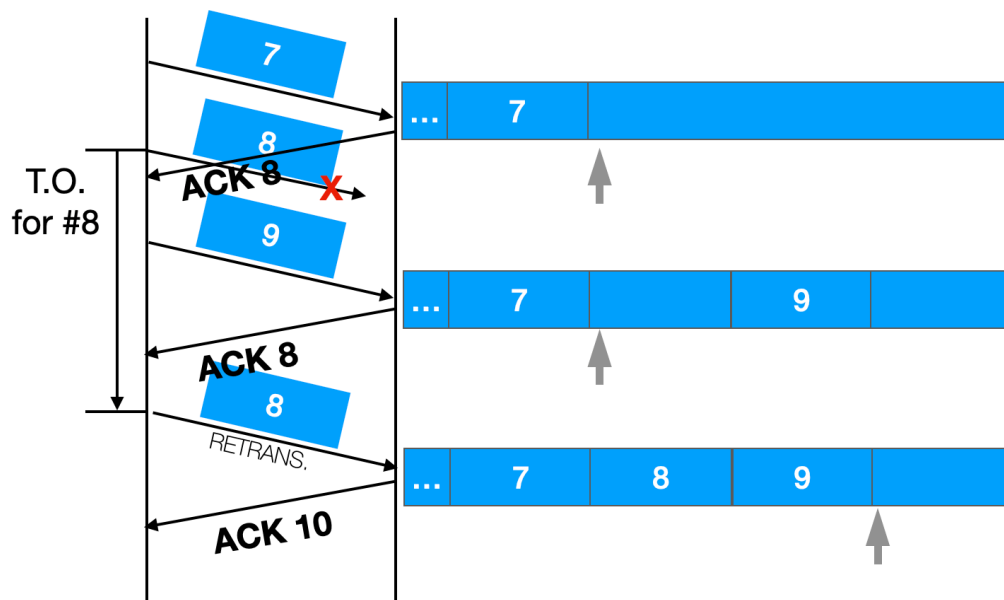


Transport Layer (II)

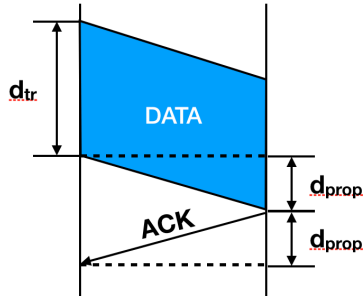
Error-control Method (cont.)

- 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



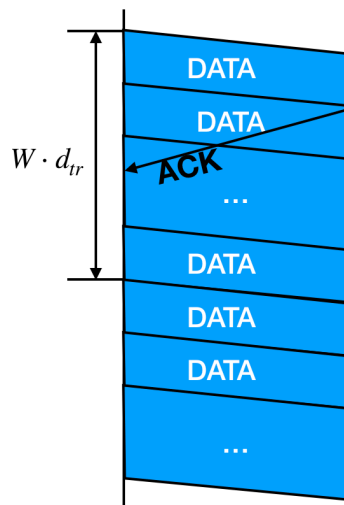
$$Utilization = \frac{d_{tr}}{d_{tr} + 2d_{prop}} = \frac{1}{1 + 2\frac{d_{prop}}{d_{tr}}}$$

