CSE 461 Lab :
Transmission Control Protocol
Layers of Internet

<table>
<thead>
<tr>
<th>OSI Model</th>
<th>Internet Model</th>
<th>Internet Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Application</td>
<td>HTTP, HTTPS, SSH, DNS, SSL, FTP, POP3, SMTP, IMAP, Telnet, NNTP</td>
</tr>
<tr>
<td>Presentation</td>
<td>Transport</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>Session</td>
<td>Network</td>
<td>IP, ICMP, ARP, DHCP</td>
</tr>
<tr>
<td>Transport</td>
<td>Network Link</td>
<td>Ethernet, PPP, ADSL</td>
</tr>
<tr>
<td>Network</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datalink</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Transport Layer – TCP vs UDP

<table>
<thead>
<tr>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable</td>
<td>Unreliable</td>
</tr>
<tr>
<td>Connection-oriented</td>
<td>Connectionless</td>
</tr>
<tr>
<td>Segment retransmission and flow control through windowing</td>
<td>No windowing or retransmission</td>
</tr>
<tr>
<td>Segment sequencing</td>
<td>No sequencing</td>
</tr>
<tr>
<td>Acknowledge segments</td>
<td>No acknowledgement</td>
</tr>
</tbody>
</table>
TCP Stream (Packet config)
Sequence number and ACK number

- **Sequence number**: byte-stream number of the first byte of the segment

- **ACK number**: the sequence number of the next byte the server is expecting.
Transport Layer – TCP vs UDP

TCP Client

```python
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE)
```

TCP Server

```python
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)
conn, addr = s.accept()
data = conn.recv(BUFFER_SIZE)
```

UDP Client

```python
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

UDP Server

```python
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))
data, addr = sock.recvfrom(1024)
```
Three Way Handshake

Client

<SYN_SENT>
connect()
SYN M
SYN N, ACK M+1
ACK N+1
<ESTABLISHED>

Server

listen() accept()
<LISTEN>
<SYN_RECV>

<XXXXX> == State of TCP state machine

ACK segment can include data
accept() returns
<ESTABLISHED>
Tearing down Connection

```
client

<table>
<thead>
<tr>
<th>close</th>
<th>FIN M</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>ACK M+1</td>
</tr>
<tr>
<td>close</td>
<td>FIN N</td>
</tr>
<tr>
<td></td>
<td>ACK N+1</td>
</tr>
</tbody>
</table>

server

(passive close)

close

read return
```
TCP States

- **CLOSED**: wait 30 seconds
- **SYN_SENT**: send SYN
- **FIN_WAIT_1**: receive ACK, send nothing
- **FIN_WAIT_2**: receive FIN, send ACK
- **TIME_WAIT**: client application initiates a TCP connection
- **ESTABLISHED**: receive SYN & ACK, send ACK
  
  **Legend**
  - Send FIN
  - Receive ACK
  - Send ACK
  - Send SYN
  - Send nothing
TCP States

- **CLOSED**
  - In this state, the server application creates a listen socket.

- **LISTEN**
  - Transition to LISTEN from CLOSED on receiving SYN.
  - Send SYN & ACK on receiving SYN.

- **SYN_RCVD**
  - Transition to SYN_RCVD from LISTEN on receiving ACK.
  - Send nothing on receiving ACK.

- **ESTABLISHED**
  - Transition to CLOSED on receiving FIN.
  - Send nothing on receiving FIN.

- **CLOSE_WAIT**
  - Transition to LAST_ACK from ESTABLISHED on receiving FIN.
  - Send ACK on receiving FIN.

- **LAST_ACK**
  - Transition to CLOSED from CLOSE_WAIT on receiving ACK.
  - Send nothing on receiving ACK.
TCP Socket Options

SO_TIMEOUT : sets a timeout value

SO_LINGER : sets a timeout to gracefully close the connection. (max is 65535)

SO_KEEPALIVE: detect if the peer host crashes

SO_REUSEPORT: allows two process to bind to the same port

SO_REUSEADDR: allows reuse of local address while it is not active
Questions

- Why do we need any connection protocol at all? Why isn't a TCP segment with sequence number 0 enough to start a connection?

- Given that we're going to do the 3-way handshake, why isn't always starting at initial sequence number 0 a good idea?

- Why randomize initial sequence numbers?

- Why the timeout state in the connection close portion of the protocol?

- Why not reuse sockets/port numbers until after a delay, by default?