University of Science and Technology
Communication and Information Engineering Program
CIE 447: Computer Networks

Reliable Transport Protocol
UDP - GBN

Mohamed Ahmed 201-801-898
Ibrahim Ibrahim 201-800-739
Khaled Elbastawisy 201-800-029
Moataz Ahmed 201-801-903

I. Sender Implementation:

In Go-Back-N the sender is allowed to send multiple messages within a specified window without waiting for acknowledgement. The sender has two important parameters; the base which is the sequence number of the oldest unacknowledged packet and NextSeqNum which is the smallest unused sequence number (the next packet to be sent). Packets with ids in the interval [base, NextSeqNum - 1] are sent immediately. When the sender receives an acknowledgement from the receiver it slides its windows depending on the packet id of the acknowledgement packet (cumulative acknowledgement).

The sender executes three types of events:

I. When it is invoked to send data. It checks to see whether the window is full. If the window is not full. The packet is created and ready to be sent and variables are modified accordingly .

II. When it receives an ACK. An acknowledgement packet with sequence n indicates that all packets up to n have been received successfully prompting the sender to slide its window and update its base to n+1

III. Timeout. If a timeout occurs, the sender is required to resend all the packets that have been already sent and they might have lost or their acknowledgements weren't received. There is one single timer which is for the oldest transmitted unacknowledged packet. The timer restarts upon each ACK reception.

II. Receiver Implementation:

In Go-Back-N the receiver deals with receiving a packet, parsing the received packet by separating the header and trailer information from the application data. The receiver also checks if the received packet ID is the expected packet to be received or not. If the packet ID is the expected packet to be received, the application data is stored. Otherwise, the data is discarded. Afterwards, the receiver sends an acknowledgment packet to the sender with the ID of the last correctly received packet. When the last packet has been received, the receiver then acknowledges the packet, writes the data to a new file, and indicates via the user interface that the file reception is complete. Packet Loss is simulated at the receiver side to indicate the channel losses by randomly dropping some packets at the receiver and assume that they were lost

The receiver executes these events:

I. The receiver starts by establishing the connection with the sender.
II. The channel loss rate is simulated using a random number generator between 0 and 1 and the loss rate is set to 10%.
III. If the random number is less than 0.1, the packet will be lost.
IV. The received packet is divided in order to obtain packet_id, file_id, data, trailer.
V. The receiver checks if the received sequence number is the expected number to be received or not.
VI. If the received packet isn't the required one, it will be discarded and the receiver will send an acknowledgment to the sender with the ID of the last correctly received packet.
VII. When the expected packet is received, the extracted data will be appended to the data array and an acknowledgement packet will be sent to the sender.
VIII. If the trailer of the received packet is "FFFF", which indicates the end of the file, the receiver stops waiting for packets and the received data will be written to a file.

III. Test Cases:

A) SmallFile.png:

Iteration 1: MSS = 1024, Window Size = 4, Timeout = 0.01 sec

- **Screenshots of the First sent packets (Sender)**

```
"F:\College\3- Fourth Year\Spring 2022\CIE 447 - Computer
Sending packet with ID: 0  |  Time Stamp: 23:15:58
Sending packet with ID: 1  |  Time Stamp: 23:15:58
Sending packet with ID: 2  |  Time Stamp: 23:15:58
Sending packet with ID: 3  |  Time Stamp: 23:15:58
Received ACK for packet: 0  |  Time Stamp: 23:15:58
Sending packet with ID: 4  |  Time Stamp: 23:15:58
```

- **Screenshots of the First received packets (Receiver)**

```
C:\Users\marad\Downloads\venv\Scripts\python.exe
Received packet with ID: 0, Time Stamp: 23:16:08
Sending ACK no: 0, Time Stamp: 23:16:08
Received packet with ID: 1, Time Stamp: 23:16:08
Sending ACK no: 1, Time Stamp: 23:16:08
Received packet with ID: 2, Time Stamp: 23:16:08
Sending ACK no: 2, Time Stamp: 23:16:08
Received packet with ID: 3, Time Stamp: 23:16:08
Sending ACK no: 3, Time Stamp: 23:16:08
```

