Module-2

Syllabus: Module-2

Introduction and Transport-Layer Services: Relationship Between Transport and Network Layers, Overview of the Transport Layer in the Internet, Multiplexing and Demultiplexing: Connectionless Transport: UDP, UDP Segment Structure, UDP Checksum, Principles of Reliable Data Transfer: Building a Reliable Data Transfer Protocol, Pipelined Reliable Data Transfer Protocols, Go-Back-N, Selective repeat, Connection-Oriented Transport TCP: The TCP Connection, TCP Segment Structure, Round- Trip Time Estimation and Timeout, Reliable Data Transfer, Flow Control, TCP Connection Management, Principles of Congestion Control: The Causes and the Costs of Congestion, Approaches to Congestion Control, Network-assisted congestion-control example, ATM ABR Congestion control, TCP Congestion Control: Fairness.

Question-1: Explain different services offered by the transport layer in TCP/IP protocol stack

Refer the answer from the module1.

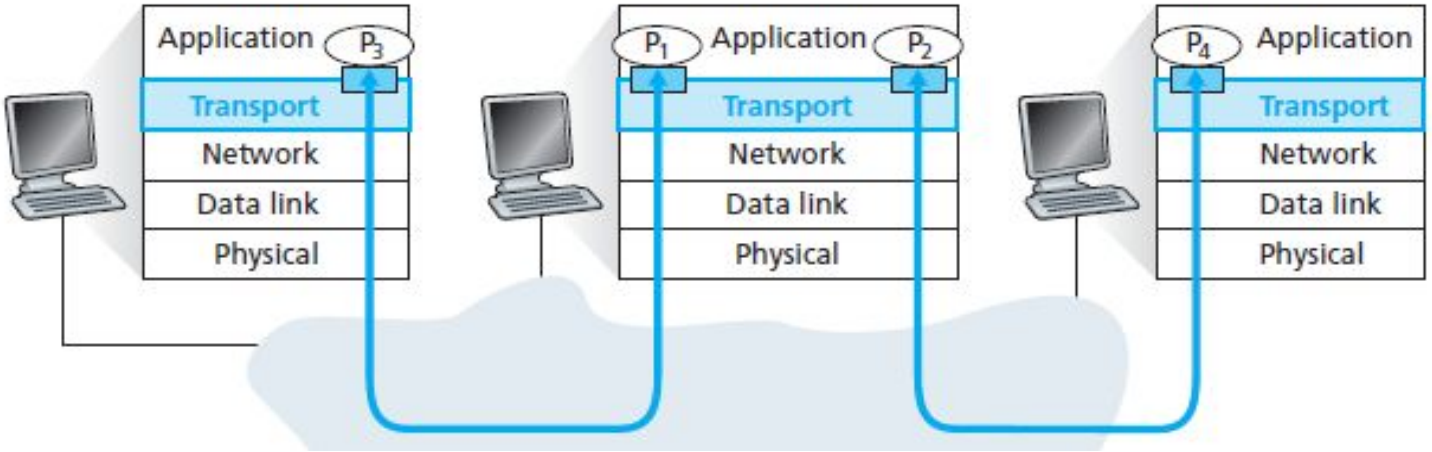
Include following points:

1. TCP and UDP based transport services.
2. Segmentation and reassembly
3. Flow control, congestion control and error control
4. Inter process communication using Sockets

2.Explain Multiplexing and Demultiplexing operation at transport Layer

Multiplexing:

- Transport layer applies multiplexing at the sender end.
- The job of gathering data chunks at the source host from different sockets, encapsulating each data chunk with header information to create segments, and passing the segments to the network layer is called multiplexing.



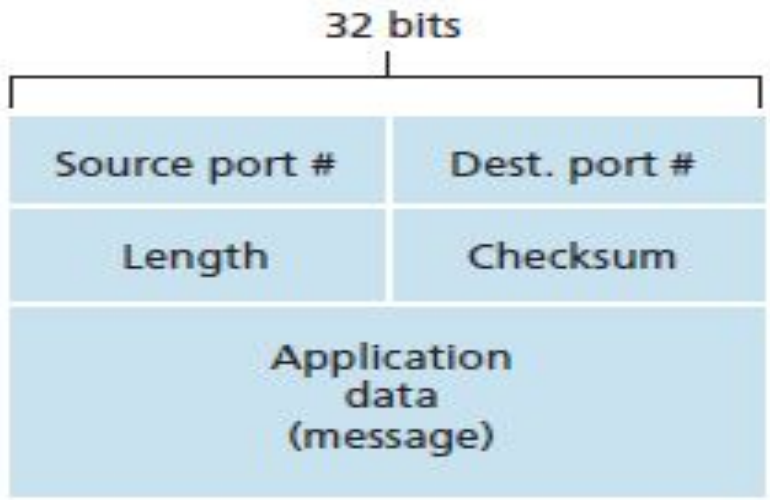
- Transport-layer multiplexing requires that sockets have unique identifiers, and that each segment have special fields that indicate the socket to which the segment is to be delivered.
- These special fields are the source port number field and the destination port number field.
- Multiplexing also enables the transport layer to receive the data from different application protocols and multiplex them and send it over network which uses only single protocol(IP) at network layer.

Demultiplexing:

- At the receiving end, the transport layer examines the received segment headers for verifying the source port number and the destination port number and to direct the data to the appropriate application.
- The process of gathering the segments (single protocol)and connecting the data to different applications at the higher layer(multiple protocols) is called as Demultiplexing.

3. Explain UDP segment structure

UDP segment structure:



- The UDP header has only four fields, each consisting of two bytes.
- The port numbers allow the destination host to pass the application data to the correct process running on the destination end system.

- The length field specifies the number of bytes in the UDP segment (header plus data).
- The checksum is used by the receiving host to check whether errors have been introduced into the segment.
- The application data occupies the data field of the UDP segment.
- Supports UDP transmission at transport layer for quick transmission services.

4. Write a short note on UDP checksum

UDP Checksum: The checksum is used to determine whether bits within the UDP segment have been altered as it moved from source to destination.

- UDP at the sender side performs the 1s complement of the sum of all the 16-bit words in the segment, with any overflow encountered during the sum being wrapped around.
- This result is put in the checksum field of the UDP segment.
- Then the segment will be transmitted by the sender.
- At the receiver, all four 16-bit words are added, including the checksum.
- If no errors are introduced into the packet, then clearly the 1's complement of the sum at the receiver will be having all the bits as zero.

Example:Suppose that we have the following three 16-bit words

Step1:Add all the data elements using binary addition (Modulo-2 addition). If you get extra bit wrap it.

```
0110011001100000
0101010101010101
1000111100001100
```

The sum of first two of these 16-bit words is

```
0110011001100000
0101010101010101
1011101110110101
```

When this sum is added with the third word

```
1011101110110101
1000111100001100
0100101011000010
```

Step 2: Take 1s complement of the result.

- The 1’s complement is obtained by converting all the 0s to 1s and converting all the 1s to 0s. Thus the 1s complement of the sum 0100101011000010 is 1011010100111101, which becomes the checksum.

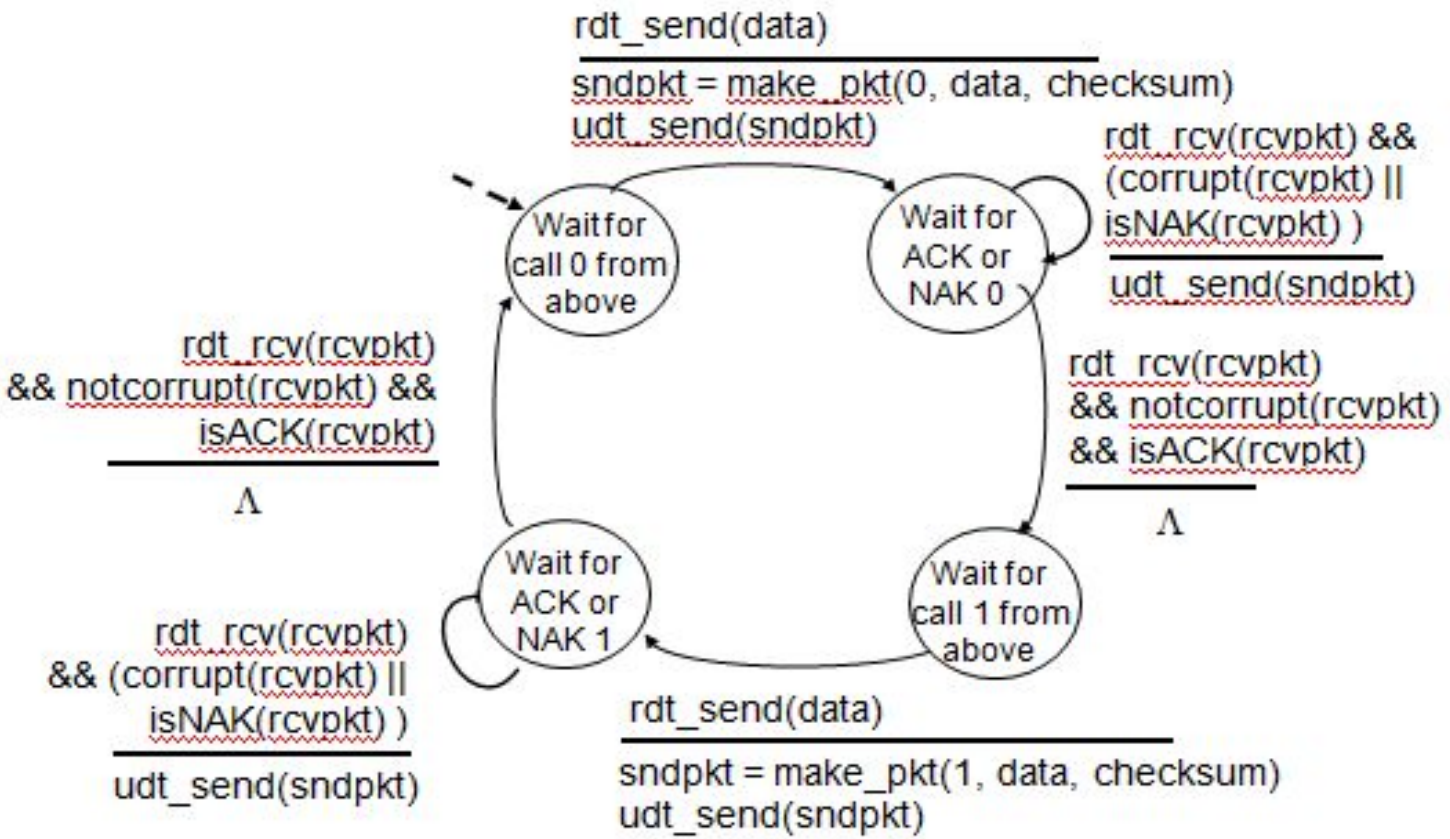
Step 3: Data along with checksum is transmitted to receiver.

Step 4: at the receiver side add all the data and checksum using binary addition. Wrap the extra bit and take 1s complement of the result. This will be the checksum. If checksum is all 0’s receiver has received error free data otherwise it has received corrupted data.

5. With FSM, explain the protocol development phase in rdt 2.1

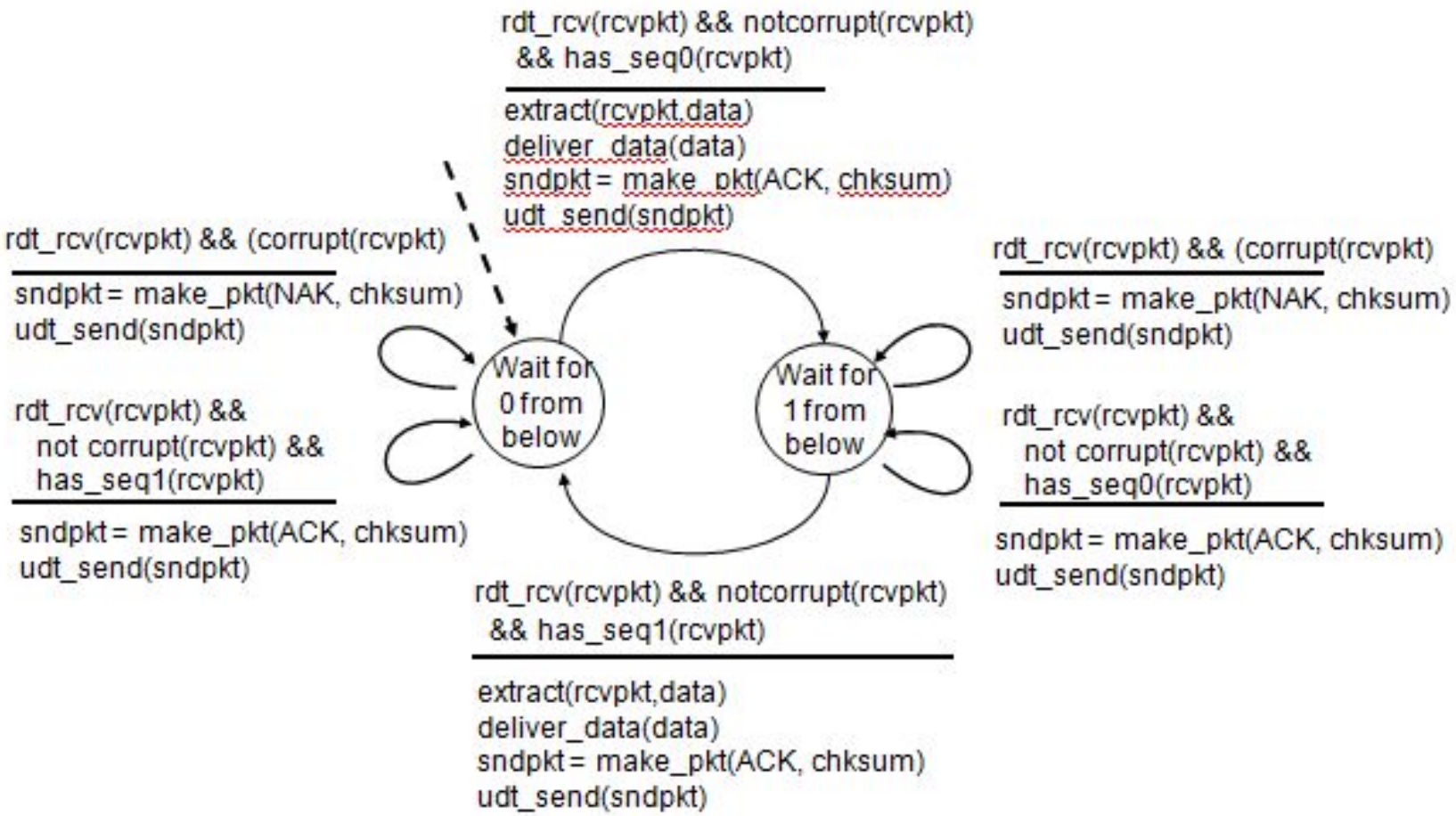
- Rdt 2.1 is designed to overcome the problem of identifying the arrival of duplicate packet at the receiver end.
- Rdt 2.1 has added the mechanism to handle the arrival of duplicate packets at the receiver.
- Handling of duplicate packets is carried out by enabling the following actions
  - sender retransmits current pkt if ACK/NAK corrupted
  - sender adds *sequence number* to each pkt
  - receiver discards (doesn't deliver up) duplicate pkt

rdt2.1 sender: handling garbled ACK/NAKs



- At the sender end, each packet before transmission, added with a sequence number ‘0’ or ‘1’ (the protocol design is for Stop and Wait).
- Checksum is added to help the receiver in detecting the transmission errors.
- Sender checks whether the received ACK/NAK is corrupted. If its got corrupted, then sends the saved copy of the current packet. It's the receiver’s job, whether to receive the arrived packet or to discard it, after verifying the sequence number of the packet it has received.

rdt2.1: receiver, handling garbled ACK/NAKs

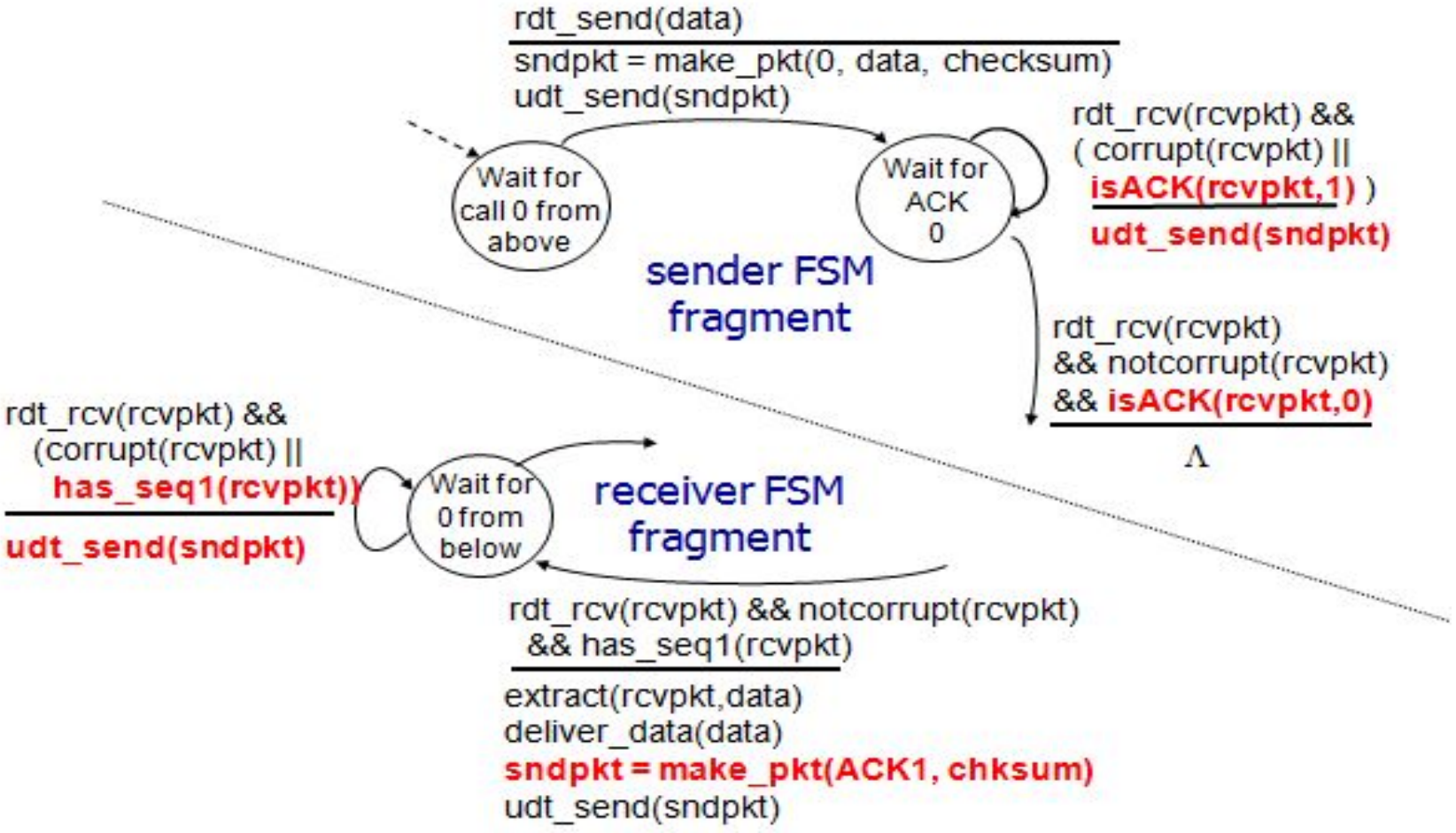


- The receiver in rdt 2.1 must check if received packet is duplicate or new one.
  - state indicates whether 0 or 1 is expected pkt seq number
- note: receiver can *not* know if its last ACK/NAK received OK at sender as sender will decide which packet to send next based on the correctness of the received reply(ACK/NAK)

6. With FSM, explain the protocol development phase in rdt 2.2

- **rdt2.2: is a NAK-free protocol.**
- Designed to implement same functionality as rdt2.1(Handling the duplicate packets), using ACKs only.
- instead of NAK, receiver sends ACK for last packet received OK
    - receiver must *explicitly* include seq. number of packet being acknowledged.
-

rdt2.2: sender, receiver fragments



- The receiver must now include the sequence number of the packet being acknowledged by an ACK message (this is done by including the ACK,0 or ACK,1 argument in make_pkt() in the receiver FSM).
- The sender must now check the sequence number of the packet being acknowledged by a received ACK message (this is done by including the 0 or 1 argument in isACK()in the sender FSM).

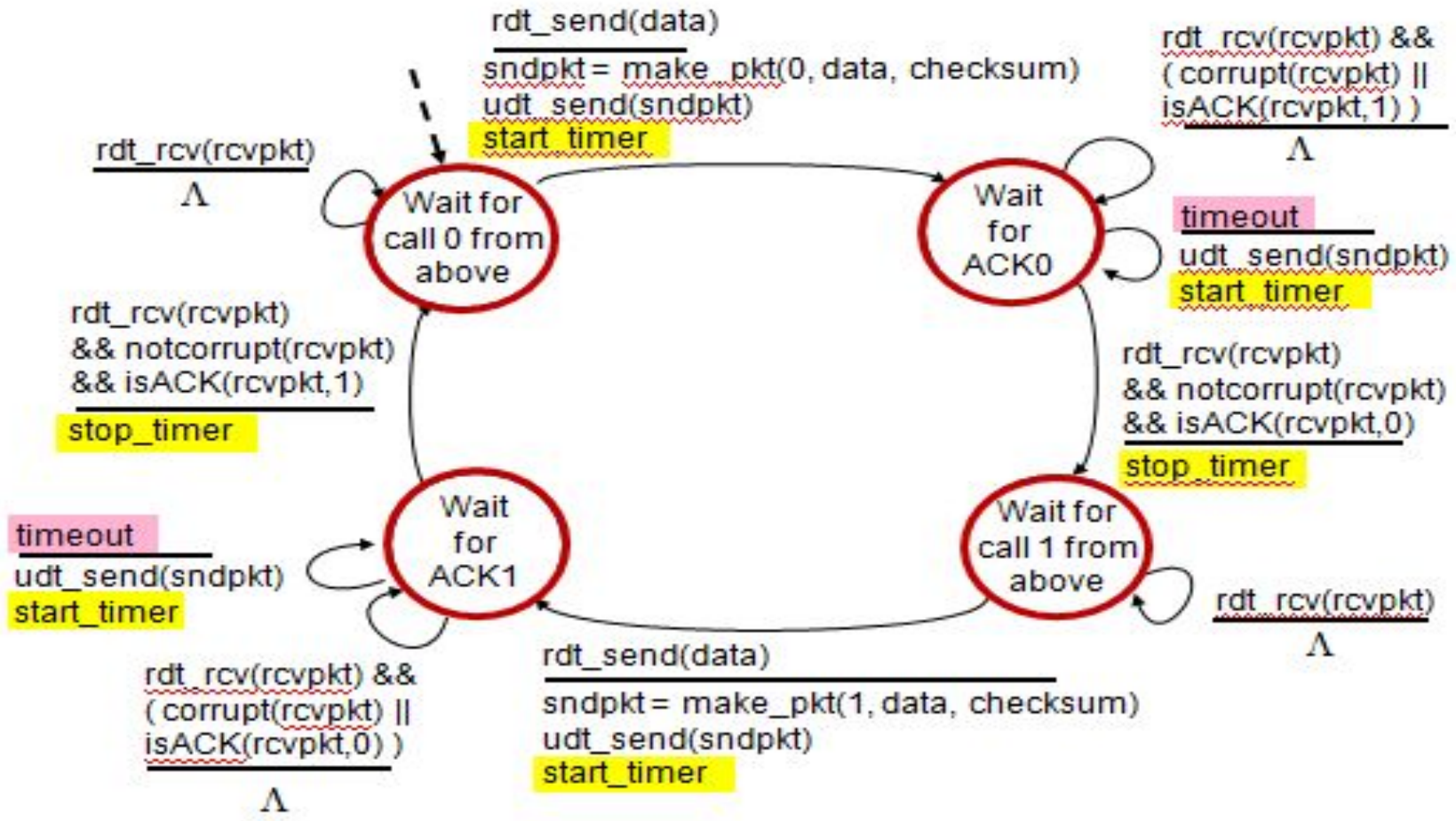
7. With FSM, explain the protocol development phase in rdt 3.0

