Transport Layer (I)

Overview

- Transport layer is an intermediate layer in computer networks, which bridges the application layer and network layer.
  - *Application Layer*: needs reliable, bidirectional stream-oriented service (continuous flow of data)
  - *Network Layer*: provides unreliable datagram service
- Transport layer follows the End-to-End Principle
  - it tries to keep the network as simple as possible
  - so, all “smarts” are implemented at the end nodes

Functions of Transport Layer

- Addressing
  - we know that transport layer uses port numbers, why?
    - transport layer: identify logical communication between processes
    - network layer: maintain logical communication between hosts
- Error control
  - packet loss detection and retransmissions
- Flow and congestion control (traffic management)
  - controlling the transmission rate
- Session management
  - establish, manage, and terminate connections
- Not all transport layer protocols implement all of these functions.

Error-control Method

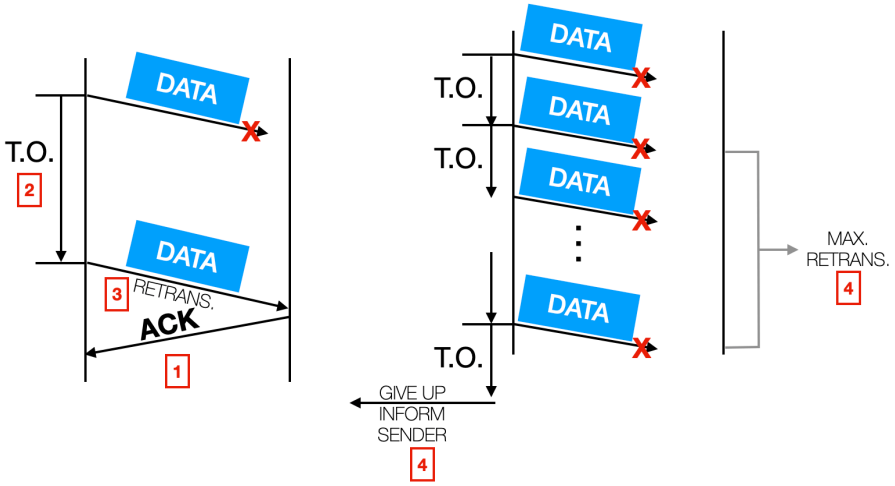
- We will first talk about the error control method in the transport layer, which provide the reliability for this layer.
- The reliability is achieved if packets are delivered despite unreliability of the network layer or it detects that delivery is impossible.

- The error-control method used in the transport layer is Automatic Repeat reQuest (ARQ). This method has six mechanisms.
  1. acknowledgement
     - receiver explicitly tells sender that a packet is received OK
     - negative acknowledgments (NAK): receiver explicitly tells sender that pkt had errors (this is unnecessary, since duplicate ACK at sender addresses this)
     - so, sender will know whether the packet is delivered or whether the packet needs to be retransmitted.

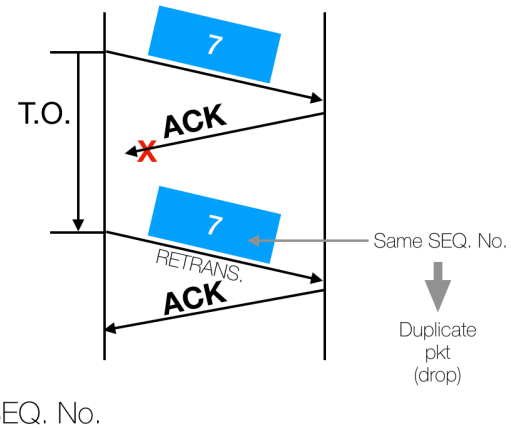
2. timeout
   - how long should the sender wait for ACK? what if ACK/data is lost? how does sender know whether the packet is received or not?
   - timeout is used to solve the problem. Sender triggers a timer immediately after transmit a packet. If sender does not receive ACK within a certain time (we call it timeout), retransmit the packet.

3. retransmission
   - sender retransmit the current packet if timeout

4. give up after k retransmissions
   - what if the link is very bad? very lossy? [sender gives up after k retransmissions]
   - prevent infinite retransmissions

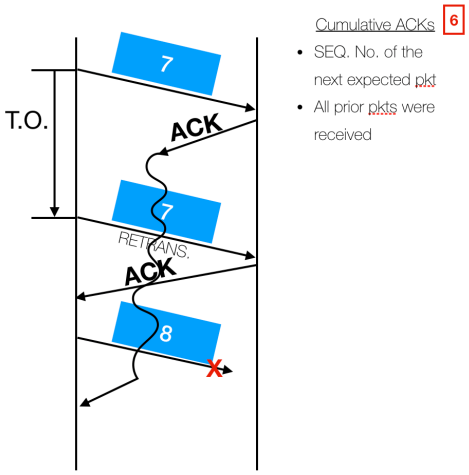


- The above components work well if DATA is lost, shown in the left figure below. What if DATA is received but ACK is lost? The sender will retransmit due to timeout. Then there will be two copies at the receiver side, which is confusing and may lead to bad user experience. The duplicate packet transmission also wastes computing and network resource. How do we solve it?

