Chapter 20: TCP

Raj Jain
Professor of CIS

Raj Jain is now at
Washington University in Saint Louis
Jain@cse.wustl.edu
http://www.cse.wustl.edu/~jain/
Overview

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

Fig 20.1
Data not acked is retransmitted.
Adaptive Retransmission

- Retransmission timeout is based on measured round-trip time

Fig 20.3
Window Flow Control

Sender Events
- send data octets 1-1000
- send data octets 1001-2000
- send data octets 2001-2500
- receive ack for 1000
- receive ack for 2000
- receive ack for 2500

Receiver Events
- advertise window=2500
- ack up to 1000, window=1500
- ack up to 2000, window=500
- ack up to 2500, window=0
- application reads 2000 octets
- ack up to 2500, window=2000
- ack up to 3500, window=1000
- ack up to 4500, window=0
- application reads 1000 octets
- ack up to 4500, window=1000
- receive ack for 4500
- receive ack for 4500

Fig 20.4

The Ohio State University

Raj Jain
The system operates at a small window even if the receiver grants a large window.

Ref: RFC0813
Transport Control Protocol (TCP)

- Key Services:
  - Send: Please send when convenient
  - Data stream push: Please send it all now
  - Urgent data signaling: Destination TCP! please give this urgent data to the user
## 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>Flags</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>
<th>Size in bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td>x</td>
<td>y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TCP Header

- Source Port (16 bits): Identifies source user process
- 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)

- **Flags (6 bits):** Urgent pointer field significant, ack field significant, push function, reset the connection, synchronize the sequence numbers, no more data from sender

- **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 adr, protocol, segment length. Protects from IP misdelivery.

- Urgent pointer (16 bits): Points to the byte following urgent data. Lets receiver know how much urgent data is coming.

- Options (variable):
  Max TPDU size (Default 536 bytes)
  Window scale, SACK permitted
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

<table>
<thead>
<tr>
<th>Source Addr</th>
<th>Dest. Addr</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
- Full passive open: Listen for connection requests from specified user
- 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.

![Diagram of Three-Way Handshake](image)

Fig 20.5
T/TCP: Transaction Oriented TCP

- Three-way handshake $\Rightarrow$ Long delays for transaction-oriented (client-server) applications.
  T/TCP avoids 3-way handshakes [RFC 1644].
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 ⇒ 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 ⇒ More overhead. Too large ⇒ Delay
  - Push ⇒ Send now.

- **Delivery Policy:**
  - May store or deliver each in-order segment.
  - Push ⇒ Send now.

- **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 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 to CWND/2
  - Set CWND to 1
  - Increment by 1 per ack until SS threshold
  - Increment by 1/CWND per ack afterwards
Fast Retransmit and Recovery

- If the same packet is acked 3 times, 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
Selective Ack (SACK)

- Initial Negotiation: Sender to receiver: “sack permitted”
  
<table>
<thead>
<tr>
<th>SYN</th>
<th>Kind = 4</th>
<th>Length = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

- Selective Ack: Variable length. Receiver to sender
  
<table>
<thead>
<tr>
<th>Kind = 5</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Left edge of 1st block</td>
<td></td>
</tr>
<tr>
<td>Right edge of 1st block</td>
<td></td>
</tr>
<tr>
<td>32 bits</td>
<td></td>
</tr>
<tr>
<td>Left edge of nth block</td>
<td></td>
</tr>
<tr>
<td>Right edge of nth block</td>
<td></td>
</tr>
</tbody>
</table>
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 $\Rightarrow$ Max $n = 4$
SACK (Cont)

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

```
<table>
<thead>
<tr>
<th>Ack 5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
</tr>
<tr>
<td>5500</td>
</tr>
<tr>
<td>6000</td>
</tr>
<tr>
<td>Ack 5500</td>
</tr>
</tbody>
</table>
```

SACK = 6000-6500
Window Scaling Option

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

  \[
  \begin{array}{ccc}
  \text{Kind} &=& 3 \\
  \text{Length} &=& 3 \\
  \text{Scale} &=& \\
  \end{array}
  \]

- Max window = $2^{16} \times 2^{255}$
- Option sent only in SYN and SYN + Ack segments
- RFC 1323
TCP/IP Tools

- nslookup
- ping
- finger
- traceroute
- People: whois, knowbot, netfind
- Files: archie, gopher, WWW
- Ref: RFC 1739, RFC 1470
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>16</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.
References


Homework

- Read RFCs 0768 (UDP) 0793 (TCP), 1323 (Large Windows), 1470+1739 (TCP/IP Tools), 2018 (SACK)
- All RFCs up to 1949 are on the CD-ROM in the book Others can be found on http://ds.internic.net/
- Read internet draft: