Transmission Control Protocol (TCP)

Srinidhi Varadarajan

TCP: Transmission Control Protocol

- **TCP must perform typical transport layer functions:**
  - **Segmentation -- breaks message into packets**
  - **Error recovery -- since IP is an unreliable service**
  - **End-to-end flow control -- to avoid buffer overflow**
  - **Multiplexing and demultiplexing sessions**

TCP: Transmission Control Protocol

- **Service provided by TCP is**
  - **Reliable**
  - **Connection-oriented -- virtual circuit**
  - **Stream-oriented -- users exchange streams of data**
  - **Full duplex -- concurrent transfers can take place in both directions**
  - **Buffered -- TCP accepts data and transmits when appropriate (can be overridden with “push”)**

TCP Addressing and Multiplexing

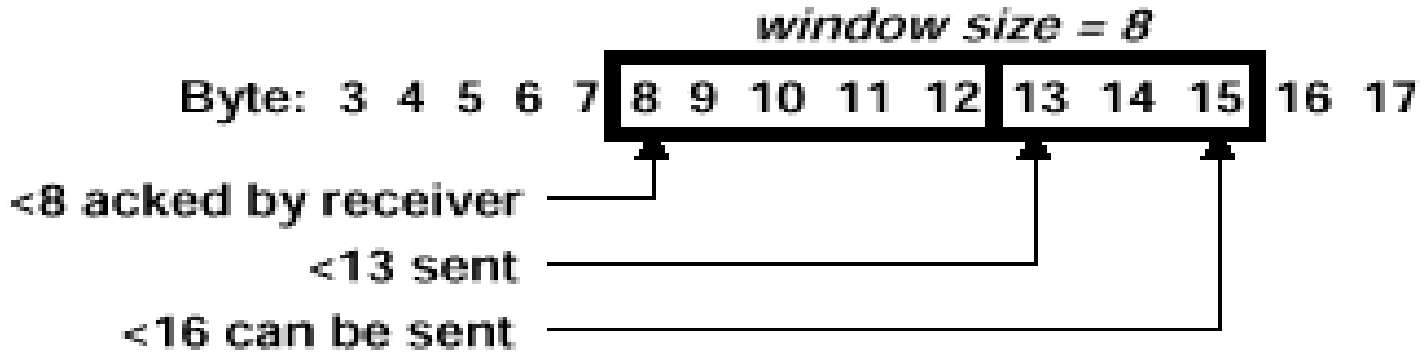
- **TCP identifies connections as socket pairs**
  - **Socket address is Internet address plus port**
  - **Host Internet address provided to IP**
  - **Port uniquely identifies user or process ID on host**
- **Example:**
  - **A connection to port 21 on 128.173.40.24 connects to ftpd (file transfer protocol daemon) on vtopus.cs.vt.edu**
  - **Port 21 is a “well known” port number and can be determined by looking at /etc/services on a UNIX machine**

TCP Sliding Window Mechanism

- **TCP is built on top of IP, an unreliable datagram service**
  - **Lost datagrams**
  - **Out-of-order datagrams**
- **TCP uses a sliding window mechanism for error recovery**
  - **Transmitted bytes are numbered**
  - **Receiver will accept bytes within the current "window"**
  - **Contiguous blocks are acknowledged by the receiver**

TCP Sliding Window Mechanism

- **Sender maintains three pointers for each connection**
  - **Pointer to bytes sent and acknowledged**
  - **Pointer to bytes sent, but not yet acknowledged**
  - **Pointer to bytes that cannot yet be sent**



TCP Sliding Window Mechanism

- **Receiver acknowledges bytes received**
  - **Specifies sequence number of next byte expected**
  - **This acknowledges all previous bytes as received error-free**
  - **Acknowledgments can be “piggy-backed” on reverse direction data packets or sent as separate packets**

TCP Sliding Window Mechanism

- **Sender sets a timer for a segment sent**
  - **On time-out, sender will retransmit the segment**
  - **Implementations send just the first unacknowledged segment -- will wait for return acknowledgment before sending more**
  - **Implementations also typically use just one timer per connection, i.e. at any given point in time, only one segment is being timed**
- **Time-out value is important**
  - **Bad values can add extra delays or result in wasted retransmissions**
  - **Time-out value is difficult to set since delays can vary greatly in an internetworking environment**

