Transmission Control Protocol (TCP)

Connection oriented
Explicit set-up and tear-down of TCP session

Stream-of-bytes service
Sends and receives a stream of bytes, not messages

Reliable, in-order delivery
Checksums to detect corrupted data
Acknowledgments & retransmissions for reliable delivery
Sequence numbers to detect losses and data reordering

Flow control
Prevents overflow of the receiver’s buffer

Congestion control
Adapts to network congestion for the greater good

Does TCP Duplicate Lower Layer Services?

Sliding-window based algorithm
Incorporates ACK, NACKs, retransmissions, timeouts, ordering

How is TCP different?
Runs over a route rather than a single physical link → Needs to establish a connection and negotiate parameters (sliding window size)
Adapts to heterogeneous conditions
- RTT varies with connected hosts, time of the day etc., unpredictable
- Flow control on computers with different resources
Reorders and retransmits packets end-to-end
Even if packets are in order on one physical link this does not guarantee end-to-end ordering
Provides congestion control
No immediate feedback from the link, no knowledge of conditions along the way
Congestion control based on feedback
TCP Byte Service

In general, TCP is a full duplex protocol, i.e. data flows in both directions.

TCP Segment

IP packet
No bigger than Maximum Transmission Unit (MTU)
E.g., up to 1500 bytes on an Ethernet

TCP packet
IP packet with a TCP header and data inside
TCP header is typically 20 bytes long

TCP segment
No more than Maximum Segment Size (MSS) bytes
E.g., up to 1460 consecutive bytes

TCP Segment

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcPort</td>
<td>Source port</td>
</tr>
<tr>
<td>DstPort</td>
<td>Destination port</td>
</tr>
<tr>
<td>SequenceNum</td>
<td>Sequence number</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>Acknowledgment (ACK)</td>
</tr>
<tr>
<td>HdrLen</td>
<td>Header length</td>
</tr>
<tr>
<td>Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>AdvertisedWindow</td>
<td>Advertised window size</td>
</tr>
<tr>
<td>Checksum</td>
<td>Checksum</td>
</tr>
<tr>
<td>UrgPtr</td>
<td>Urgent pointer</td>
</tr>
<tr>
<td>Options (variable)</td>
<td>Options that can vary</td>
</tr>
<tr>
<td>Data</td>
<td>Data (variable)</td>
</tr>
</tbody>
</table>
Simplified TCP Flow

Data Sequence Numbers flow toward the receiver
Acks and Window Size flow the opposite direction

Header Fields in a Segment

<SrcPort, SrcIPAddr, DestPort, DestIPAddr> define a TCP connection

SequenceNum: Sequence number of the first byte in the segment

Flags: 6-bit field: SYN, FIN, RESET, PUSH, URG, ACK
    SYN, FIN: Establish and terminate a TCP connection
    ACK: request for an acknowledgement
    URG: Urgent Data UrgPtr points to where the urgent data ends
    PUSH: a notification to the application running over TCP
    RESET: some error occurred and the connection needs to be aborted

Sequence Numbers

Host A

ISN (initial sequence number)

Sequence number = 1st byte

Host B

TCP Data

ACK sequence number = next expected byte
**Initial Sequence Number (ISN)**

Sequence number for the very first byte

E.g., Why not a de facto ISN of 0?

**Practical issue**

- IP addresses and port #s uniquely identify a connection
- Same port #s may get used again to establish new connection
- and there is a chance an old packet is still in flight
- and might be associated with the new connection

So, TCP requires changing the ISN over time

- Set from a 32-bit clock that ticks every 4 $\mu$s
- Wraps around once every 4.55 hours!

But, this means the hosts need to exchange ISNs

---

**Connection Establishment**

**Three-Way Handshake**

Exchange of three messages

---

**Client’s SYN Packet**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>Client’s Port</td>
<td>Server’s Port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client’s ISN</td>
<td>Acknowledgment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>SYN</td>
<td>AdvertisedWindow</td>
</tr>
<tr>
<td>Checksum</td>
<td>UrgPtr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (variable)</td>
<td>Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tearing Down the Connection

Closing the connection
- Finish (FIN) to close and receive remaining bytes
- And other host sends a FIN ACK to acknowledge
- Reset (RST) to close and not receive remaining bytes