Lab 7 Transmission Control Protocol

1. TCP introduction

TCP (Transmission control protocol) is a connection-orient, reliable transport layer protocol. Different with UDP, TCP allows to send and to receive data in byte stream. TCP offers sending buffer and receiving buffer in order to balance the different speed of sending and receiving data in client and server. TCP offers full duplex service. Each process of communication has two buffers: sending buffer and receiving buffer. TCP adds an acknowledge number in message to ask the sender to send the next message. If not receiving an ACK of the message in specified time, the sender will send it again. That ensures TCP to be a reliable transport layer protocol.

Well-known ports used by TCP:

Port	Protocol	Port	Protocol
20	FTP data	80	HTTP
21	FTP Control	110	POP3
23	TELNET	143	IMAP
25	SMTP		

Table 7-1

2. TCP segment format

TCP segment format is as follow:

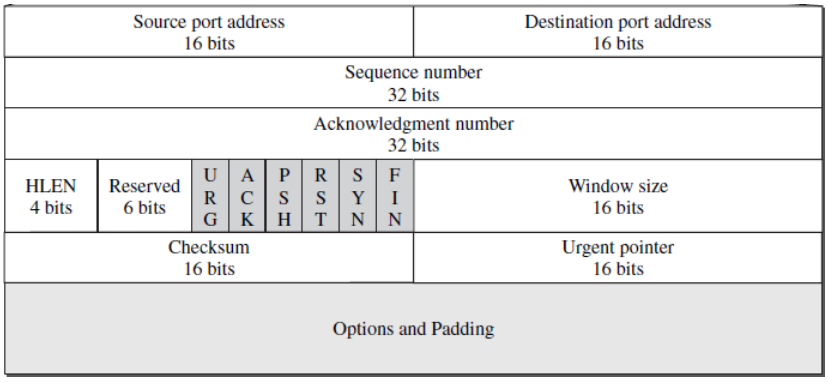

Figure 7-1 TCP segment format

The segment consisits of 20-60 bytes header, followed by data from the application program. The header is 20 bytes if there is no options and up to 60 bytes if it contains options.

(1) Source port address: it defines the port number of the application program in the host that is sending the segment.

(2) Destination port address: it defines the port number of the application program in the host that is receiving the segment.

(3) Sequence number: it defines the number assigned to the first byte of data contained in this segment. TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which

byte in this sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial sequence number, which is usually different in each direction.

(4) Acknowledgment number: it defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number x (suppose this is the sequence number of the last data byte) from the other party, it defines x+1 as the acknowledgment number. Acknowledgment and data can be piggybacked together.

(5) Header length: it indicates the number of 4 bytes words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 and 15.

(6) Reserved: this is a 6 bits field reserved for future use. (**Two of them can be used for CWR and ECE**)

(7) Control bits: this field defines 8 different control bits or flags as shown in Figure 7-2. One or more of these bits can be set at a time.

CWR: reduce congestion window
ECE: experience congestion loopback
URG: the value of the urgent pointer field is valid
ACK: the value of the acknowledgment field is valid
PSH: push the data
RST: reset the connection
SYN: synchronize sequence numbers during connection
FIN: terminate the connection

Figure 7-2 control segment

These bits enable flow control, connection establishment and ternination, connection abortion and the mode of data transfer in TCP. See table 7-2

Flag	Description
CWR	Decrease congestion window (it indicates that the host receives a TCP packet which is set ECE flag. This window that maintained in TCP is used to manage size of sending window.)
ECN	Experience congestion loopback (used to indicate whether a TCP port has ECN function. It also indicates ECN in TCP packet IP header is set at 11)
URG	The value of the urgent pointer field is valid
ACK	The value of the acknowledgment field is valid.
PSH	Push the data
RST	Reset the connection
SYN	Synchronize sequence numbers during connection.
FIN	Terminatie the connection.

(8) Window size: this field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65535 bytes. This value normally referred to as the receiving window and is determined by the receiver. The sender must obey the dictation of the recever in this case.

(9) Checksum: the calculation of the checksum contains pseudo header, TCP header and TCP data.

(10) Urgent pointer: this 16 bits field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.

(11) Option: there can be up to 40 bytes of optional information in the TCP header.

3. TCP encapsulation

A TCP message is encapsulated in an IP message and the IP message is encapsulated in a frame of data link layer. See Figure 7-3:

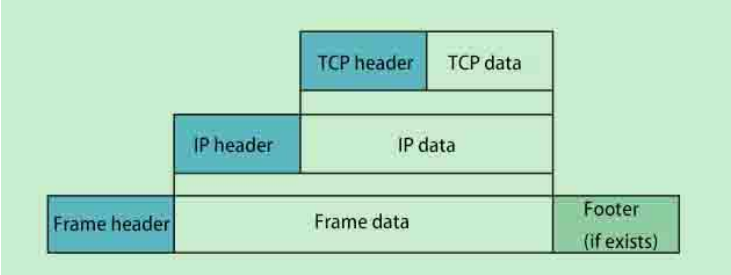

Figure 7-3 TCP encapsulation

4. TCP checksum

The computation process of TCP checksum is the same as UDP. Whether use checksum or not in UDP is optional but use checksum in TCP is compulsory. Add pseudo header in message when computing TCP checksum. The upper layer protocol type field value is 6 for TCP pseudo header. See Figure 7-4:

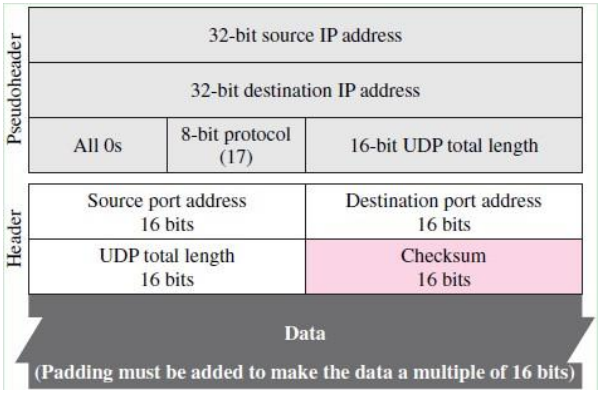

Figure 7-4 Add pseudo header to TCP message

5. TCP connection establishment and release

Please refer to more details in the lecture slides and textbook!

(1) Connection establishment

TCP transmits data in full duplex way. Two processes can send data to each other when they have established TCP connection. Before transmitting data, they have to initiate the communication.

(2) Three-way handshaking

The server must first get ready to receive TCP connection. This is called a passsive open request. Then the server is getting ready to accept TCP connection from any one host.

A client may intiate an active open. Then the server and the client begin a three-way handshaking process. See Figure 7-5:

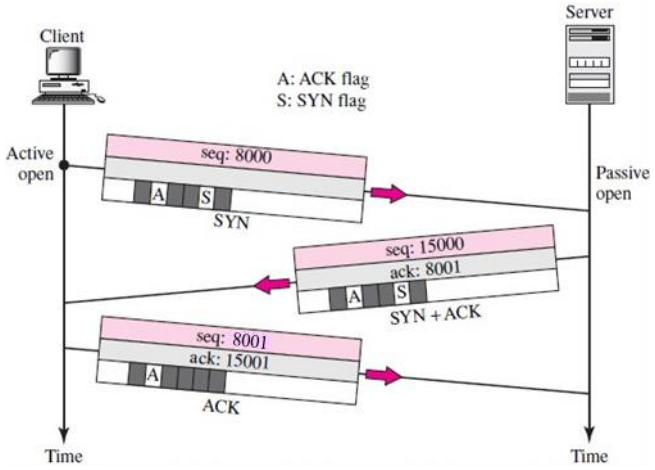

Figure 7-5 Connection establishment using three-way handshaking

1) The client sends the first message which is a SYN message. Only the SYN flag set at 1 in this message. The function of it is to sychronize sequence.
2) Ther server sends the second message which is a SYN+ACK message. SYN and ACK flag set at 1 in the message. The first function of it is communicate with the client. And the other function is define the size of receive window of the client.
3) The client sends the third message. It is only a ACK message which is used to acknowledge the second received message.

