# Computer Network

## Lecture-40

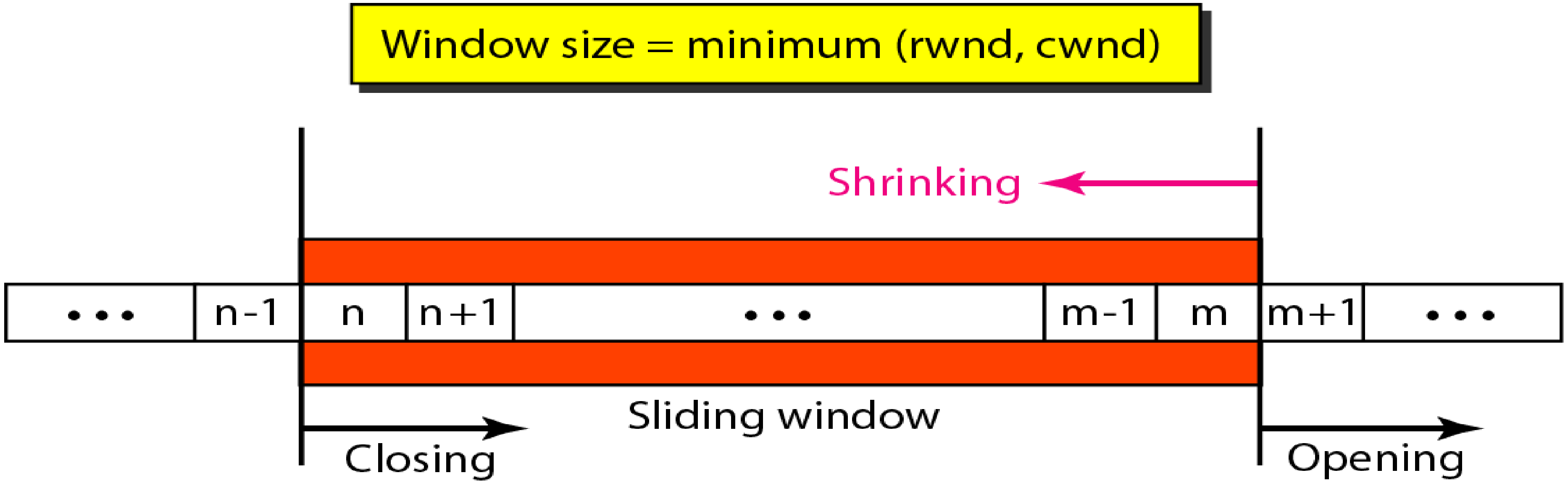
Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

# Process-to-Process Delivery

**Flow Control or TCP Sliding Window**

- ❖ TCP uses a sliding window, to handle flow control. The sliding window protocol used by TCP, however, is something between the Go-Back-N and Selective Repeat sliding window.
- ❖ The sliding window protocol in TCP looks like the Go-Back-N protocol because it does not use NAKs; it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive.
- ❖ There are two big differences between this sliding window and the one we used at the data link layer.
  - ❖ The sliding window of TCP is byte-oriented; the one we discussed in the data link layer is frame-oriented.
  - ❖ The TCP's sliding window is of variable size; the one we discussed in the data link layer was of fixed size.

# Process-to-Process Delivery



The window is opened, closed, or shrunk. These three activities, are in the control of the receiver (and depend on congestion in the network), not the sender.

## Process-to-Process Delivery

The sender must obey the commands of the receiver in this matter.

**Opening** a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending.

**Closing** the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender need not worry about them anymore.

**Shrinking** the window means moving the right wall to the left.

# Process-to-Process Delivery

The size of the window at one end is determined by the lesser of two values: receiver window (rwnd) or congestion window (cwnd).

The **receiver window** is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded.

The **congestion window** is a value determined by the network to avoid congestion

# Process-to-Process Delivery

**Silly Window Syndrome**

Silly Window Syndrome is a problem that arises due to the poor implementation of TCP flow control.

It degrades the TCP performance and makes the data transmission extremely inefficient.

It causes the sender window size to shrink to a silly value.

The window size shrinks to such an extent where the data being transmitted is smaller than TCP Header.

## Process-to-Process Delivery

**Causes-**

The problem arises due to following causes-

1. Sender transmitting data in small segments repeatedly
2. Receiver accepting only few bytes at a time repeatedly

**Cause-01: Sender Transmitting Data In Small Segments Repeatedly-**

- Consider application generates one byte of data to send at a time.
- The poor implementation of TCP causes the sender to send each byte of data in an individual TCP segment.

This problem is solved using Nagle’s Algorithm.

**Nagle’s Algorithm-**

## Process-to-Process Delivery

**Nagle’s Algorithm-**

Nagle’s algorithm suggests-

- Sender should send only the first byte on receiving one byte data from the application.
- Sender should buffer all the rest bytes until the outstanding byte gets acknowledged.
- In other words, sender should wait for 1 RTT.
- After receiving the acknowledgement, sender should send the buffered data in one TCP segment.
- Then, sender should buffer the data again until the previously sent data gets acknowledged.

