TCP

• Transmission Control Protocol (TCP): defined in 1973 by Cerf and Kahn. RFC 793 specifies the protocol.

• Roadmap:
  • Lecture 1: the basics
    • sliding window mechanism
    • message formats
    • TCP state machine
  • Lecture 2: operation
    • congestion control algorithms
  • Lecture 3
    • Performance
• Other topics
  • TCP enhancements
  • TCP Friendly congestion control
TCP

• Provides a **reliable** end-to-end delivery service
  • Sliding window protocol

• Properties of TCP:
  • Stream orientation

  • Virtual circuit connection
    • Transport vs network level

• Buffered transport
  • for reliability and to handle speed mismatch

• Unstructured stream

• Full duplex connection
TCP

• **Reliability:** objective is to deliver a stream from one machine to another without duplication or loss of data.
• Requires error detection and recovery algorithms.
• General approach: automatic repeat and request (ARQ)
• Example: Stop-and-Wait ARQ:
TCP

• Go-Back-n (or sliding window) ARQ protocols are more efficient
TCP

• Select Repeat ARQ even more efficient
TCP

• A sliding window algorithm has:
  • sequence numbers
  • acknowledgement numbers
  • a send window size
  • an ack strategy
• Go-back-n can substantially increase throughput compared with a stop and wait protocol.
• Selective repeat optimal for high loss networks.
• TCP’s sliding window is a hybrid:
  • enhanced error recovery
  • a window size that changes over time
ACK/SNUM plot: based on tcpdump at a sender

A “+” is a data packet transmission
A “o” is an Ack packet arrival

Copyright Jim Martin 2007
TCP

• TCP is a protocol, not a piece of software.
• What does it do?
  • It specifies the format of the data and acknowledgements that two computers use to exchange data reliably.
  • It breaks application data into ‘segments’ and presents the original stream to the receiver.
  • It specifies the rules for error recovery.
  • It specifies how a connection is started and closed.
• What does it not do?
  • It does not specify a programming interface.
TCP

• TCP uses protocol numbers to demultiplex inbound traffic to applications.
• A UDP server will receive all packets destined for the server address/port.
• TCP uses a connection as the fundamental abstraction, not the port.
  • A connection is identified by a pair of endpoints.
  • Unlike UDP, a TCP server can multiplex multiple Cxs over a single port.
TCP

• Both ends of a TCP application program must agree that the connection is desired.
  • The “initiator” does this with an explicit connect (an active open).
  • The “receiver” does this with a passive open (the listen and accept).
TCP

- UDP Datagram vs TCP Segment
TCP

• TCP’s window varies dynamically, determined by congestion control algorithms.
• Many forms of congestion control— one way to classify is by location:
  • End to end flow control vs network level congestion.
    • End-to-end: sender and receiver participate.
      • A receiver can inform a sender when it runs out of buffers.
    • Network: routers participate in one of two ways:
      • implicit feedback indication such as dropped packets
      • explicit feedback indication such as source quench or setting a
        packet bit (a congestion indication bit).
TCP

- TCP Segment header (minimum 20 byte header)

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Dest. Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Seq. Number</td>
<td></td>
</tr>
<tr>
<td>Ack Number</td>
<td></td>
</tr>
<tr>
<td>HLEN</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>U A P R S F</td>
</tr>
<tr>
<td>TCP Checksum</td>
<td>window size</td>
</tr>
<tr>
<td></td>
<td>Urgent Ptr</td>
</tr>
<tr>
<td></td>
<td>Options if any</td>
</tr>
<tr>
<td></td>
<td>data if any</td>
</tr>
</tbody>
</table>

HLEN: header length in 32 bit multiples. Actual size depends on number of options.

Code bites:
- URG: Urgent pointer field is valid
- ACK: Ack field is valid (always except first syn packet)
- PSH: Segment requests a push
- RST: Reset the connection
- SYN: Synchronize sequence numbers
- FIN: Sender has reached end of its byte stream

Window size: Receiver’s advertised window (flow control)
TCP

Sequence and ack numbers:

- 32-bit unsigned long, wraps around to 0 after reaching $2^{32}-1$.
- Sequence number: identifies the byte in the stream of data that the first byte of data in the segment represents.
  - First segment contains the ‘initial sequence number’.
  - Subsequent Seq number’s relative to the ISN.
- Ack number: indicates the next sequence number that the receiver expects to receive.

- The ISN cycles, repeating roughly every 4-9 hours, why?
TCP

• Out of Band data: TCP allows a sender to specify data as urgent.
  • The receiver sends urgent data immediately to the receiver (regardless of its position in the stream).
  • When URG is set, the URG pointer field specifies the position in the segment where urgent data ends (i.e., it points to the last byte).
• But not a true out of band mechanism…. 
• Example: If a user aborts (CNT-C’s) an FTP data transfer, urgent data (an ABORT message) is sent by the client instructing the server to cancel.
TCP

• Push bit: A notification from the sender to the receiver for the receiver to pass all the data that it has to the receiving process.
  • BSD systems ignore it.
  • Most API’s don’t allow a program to set the push bit.

• TCP Options
  • Maximum Segment Sized: specifies the largest “chunk” of data that will be sent to the other end.
    • Option can only appear in a SYN segment
  • Timestamp
  • Window scale
TCP

• TCP connection setup: 3 way handshake
  • guarantees that both sides are ready to transfer data (handles simultaneous opens)
  • Allows both sides to agree on initial sequence numbers (ISNs).
TCP

• TCP connection termination: modified 3 way handshake
• The TCP close requires 4 flows rather than 3
• A FIN leads to an EOF to a receiving application
• Handles simultaneous close
• Supports a half close: one side terminates its output but still receives data from the other side.
  • Sockets supports this- but most applications don’t use it.
Figure 13.15 The TCP finite state machine. Each endpoint begins in the closed state. Labels on transitions show the input that caused the transition followed by the output if any.
TCP

• TCP Acks are cumulative
• Advantage:
  • Acks easy to generate and not ambiguous.
  • Lost Acks don’t necessarily force a retransmission.
• Disadvantage:
  • Receiver does not get all the information about successful retransmissions.