(3) Connection release

Both sides of communication can terminate the connection. When connection of one side is terminated, the other one can keep sending data. TCP connection termination has two methods: three-way handshaking and half-close four-way handshaking.

(4) Three-way handshaking connection release

See Figure 7-6

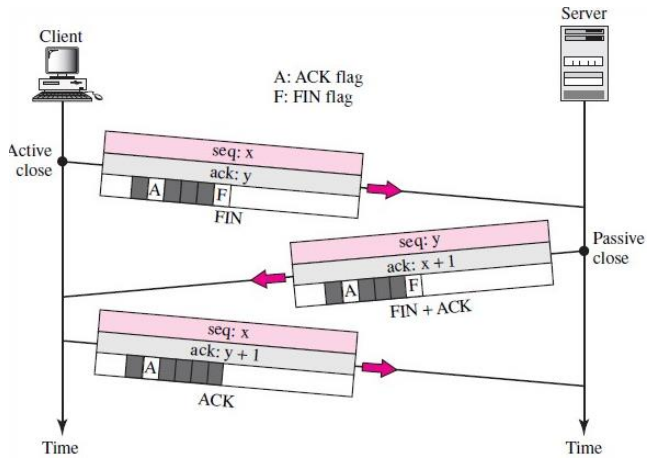

Figure 7-6 three-way handshaking termination

1) When the client wants to close TCP connection, it sends a TCP message and sets the FIN flag at 1.
2) When receiving the TCP message sent by step 1), The server passes the closing message of TCP connection to the corresponding process and send the second message (FIN+ACK

message) to acknowledge receiving FIN message from the client. Now the other direction of connection is closed.

3) The client send the last message to acknowledge receiving FIN message from the server. The message contains a acknowledge number which equals to the sequence number of FIN message from the server plus one.

(5) Half-close four-way handshaking connection termination

In TCP connection, one side can stop transmitting data, but keep receiving data from the other side. This is called half-close. Half-close is always started by the client. See Figure 7-7. The client sends FIN message and half closes the connection. The server sends ACK message and accepts the half-close. However, the server can still send message. When the server finishes transmiting data, it sends a FIN message and the client acknowledge it.

When half close a connection, the client can still receive data from the server and the server can receive acknowledge from the client. But the client can not send data to the server.

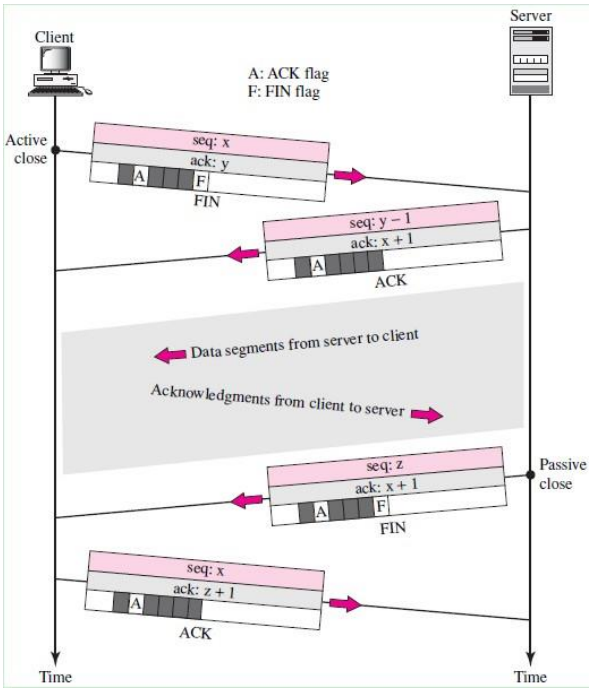

Figure 7-7 Half close

6. Traffic control

Before the sender receives acknowlege message from the receiver, traffic control can manage the number of sending data of the sender.