TCP Flow Control

- **Flow control is needed to**
  - **Prevent sender from “swamping” receiver with data, e.g. a fast server sending to a slow client**
  - **Provide congestion control inside the network, e.g. at gateways or routers**
  - **In either case, a node can be forced to discard packets due to lack of buffer space**
- **TCP provides end-to-end flow control**
  - **Can solve overload problems at the end node**
- **Flow control is provided by varying the size of the sliding window**

TCP Flow Control

- **Receiver “advertises” its window size in acknowledgments**
  - **Window size specifies how many more bytes the receiver is willing to accept**
  - **Receiver is not allowed to shrink the window beyond previously accepted bytes**
  - **Window size of 0 causes sender to stop transmission, later advertisement of a non-zero window resumes transmission**
- **Sender will adjust its “allowed to send” pointer only as far as the advertised window**

Packet Capture (tcpdump)

2:60:8c:9e:ca:b 8:0:2b:b:6c:1f 0800 62:
128.173.5.244.1524 > 128.173.5.221.21:
P 171:179(8) ack 1156928647 win 2048 ← **Receiver Window**

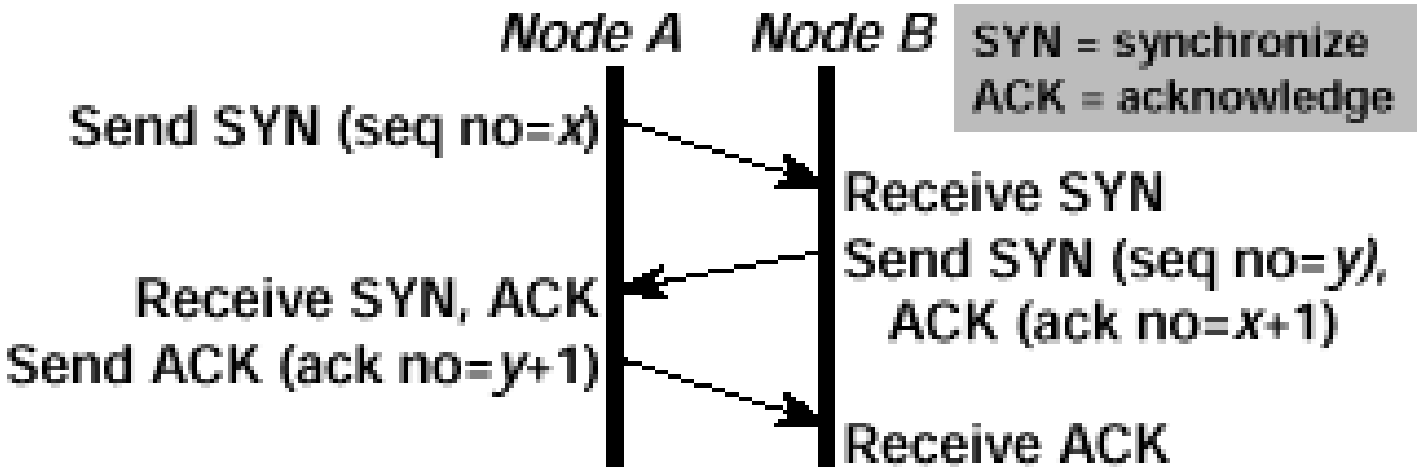
8:0:2b:b:6c:1f 2:60:8c:9e:ca:b 0800 74:
128.173.5.221.21 > 128.173.5.244.1524:
P 1156928647:1156928667(20) ack 179 win 16384

2:60:8c:9e:ca:b 8:0:2b:b:6c:1f 0800 60:
128.173.5.244.1524 > 128.173.5.221.21:
P ack 1156928667 win 2048

2:60:8c:9e:ca:b 8:0:2b:b:6c:1f 0800 80:
128.173.5.244.1524 > 128.173.5.221.21:
P 179:205(26) ack 1156928667 win 2048

