TCP

Raj Jain
Professor of CIS
The Ohio State University
Columbus, OH 43210
Jain@ACM.Org

http://www.cis.ohio-state.edu/~jain/
Overview

- Key features, Header format
- Mechanisms, Implementation choices
- Slow start congestion avoidance, Fast Retransmit/Recovery
- Selective Ack and Window scaling options
- UDP

Ref: RFCs, Thomas
Key Features of TCP

- Connection oriented
- Point-to-point communication: Two end-points
- Reliable transfer: Data is delivered in order
- Full duplex communication
- Stream interface: Continuous sequence of octets
- Reliable connection startup: Data on old connection does not confuse new connections
- Graceful connection shutdown: Data sent before closing a connection is not lost.
Transport Control Protocol (TCP)

- Key Services:
  - Send: Please send when convenient
  - Data stream push: Please send it all now, if possible.
  - Urgent data signaling: Destination TCP! please give this urgent data to the user (Urgent data is delivered in sequence. Push at the should be explicit if needed.)
  - Note: Push has no effect on delivery. Urgent requests quick delivery
## TCP Header Format

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Dest Port</th>
<th>Seq No</th>
<th>Ack No</th>
<th>Data Offset</th>
<th>Resvd</th>
<th>Control</th>
<th>Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td>32</td>
<td>32</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checksum</th>
<th>Urgent</th>
<th>Options</th>
<th>Pad</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td>x</td>
<td>y</td>
<td>Size in bits</td>
</tr>
</tbody>
</table>

- Size in bits
TCP Header

- **Source Port (16 bits):** Identifies source user process
  20 = FTP, 23 = Telnet, 53 = DNS, 80 = HTTP, ...

- **Destination Port (16 bits)**

- **Sequence Number (32 bits):** Sequence number of the first byte in the segment. If SYN is present, this is the initial sequence number (ISN) and the first data byte is ISN+1.

- **Ack number (32 bits):** Next byte expected

- **Data offset (4 bits):** Number of 32-bit words in the header

- **Reserved (6 bits)**
TCP Header (Cont)

- Control (6 bits): Urgent pointer field significant, Ack field significant, Push function, Reset the connection, Synchronize the sequence numbers, No more data from sender

```
| URG | ACK | PSH | RST | SYN | FIN |
```

- Window (16 bits): Will accept [Ack] to [Ack]+[window]
TCP Header (Cont)

- Checksum (16 bits): covers the segment plus a pseudo header. Includes the following fields from IP header: source and dest addr, protocol, segment length. Protects from IP misdelivery.

- Urgent pointer (16 bits): Points to the byte following urgent data. Lets receiver know how much data it should deliver right away.

- Options (variable):
  - Max segment size (does not include TCP header, default 536 bytes), Window scale factor, Selective Ack permitted, Timestamp, No-Op, End-of-options
TCP Options

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>End of Valid options in header</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>No-op</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Maximum Segment Size</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Window Scale Factor</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>Timestamp</td>
</tr>
</tbody>
</table>

- **End of Options**: Stop looking for further option
- **No-op**: Ignore this byte. Used to align the next option on a 4-byte word boundary
- **MSS**: Does not include TCP header
TCP Checksum

- Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the TCP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.
- Checksum field is filled with zeros initially
- TCP length (in octet) is not transmitted but used in calculations.
- Efficient implementation in RFC1071.

<table>
<thead>
<tr>
<th>Source Adr</th>
<th>Dest. Adr</th>
<th>Zeros</th>
<th>Protocol</th>
<th>TCP Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32</td>
<td>8</td>
<td>8</td>
<td>16</td>
</tr>
</tbody>
</table>

TCP Header | TCP data
TCP Service Requests

- Unspecified passive open:
  Listen for connection requests from any user (port)
- Full passive open:
  Listen for connection requests from specified port
- Active open: Request connection
- Active open with data: Request connection and transmit data
- Send: Send data
- Allocate: Issue incremental allocation for receive data
- Close: Close the connection gracefully
- Abort: Close the connection abruptly
- Status: Report connection status
TCP Service Responses

- Open ID: Informs the name assigned to the pending request
- Open Failure: Your open request failed
- Open Success: Your open request succeeded
- Deliver: Reports arrival of data
- Closing: Remote TCP has issued a close request
- Terminate: Connection has been terminated
- Status Response: Here is the connection status
- Error: Reports service request or internal error
TCP Mechanisms

- Connection Establishment
  - Three way handshake
  - SYN flag set ⇒ Request for connection
    
    SYN, ISN = 100
    SYN, ISN = 350, Ack 101
    Ack 351

- Connection Termination
  - Close with FIN flag set
  - Abort
Three-Way Handshake

- 3-way handshake for opening and closing connections. Necessary and sufficient for unambiguity despite loss, duplication, and delay.

Fig 20.5
TCP Retransmission

Seq=3
300 bytes sent
Ack 303
Seq=303
300 bytes sent
Ack 303
Seq=603
300 bytes sent
Ack 303
Seq=303
300 bytes sent
Ack 903

X Not delivered

Timeout

The Ohio State University

Raj Jain
T/TCP: Transaction Oriented TCP

- Three-way handshake ⇒ Long delays for transaction-oriented (client-server) applications.
  T/TCP avoids 3-way handshakes [RFC 1644].

