# Wide Area Networks: Project 1

## Reliable UDP Transport

## Final Design

**Ashwini Ananda**

**Jayati Halder Jui**

1. **Message Types**: We have three types of messages; Data messages and Acknowledgements (ACK and NACK). We have just one data structure to represent them all.

struct packet:

| **Data Type** | **Field** | **Comment** |
|---|---|---|
| Int | Sequence No. | Sequence Number of packets |
| Int | ACK No | Acknowledgement number based on sequence no. |
| Int | Size | Data length |
| Struct timeval | Ts | Timestamp |
| Int | ACK Flag | ACK = 1, NACK = 0 |
| Char* [1368] | Data_buf | Data read from file |

2. **Data Structure for Sender and Receiver:**
   The Sender and Receiver use ARRAY (packet*) of size 200 to store the packets.

3. **Design (Sender)**

```
1.  Initialize WINDOW
2.  Send Filename and Wait for Acknowledgement.
        A.  If ACK doesn’t come in 1000 microsecond, send again.
        B.  IF ACK.ACK_NO == -1, Receiver is Busy. Sleep for 3 minutes and try again.
        C.  If ACK comes for the filename, continue to next step.
        D.  If NACK comes for filename, re-send filename again.
3.  While(true)
        A.  Read Segments from file to fill the WINDOW Buffer.
        B.  Create packets using the segments read from file, add sequence numbers, sizes of data segments and
            timestamps.
        C.  Fill the WINDOW using the packets.
        D.  Listen to socket for acknowledgements (ACKS/NACKS):
                i.   If explicit NACK comes for a packet, re-send the packet from WINDOW Buffer
                ii.  If an ACK comes:
                        a)  If the ACK is for a sequence less than the last sequence in the WINDOW, re-send the
                            rest of the packets from the WINDOW
                        b)  If an ACK comes for the last packet in the WINDOW,
                            Clear WINDOW.
        E.  If EOF reached and ACK has arrived for the last packet, exit. Else, Continue.
```

## 4. Design (Receiver)

```
1. Initialize WINDOW
2. Next_Expected_Seq = 0;
3. While(true)
    A. If another packet come from another Sender, Block it if the previous transfer isn't
       complete. Else, allow the transfer and move to next step.
    B. Listen to socket for getting the Destination Filename.
        i. If Timeouts, send a NACK for filename.
        ii. Else Send ACK for filename.
    C. Listen to the socket for data packet.
    D. Place packet in WINDOW buffer at index Packet.Seq_No % WINDOW_SIZE.
    E. If Packet.Seq_no >= Next_Expected_Sequence:
        i. Send explicit NACKS for missing packets.
        ii. Next_Expected_Sequence = Packet.Seq_No + 1
    F. If, Packet.Seq_No < Next_Expected_Sequence:
        i. If BUFFER index at index Packet.Seq_No % WINDOW_SIZE isn't empty,
           discard the packet. It is a duplicate packet. Else,
        ii. Put the packet in the BUFFER at index Packet.Seq_No % WINDOW_SIZE.
            This Packet is response to a NACK.
    G. If WINDOW is FULL of EOF reached:
        i. Write Data from packets in order to Destination File.
        ii. Send Cumulative ACK for last received sequence..
    H. If Timeouts:
        i. Send Explicit NACKS for Missing Sequences.
        ii. If WINDOW is full or EOF reached, send cumulative ACK for the last
```