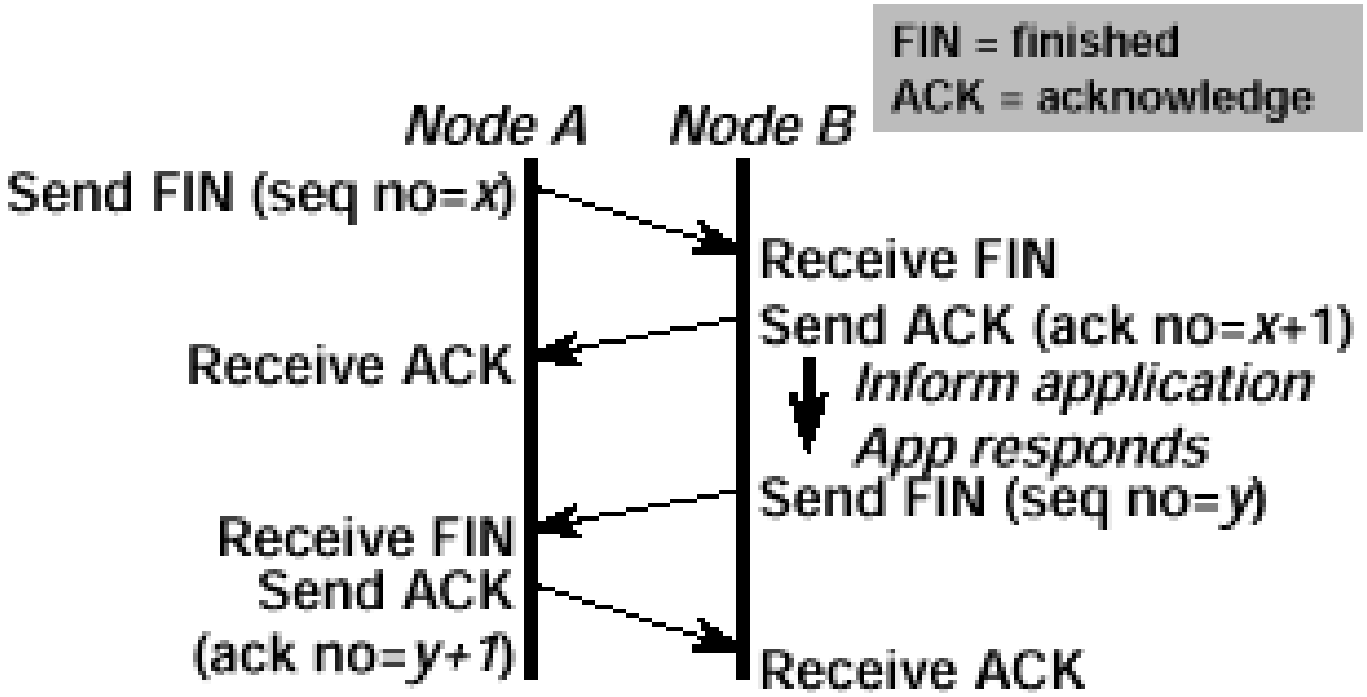
TCP Connection Establishment

- **TCP uses a “three-way handshake” (balanced protocol) to establish a connection**
- **Ensures that both nodes are ready and synchronizes sequence numbers**