Rdt 3.0 is designed to achieve reliable data transfer over a lossy channel with bit errors

Approach: sender waits “reasonable” amount of time for ACK, retransmits if no ACK received in this time. If pkt (or ACK) just delayed (not lost):

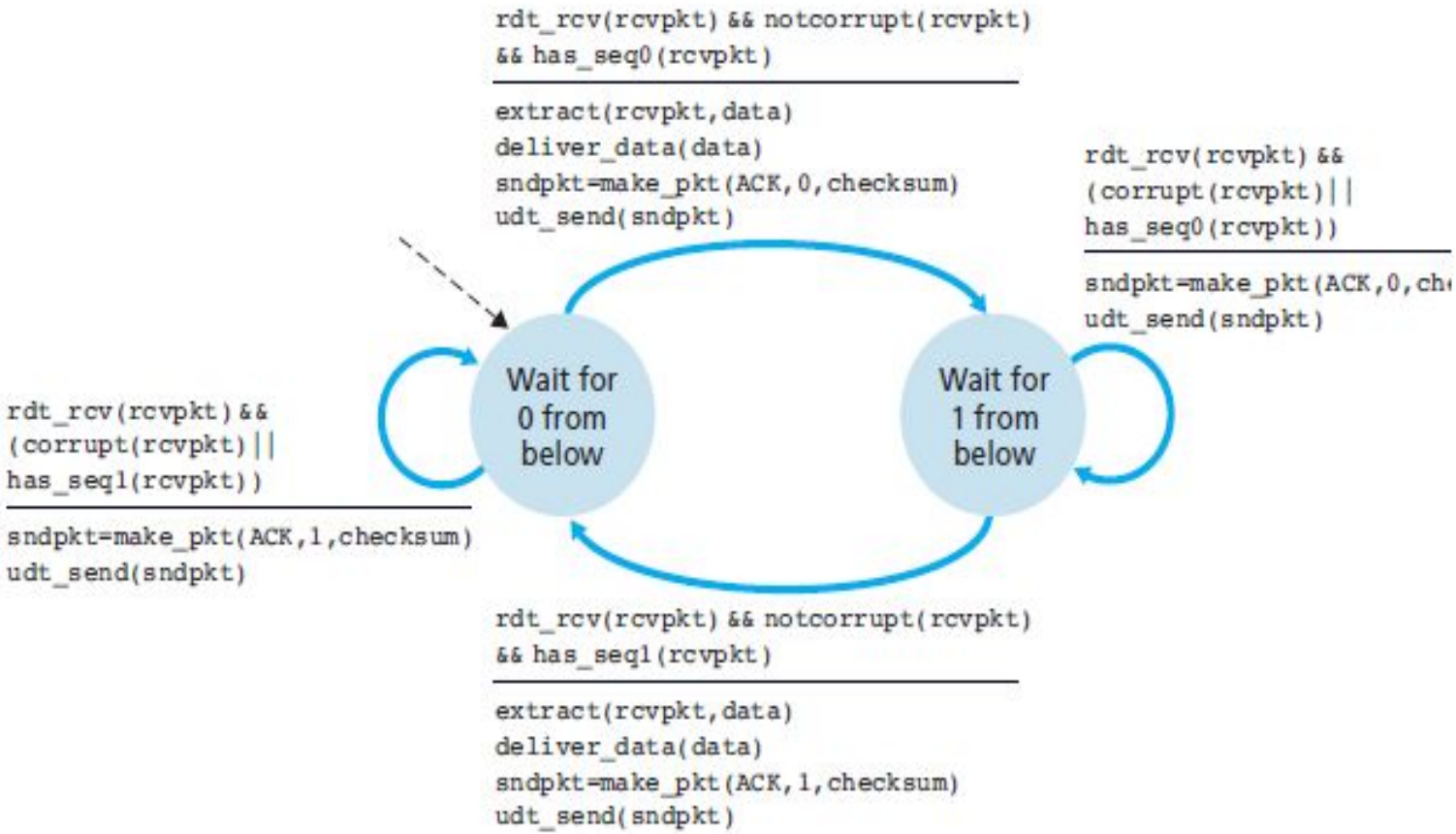
- retransmission will be duplicate, but seq numbers already handles this.
- receiver must specify seq number of packet being Acknowledged.
- uses countdown timer to interrupt after “reasonable” amount of time

Rdt 3.0 Sender



- Suppose that the sender transmits a data packet and either that packet, or the receiver's ACK of that packet, gets lost. In either case, no reply is forthcoming at the sender from the receiver.
- After waiting for the set amount of time(RTT-Timeout), If an ACK is not received at the sender, then the same packet is retransmitted. Note that if a packet experiences a particularly large delay, the sender may retransmit the packet even though neither the data packet nor its ACK have been lost.

Rdt 3.0 Receiver



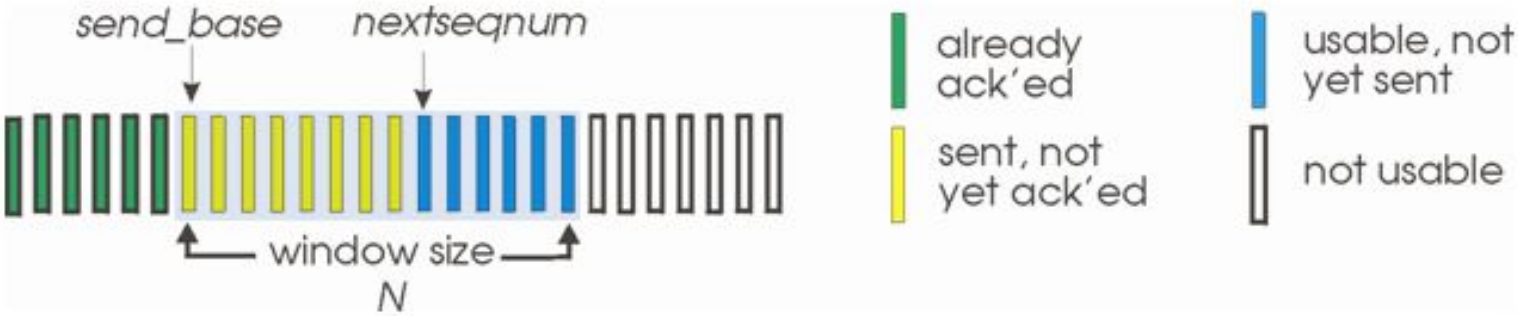
- When the receiver generates an ACK, it will copy the sequence number of the data packet being ACK'ed into this acknowledgement field.
- By examining the contents of the acknowledgment field, the sender can determine the sequence number of the packet being positively acknowledged.

8. Explain the working of Go Back-N protocol

- GO-Back-N is a pipelined reliable data transmission protocol.
- Works using Sliding-Window approach. Uses cumulative acknowledgements.
- In Go back N, sender window size is N and receiver window size is always 1.
- The sender is allowed to transmit multiple packets (when available) without waiting for an acknowledgment, but is constrained to have no more than some maximum allowable number, N, of unacknowledged packets in the pipeline.
- Sender decides how many packets to send, based on the allowed window size.

