Transport Layer Service Model

- provide logical communication between application processes running on different hosts
- transport layer protocols are implemented in the end systems

Layering: Protocol Stack

- IP address
- MAC address
- Ethernet
- Network Adapter (NIC)
- Frame
- Datagram
- Segment/Datagram
- Message
- Data
- Port no.
Transport Layer Services

- Decompose/compose message into/from 4-PDU (TPDU)
- Demultiplexing/multiplexing
- Connection-oriented Service (TCP)  
  - Reliable data transfer (RDT)
  - Flow control
  - Congestion control
  - TCP connection management
- Connectionless Service (UDP)  
  - checksum

Transport Services (1)

- Decompose/compose message into/from 4-PDU (Protocol Data Unit)
  - breaking application message into smaller chunks
  - Add transport-layer header to each chunk
  - 4-PDU for TCP is called segment
  - 4-PDU for UDP is called datagram

Transport Service (2)

- Demultiplexing/multiplexing
  - IP only delivers data between end systems identified with unique IP address
  - IP does not deliver data between the application processes
  - Demultiplexing: delivering the data in a transport-layer segment to the correct application process
  - Multiplexing: gathering data at the source host from different application processes, enveloping data with header information to create segments and passing the segments to the network layer
Transport Service (3)

- Connection-oriented Service
  - TCP
    - Reliable data transfer
    - TCP flow control
    - TCP congestion control
    - TCP connection management

- Connectionless Service
  - UDP

Through socket programming

Multiplexing/demultiplexing: Examples

Web client host A

Source IP: C
Source port: x
Dest IP: B
Dest port: 80

Source IP: C
Source port: y
Dest IP: B
Dest port: 80

Web server B

port use: Web server

UDP (User Datagram Protocol)

- Bare bone transport protocol
- Best-effort service
- Connectionless
- RFC 768
- Eg. DNS

Pros:
- No connection delay
- No connection state
- Small packet header overhead (8 bytes vs. TCP’s 20 bytes)
- Unregulated send rate

UDP datagram format

<table>
<thead>
<tr>
<th>Source port</th>
<th>Dest port</th>
<th>Length</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 bits</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Application data (message)
Applications Using UDP

- Multicast
- Real-time applications
- Multimedia applications
- Tolerate a small fraction of packet loss

TCP Segment Structure

```
<table>
<thead>
<tr>
<th>0</th>
<th>16</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledge Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLEN</td>
<td>Reserved</td>
<td>Code Bits</td>
<td>Window</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent Pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Padding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

TCP Segment Structure

```
<table>
<thead>
<tr>
<th>0</th>
<th>16</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledge Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLEN</td>
<td>Reserved</td>
<td>Code Bits</td>
<td>Window</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent Pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Padding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
TCP Flow Control

- **Flow Control**
  - Sender: explicitly informs sender of (dynamically changing) the amount of free buffer space
  - Receiver: keeps the amount of transmitted, unACKed data less than most recently received

- **RcvWindow** field in TCP segment

Congestion Control Problem

- When intermediate machines (routers) become overloaded, the condition is called **congestion**, the mechanisms to solve the problem are called **congestion control mechanism**
- Difference from flow control?

Congestion Control Approaches

- **End-end congestion control**
  - Network layer provides no explicit support to the transport layer for congestion-control purposes

- **Network-assisted congestion control**
  - Routers provide explicit feedback to the sender regarding the congestion state in the network
Congestion Window

- At any time, TCP window size:
  \[ \text{LastByteSent} - \text{LastByteAcked} \leq \text{Allowed window} = \min(\text{receiver advertisement}, \text{congestion window}) \]

Response To Congestion

- Two recommended techniques:
  - Slow start
  - Congestion avoidance
    - Additive increase
    - Multiplicative techniques

TCP Slowstart

- **Slowstart algorithm**
  - initialize: \( \text{Congwin} = 1 \)
  - for (each segment ACKed)
    - \( \text{Congwin}++ \)
  - until (loss event OR \( \text{CongWin} > \text{threshold} \))

  - exponential increase (per RTT) in window size
  - loss event: timeout and/or three duplicate ACKs

  How many RTT is needed before TCP can send N segments?
TCP Congestion Avoidance
/* slowstart is over */
/* Congwin > threshold */
Until (loss event) {
  every w segments ACKed: Congwin++
  threshold = Congwin/2
}
Congwin = 1
perform slowstart

AIMD – TCP Congestion Avoidance
AIMD: additive increase, multiplicative decrease
- increase window by 1 per RTT
- decrease threshold by factor of 2 on loss event

TCP Timeout and Retransmission
Adaptive retransmission algorithm
- Next_RTT = (α * last_estimated_RTT) + (1 - α) * newly_collected_RTT_sample, 0 < α < 1
- Typically, α = 7/8
TCP Connection Management

- Establishing a connection
- Initial Sequence Number (ISN)
- Closing a TCP connection
- Connection Reset
- TCP state machine

Three-way Handshake Protocol

A node chooses x, B node ACKs x+1, which specifies that B will expect octet from x+1
B node piggybacks his chosen ISN, y, on second message, with SYN set
A node ACKs this y by y+1

ISN: Initial Sequence Number

- Two tasks have been accomplished by 3-way handshaking
  - Both sides are ready to transfer data
  - Both sides agree on ISN, ISN is chosen randomly
- Action taken in the three messages
  - A node chooses x, B node ACKs x+1, which specifies that B will expect octet from x+1
  - B node piggybacks his chosen ISN, y, on second message, with SYN set
  - A node ACKs this y by y+1
TCP: Closing Connection

The Two-Army Problem

Connection Release (4 Scenarios)

TCP: Close Connection
TCP: Connection Reset

- Normally, TCP closes connection, analogous to closing a file
- Under abnormal conditions, an RST bit indicates immediate abort, resources such as buffers are released

TCP Segment Structure

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>16</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledge Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLEN</td>
<td>Reserved</td>
<td>Code Bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent Pointer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Padding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TCP Code Bits

<table>
<thead>
<tr>
<th>Bit (left to right)</th>
<th>Meaning if bit set to 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>URG</td>
<td>Urgent pointer field is valid</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledgement field is valid</td>
</tr>
<tr>
<td>PSH</td>
<td>The segment requires a push</td>
</tr>
<tr>
<td>RST</td>
<td>Reset the connection</td>
</tr>
<tr>
<td>SYN</td>
<td>Synchronize sequence number</td>
</tr>
<tr>
<td>FIN</td>
<td>Sender has reached end of its byte stream</td>
</tr>
</tbody>
</table>
TCP: Finite State Machine

TCP client lifecycle

TCP server lifecycle