Reliability and Flow Control

TASK

TCP Reliability – Guaranteed and Ordered Delivery

The reason that TCP is the better protocol for some applications is because, under UDP, it rewards dropped packets and numbers packets to indicate their proper order before delivery. TCP can also help maintain the flow of packets so that devices do not become overloaded. This topic covers these features of TCP in detail.

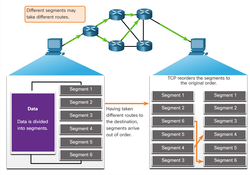
There may be times when TCP segments do not arrive at their destination. Other times, the TCP segments might arrive out of order. For the original message to be understood by the recipient, all the data must be received and the data in these segments must be reassembled into the original order. Sequence numbers are assigned in the header of each packet to achieve this goal. The sequence number represents the first data byte of the TCP segment.

During session setup, an initial sequence number (ISN) is set. This ISN represents the starting value of the bytes that are transmitted to the receiving application. As data is transmitted during the session, the sequence number is incremented by the number of bytes that have been transmitted. This data byte tracking enables each segment to be uniquely identified and acknowledged. Missing segments can then be identified.

The ISN does not begin at one but is effectively a random number. This is to prevent certain types of malicious attacks. For simplicity, we will use an ISN of 1 for the examples in this chapter.

Segment sequence numbers indicate how to reassemble and reorder received segments, as shown in the figure.

TCP Segments Are Reordered at the Destination



The receiving TCP process places the data from a segment into a receiving buffer. Segments are then placed in the proper sequence order and passed to the application layer when reassembled. Any segments that arrive with sequence numbers that are out of order are held for later processing. Then, when the segments with the missing bytes arrive, these segments are processed in order.

TASK

Video – TCP Reliability – Sequence Numbers and Acknowledgments

One of the functions of TCP is to ensure that each segment reaches its destination. The TCP services on the destination host acknowledge the data that have been received by the source application.

Click Play in the figure to view a lesson on TCP sequence numbers and acknowledgments.

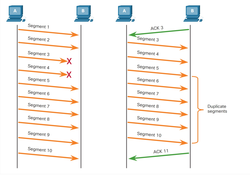
TASK

TCP Reliability – Data Loss and Retransmission

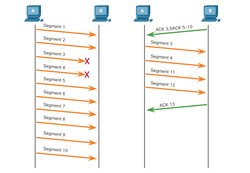
No matter how well designed a network is, data loss occasionally occurs. TCP provides methods of managing these segment losses. Among these is a mechanism to retransmit segments for unacknowledged data.

The sequence (SEQ) number and acknowledgment (ACK) number are used together to confirm receipt of the bytes of data contained in the transmitted segments. The SEQ number identifies the first byte of data in the segment being transmitted. TCP uses the ACK number sent back to the source to indicate the next byte that the receiver expects to receive. This is called acknowledgment acknowledgment.

Prior to later enhancements, TCP could only acknowledge the next byte expected. For example, in the figure, using segment numbers for simplicity, host A sends segments 1 through 10 to host B. If all the segments arrive except for segments 3 and 4, host B would reply with acknowledgment specifying that the next segment expected is segment 3 (host B has no idea if any other segments arrived or not. Host A would, therefore, resend segments 3 through 10. If all the seven segments arrived successfully, segments 5 through 10 would be duplicates. This can lead to delays, congestion, and inefficiencies.



Most operating systems today typically employ an optional TCP feature called selective acknowledgment (SACK), negotiated during the three way handshake. If both hosts support SACK, the receiver can explicitly acknowledge which segments (bytes) were received including any discontinuous segments. The sending host would then only need to retransmit the missing data. In addition, TCP may infer that more segments have been lost if acknowledgments arrive for the last segment. In the figure, host B of the segments arrive except for segments 3 and 4, host B can acknowledge that it has received segments 1 and 2 (ACK 3), and selectively acknowledge segments 5 through 10 (SACK 5-10). Host A would only need to resend segments 3 and 4.



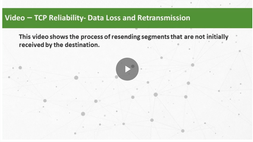
Note: TCP typically sends ACKs for every other packet, but other factors beyond the scope of this topic may alter this behavior.

TCP uses timers to know how long to wait before resending a segment. In the figure, play the video and click the link to download the PDF file. The video and PDF file examine TCP data loss and retransmission.

TASK

Video – TCP Reliability – Data Loss and Retransmission

Click Play in the figure to view a lesson on TCP retransmission.



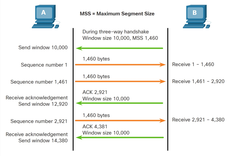
TASK

TCP Flow Control – Window Size and Acknowledgments

TCP also provides mechanisms for flow control. Flow control is the amount of data that the destination can receive and process reliably. Flow control helps maintain the reliability of TCP transmission by adjusting the rate of data flow between source and destination for a given session. To accomplish this, the TCP header includes a 16-bit field called the window size.

The figure shows an example of window size and acknowledgments.

TCP Window Size Example



The window size determines the number of bytes that can be sent before expecting an acknowledgment. The acknowledgment number is the number of the next expected byte.

