TCP & UDP

C.K. Kim
Transport Layer

- Functions

- Transport

  - Unreliable End-to-end Packet Delivery

  - Reliable Byte Stream
    - Error control
    - Flow control
    - Congestion control

  Connection Management
  Multiplexing/Demultiplexing

- Network
Transport Layer Protocols

- The Internet has two transport protocols; **TCP** and **UDP**

- **Why two protocols??**
  - Applications with diverse requirements

- **TCP**
  - Connection-oriented
  - **Reliable** end-to-end byte stream over IP
  - Application Multiplexing/Demultiplexing

- **UDP**
  - Connectionless
  - **Unreliable**
  - Application Multiplexing/Demultiplexing
UDP

- Reliable delivery is not required
- Real-time communication
- Multicast/Broadcast
- Connection setup/close overhead is critical
- Application itself has error/flow control mechanisms
De/Multiplexing

- Use **port** to identify processes
  - 16 bit integer
  - Each application process is associated with a specific port
  - Application-to-port association should be known to senders

- **Well-known port**: An app. uses the same port number across all hosts (servers)

**Well-known UDP port examples**
- Echo (7)
- DNS (53)
- DHCP (67)
TCP Format

<table>
<thead>
<tr>
<th>SrcPort</th>
<th>DestPort</th>
</tr>
</thead>
<tbody>
<tr>
<td>SN</td>
<td></td>
</tr>
<tr>
<td>AN</td>
<td></td>
</tr>
<tr>
<td>HL</td>
<td>0</td>
</tr>
<tr>
<td>CheckSum</td>
<td>UrgPtr</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
</tbody>
</table>

**Error control**

**AdvertiseWindow**

**Flow control**

**Flags**

SYN, FIN, RST, PSH, URG, ACK
Error Control

- **End-to-end issues**
  - Transport layer is similar to data link layer
    - Both layers provide error control function

- **Differences**
  - **Dynamic connection setup/release**
    - Unknown peer capability
    - Unknown path capacity, quality
  - Variable and unpredictable RTT
  - Out-of-sequence deliveries
  - Dynamic resource sharing
RTT & TO (Time Out)

- Need to estimate exact RTT
  - TO & Retransmission
- Original RTT estimation method
  - \( RTT = a \times RTT + (1-a) \times \text{SampleRTT} \)
  - \( TO = b \times RTT \)

- Karn/Partridge algorithm
  - What if a segment is lost?

- Jacobson/Karels algorithm

\[
\begin{align*}
\text{Diff} &= \text{SampleRTT} - \text{RTT} \\
\text{RTT} &= \text{RTT} + (d \times \text{Diff}) \\
\text{Dev} &= \text{Dev} + d \times (\text{Diff} - \text{Dev}) \\
\text{TO} &= u \times \text{RTT} + p \times \text{Dev}
\end{align*}
\]
Flow Control

- Link-level flow control vs End-to-end flow control
  - No information on receiver’s resource
  - Delayed feedback
- TCP byte stream delivery model
Flow Control

- The receiver informs the source of the available buffer size

- A receiver sends permits to a sender
  - \( AW = \text{MaxRcvBuffer} - (\text{LastByteExpected} - \text{LastByteRead}) \)

- A sender adjust its window size up to \( AW \)
  - \( \text{LastByteSent} - \text{LastByteAcked} \leq AW \)
Connection Management

- **Connection**
  - Synchronization of hosts before data exchange
  - Connection setup & teardown (termination)

- **Client-Server model**
  - Client: Initiate actions, active mode
  - Server: Wait for service request, passive mode

- **Connection establishment procedure is asymmetric**
  - Passive open - Server
  - Active open - Client
Connection Setup

- Use **TWH (Three-Way Handshaking)**
  - Three message exchanges for connection establishment
- Message contains connection id (SN)
  - Random

**Diagram:**

- **Client**
  - SYN, seq=x
  - SYN+ACK, seq=y, ack=x+1
  - ACK, seq=y+1

- **Server**
  - Passive open/Listen
  - Active open/SYN
  - SYN/ACK, SYN+ACK
  - Send/SYN
  - SYN_SENT
  - ESTABLISHED
  - Close/FIN

**States:**

- CLOSED
- LISTEN
- SYN_RCVD
- SYN_SENT
- ESTABLISHED
- Passive open
- Close
- Active open/SYN
- SYN/ACK, SYN+ACK
- Send/SYN
- SYN_SENT
- ESTABLISHED
- Close/FIN
Connection Termination

- **Symmetric**
  - Either server or client initiates connection termination

- **Graceful termination**
  - Even after you finish sending, listen until the other side finishes

```
FIN seq=x
Rcv FIN
ACK x+1

Rcv ACK

Rcv FIN+ACK
ACK y+1

Rcv ACK

ACK x+1
```

```plaintext
FIN seq=x
Rcv FIN
ACK x+1

FIN seq=y,ACK x+1

Rcv FIN+ACK

Rcv FIN

Rcv ACK

Rcv ACK

ACK x+1
```