UDP and TCP

- **Recommended reading:** This note is based on Chapters 10,11 of the text book.

- **User Datagram Protocol (UDP)**
  
  UDP provides unreliable connectionless data transfer between applications. It uses IP to carry message and *protocol port* to identify application processes. An application using UDP for data transfer handles the reliability issues itself.

- **Protocol port**
  
  In TCP/IP protocol suite, an IP address defines a destination host and a protocol port defines a destination process. A protocol port is identified by an integer between 0 and 65535.

- **UDP message format**
  
  A UDP message is called a user datagram and consists of a UDP header and a UDP data area. The UDP header has four fields.
  
  - UDP source port, specify the source port.
  - UDP destination port, specify the destination port.
  - UDP message length, give the length of the whole datagram (in byte).
  - UDP checksum, optionally used.

- **UDP encapsulation and layering**
  
  In the TCP/IP layering model, UDP lies in the layer above IP layer and below application layer. A UDP message, including UDP header and data, is encapsulated in an IP datagram. The IP layer is responsible only for transferring data between a pair of hosts on an internet, while UDP is responsible only for differentiating among multiple sources/destinations within one host.

- **Multiplexing/demultiplexing, port**
  
  Multiplexing/demultiplexing here means multiple users employ the same transport protocol UDP. Users are identified by port number or service access point. An application program must get a protocol port and the number for the port first and then send a UDP message. On receiving message, a protocol port can be considered as a queue. Once a datagram comes in, UDP checks the port number, if there is a corresponding port, then enqueue the datagram, otherwise discards the datagram.
  
  There are two approaches for assigning a port, universal port assignment and dynamic binding.
  
  Universal port assignment: A central authority assigns ports and publishes the list of assigned port numbers.
Dynamic binding: In this approach, ports are not globally known. When a program needs a port, the network software assigns one. To know the current port assignment on a remote computer, a request message is sent and the target machine replies the correct port number.

TCP/IP uses both approaches, universal one (the central authority is Internet Corporation for Assigned Names and Numbers ICANN) for common applications and dynamic one for others. The port numbers are classified into

1. **well known port numbers**, from 0 to 1023, assigned to the key services provided by computer systems;
2. **registered port numbers**, from 1024 to 49151, assigned to industry applications and processes; and
3. **dynamic port numbers**, from 49152 to 65535, used as temporary ports for specific communications.

The well known and registered port numbers are assigned universally.

- **Transmission Control Protocol (TCP)**

TCP provides reliable stream delivery network service which has the following properties.

Stream orientation: Data are considered as a stream of bytes. The stream delivery service on the destination passes to the receiver exactly the same sequence of bytes that the sender passes to source.

Virtual circuit connection: Protocols on two machines communicate with each other by sending messages across an internet to make an agreement (virtual connection) on transfer data. It looks like to set-up a dedicated path between the two machines, although no dedicated physical path is created. When the transfer is finished, the connection is terminated.

Buffered transfer: the data stream is buffered at source and partitioned into packets of reasonable size for efficient transmission. An application can use *push* mechanism to force sending the data even if the data does not fill one packet.

Unstructured stream: The service does not give structured data streams. Applications must understand stream content and agree on stream format before they initiate a connection.

Full duplex connection: The service allows concurrent transfer in both directions.

- **Reliability**

A technique used for reliable stream delivery on the unreliable datagram transfer is *positive acknowledgement with retransmission*. The destination sends source an ACK message when it receives a packet. The source waits for an ACK for the sent message. If no ACK (use a timer to check), the source retransmits the packet.

- **Sliding windows**

Simple positive ACK is not efficient. Sliding window technique improves the efficiency.
Source places a fixed size window on the sequence of packets. It can transmit all packets within the window in sequence. Once a packet is sent, the window shrinks in one packet position from the trailing edge. Once an ACK is received, the window expands one packet position from leading edge. It looks like the window sliding from left to right.

Destination also has a window of the same size. The packet numbers within the window indicate the packets expected. Once a packet is received, the window shrinks from the left side. Once an ACK is sent, the window expands from the right side.

- **Segment, stream, sequence number**

  TCP views the data stream as a sequence of bytes and partitions the sequence into segments. Each segment is transmitted in an IP datagram. TCP uses a specialized sliding window which operates at byte level to solve two important problems: efficient transmission and flow control.

  In TCP sliding window, bytes of the data stream are numbered sequentially (each byte in the data stream has a *sequence number*), and a sender uses three pointers for each connection. The 1st pointer marks the left of the window, separates the bytes which have been set and acknowledged from the bytes which have not been acknowledged. The 2nd pointer marks the right of the window, defines the highest bytes in the sequence that can be sent. The 3rd point is within the window and separates the bytes which have been set and the bytes have not been sent.