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

CASE 1

$$2d_{prop} < (W - 1) \cdot d_{tr}$$

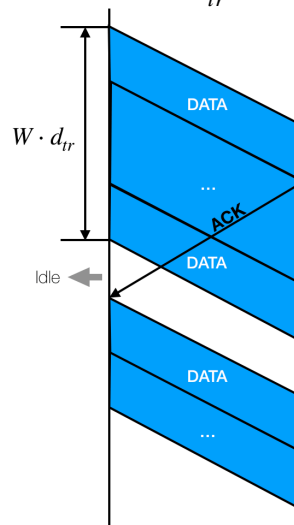
$$Utilization = 100\%$$



CASE 2

$$2d_{prop} \geq (W - 1)d_{tr}$$

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

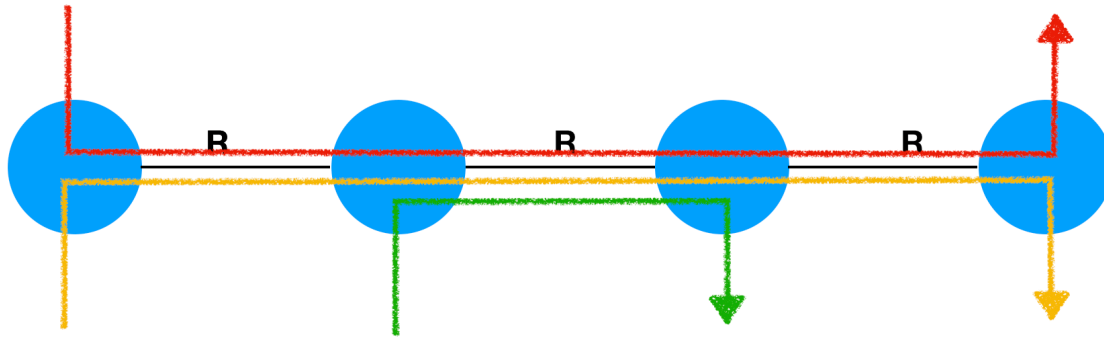


Flow control

- Flow control, congestion control, traffic management etc. (same fundamental issue)
- **Goal:** Make the most effective use of the network capacity
 - avoid congestion

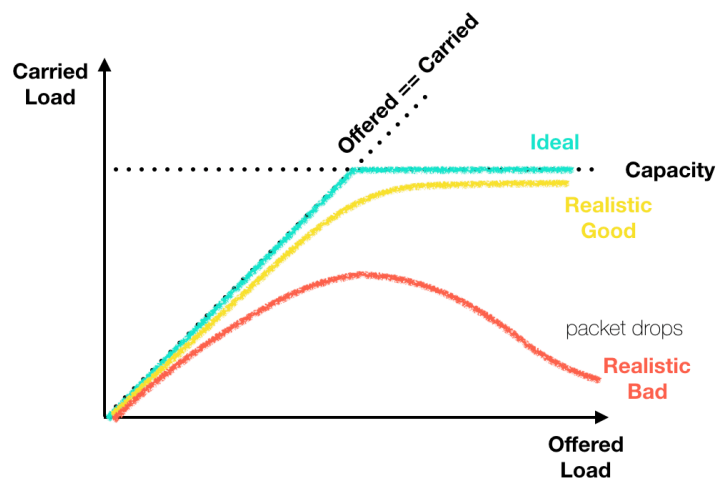
- maximize utilization
- maintain fairness (or deliver promised service level)

Fairness VS Utilization



Traffic	Max Utilization	Fair
—	$R/2$	$R/3$
—	$R/2$	$R/3$
—	θ	$R/3$

- If the red and yellow traffic join the network first, they will maximize each of their utilization while keeping the fairness of the usage. Therefore, each of them will get a half of the bandwidth. When the green traffic join the network, they have to adjust the usage to keep the fairness while maximizing utilization among them.



- Carried load is the actual traffic volume in the network over a given period of time, while the offered load is the total traffic volume the network can provide.