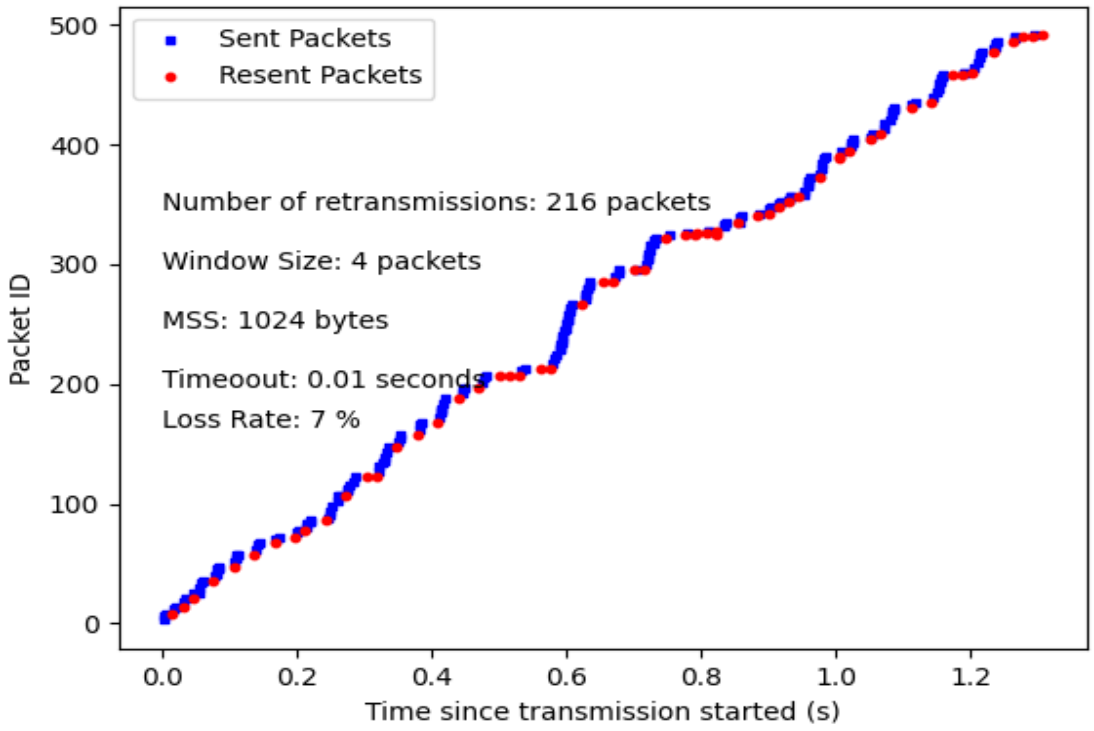
- **Screenshots of the Last sent packets (Sender)**

```
Retransmitting Packets
Resending packet with ID: 488  |  Time Stamp: 23:15:59
Resending packet with ID: 489  |  Time Stamp: 23:15:59
Resending packet with ID: 490  |  Time Stamp: 23:15:59
Resending packet with ID: 491  |  Time Stamp: 23:15:59
Received ACK for packet: 488  |  Time Stamp: 23:15:59
Received ACK for packet: 489  |  Time Stamp: 23:15:59
Received ACK for packet: 490  |  Time Stamp: 23:15:59
Received ACK for packet: 491  |  Time Stamp: 23:15:59
Retransmitting Packets
All Packets Sent
```

- **Screenshots of the Last received packets (Receiver)**

```
Received packet with ID: 489, Time Stamp: 23:16:09
Sending ACK no: 489, Time Stamp: 23:16:09
Received packet with ID: 490, Time Stamp: 23:16:09
Sending ACK no: 490, Time Stamp: 23:16:09
Received packet with ID: 491, Time Stamp: 23:16:09
Sending ACK no: 491, Time Stamp: 23:16:09
All packets are received.
Writing data into file: received.png.
```

- **Plot of the received packet ID versus time**



- **Statistics (Sender)**

```
============================================================
Statistics:   MSS = 1024,  Window Size = 4,  Timeout = 0.01 sec
============================================================
Transfer Start Time:   Date: 20-May-2022  |  Time: 23:15:58
Transfer End Time:   Date: 20-May-2022  |  Time: 23:15:59
Elapsed Time in Seconds:   1.31 sec
Number of Packets:   708
Number of Bytes:   727846
Number of Retransmitted Packets:   216
Average Transfer Rate (bytes/sec):   556227.55
Average Transfer Rate (packets/sec):   541.06
```