  Receiver has a similar sliding window. Receiver sends sender an ACK with sequence numbers for the bytes received.

  For a full-duplex connections, two pairs of windows are maintained.

- **Variable window size and flow control**

  TCP allows the size of sliding windows change dynamically. A receiver can specify how many bytes it is ready to receive in an ACK message. The number of bytes specified in the ACK message is sometimes called the *number of credits*. The sender can have a sliding window of the size defined by the number of credits.

  The variable window size provides a mechanism for flow control.

- **TCP segment format**

  A TCP segment has a segment header (also called TCP header) followed by data. In the TCP header, there are following fields.

  - Source port, the protocol port number for an application at the source.
  - Destination port, the protocol port number for an application at the destination.
  - Sequence number, the sequence number of the 1st byte in the segment. During the connection establishment, source creates an initial sequence number. The sequence number of the last byte in the segment can be determined from the segment size.
  - Acknowledge number, the sequence number of the byte the destination expects to receive.
Header length, the length of the header.
Reserved, reserved for future use.
Control, has six bits.
   - URG, Urgent pointer field is valid
   - ACK, Acknowledgement field is valid
   - PSH, Push the segment
   - RST, Reset the connection
   - SYN, Synchronize sequence numbers
   - FIN, Terminate the connection
Window, gives the number of credits.
Checksum, for error check.
Urgent pointer.

- **Acknowledgements**
  A destination uses sequence numbers to record the segments received and acknowledges the longest contiguous prefix of the stream received. The ACK specifies the sequence number of the next byte to the prefix (the next byte the destination expects to receive).

- **Timeout and retransmission**
  If a segment gets lost, TCP retransmits it. TCP uses timer for retransmission. Once a segment is sent, a timer is set-up, and if the ACK has not been received when the timer expires, TCP retransmits the segment.

  TCP uses dynamic timer values based on delays of segments. TCP records the round trip time, RTT, for delivering each segment (from sending to receiving ACK) and computes a smoothed round trip time, SRTT as follows:

  $$ SRTT(k + 1) = \alpha \times SRTT(k) + (1 - \alpha) \times RTT(k + 1). $$

  The timeout value is set to

  $$ Timeout = \beta \times SRTT. $$

  How to choose $\beta$ is not easy. The original specification recommended $\beta = 2$. More discussions on choosing $\beta$ can be found in Sections 13.17-13.19 of the text.

- **Ports, connections, and end points**
  TCP resides above IP layer and below application layer. TCP allows multiple applications on the same host to communicate concurrently. Because TCP is connection-oriented, it uses connection not a single object (a datagram) as its fundamental abstraction. A connection is identified by a pair of endpoints. Each endpoint is a pair $(\text{host}, \text{port})$ of integers.
• Establish a connection

A connection can be set-up only if both endpoints agree to do so. One endpoint uses active open to request the connection and the other endpoint uses passive open to accept the request.

TCP uses a three-way handshake technique to establish a connection. The three-way handshake guarantees that both sides are ready to transfer data (and they know they are both ready), and allows both sides to agree on initial sequence numbers.

• Terminate a connection

TCP uses a modified three-way handshake to terminate a connection.

• Response to congestion

Sliding windows and ACK are used to handle the congestion at the end point. TCP also deals with the congestion in the Internet. The TCP standard recommends slow-start and multiplicative decrease techniques to avoid congestion. In addition to the window size of the source, there is a second limit, called congestion window limit. The number of segments can be sent is limited by

\[ awnd = \min\{credit, cwnd\}, \]

where credit is the window size of the source and cwnd is the congestion window limit.

Slow-start: Start from \( cwnd = 1 \) and on receiving each ACK message, \( cwnd \) is doubled until a predefined threshold value \( t \). When \( cwnd \geq t \), \( cwnd \) is increased by one per ACK message.

Multiplicative decrease: When a timeout occurs, \( t = cwnd/2 \) and goto slow-start (from \( cwnd = 1 \)).

• Duplication detection

If an ACK message is lost, the segment is retransmitted. The receiver must recognize duplicates. If the duplicate is got prior to the close of the connection, the receiver knows ACK is lost and retransmits the ACK. The source will not be confused by multiple same ACKs if the sequence number space is big enough to not cycle within segment lifetime. If duplicate is got after the close of the connection, the segment is discarded.

• TCP state machine

See Figure 11.16 (page 235) of the text book.