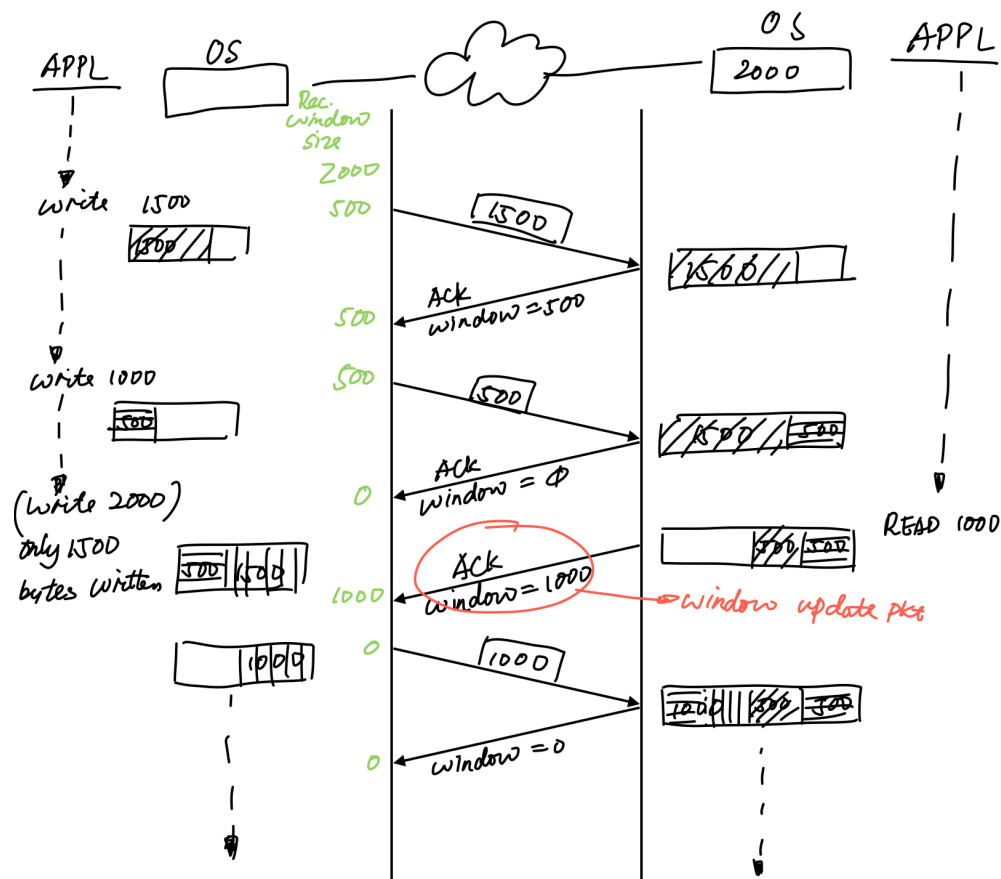
- The ideal case is that the carried load is the same as the offered load, so that the utilization can be best maximized. In reality, the good case is that the carried load can be really close to the offered load. However, when congestion happens, the carried load can drop quickly.

Two Types of Congestions

- There are two types of congestions in computer networks: **Receiver Congestion** and **Network Congestion**.

- **Receiver Congestion:**

- receiver congestion happens when the receiver is unable to keep up with incoming data
- this congestion can be solved by explicit feedback from receiver to sender



- In the above example, after building up the connection, the sender will know that the receiver window size is 2000 bytes. So, the sender won't transmit more than the available receiver window size per time. Suppose the sender's window size is also 2000 bytes in this example. The sender wants to send three messages to the receiver in this example. The three message sizes are 1500 bytes, 1000 bytes, and 2000 bytes. The first message can be sent all at once, as the receiver has 2000-byte window available. After receiving the first message, the receiver only has 500-byte available window, and it will inform the sender the size of the

available window via acknowledgment. So, when the sender sends the second message, it can only send 500 bytes out of the 1000-byte message to the receiver first, and then wait for the window becomes available again. Once the receiver READs 1000 bytes from the window, there are 1000 bytes in the receiver window become available again. Meanwhile, the sender wants to send the third message that is 2000 bytes. The sender window is then filled up by the remaining 500 bytes from the second message and 1500 bytes from the third message. The remaining 500 bytes of the third message will be written into the sender window when there is available space. Since only 1000 bytes become available in the receiver window, the sender will transmit 1000 bytes of its window to the receiver. The remaining 1000 bytes in the sender's window coupled with the remaining 500 bytes of the third message will be transmitted when the receiver window becomes available.

Disclaimer: Notes adapted from the textbook and online resources.