Go-Back-N: sender

- sender: "window" of up to N, consecutive transmitted but unACKed pkts
  - k-bit seq # in pkt header

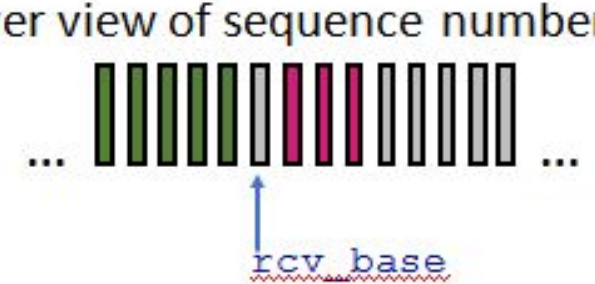


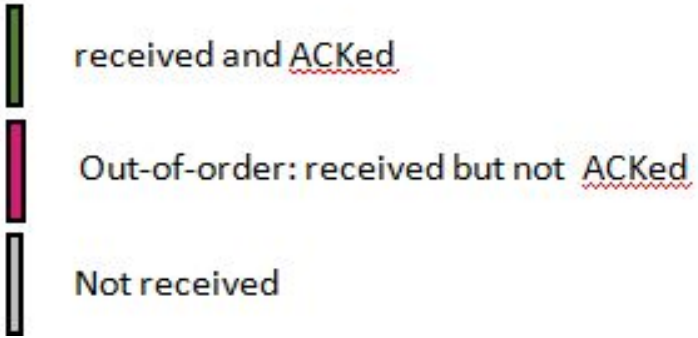
- *cumulative ACK:* ACK(*n*): ACKs all packets up to, including seq # *n*
  - on receiving ACK(*n*): move window forward to begin at *n+1*
- timer for oldest in-flight packet
- *timeout(n):* retransmit packet n and all higher seq # packets in window

Go-Back-N: receiver

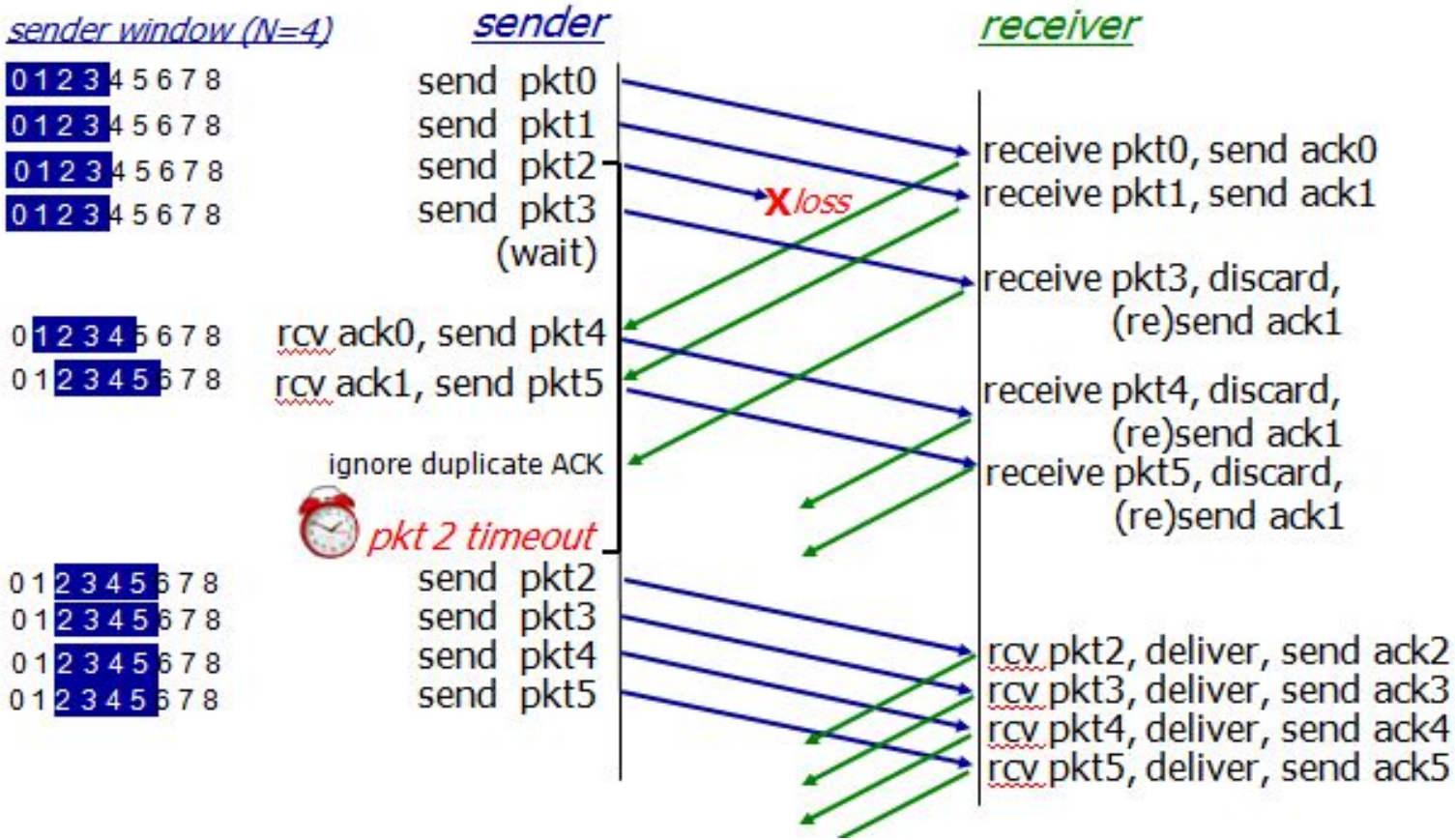
- ACK-only: always send ACK for correctly-received packet so far, with highest *in-order* seq #
  - may generate duplicate ACKs
  - need only remember `rcv_base`
- on receipt of out-of-order packet:
  - can discard (don't buffer) or buffer: an implementation decision
  - re-ACK pkt with highest in-order seq #

Receiver view of sequence number space:





Go-Back-N in action



Add explanation based on the fig. Given in the previous slide

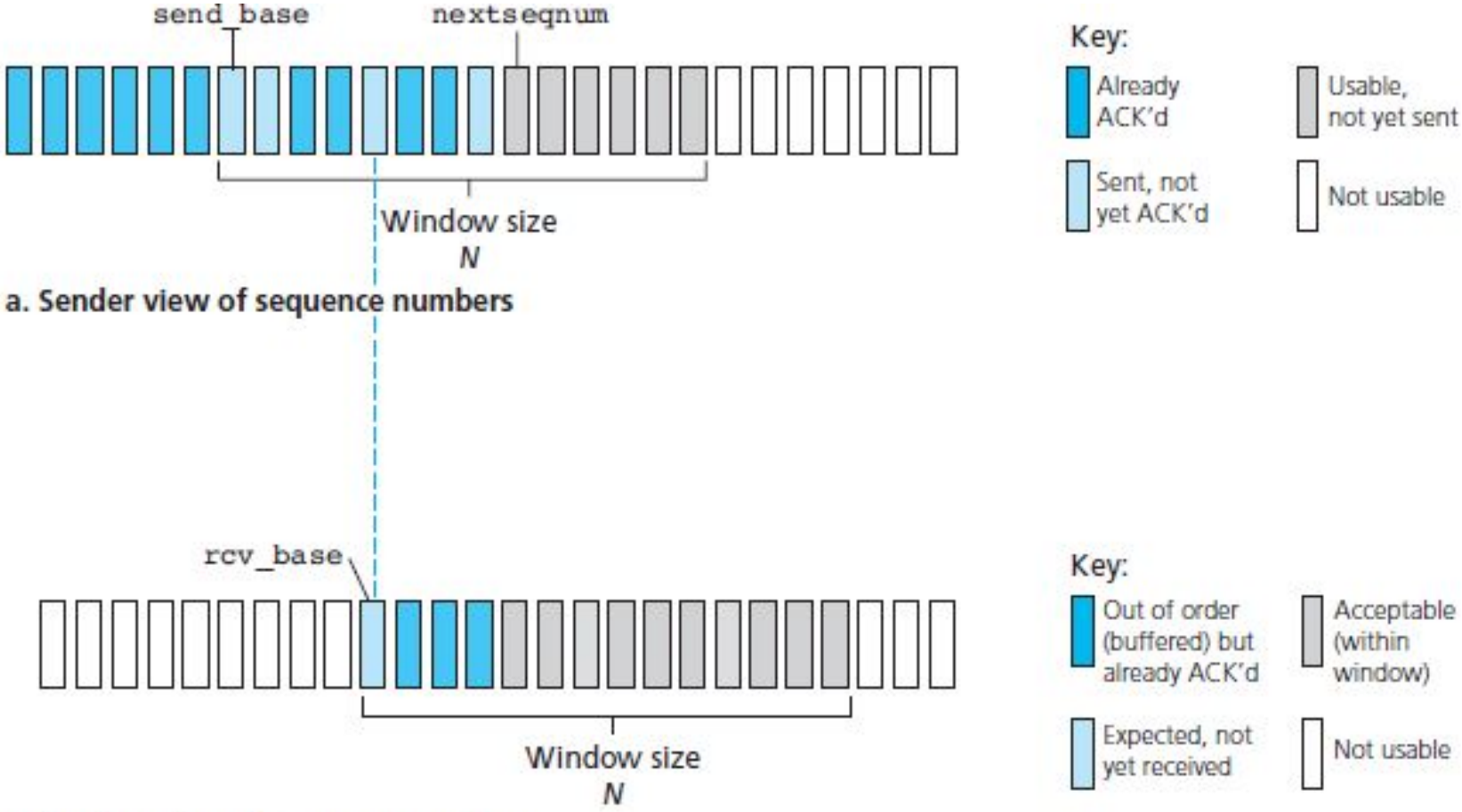
Or

Even you can write the answer by referring the FSM given in the text book.

9. Explain the working of Selective-Repeat Protocol

Selective-Repeat is a pipelined reliable data transmission protocol.

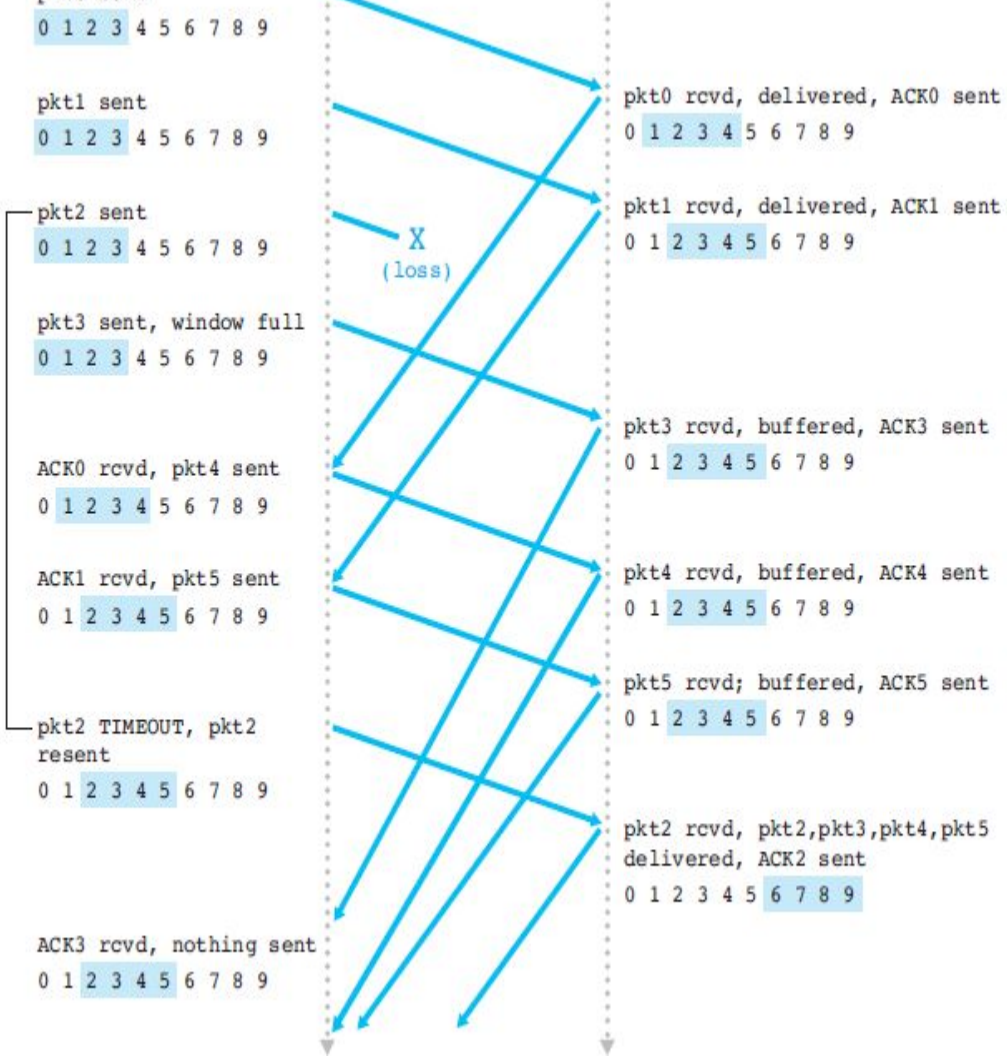
- Works using Sliding-Window approach. Uses cumulative acknowledgements.
- Developed to overcome the drawbacks of Go-Back-N, to avoid unnecessary retransmissions.
- A window of size-N is defined to limit the number of outstanding and unacknowledged packets in the pipeline.
- The SR receiver will acknowledge a correctly received packet whether or not it is in order. Out-of-order packets are buffered until any missing packets are received, at which point a batch of packets can be delivered to the upper layer in the sequence order.



a. Sender view of sequence numbers

b. Receiver view of sequence numbers

Working Example:



SR-Sender: Events and Actions

1. Data received from application layer: When receives data from the application, adds next available sequence number to it. Adds checksum for error detection at the receiver end. Checks whether the sequence number is within the limit of the window. If so, data is packetized and sent. If not, it is buffered for later transmission.
2. ACK Received: If the sender receives the ACK, marks the packet as acknowledged and then slides the window toward the new unacknowledged frame having smallest number. The untransmitted packets from the slided window would be transmitted by the sender.
3. Timeout: If the sender has not received ACK within the anticipated time and the timer expires, the buffered copy of the current frame will be retransmitted.

SR-Receiver:Events and Actions

1. Receiving the packet having the sequence number within the logical window limit, buffers it and sends the acknowledgement to the sender. Received in-sequence packets are pushed to the upper layer. Then slides the window to the next set of expected sequence numbers.
2. Discarding of duplicate packet: If the arrived packet has been already acknowledged and buffered at the end of receiver, then it would be discarded silently without pushing it to the higher layer. This mechanism reduces the generation of duplicate packets at the receiving application.

10. Problem solving on RTT estimation(RTT estimation, Timeout calculations)

Problem: Suppose that two measured sample RTT values are 116 ms and 130ms. Compute: i) Estimated RTT after each of these sample RTT value is obtained. Consider weight factor, α=0.125 and estimated RTT is 100 ms just before first of the samples obtained. ii) Compute DevRTT and iii).Timeout values

Consider β=0.25 and DevRTT was 5 ms before first of these samples are obtained.

