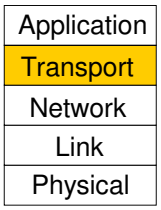
CSE 461 – The Transport Layer

The Transport Layer

- Focus
  - How do we (reliably) connect processes?
  - This is the transport layer
- Topics
  - Naming end points
  - UDP: unreliable transport
  - TCP: reliable transport
    - Connection setup / teardown



The Transport Layer

- Builds on the services of the Network layer
  - "TCP/IP"
- Communication between processes running on hosts
  - Naming/Addressing
- Guarantees about message delivery make (more) sense
  - This is the first layer that is talking "end-to-end"

The Layers Below: Network

- Network layer provides:
  - A name space for hosts (interfaces)
    - IP addresses
  - Framing (message boundaries)
    - Packets
  - "0 or more" delivery semantics
    - Packets may be lost
    - Packets *usually* aren't duplicated by the network
  - Byte transmission
    - No data types (except in headers)

The Layers Below: Network

- Network layer imposes:
  - Not all hosts are reachable
    - NATs
  - Packets may arrive out of order
    - Routing table updates
  - Maximum packet size
    - 65,535 bytes, including header ($2^{16} - 1$)
      - IPv6: optionally, up to 4GB
  - Variable transmission speeds
    - 100's of bits/sec to 10's of gigabits/sec
  - Variable delays
    - 10's of microseconds to seconds

We want the transport "to work" despite all this.

5

The Layers Below: Link/Physical

- Link layer provides:
  - A name space
    - MAC addresses
  - May, or may not, provide data integrity
    - Checks sums (in the general sense of error detection/correction)
  - Broadcast/multicast
    - Maybe
- Link layer imposes:
  - Variable delivery probability
    - Wired vs. wireless
  - Maximum frame size
    - Ethernet: 1500 bytes
      - Jumbo: 9K

6

Effect on Transport Protocols

- A transport layer packet may traverse many different kinds of networks
- Functionality:
  - Can't rely on very much from the lower layers
    - "Best effort" delivery, hop by hop
- Performance:
  - What can the transport layer do about delay?
  - What can the transport layer do about throughput?
  - Notion of a *bottleneck* link
    - The hop most limiting performance

7

Internet Transport Protocols

- UDP
  - Datagram abstraction between processes
  - Reliability: Never delivers data that wasn't sent
    - Error detection (data integrity)
- TCP
  - Bytestream abstraction between processes
  - Reliability: In order, at-most-once delivery
    - ARQ with a sliding window
      - Connections
  - Flow control
    - Throttle sender based on receiver's capabilities
  - Congestion control (later!)
    - Throttle sender based on network's capabilities

8

Comparison of TCP/UDP/IP properties

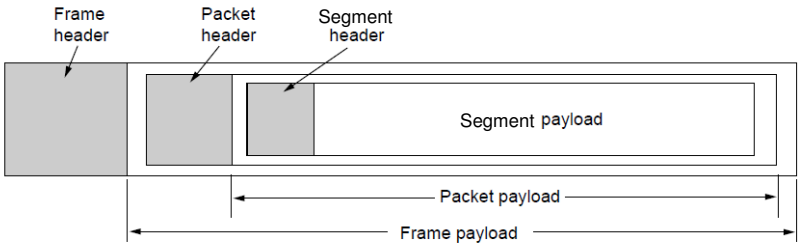
TCP
- Connection-oriented
- Reliable byte-stream
  - In-order delivery
  - Single delivery
  - Arbitrarily length
- Synchronization
- Flow control
- Congestion control

UDP
- Datagram oriented
- Lost packets
- Reordered packets
- Duplicate packets
- Limited size packets

IP
- Datagram oriented
- Lost packets
- Reordered packets
- Duplicate packets
- Limited size packets

Relation to layers

Transport layer sends segments in packets (in frames)



Naming: IP address + port number

- You know these from the projects…
- How does client determine the name of the receiver?
  - IP address: use the DNS name service
    - Project 4 is very like DNS; DNS is very like Project 4
    - We'll look at the Internet's DNS a bit later
  - Port: ?
    - Project 4 provides a translation from a "well known name" to a port
    - DNS does not (mostly)