- **Statistics (Receiver)**

```
================================================
Statistics:
================================================
Transfer Start Time:   Date: 20-May-2022  |  Time: 23:16:08
Transfer End Time:   Date: 20-May-2022  |  Time: 23:16:09
Elapsed Time in Seconds:   1.31 sec
Number of Received Packets (Data):   633
Number of Received Bytes (Data):   651293
Number of Transmitted Packets (ACK):   633
Number of Transmitted Bytes (ACK):   2532
Number of Lost Packets:   141
Average Transfer Rate (bytes/sec):   497508.99
Average Transfer Rate (packets/sec):   483.54
```

<u>Iteration 2:</u> MSS = 1024, Window Size = 8, Timeout = 0.001 sec

- **Screenshots of the First sent packets (Sender)**

```
"F:\College\3- Fourth Year\Spring 2022\CIE 447 - Comput
Sending packet with ID: 0  |  Time Stamp: 23:30:33
Sending packet with ID: 1  |  Time Stamp: 23:30:33
Sending packet with ID: 2  |  Time Stamp: 23:30:33
Sending packet with ID: 3  |  Time Stamp: 23:30:33
Sending packet with ID: 4  |  Time Stamp: 23:30:33
Sending packet with ID: 5  |  Time Stamp: 23:30:33
Sending packet with ID: 6  |  Time Stamp: 23:30:33
Sending packet with ID: 7  |  Time Stamp: 23:30:33
Received ACK for packet: 0  |  Time Stamp: 23:30:33
```

- **Screenshots of the First received packets (Receiver)**

```
C:\Users\marad\Downloads\venv\Scripts\python.exe C:/Us
Received packet with ID: 0, Time Stamp: 23:30:44
Sending ACK no: 0, Time Stamp: 23:30:44
Received packet with ID: 1, Time Stamp: 23:30:44
Sending ACK no: 1, Time Stamp: 23:30:44
Received packet with ID: 2, Time Stamp: 23:30:44
Sending ACK no: 2, Time Stamp: 23:30:44
Received packet with ID: 3, Time Stamp: 23:30:44
Sending ACK no: 3, Time Stamp: 23:30:44
Received packet with ID: 4, Time Stamp: 23:30:44
```

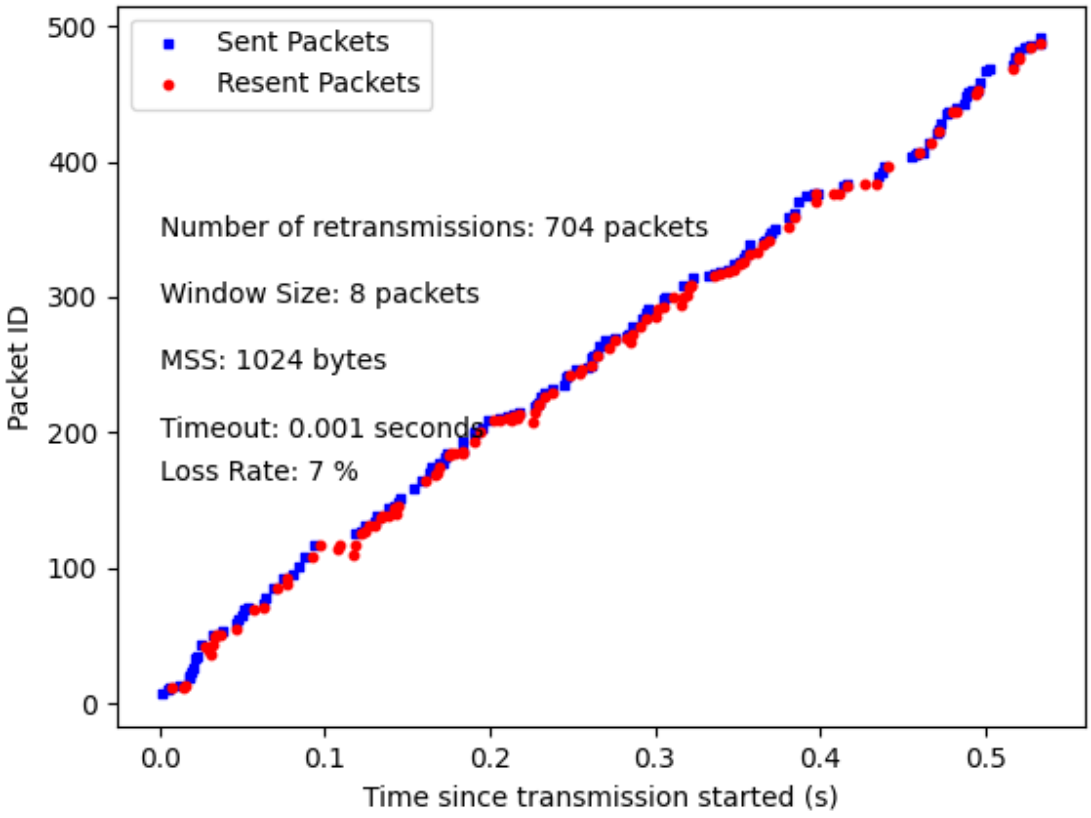
- **Screenshots of the Last sent packets (Sender)**

```
Received ACK for packet: 487  |  Time Stamp: 23:30:34
Received ACK for packet: 487  |  Time Stamp: 23:30:34
Received ACK for packet: 488  |  Time Stamp: 23:30:34
Received ACK for packet: 489  |  Time Stamp: 23:30:34
Received ACK for packet: 490  |  Time Stamp: 23:30:34
Received ACK for packet: 491  |  Time Stamp: 23:30:34
```