## Process-to-Process Delivery

**Cause-02: Receiver Accepting Only Few Bytes Repeatedly-**

- Consider the receiver continues to be unable to process all the incoming data.
- In such a case, its window size becomes smaller and smaller.
- A stage arrives when it repeatedly sends the window size of 1 byte to the sender.

This problem is solved using Clark's Solution.

**Clark's Solution-**

Clark's solution suggests-

- Receiver should not send a window update for 1 byte.
- Receiver should wait until it has a decent amount of space available.
- Receiver should then advertise that window size to the sender.

# Process-to-Process Delivery

**Error Control**

TCP is a reliable transport layer protocol. This means that an application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order, without error, and without any part lost or duplicated.

TCP provides reliability using error control. Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments. Error control also includes a mechanism for correcting errors after they are detected. Error detection and correction in TCP is achieved through the use of three simple tools: **checksum, acknowledgment, and time-out.**

## Process-to-Process Delivery

**Checksum**

Each segment includes a checksum field which is used to check for a corrupted segment. If the segment is corrupted, it is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment.

**Acknowledgment**

TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data but consume a sequence number are also acknowledged. ACK segments are never acknowledged.

ACK segments do not consume sequence numbers and are not acknowledged.

# Process-to-Process Delivery

**Retransmission**

The heart of the error control mechanism is the retransmission of segments. When a segment is corrupted, lost, or delayed, it is retransmitted.

In modern implementations, a segment is retransmitted on two occasions: when a retransmission timer expires or when the sender receives three duplicate ACKs.

**Out-of-Order Segments**

- When a segment is delayed, lost, or discarded, the segments following that segment arrive out of order.
- Originally, TCP was designed to discard all out-of-order segments, resulting in the retransmission of the missing segment and the following segments.
- Most implementations today do not discard the out-of-order segments. They store them temporarily and flag them as out-of-order segments until the missing segment arrives. TCP guarantees that data are delivered to the process in order.

# Process-to-Process Delivery

**Exercise**

1. The following is a dump of a UDP header in hexadecimal format.

```
0632000D00 1CE217
```

a. What is the source port number?

b. What is the destination port number?

c. What is the total length of the user datagram?

d. What is the length of the data?

e. Is the packet directed from a client to a server or vice versa?

f. What is the client process?

# Process-to-Process Delivery

2. The following is a dump of a TCP header in hexadecimal format.

```
05320017 00000001 00000000 500207FF 00000000
```

a. What is the source port number?

b. What is the destination port number?

c. What the sequence number?

d. What is the acknowledgment number?

e. What is the length of the header?

f. What is the type of the segment?

g. What is the window size?

# Process-to-Process Delivery

3. If WAN link is 2 Mbps and RTT between source and destination is 300 msec, what would be the optimal TCP window size needed to fully utilize the line?
   a) 60,000 bits
   b) 75,000 bytes
   c) 75,000 bits
   d) 60,000 bytes
4. Suppose host A is sending a large file to host B over a TCP connection. The two end hosts are 10 msec apart (20 msec RTT) connected by a 1 Gbps link. Assume that they are using a packet size of 1000 bytes to transmit the file. For simplicity, ignore ack packets. At least how big would the window size (in packets) have to be for the channel utilization to be greater than 80%?
   a) 1000
   b) 1500
   c) 2000
   d) 2500

# Process-to-Process Delivery

5. A TCP machine is sending windows of 65535 B over a 1 Gbps channel that has a 10 msec one way delay.
   a) What is the maximum throughput achievable?
   b) What is the line efficiency?

6. Consider the three-way handshake mechanism followed during TCP connection establishment hosts P and Q. Let X and Y be two random 32-bit starting sequence numbers chosen by P and Q respectively. Suppose P sends a TCP connection request message to Q with a TCP segment having SYN bit = 1, SEQ number = X, and ACK bit = 0. Suppose Q accepts the connection request. Which one of the following choices represents the information present in the TCP segment header that is sent by Q to P?

(A) SYN bit = 0, SEQ number = X + 1, ACK bit = 0, ACK number = Y, FIN bit = 1
(B) SYN bit = 1, SEQ number = Y, ACK bit = 1, ACK number = X + 1, FIN bit = 0
(C) SYN bit = 1, SEQ number = Y, ACK bit = 1, ACK number = X, FIN bit = 0
(D) SYN bit = 1, SEQ number = X + 1, ACK bit = 0, ACK number = Y, FIN bit = 0

# Process-to-Process Delivery

5. Consider a TCP connection in a state where there are no outstanding ACKs. The sender sends two segments back to back. The sequence numbers of the first and second segments are 230 and 290 respectively. The first segment was lost but the second segment was received correctly by the receiver. Let X be the amount of data carried in the first segment (in bytes) and Y be the ACK number sent by the receiver. The values of X and Y are-

   a) 60 and 290
   b) 230 and 291
   c) 60 and 231
   d) 60 and 230