Transport Layer Protocols

→ end-to-end
→ runs on top of network layer protocols
→ treat network layer & below as black box

Three-level encapsulation: e.g., TCP and UDP

Meaning of protocol “stack”
→ push/pop headers and trailers
Network layer (IP) assumptions:
  • unreliable
  • out-of-order delivery
  • absence of QoS guarantees (delay, throughput, etc.)
  • insecure (IPv4)
    → IPsec

Additional performance properties:
  • works “ok”
  • can break down under high load conditions
    → e.g., flash crowds, DDoS and worm attacks
  • wide behavioral range
    → from good to bad
Goal of UDP (User Datagram Protocol):

\[\rightarrow \] process identification

\[\rightarrow \] port number as demux key

\[\rightarrow \] minimal support beyond IP
UDP packet format:

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

Payload

Checksum calculation: pseudo header

<table>
<thead>
<tr>
<th>Source Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Address</td>
</tr>
<tr>
<td>00 ··· 0</td>
</tr>
</tbody>
</table>

→ pseudo header, UDP header and payload
UDP usage:

- multimedia streaming
  → lean and nimble
  → at minimum requires process identification
  → reliability addressed above UDP if needed: FEC or ARQ
  → congestion control addressed above UDP

- lightweight client/server applications
  → persistent state leads to overhead
Goals of TCP (Transmission Control Protocol):

- process identification
- reliable communication: ARQ
- speedy communication: congestion control
- segmentation and MTU
- connection establishment and tear-down

→ complex mixture of functionalities
Provide stream interface to higher level protocols
→ exported semantics: contiguous byte stream
→ e.g., accessed using read(), write() system calls

Segmentation:
• segment stream of bytes into blocks of fixed size
• segment size determined by TCP MTU (Maximum Transmission Unit)
• actual unit of transmission in ARQ
• efficiency and reduced fragmentation
TCP packet format:

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acknowledgement Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Header Length</th>
<th>U A P R S F</th>
<th>Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G K H T N</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checksum</th>
<th>Urgent Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options (if any)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA (if any)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
• Sequence Number: position of first byte of payload
• Acknowledgement: next byte of data expected (receiver)
• Header Length (4 bits): 4 B units
• URG: urgent pointer flag
• ACK: ACK packet flag
• PSH: override TCP buffering
• RST: reset connection
• SYN: establish connection
• FIN: close connection
• Window Size: receiver’s advertised window size
• Checksum: prepend pseudo-header
• Urgent Pointer: byte offset in current payload where urgent data begins
• Options: MTU; take min of sender and receiver
Checksum calculation: pseudo header

<table>
<thead>
<tr>
<th>Source Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Address</td>
</tr>
<tr>
<td>00 ··· 0</td>
</tr>
</tbody>
</table>

$\rightarrow$ pseudo header, TCP header and payload
TCP connection establishment (3-way handshake):

- $X$, $Y$ are chosen randomly
  - $\rightarrow$ mitigate sequence number prediction
- piggybacking
2-party consensus problem: are $A$ and $B$ in agreement about the state of affairs after 3-way handshake?

→ in general: in networks with unbounded delay, there is no solution

→ acknowledging the ACK . . .

→ lunch date problem

→ also TCP session termination
TCP connection termination:

- **full duplex**
- **half duplex**
Finite state machine representation: