNo: 20

```
PACKAGE SENT 7604
Recipient's response:1package number:7604
The wait is interrupted. last_sent_packet=7604
PACKAGE SENT 7605
Recipient's response:1package number:7605
The wait is interrupted. last_sent_packet=7605
PACKAGE SENT 7606
Recipient's response:1package number:7606
The wait is interrupted. last_sent_packet=7606
```

File:

Sender UDP port: 54322

Recipient UDP port: 54321

DataSize: 1024

ErrorRate: 351

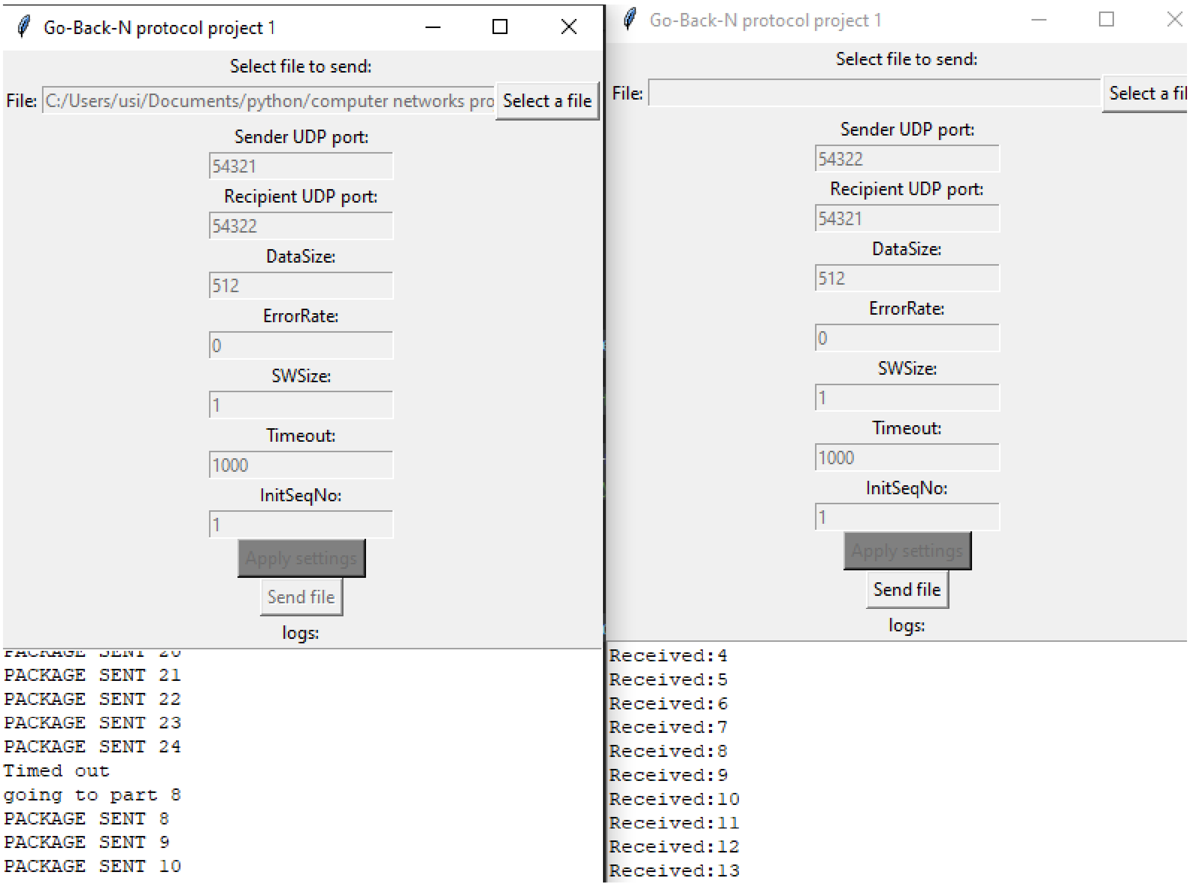
SWSize: 4

Timeout: 1000

InitSeqNo: 20

```
Received:7600
Received:7601
Received:7602
Received:7603
Received:7604
Received:7605
Received:7606
THIS WAS THE LAST PACKET. BEGINNING TO CREATE A FI
LE 935
```

Pic.5. with errors ErrorRate 351. Window Size > 1.



File: C:/Users/usi/Documents/python/computer networks pro

Sender UDP port: 54321

Recipient UDP port: 54322

DataSize: 512

ErrorRate: 0

SWSize: 1

Timeout: 1000

InitSeqNo: 1

```
PACKAGE SENT 20
PACKAGE SENT 21
PACKAGE SENT 22
PACKAGE SENT 23
PACKAGE SENT 24
Timed out
going to part 8
PACKAGE SENT 8
PACKAGE SENT 9
PACKAGE SENT 10
```

File:

Sender UDP port: 54322

Recipient UDP port: 54321

DataSize: 512

ErrorRate: 0

SWSize: 1

Timeout: 1000

InitSeqNo: 1

```
Received:4
Received:5
Received:6
Received:7
Received:8
Received:9
Received:10
Received:11
Received:12
Received:13
```

Pic. 6. Sending window 1 and timeout

## 6. Performance and Analysis

When SWsize > 1. The algorithm shows good performance, data is transmitted continuously. But when SWsize = 1 there is useless waiting. Also, the algorithm copes well with error correction during packet sending.

## 7. Summary or Conclusions

While working on the project, it turned out to be not easy to combine parallel processing of sending and receiving returns to N. We solved this problem by using return flags on N.

The developed training project showed us the possibility and simplicity of realizing our own information transfer protocols, and helped us to understand more deeply the workings of information transfer protocols.

## 8. References

5 or more references.

1. Computer Networks / Andrew S. Tanenbaum, David J. Wetherall. – 5th ed.
2. Computer Networks – A System Approach / Larry L. Peterson, Bruce S. Davie – 6th ed.
3. Computer Networking – A Top-Down Approach / James F. Kurose, Keith W. Ross – 7th ed.
4. struct — Interpret bytes as packed binary data. Documentation. URL: https://docs.python.org/3/library/struct.html
5. socket — Low-level networking interface. Documentation. URL: https://docs.python.org/3/library/socket.html
6. tkinter — Python interface to Tcl/Tk. Documentation. URL: https://docs.python.org/3/library/tkinter.html
7. Zlib python. Documentation. URL: https://docs.python.org/3/library/zlib.html