Given data:

Sample RTT=100ms

Sample DevRTT=5ms

$$\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$$

Where, α=0.125

$$\text{EstimatedRTT} = 0.875 \cdot \text{EstimatedRTT} + 0.125 \cdot \text{SampleRTT}$$

$$\text{DevRTT} = (1 - \beta) \cdot \text{DevRTT} + \beta \cdot |\text{SampleRTT} - \text{EstimatedRTT}|$$

Where β=0.25

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \cdot \text{DevRTT}$$

For the sample RTT 116ms,

Estimated RTT=0.125*116+0.875*100 =102ms

DevRTT= 0.25*(116-102)+0.75*5=7.25ms

Timeout RTT=102+4*7.25=131ms

For the sample RTT 130ms

Estimated RTT=0.125*130+0.875*102 =105.5ms

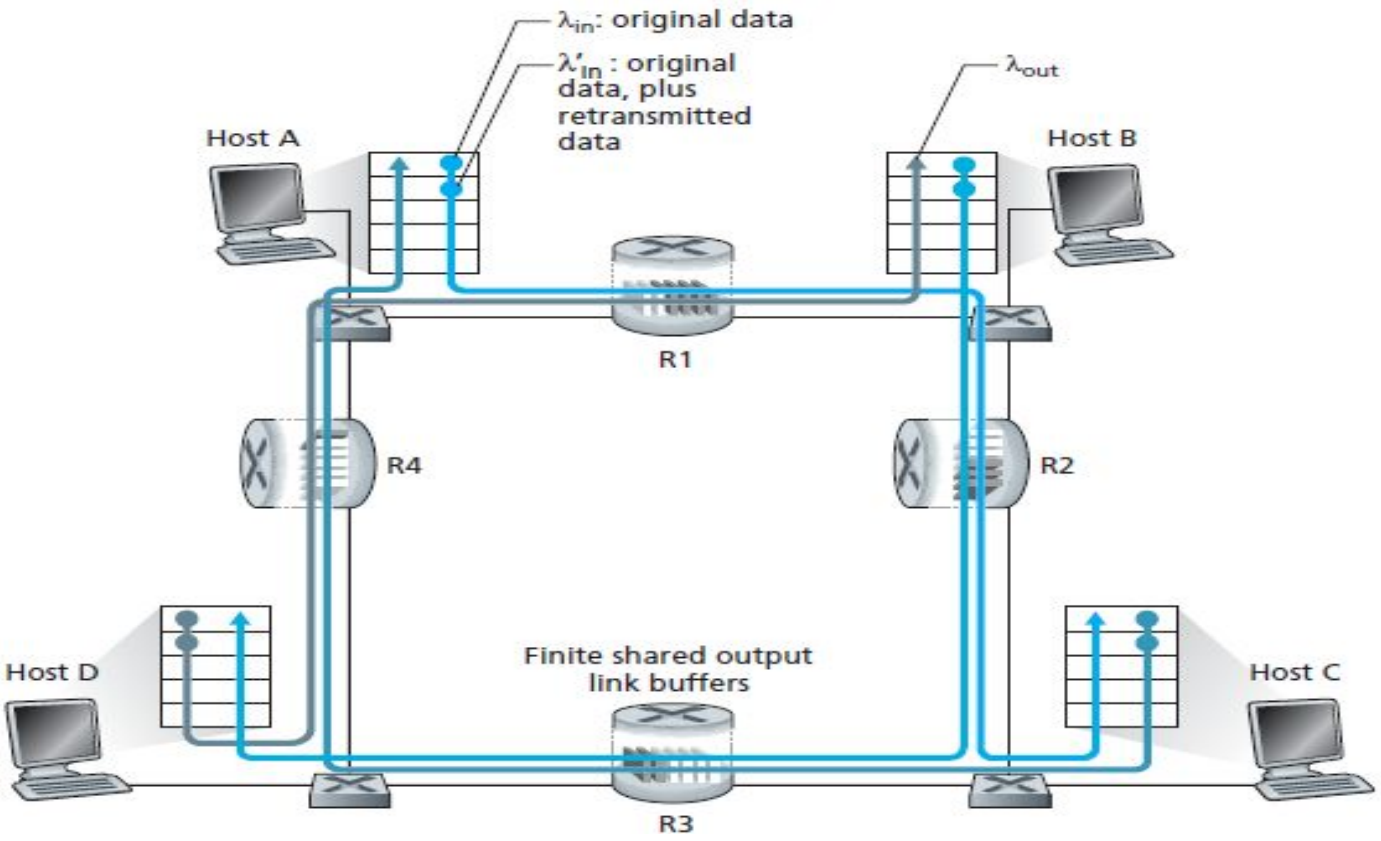
DevRTT= 0.25*(130-105.5)+0.75*7.25=11.5625ms

Timeout RTT=105.5+4*11.5625=151.75ms

11. Explain causes and cost of congestion with 4 routers with limited buffer size. (Explain Delay and throughput graphs also)

Scenario 3: Four Senders, Routers with Finite Buffers, and Multihop Paths

- In this congestion scenario, four hosts transmit packets, each over overlapping two-hop paths, as shown in Figure below.
- The major causes of congestion are:
  - four senders (Number of nodes)
  - multi-hop paths
  - timeout/retransmit(Transmission delays, packet drops or duplications)
- Costs(consequences) of Congestion are:
  - Decrease in throughput
  - Delay increases as capacity approached
  - loss/retransmission
  - un-needed duplicates of packets
  - upstream transmission capacity / buffering wasted for packets lost downstream

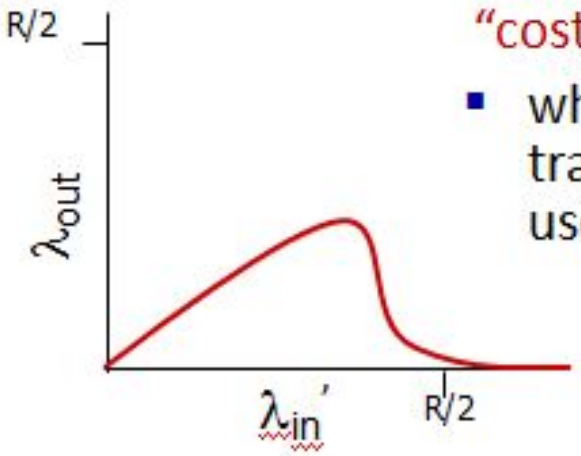

λ_{in}: original data
λ'_{in} : original data, plus retransmitted data
λ_{out}
Host A
Host B
R1
R4
R2
Host D
Finite shared output link buffers
Host C
R3

Let's consider the connection from Host A to Host C, passing through routers R1 and R2. The A–C connection shares router R1 with the D–B connection and shares router R2 with the B–D connection. For extremely small values of λ_{in}, buffer overflows are rare (as in congestion scenarios 1 and 2), and the throughput approximately equals the offered load. For slightly larger values of λ_{in}, the corresponding throughput is also larger, since more original data is being transmitted into the network and delivered to the destination, and overflows are still rare. Thus, for small values of λ_{in}, an increase in λ_{in} results in an increase in λ_{out}.

Having considered the case of extremely low traffic, let's next examine the case that λ_{in} (and hence λ'_{in}) is extremely large. Consider router R2. The A–C traffic arriving to router R2 (which arrives at R2 after being forwarded from R1) can have an arrival rate at R2 that is at most R, the capacity of the link from R1 to R2, regardless of the value of λ_{in}. If λ'_{in} is extremely large for all connections (including the B–D connection), then the arrival rate of B–D traffic at R2 can be much larger than that of the A–C traffic. Because the A–C and B–D traffic must compete at router R2 for the limited amount of buffer space, the amount of A–C traffic that successfully gets through R2 (that is, is not lost due to buffer overflow) becomes smaller and smaller as the offered load from B–D gets larger and larger. In the limit, as the offered load approaches infinity, an empty buffer at R2

is immediately filled by a B–D packet, and the throughput of the A–C connection at R2 goes to zero. This, in turn, implies that the A–C end-to-end throughput goes to zero in the limit of heavy traffic.

Causes/costs of congestion: scenario 3



"cost" of congestion:

- when packet dropped, any upstream transmission capacity and buffering used for that packet was wasted!