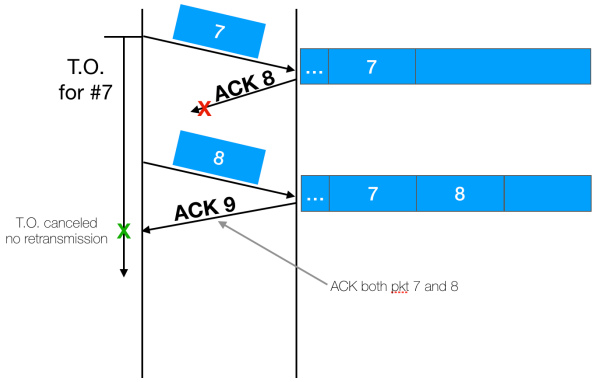


5. sequence numbers on data packets
   - handle duplicates happened in retransmission
   - sender adds a unique sequence number to each packet
   - receiver can detect duplicate packets based on their sequence number and discard duplicate packets (shown as the right figure above)

- So far so good. However, it's possible that ACK will be received by the sender out of order. If so, it's hard to tell which packets have been received. As shown in the figure below, it's hard to tell wether the last received ACK is for packet 7 or packet 8, and packet 8 is lost.



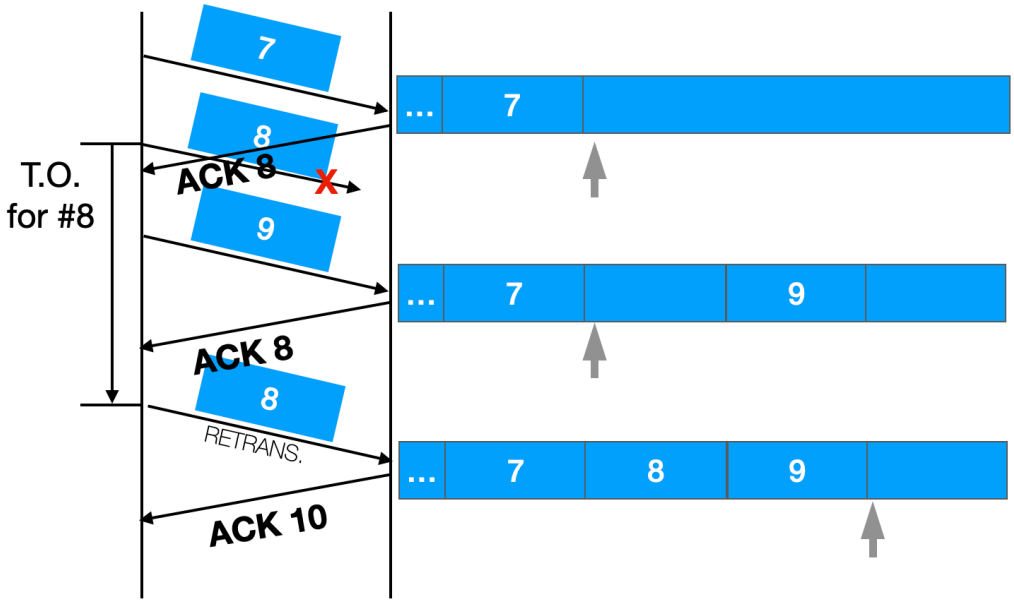
6. cumulative acknowledgment numbers
   - add sequence numbers on ACKs
   - the sequence numbers on ACKs are not the received packet numbers but the next expected packet. This improves the efficiency, as an ACK indicates all prior packets were received even if ACK for a prior received packet is lost, show as the figure below.



- We know that ARQ is the error-control method used in computer network. It has six components working together to provide reliability. These components are: acknowledgment, timeout, retransmission, giving up after k retransmissions, sequence numbers on data packets, and cumulative acknowledgement number.

- All the scenarios discussed above use the one of the two ways to fill the communication pipe, Stop and Wait and Sliding Window:
  - Stop and Wait lets senders wait for acknowledgement before seeding next packet.
  - Sliding Window, widely used in nowadays, lets senders send up to W (window size) packets before waiting for acknowledgement

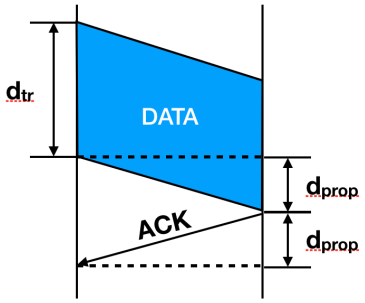
- Now, let us consider a scenario, where the sender is allowed to send 3 packets before waiting for acknowledgment. The sender sends packets 7, 8, and 9. Packets 7 and 9 are received. But, packet 8 is lost. How would the cumulative acknowledgement number work?



- So, why sliding window protocols are wildly used? [Better Utilization]

- Utilization is also called *normalized throughput*.
  - It's the ratio between goodput and throughput.
    - goodput: "useful" bytes delivered per second
    - throughput: total number of bytes that can be delivered per second

- Stop and Wait Utilization



$$Utilization = \frac{d_{tr}}{d_{tr} + 2d_{prop}} = \frac{1}{1 + 2\frac{d_{prop}}{d_{tr}}}$$

- Please note that the transmission rate is a constant, so the utilization can be calculated using transmission delay and propagation delay.

Disclaimer: Notes adapted from the textbook and online resources.