The window size is the number of bytes that the destination device of a TCP session can accept and process at one time. In this example, the PC B initial window size is 10,000 bytes starting with byte number 1,460. PC A can send up to 10,000 bytes before expecting an acknowledgment from the destination. The window size is known as the send window of PC A. The window size is included in every TCP segment so the destination can modify the window size at any time depending on buffer availability.

The initial window size is agreed upon when the TCP session is established during the three way handshake. The source device must limit the number of bytes sent to the destination device based on the window size of the destination. Only after the source device receives an acknowledgment that the bytes have been received, can it continue sending more bytes. The source device also limits its send window based on the speed and acknowledgments received from the destination. If the bytes are acknowledged, the bytes are received and processed. If the destination will send acknowledgments to inform the source that it can continue to send additional bytes.

For example, it is typical that PC B would not wait until all 10,000 bytes have been received before sending an acknowledgment. This means PC A can adjust its send window as it receives acknowledgments from PC B. As shown in the figure, when PC A receives an acknowledgment with the acknowledgment number 2,921, which is the next byte that PC B is expecting, PC A can send another 1,460 bytes. This leaves PC A with 12,920 bytes to 12,920. PC A can now continue to send up to another 10,000 bytes to PC B as long as it does not send more than its new send window of 12,920.

A destination sending acknowledgments as it processes bytes received, and the continual adjustment of the source send window, is known as sliding windows. In the previous example, the send window of PC A increases after another TCP segment arrives from 10,000 to 12,920.

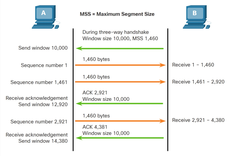
If the availability of the destination's buffer space decreases, it may reduce its window size to inform the source to reduce the number of bytes it should send without receiving an acknowledgment.

Note: Devices today use the sliding windows protocol. The receiver typically sends an acknowledgment after every two segments it receives. The number of segments received before being acknowledged may vary. The advantage of sliding windows is that it allows the sender to continuously transmit segments, as long as the receiver is acknowledging previous segments. The details of sliding windows are beyond the scope of this course.

TASK

TCP Flow Control – Maximum Segment Size (MSS)

In the figure, the source is transmitting 1,460 bytes of data within each TCP segment. This is typically the Maximum Segment Size (MSS) that the destination device can receive. The MSS is part of the options field in the TCP header that specifies the largest amount of data, in bytes, that a device can receive in a single TCP segment. The MSS size does not include the TCP header. The MSS is typically included during the three way handshake.



A common MSS is 1,460 bytes when using IPv4. A host determines the value of its MSS field by subtracting the IP and TCP headers from the Ethernet maximum transmission unit (MTU). On an Ethernet interface, the default MTU is 1500 bytes. Subtracting the IPv4 header of 20 bytes and the TCP header of 20 bytes, the default MSS size will be 1460 bytes, as shown in the figure.

TASK

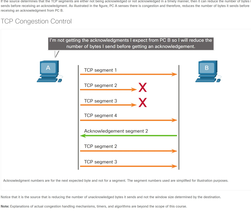
TCP Flow Control – Congestion Avoidance

When congestion occurs on a network, it results in packets being discarded by the overloaded router. When packets containing TCP segments do not reach their destination, they are lost unacknowledged. By determining the rate at which TCP segments are sent but not acknowledged, the source can assume a certain level of network congestion.

Whenever there is congestion, retransmission of lost TCP segments from the source will occur. If the retransmission is not properly controlled, the additional retransmission of the TCP segments can make the congestion even worse. Not only are new packets with TCP segments introduced into the network, but the feedback from the destination hosts regarding that same lost set of data will be the congestion. To avoid and control congestion, TCP employs several congestion handling mechanisms, timers, and algorithms.

If the source determines that the TCP segments are either not being acknowledged or not acknowledged in a timely manner, then it can reduce the number of bytes it sends before receiving an acknowledgment. As illustrated in the figure, PC A senses there is congestion and therefore, reduces the number of bytes it sends before receiving an acknowledgment from PC B.

TCP Congestion Control



Acknowledgment numbers are for the next expected byte and not for a segment. The segment numbers used are simplified for illustration purposes.

Notice that it is the source that is reducing the number of unacknowledged bytes it sends and not the window size determined by the destination.

Note: Explanations of actual congestion handling mechanisms, timers, and algorithms are beyond the scope of this course.

TASK

Check Your Understanding – Reliability and Flow Control

Check your understanding of the TCP reliability and flow control process by choosing the correct answer to the following questions.

1. What field is used by the destination host to reassemble segments into the original order?
   - Control Bits
   - Destination Port
   - Sequence Number
   - Source Port
   - Window Size

2. What field is used to provide flow control?
   - Control Bits
   - Destination Port
   - Sequence Number
   - Source Port
   - Window Size

3. What happens when a sending host senses there is congestion?
   - The receiving host increases the number of bytes it sends before receiving an acknowledgment from the sending host.
   - The receiving host reduces the number of bytes it sends before receiving an acknowledgment from the sending host.
   - The sending host increases the number of bytes it sends before receiving an acknowledgment from the destination host.
   - The sending host reduces the number of bytes it sends before receiving an acknowledgment from the destination host.

Check

Show Me

Reset