**Go-Back-N and Selective Repeat Protocols**

# **Programming Language**

**Python Version 2.7**

***(Note:*** *The code does not work in Python 3.5****)***

# **Objective**

The objective of this project is to implement sliding window protocols – Go-Back-N and Selective Repeat protocols using an Unreliable Datagram Protocol (UDP) method. Our goal in project is to simulate checksum error, packet loss and ACK loss during the data transfer between sender and receiver.

# **Unreliable Datagram Protocol (UDP)**

UDP is a transport layer protocol that provides best effort is transferring data between systems. UDP offers unreliable service that provides no guarantee in data delivery and prevention of data duplication. Compared to other reliable data transfer methods like TCP, UDP has unique way of data transfer. UDP does not establish connection between End-to-End systems and does not provide any form of congestion control or communication security for the data transfer. These characteristics contributes to efficient data communication for some application like media streaming and online gaming.

We used existing socket programming from Project-1 and UDP connection on top of that to achieve our goal in this project.

# **Difference between Go-Back-N and Selective Repeat Protocols**

|  |  |  |
| --- | --- | --- |
|  | **Go-Back-N Protocol** | **Selective Repeat Protocol** |
| **Functionality** | Go-Back-N protocol retransmits all frames that are sent after a frame that was suspected to be lost or corrupted | Selective Repeat protocol retransmit only the frames that are suspected to be lost or corrupted |
| **Sender Window Size** | Maximum value of N-1, where N is number of sequence numbers | Maximum value of (N/2), where N is number of sequence numbers |
| **Receiver Window Size** | 1 | Maximum value of (N/2), where N is number of sequence numbers |
| **Receiver Storage** | Since receiver’s maximum window size is 1, it does not require any storage for received frames | Receiver stores all frames that are received after receiving a corrupted frame until the corrupted frame is replaced by a valid frame |
| **Bandwidth Utilization** | Bandwidth utilization is high if the error rate is high since the protocol need to send all frames even though received correctly after the reception of corrupted frame | Bandwidth utilization is minimum since the protocol retransmits only corrupted frames |

***Reference:*** [***http://techdifferences.com/difference-between-go-back-n-and-selective-repeat-protocol.html***](http://techdifferences.com/difference-between-go-back-n-and-selective-repeat-protocol.html)

# **Implementation and Execution**

As mentioned in the project requirements, following probability setup is made in both protocols

* **Probability of checksum error** is set to **0.1**
* **Probability of packet loss** is set to **0.1**
* **Probability of ACK loss** is set to **0.05**
* Checksum error is introduced by the sender before sending the packet to the receiver. Receiver detects problem in the packet and requests the same packet
* Similarly, ACK loss is handled in the Sender side. When ACK is received, sender purposefully drops the ACK and waits for timeout of the packet. Once the timeout occurs, sender resends the packet
* Packet loss is handled in the Receiver side. When a packet is received, receiver purposefully drops the packet. Sender waits for ACK for the sent packet until timeout. Once the timeout occurs, sender resends the packet

## **1. Go-Back-N**

* Start Receiver with the command:

**python ServerApp.py -a <Sender-IP> -b <Sender-Port> -x <Receiver-IP> -y <Receiver-Port> -m <Max\_Sequence\_Number\_Bits> -t <Timeout>**

* Start Sender with the command:

**python ClientApp.py -f <FileName>-a <Sender-IP> -b <Sender-Port> -x <Receiver-IP> -y <Receiver-Port> -m <Max\_Sequence\_Number\_Bits> -s <Max\_Segment\_Size> -n <Total\_Packets> -t <Timeout>**

where **Max\_Sequence\_Number\_Bits** is number of sequence numbers to be considered, **Max\_Segment\_Size** is number of segments the given file have to be split and **Total\_Packets** is total packets of the file to be sent

* + All the above values are set to default values. Default values are:
* Sender-IP: 127.0.0.1
* Receiver-IP: 127.0.0.1
* Sender-Port: 8081
* Receiver-Port: 8080
* Max\_Sequence\_Number\_Bits: 2
* Max\_Segment\_Size: 1500
* Total\_Packets: ALL
* Timeout: 10
* File\_Name: Index.html (Attached in the project)

**Note:** The project also contains default file locations (Data folder in the project) for the sender and receiver.

* + Simple execution commands:

**Receiver:**

**python ServerApp.py -a 127.0.0.1 -b 8081 -x 127.0.0.1 -y 8080 -m 3**

**Sender:**

**python ClientApp.py -f index.html -a 127.0.0.1 -b 8081 -x 127.0.0.1 -y 8080 -m 3 -s 1600 -t 15**

**(*Note:*** *-m <Max\_Sequence\_Number\_Bits> has to be set as equal values in both sender and receiver end***)**

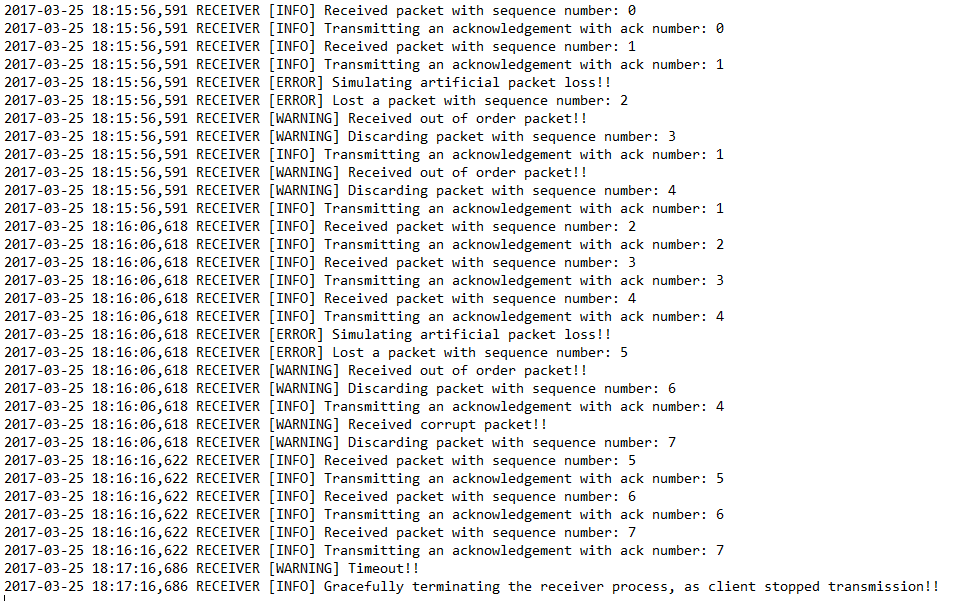
* + Results for above commands are given in Figure-1 for Sender and Figure-2 for Receiver:

**Sender:**



*Figure 1: Go-Back-N Client Execution for the given command with bit error and ACK loss simulation*

* + **Receiver:**



*Figure 2: Go-Back-N Receiver side execution for the given command with packet loss simulation*

**Note:** Once the Receiver terminates, the transferred file will be available in Receiver’s location.

## **2. Selective Repeat**

* + Start Receiver with the command:

**python ServerApp.py -a <Sender-IP> -b <Sender-Port> -x <Receiver-IP> -y <Receiver-Port> -m <Max\_Sequence\_Number\_Bits> -t <Timeout>**

* Start Sender with the command:

**python ClientApp.py -f <FileName>-a <Sender-IP> -b <Sender-Port> -x <Receiver-IP> -y <Receiver-Port> -m <Max\_Sequence\_Number\_Bits> -s <Max\_Segment\_Size> -n <Total\_Packets> -t <Timeout>**

where **Max\_Sequence\_Number\_Bits** is number of sequence numbers to be considered, **Max\_Segment\_Size** is number of segments the given file have to be split and **Total\_Packets** is total packets of the file to be sent

* All the above values are set to default values. Default values are:
* Sender-IP: 127.0.0.1
* Receiver-IP: 127.0.0.1
* Sender-Port: 8081
* Receiver-Port: 8080
* Max\_Sequence\_Number\_Bits: 2
* Max\_Segment\_Size: 1500
* Total\_Packets: ALL
* Timeout: 10
* File\_Name: Index.html (Attached in the project)

**Note:** The project also contains default file locations (Data folder in the project) for the sender and receiver.

* Simple execution commands:

**Receiver:**

**python ServerApp.py -a 127.0.0.1 -b 8081 -x 127.0.0.1 -y 8080 -m 3**

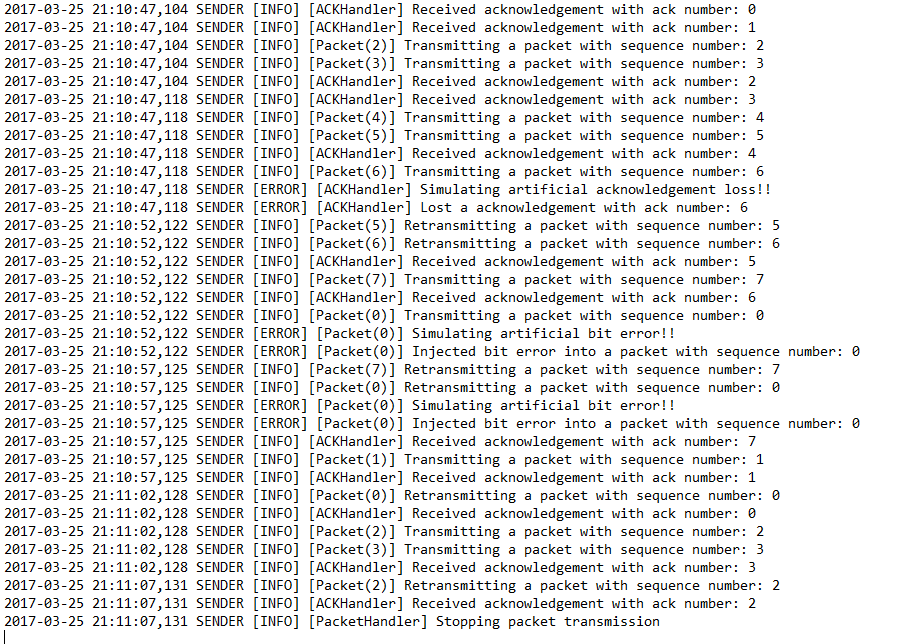
**Sender:**

**python ClientApp.py -f index.html -a 127.0.0.1 -b 8081 -x 127.0.0.1 -y 8080 -m 3 -s 1000 -n 20 -t 5**

**(*Note:*** *-m <Max\_Sequence\_Number\_Bits> has to be set as equal values in both sender and receiver end***)**

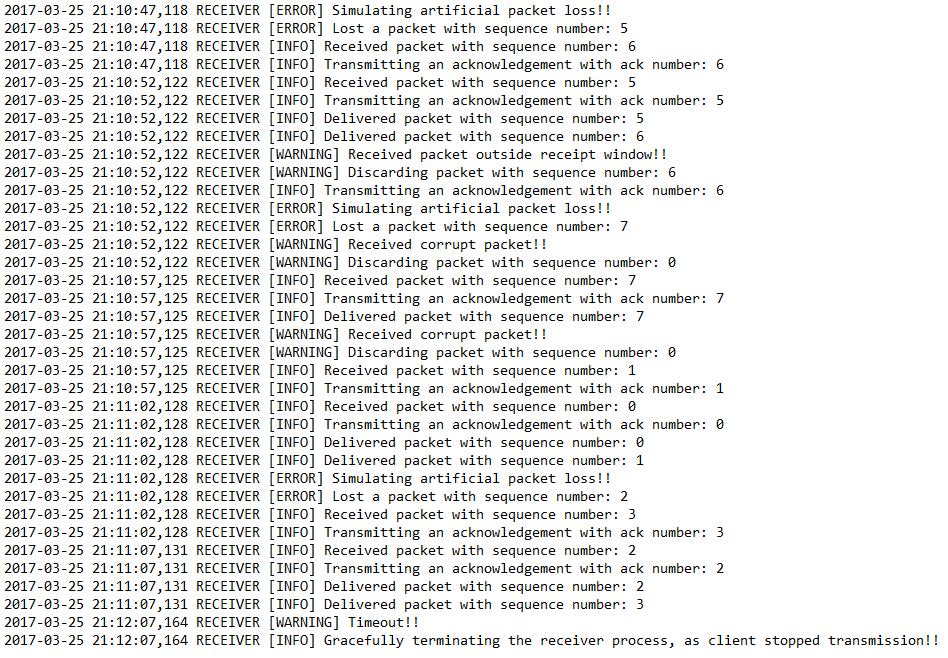
* Results for above commands are given in Figure-3 for Sender and Figure-4 for Receiver:

**Sender:**



*Figure 3 Selective Repeat Sender side execution with ACK loss and bit error simulation*

**Receiver:**



*Figure 4: Selective Repeat Receiver side execution for the given command with Packet loss simulation*

**Note:** Once the Receiver terminates, the transferred file will be available in Receiver’s location.