- **Screenshots of the Last received packets (Receiver)**

```
Received packet with ID: 489, Time Stamp: 23:30:44
Sending ACK no: 489, Time Stamp: 23:30:44
Received packet with ID: 490, Time Stamp: 23:30:44
Sending ACK no: 490, Time Stamp: 23:30:44
Received packet with ID: 491, Time Stamp: 23:30:44
Sending ACK no: 491, Time Stamp: 23:30:44
All packets are received.
Writing data into file: received.png.
```

- **Plot of the received packet ID versus time**



- **Statistics (Sender)**

```
================================================================
Statistics:   MSS = 1024,  Window Size = 8,  Timeout = 0.001 sec
================================================================
Transfer Start Time:   Date: 20-May-2022  |  Time: 23:30:33
Transfer End Time:   Date: 20-May-2022  |  Time: 23:30:34
Elapsed Time in Seconds:   0.54 sec
Number of Packets:   1196
Number of Bytes:   1231183
Number of Retransmitted Packets:   704
Average Transfer Rate (bytes/sec):   2297471.85
Average Transfer Rate (packets/sec):   2231.82
```

- **Statistics (Receiver)**

```
================================================
Statistics:
================================================
Transfer Start Time:   Date: 20-May-2022  |  Time: 23:30:44
Transfer End Time:   Date: 20-May-2022  |  Time: 23:30:44
Elapsed Time in Seconds:   0.53 sec
Number of Received Packets (Data):   1093
Number of Received Bytes (Data):   1125093
Number of Transmitted Packets (ACK):   1093
Number of Transmitted Bytes (ACK):   4372
Number of Lost Packets:   601
Average Transfer Rate (bytes/sec):   2110698.39
Average Transfer Rate (packets/sec):   2050.49
```

<u>Iteration 3:</u> MSS = 2048, Window Size = 8, Timeout = 0.001 sec

- **Screenshots of the First sent packets (Sender)**

```
"F:\College\3- Fourth Year\Spring 2022\CIE 447 - Compu
Sending packet with ID: 0  |  Time Stamp: 23:48:13
Sending packet with ID: 1  |  Time Stamp: 23:48:13
Sending packet with ID: 2  |  Time Stamp: 23:48:13
Sending packet with ID: 3  |  Time Stamp: 23:48:13
Sending packet with ID: 4  |  Time Stamp: 23:48:13
Sending packet with ID: 5  |  Time Stamp: 23:48:13
Sending packet with ID: 6  |  Time Stamp: 23:48:13
Sending packet with ID: 7  |  Time Stamp: 23:48:13
```

- **Screenshots of the First received packets (Receiver)**

```
C:\Users\marad\Downloads\venv\Scripts\python.exe C:/Users/ma
Received packet with ID: 0, Time Stamp: 23:48:23
Sending ACK no: 0, Time Stamp: 23:48:23
Received packet with ID: 1, Time Stamp: 23:48:23
Sending ACK no: 1, Time Stamp: 23:48:23
Received packet with ID: 2, Time Stamp: 23:48:23
Sending ACK no: 2, Time Stamp: 23:48:23
Received packet with ID: 3, Time Stamp: 23:48:23
Sending ACK no: 3, Time Stamp: 23:48:23
```