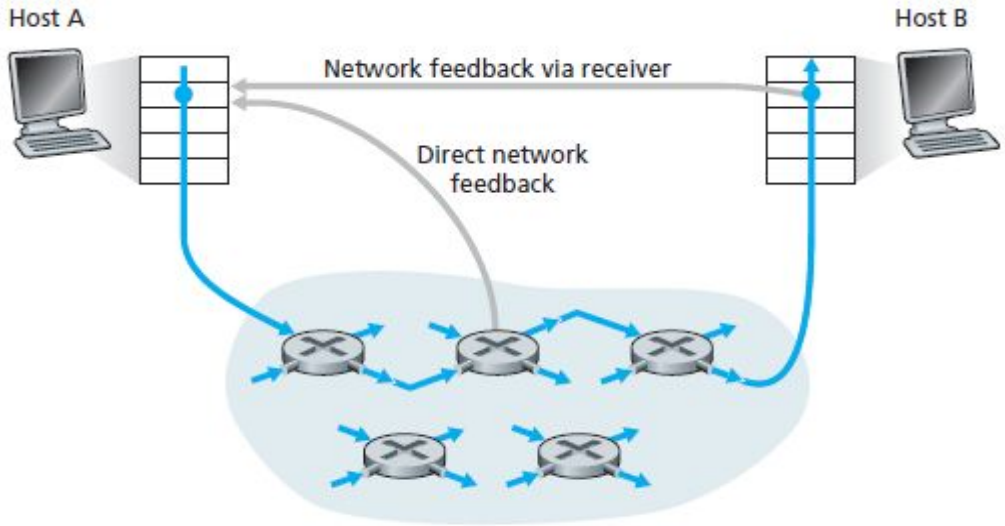
12. Explain network assisted congestion control mechanisms

Network-assisted congestion control:

- With network-assisted congestion control, network-layer components (that is, routers) provide explicit feedback to the sender regarding the congestion state in the network.
- For network-assisted congestion control, congestion information is typically fed back from the network to the sender in one of two ways,
- Direct feedback may be sent from a network router to the sender.
- This form of notification typically takes the form of a choke packet (essentially saying, “I’m congested!”).

- The second form of notification occurs when a router marks/updates a field in a packet flowing from sender to receiver to indicate congestion.
- Upon receipt of a marked packet, the receiver then notifies the sender of the congestion indication. Note that this latter form of notification takes at least a full round-trip time.

Example:congestion-control algorithm in ATM ABR based transmission.

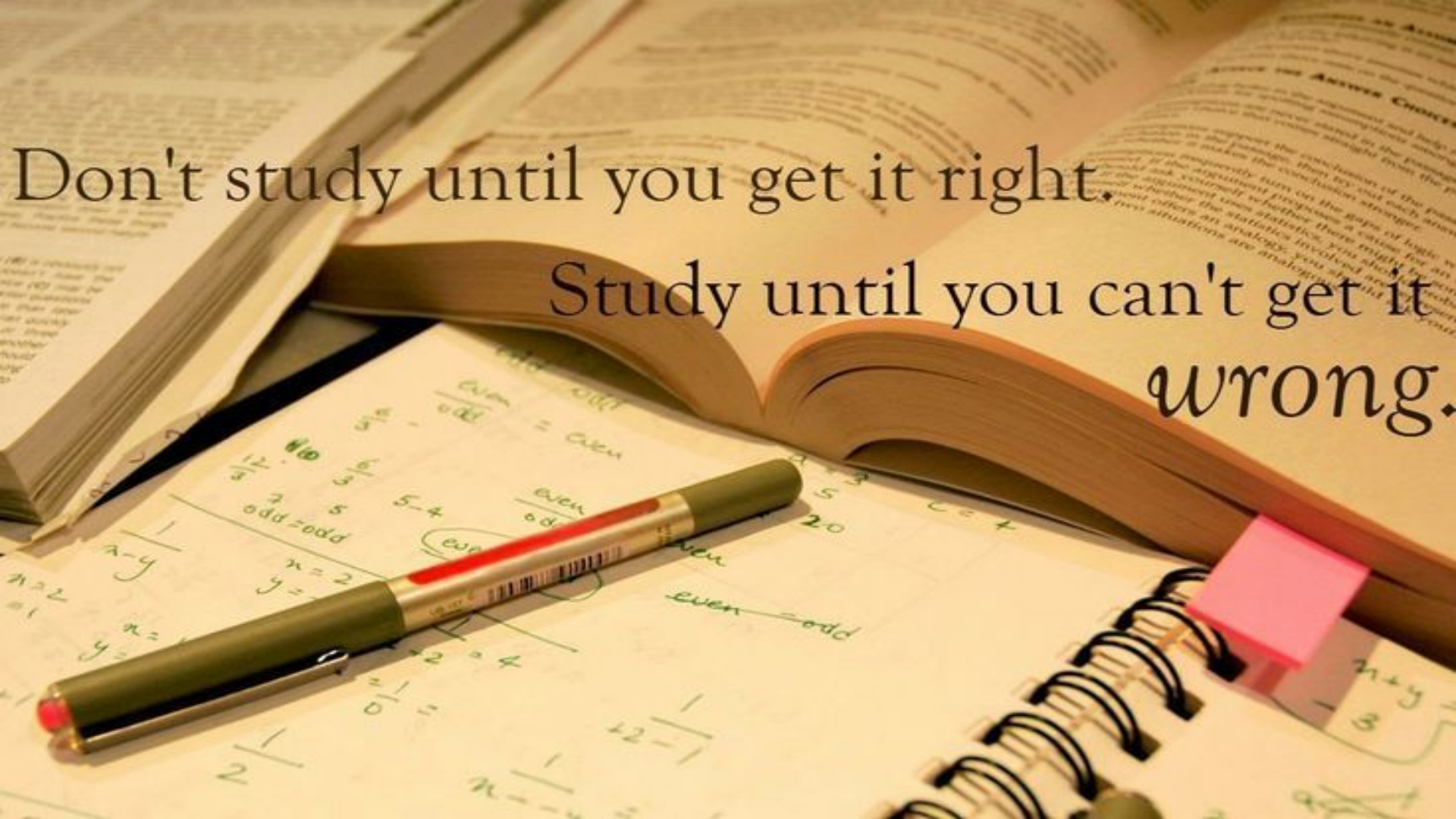


- Data cells are transmitted from a source to a destination through a series of intermediate routers.
- Resource management(RM) cells are embedded within the data cells.
- These RM cells can be used to convey congestion-related information
- among the hosts and routers.
- When an RM cell arrives at a destination, it will be turned around and sent back to the sender(Network feedback via receiver) (possibly after the destination has modified the contents of the RM cell).
- It is also possible for a switch to generate an RM cell itself and send this RM cell directly to a source(Direct feedback)
- RM cells can thus be
- used to provide both direct network feedback and network feedback via the
- Receiver.
- ABR provides three mechanisms for signaling congestion-related information
- from the switches to the receiver:
-

EFCI bit: Each data cell contains an explicit forward congestion indication (EFCI) bit. A congested network switch can set the EFCI bit in a data cell to 1 to signal congestion to the destination host. The destination must check the EFCI bit in all received data cells. When an RM cell arrives at the destination, if the most recently received data cell had the EFCI bit set to 1, then the destination sets the congestion indication bit (the CI bit) of the RM cell to 1 and sends the RM cell back to the sender.

CI and NI bits: RM cells have a congestion indication (CI) bit and a no increase (NI) bit that can be set by a congested network switch. Specifically, a switch can set the NI bit in a passing RM cell to 1 under mild congestion and can set the CI bit to 1 under severe congestion conditions. When a destination host receives an RM cell, it will send the RM cell back to the sender

- **ER setting.** Each RM cell also contains a 2-byte explicit rate (ER) field. A congested switch may lower the value contained in the ER field in a passing RM cell. In this manner, the ER field will be set to the minimum supportable rate of all switches on the source-to-destination path.


Don't study until you get it right.
Study until you can't get it
wrong.