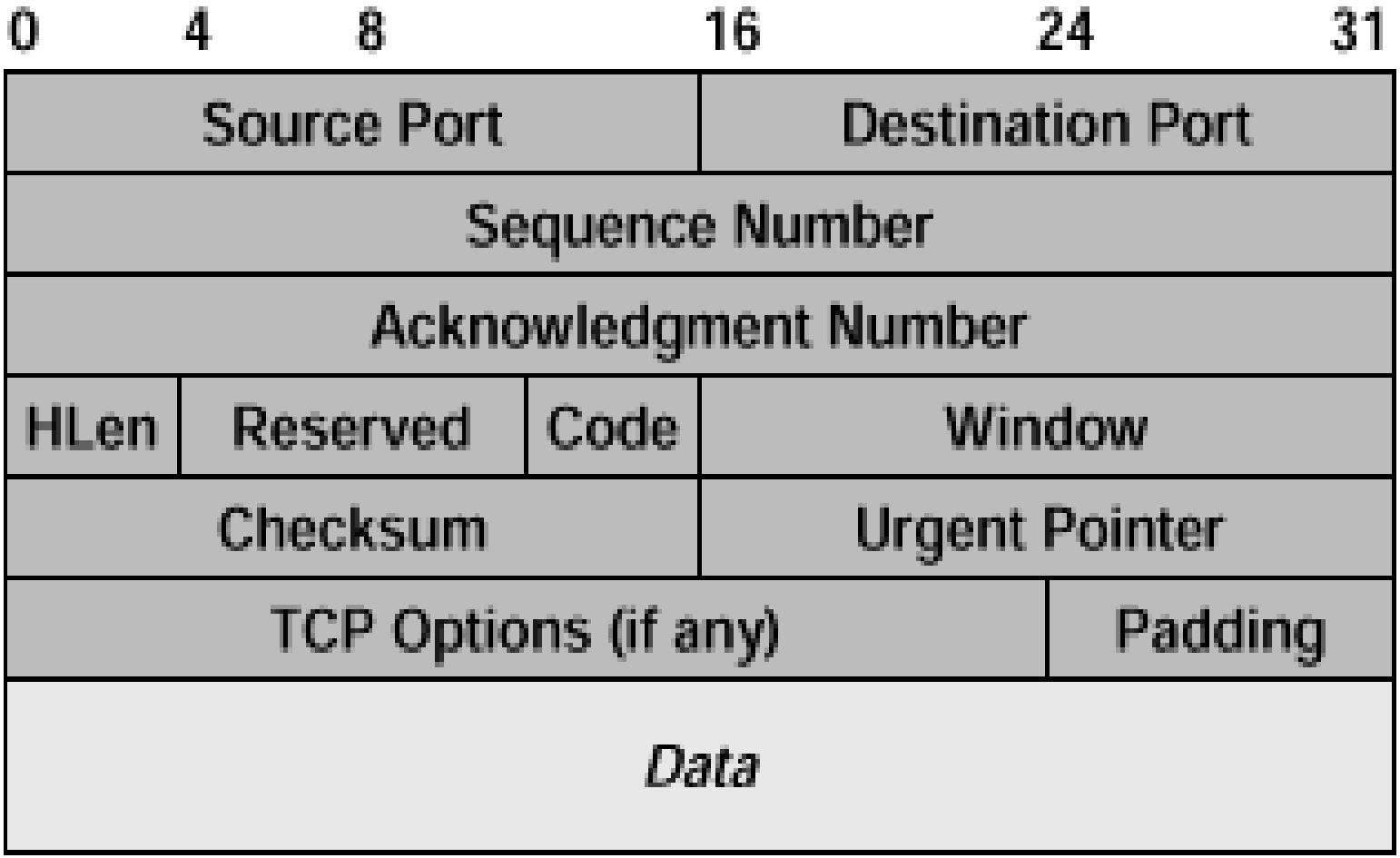


Closing a TCP Connection

- **A modified three-way handshake is used to gracefully close a connection**



TCP Packet Format



TCP Header Fields

- **Source Port and Destination Port: identify applications at ends of the connection**
- **Code Bits:**
  - **URG urgent (skip over data to urgent data)**
  - **ACK acknowledgment**
  - **PSH push request (send data to application)**
  - **RST reset the connection**
  - **SYN synchronize sequence numbers**
  - **FIN sender at end of byte stream**

TCP Header Fields

- **Sequence Number: position of the data in the sender's byte stream in bytes**
- **Acknowledgment Number: position of the byte that the source expects to receive next (valid if ACK bit set)**
- **Header Length: header size in 32-bit units**
- **Window: advertised window size in bytes**
- **Urgent: number of bytes to skip over in window to reach urgent (or "out-of-band") data -- valid only if URG bit is set**
- **Checksum: 16-bit CRC over header and data**