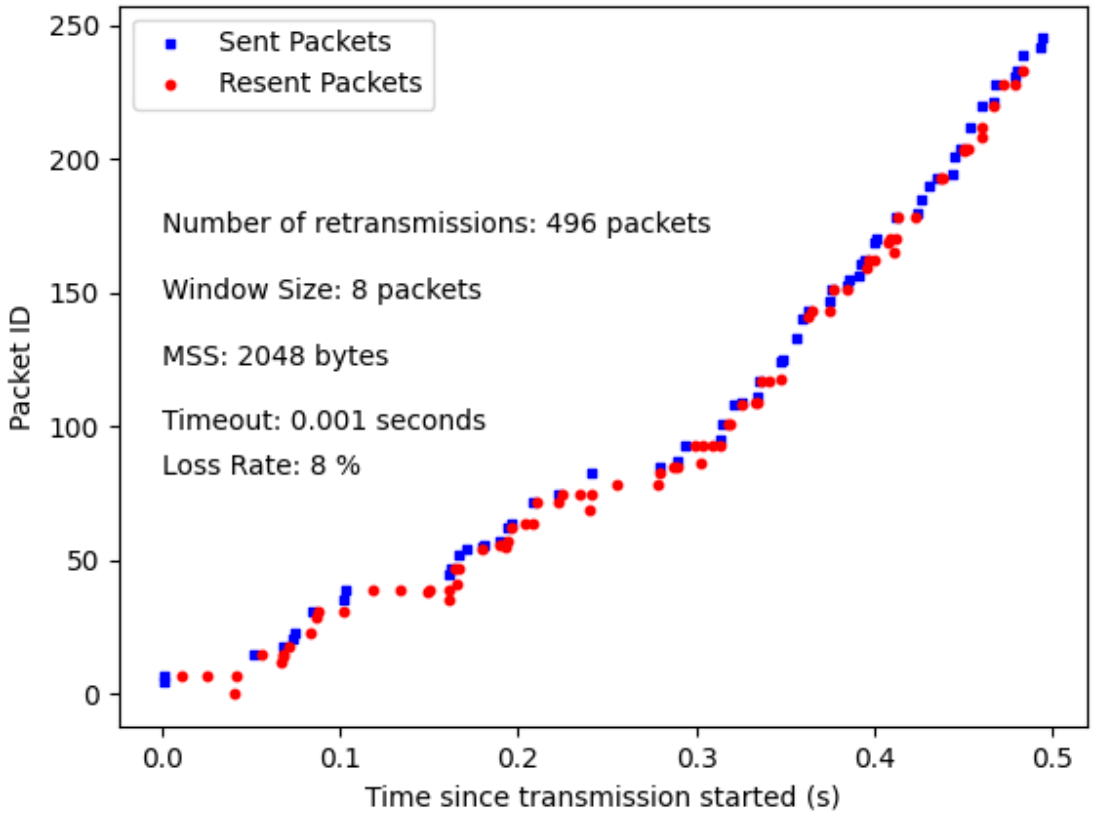
- **Screenshots of the Last sent packets (Sender)**

```
Received ACK for packet: 238  |  Time Stamp: 23:48:13
Received ACK for packet: 239  |  Time Stamp: 23:48:13
Received ACK for packet: 240  |  Time Stamp: 23:48:13
Received ACK for packet: 241  |  Time Stamp: 23:48:13
Received ACK for packet: 242  |  Time Stamp: 23:48:13
Received ACK for packet: 243  |  Time Stamp: 23:48:13
Received ACK for packet: 244  |  Time Stamp: 23:48:13
Received ACK for packet: 245  |  Time Stamp: 23:48:13
```

- **Screenshots of the Last received packets (Receiver)**

```
Received packet with ID: 242, Time Stamp: 23:48:23
Sending ACK no: 242, Time Stamp: 23:48:23
Received packet with ID: 243, Time Stamp: 23:48:23
Sending ACK no: 243, Time Stamp: 23:48:23
Received packet with ID: 244, Time Stamp: 23:48:23
Sending ACK no: 244, Time Stamp: 23:48:23
Received packet with ID: 245, Time Stamp: 23:48:23
Sending ACK no: 245, Time Stamp: 23:48:23
All packets are received.
Writing data into file: received.png.
```

