# Lecture 7 (Transport Layer pt 2)

## 1 Reliable Data Transfer

Different "functions" are implemented for different components of the communication:

1. rdt_send() - reliable data transfer protocol
2. udt_send() - simple UDT transfer
3. red_rcv() - receiver implementation of rdt
4. deliver_data() - parse message

For incremental development of rdt, FSMs are used

### 1.1 rdt 1.0

Assumes no bit error and no loss of packets. The FSM is simply waiting for call from above/below and processing on call

### 1.2 rdt 2.0

1. Handles bit errors
2. To recover from errors, ACKs (acknowledgements) are sent by receiver
3. Retransmission happens on receiving NAKs (negative acknowledgements)
4. It is a stop and wait protocol
5. FSM has two states for sender - wait for call and then wait for ACK/NAK
6. Only one state for receiver but two edges (Mealy FSM)
7. Issue occurs when ACK/NAK packet is corrupted, retransmission might lead to corruption of the packet

### 1.3 rdt 2.1

1. To resolve the issue of `rdt2.0`, sequence number is added to the packet and retransmission is done for corrupted ACK/NAK
2. FSM has 4 states for sender and 2 states for receiver (for receiving sequence 0 and 1)
3. Two sequence numbers suffice since it avoids receiving duplicate packets

## 1.4 rdt 2.2

1. The usage of NAK is removed by sending a duplicate ACK (this is used by TCP)
2. The FSM is modified accordingly

## 1.5 rdt 3.0

1. Handling both data loss and packet loss
2. Sender waits for some time for receiving ACK and then resends packet
3. Timeout is added on sender's side and no change in receiver's FSM

### 1.5.1 Performance of rdt 3.0

The utilisation is given as:

$$U_{sender} = \frac{d_{trans}}{d_{trans} + 2d_{prop}}$$

This ratio is very very small and hence the sender is being under utilised

### 1.5.2 Pipelining in rdt 3.0

1. Sending multiple in-flight packets
2. Requires buffers
3. Utilisation increases by a factor of number of packets that can be sent at once

# 2 Go-Back-N Protocol

1. Sender has a window of N packets which are yet to be ACK-ed, they can be in transit or yet to be sent
2. Receiver also sends cumulative ACK, upto the largest sequence number it has received
3. Timeout happens for oldest in-flight packet
4. Retransmission happens for all n packets on timeout

## 2.1 Selective Repeat

1. Receiver individually ACKs each packet it has received
2. Sender individually sends the un-ACKed packets instead of all
3. Modulo n sequence numbers are used
4. Issue happens when ACK arrives late and retransmission occurs, receiver doesn't know which packet it has received, older or newer
5. To resolve the issue, window size should be $\leq$ half of sequence number size