When not considering about traffic control, the transport layer can transmit one byte per time and waits for a acknowledge meassage from the receiver before sending next byte. This is a slow process, if the data will be transmitted for a long distance, then the sender will stay in a idle situation for a long time. The other case is that the transport layer protocol sends all data out in one time and ignores acknowledge message. It will accelerate the process of transmission. But the receiver may be paralyzed for receiving data in time. Moreover, the sender will know the situation that a part of data get lost, repeat, disorder and damage until the receiver check out all data.

Traffic control of TCP uses a compromise method. It defines a window in cache which is used to store the data that are ready to send. How much data should be sent depend on this window.

(1) Sliding window protocol

In order to control traffic, TCP uses a sliding window protocol. The window spans a portion of the buffer containing bytes received from the process. The window has two walls: one left and one right. Because both walls can slide, the window is called sliding window. See figure 7-8:

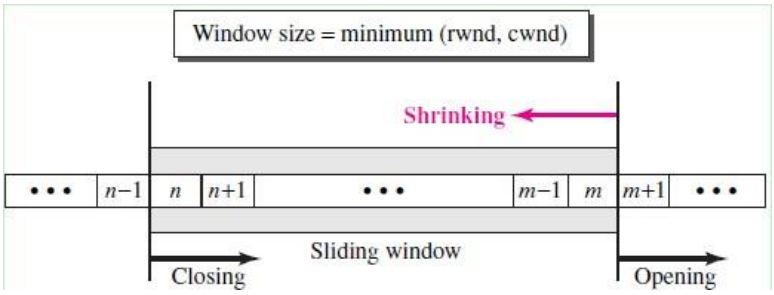

Figure 7-8 Sliding window

The window has three actions: opened, closed and shrunk. These actions are in the control of the receiver.

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. This is strongly discouraged and not allowed in some implementations because it means revoking the eligibility of some bytes for sending. This is a problem if the sender has already sent these bytes. Note that the left wall cannot move to the left because this would revoke some of the previously sent acknowledgments.

The size of the window at one end is determined by the minimum 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.

7. 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 dectection and correction in TCP is achieved through the use of three simple tools: checksum, acknowledgement, and time-out.

(1) Checksum

Each segment includes a checksum field which is used to check for a corrupted segment. If the segment is corrupted, it's discarded by the destination TCP and is considered as lost.

(2) Acknowledgement

TCP uses acknowledgements to confirm the receipt of data segments. Control segments thatcarry no data but consume a sequence number are also acknowledged.

(3) Retransmission

The key of the error control mechanism is the retransmission of segments. When a segment is corrupted, lost or delayed, it's retransmitted. In modern implementations, a segment is retransmitted in two cases: when a retransmission timer expires or when the sender receives three duplicate ACKs.

1) Retransmission after retransmission time-out

The sender maintains a retransmission timer for each TCP message. When the timer expires, the outstanding segment is retransmitted, even though it may casused by a delayed ACK , or a lost acknowledgment. The value of retransmission time-out is dynamic in TCP and is updated based on the round-trip time of segments. The round-trip time is the time needed for a segment to reach a destination and for an ackowledgement to be received.

2) Retransmission after three duplicate ACK segments

One segment is lost and the receiver receives so many out-of-order segments. To aleviate this situation, most implementations today follow the three-duplicate-ACKs rule and retransmit the missing sement immediately. This is fast retransmission.

Do not retransmit segments which comsume no sequence number. Do not retransmit all ACK segments.