- **Plot of the received packet ID versus time**



- **Statistics (Sender)**

```
================================================================
Statistics:   MSS = 2048,  Window Size = 8,  Timeout = 0.001 sec
================================================================
Transfer Start Time:   Date: 20-May-2022  |  Time: 23:48:13
Transfer End Time:   Date: 20-May-2022  |  Time: 23:48:13
Elapsed Time in Seconds:   0.50 sec
Number of Packets:   742
Number of Bytes:   1523371
Number of Retransmitted Packets:   496
Average Transfer Rate (bytes/sec):   3065805.18
Average Transfer Rate (packets/sec):   1493.29
```

- **Statistics (Receiver)**

```
================================================
Statistics:
================================================
Transfer Start Time:   Date: 20-May-2022  |  Time: 23:48:23
Transfer End Time:   Date: 20-May-2022  |  Time: 23:48:23
Elapsed Time in Seconds:   0.47 sec
Number of Received Packets (Data):   667
Number of Received Bytes (Data):   1369321
Number of Transmitted Packets (ACK):   667
Number of Transmitted Bytes (ACK):   2668
Number of Lost Packets:   421
Average Transfer Rate (bytes/sec):   2944497.72
Average Transfer Rate (packets/sec):   1434.27
```

IV. Protocol Optimization:

Based on the Test Cases, iteration (3) showed the highest transmission rate using parameters as follows:

- MSS: 2048
- Window Size: 8
- Timeout: 0.001

As the MSS increases the number of packets to be transmitted is significantly reduced which increases the rate of transmission. Increasing the window size should increase the transmission rate unless the loss rate is too high so there would be too much unnecessary transmissions which comprises the transmission rate. Decreasing the timeout interval also increases the transmission rate.

V. Protocol Modification:

We can implement congestion control and some needed reliability to our UDP based protocol.

Congestion Control:

Our approach of applying congestion control can be executed by simply letting the receiver send its receiving rate after a specific amount of time of starting the communication. The receiving rate can be communicated within the acknowledgment packet. Consequently, the sender can modify its packet transmission in order to be sufficiently higher than that of the receiver (taking the latency into account) and reduces the possibility of congestion control taking place. The sender can freely alter the MSS accordingly. Besides, the protocol can be modified such that it is not obligated to retransmit the whole window, yet it can send a certain packet instead.

Reliability:

It can be done if reliability is built into applications itself. Adding acknowledgment and retransmission mechanisms is a way for having reliable data transfer; those mechanisms are already implemented in our protocol. Thus, the processes are able to communicate reliably ignoring the restrictions constraints imposed by TCP's congestion control mechanisms. In addition, to add more reliability, we can allow the receiver to send a signal to the sender in case the receive buffer is full in order not to lose any packets.

VI. Protocol Attack:

Two types of possible attacks are:

UDP Flood:
A UDP flood occurs when a large number of faked UDP packets are transmitted to various ports on a single server, with no mechanism to authenticate the packets' true origin. The server overwhelms its resources by responding to all requests with ICMP 'Destination Unreachable' signals.

DDoS attackers frequently perform generic network layer attacks, in addition to the typical UDP flood, by delivering large volumes of false UDP packets to cause network congestion. Only by scaling up a network's resources on demand, like with a cloud DDoS mitigation system, can these attacks be neutralized.

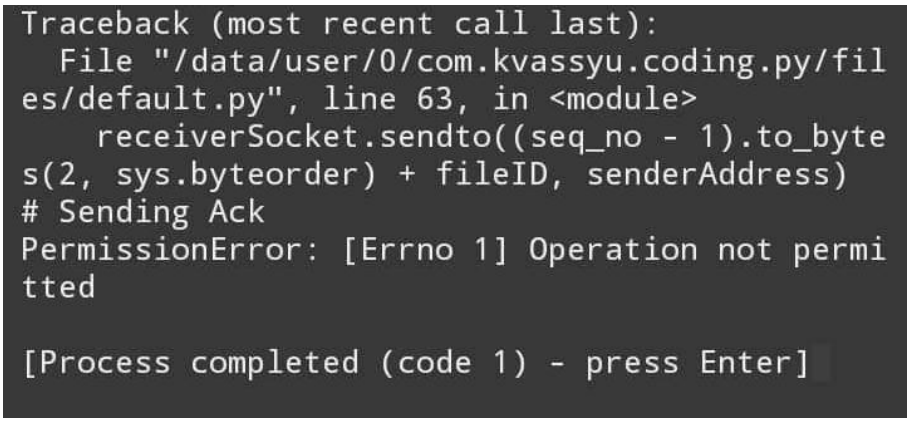
Amplification of DNS:
A DNS amplification attack entails a perpetrator sending UDP packets to the victim's IP address with a fake IP address.

The sender was modified to continuously send fake packets that contain no data and only iterate in ID so that the receiver is overwhelmed and eventually crippled. The loop iterates the ID starting at 0 and ending at 65,535. At this point the ID counter restarts and it goes over all IDs again.

The loop used to execute the attack is as follows:

```
print("Starting attack")
fakeid=0
while True:
   trailer = "0000"
   data = 10101
   pckt_id = fakeid.to_bytes(2,sys.byteorder)
   file_id = 0
   data = data.to_bytes(2,sys.byteorder)
   pcktfile = file_id.to_bytes(2, sys.byteorder)
   pckttrailer = bytearray.fromhex(trailer)
   sndpckt = pckt_id + pcktfile + data + pckttrailer
   print('Sending fake packet with ID: ', fakeid)
   senderSocket.sendto(sndpckt, (rec_IP, rec_port))
   if fakeid>=65535:
       fakeid=0
   else:
       fakeid += 1
```

Screenshot of the error shown at the receiver:



```
Traceback (most recent call last):
  File "/data/user/0/com.kvassyu.coding.py/fil
es/default.py", line 63, in <module>
    receiverSocket.sendto((seq_no - 1).to_byte
s(2, sys.byteorder) + fileID, senderAddress)
# Sending Ack
PermissionError: [Errno 1] Operation not permi
tted

[Process completed (code 1) - press Enter]
```

VII. Mitigation Methods and Possible protocol updates:

The attack pays no regard to any acknowledgements sent by the receiver, which reinstate the needed packets for the sender to send. For example, the sender will continue to send packets with higher IDs even if the receiver keeps telling it that it is waiting for packet no.11. Therefore, we can adjust the receiver so that if it receives incorrect packets for a repeated number of times, it will shut off the socket. Another way to secure the receiver is to set a maximum file size to be accepted. This is to prevent more malicious attacks where one file is being resent constantly. The maximum file size would be set based on the expected range of file sizes for the specific use case of the receiver.

VIII. Regulations and Laws:

EU regulations and laws:

Cybercrime tariffs are now patchwork throughout the 28 EU member states. For attempting to illegally access information systems, the ruling sets countrywide maximum punishments of at least two years in jail. Attacks against infrastructure such as power plants, transportation, or government networks will carry a maximum penalty of five years or more, which is higher than the present tariff in most member states. The verdict also strengthens the penalty for illegally intercepting communications, as well as developing and selling the instruments to do so.

Local regulations:

Imprisonment for a period of not less than six months and a fine of not less than one hundred thousand pounds and not exceeding five hundred thousand pounds, or either of these two penalties, whoever intentionally causes the suspension or disruption of an information network, or limits the efficiency of its work, or disrupts it, or obstructs it, or Intercepting its work, or unlawfully conducting electronic processing of its data.

Economic impact:

Although not limited to this, cyber-attacks might result in unexpectedly substantial economic and productivity losses. The cost of malware eradication, investigation, and incident management is the responsibility of the affected firms. Furthermore, businesses may not be able to recover from all cyber-attacks; data loss or trade secret theft can be fatal for industries that rely on product quality and secrecy heavily. Many businesses will also have to deal with their weakened market position and credibility.

The impact of cybercrime on the financial industry and small and medium businesses (SMEs) comes at a delicate time, particularly in Europe, when businesses are striving to cope with strict austerity measures and low revenues.

Social Impact:

While the financial impact of cybercrime has been debated, the communal implications of cybercrime have received far less attention. Psychologists and psychiatrists can help victims cope with the consequences of identity theft, sexual abuse, or financial ruin, while sociologists are well-positioned to examine the broader community implications and interpretations of cyber crime.

Cyber criminals take use of the Internet's anonymity, secrecy, and interconnection to assault the core underpinnings of our modern information society. Botnets, computer viruses, cyberbullying, cyberstalking, identity theft, malware, and spam are all examples of cybercrime.