Client

Server

Request

Ack

Response

Ack+FIN
Data Transfer

- Stream: Every byte is numbered modulo $2^{32}$.
- Header contains the sequence number of the first byte.
- Flow control: Credit = number of bytes.
- Data transmitted at intervals determined by TCP.
  Push $\Rightarrow$ Send now.
- Urgent: Send this data in ordinary data stream with urgent pointer.
- If TPDU not intended for this connection is received, the “reset” flag is set in the outgoing segment.
Implementation Policies (Choices)

- **Send Policy:**
  - Too little $\Rightarrow$ More overhead. Too large $\Rightarrow$ Delay
  - Push $\Rightarrow$ Send now, if possible.

- **Delivery Policy:**
  - May store or deliver each in-order segment.
  - Urgent $\Rightarrow$ Deliver now, if possible.

- **Accept Policy:**
  - May or May not discard out-of-order segments
Implementation Policies (Cont)

- Retransmit Policy:
  - First only
  - Retransmit all
  - Retransmit individual
    (maintain separate timer for each segment)

- Ack Policy:
  - Immediate (no piggybacking)
  - Cumulative (wait for outgoing data or timeout)
Slow Start Flow Control

- Window = Flow Control Avoids receiver overrun
- Need congestion control to avoid network overrun
- The sender maintains two windows:
  - Credits from the receiver
  - Congestion window from the network
  - Congestion window is always less than the receiver window
- Starts with a congestion window (CWND) of 1 segment (one max segment size)
  ⇒ Do not disturb existing connections too much.
- Increase CWND by 1 every time an ack is received.
Slow Start (Cont)

- If packets lost, remember slow start threshold (SSThresh) to CWND/2
  - Set CWND to 1
  - Increment by 1 per ack until SSthresh
  - Increment by 1/CWND per ack afterwards

![Diagram showing the Slow Start process with timelines for congestion window (CWND), receiver window, idle interval, timeout, and SSThresh.]
Slow Start (Cont)

- At the beginning, SSThresh = Receiver window
- After a long idle period (exceeding one round-trip time), reset the congestion window to one.
- Exponential growth phase is also known as “Slow start” phase
- The linear growth phase is known as “congestion avoidance phase”
Fast Retransmit and Recovery (FRR)

- If 3 duplicate acks are received for the same packet, assume that the next packet has been lost. Retransmit it right away. Retransmit only one packet.

- Helps if a single packet is lost.
  Does not help if multiple packets lost.

- Ref: Stevens, Internet draft
FRR (Cont)

- Upon receiving the third duplicate Ack:
  - Set SSThresh to 1/2 of current CWND
  - Retransmit the missing segment
  - Set CWND to SSthresh+3
- For each successive duplicate Ack:
  - Increment CWND by 1 MSS
  - New packets are transmitted if allowed by CWND
- Upon receiving the next (non-duplicate) Ack:
  - Set CWND to SSthresh $\Rightarrow$ Enter linear growth phase
- Receiver caches out-of-order data.
Selective Ack (SACK)

- Initial Negotiation: Sender to receiver: “sack permitted”
  
  - SYN  Kind = 4  Length = 2
  
  - Size in bits

- Selective Ack: Variable length. Receiver to sender
  
  - Kind = 5  Length
    - Left edge of 1st block
    - Right edge of 1st block
    - 32 bits
    - Left edge of nth block
    - Right edge of nth block
SACK (Cont)

- Left edge = 1st sequence number in this block
- Right edge = sequence number immediately after the last sequence number in this block
- Ack field meaning is same as before. It is the next byte the receiver is expecting.
- When missing segments are received, ack field is advanced.
- Receiver can send SACK only if sender has “sack permitted” option in the SYN segment of the connection.
- Option Length = 8*n+2 byte for n blocks. 40 Bytes max options ⇒ Max n = 4
SACK (Cont)

- Data receiver can discard SACKed (queued) data. Sender must not discard data until acked.
- Example: 500 byte segments

![Diagram showing SACK acknowledgment example]

Ack 5000
5000
5500
6000

Ack 5500

SACK = 6000-6500
Window Scaling Option

- Long Fat Pipe Networks (LFN): Satellite links
  Pronounced elephan(t)
- Need very large window sizes.
- Normally, Max window = $2^{16} = 64$ KBytes
- Window scale option: $\text{Window} = W \times 2^\text{Scale}$
  
  Kind = 3  
  Length = 3  
  Scale

- Max window = $2^{16} \times 2^{255}$
- Option sent only in SYN + Ack segments
- RFC 1323
Random Early Drop (RED)

- Routers compute average queue size using an exponential weighted average.
- If the average queue size is more than a high-threshold, drop all arriving packets.
- If the average queue size is between the low and high threshold, drop the arriving packet with a probability $p = fn(\text{avg } q, \# \text{ of packets since the last dropped packet})$.
- High-rate sources are more likely to be dropped.
User Datagram Protocol (UDP)

- Connectionless end-to-end service
- No flow control. No error recovery (no acks)
- Provides port addressing
- Error detection (Checksum) optional. Applies to pseudo-header (same as TCP) and UDP segment. If not used, it is set to zero.
- Used by network management

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Dest Port</th>
<th>Length</th>
<th>Checksum</th>
<th>Size in bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>
TCP provides reliable full-duplex connections.
TCP Streams, credit flow control, 3-way handshake
Slow-start, Fast retransmit/recovery, SACK, Scaling
UDP is connectionless and simple. No flow/error control.