Determining Internet Port Numbers

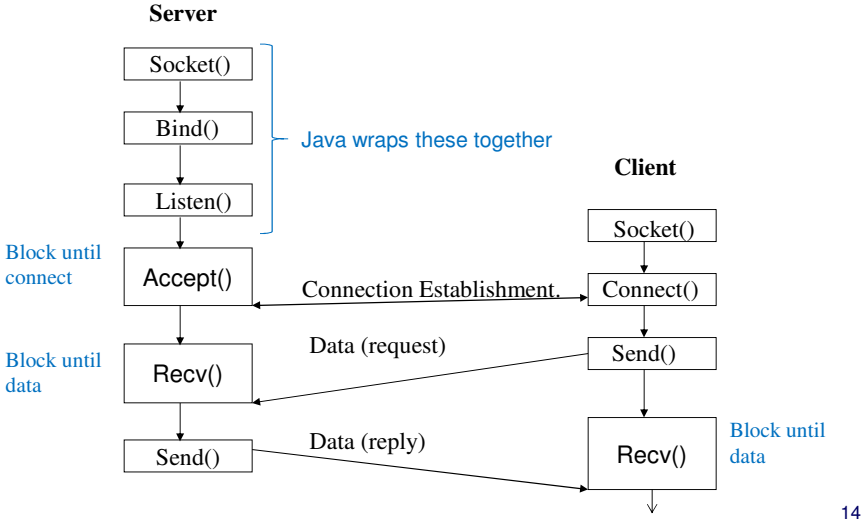
- Servers typically bind to well-known port numbers
  - Ports below 1024 reserved for well-known services
    - look in /etc/services
- Clients use OS-assigned temporary (ephemeral) ports
  - Above 1024, recycled by OS when client finished
    - (Some clients try to bind to a particular ephemeral port, to act well-known within some restricted community.)

Some well-known (TCP) ports

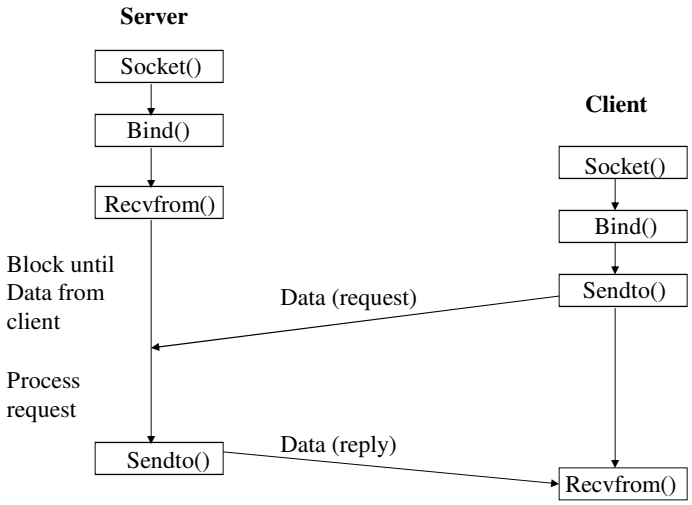
Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

Is there a mechanism to verify you've connected to what you intended? No.
Can a well-known server use a non-standard port? Yes.
What's the security provided by using a non-standard port? None.
What's the benefit of using a non-standard port? Minimal.
Why are there well-known ports, again?

TCP Socket Interface (connection-oriented)



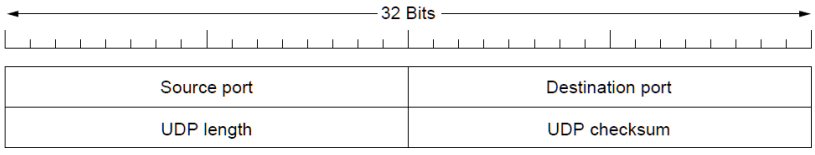
UDP Socket Interface (connectionless)



15

User Datagram Protocol (UDP)

- Provides message delivery between processes
  - Source port filled in by OS as message is sent
  - Destination port identifies UDP delivery queue at endpoint

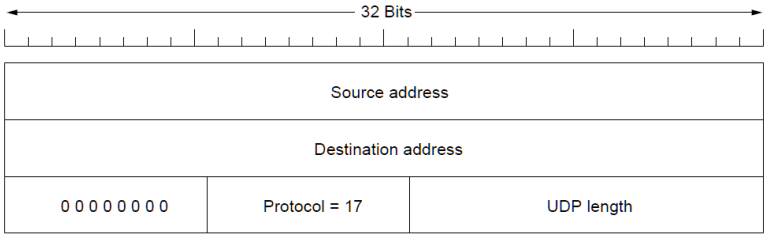


16

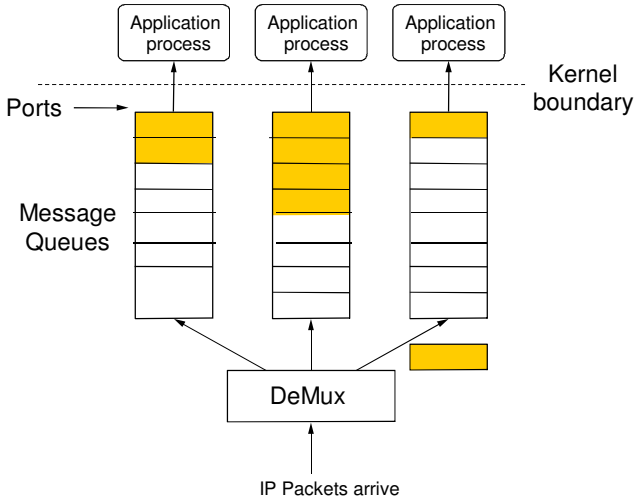
UDP checksum

Checksum covers UDP segment and IP pseudoheader

- Fields that change in the network are zeroed out
- Provides an end-to-end delivery check



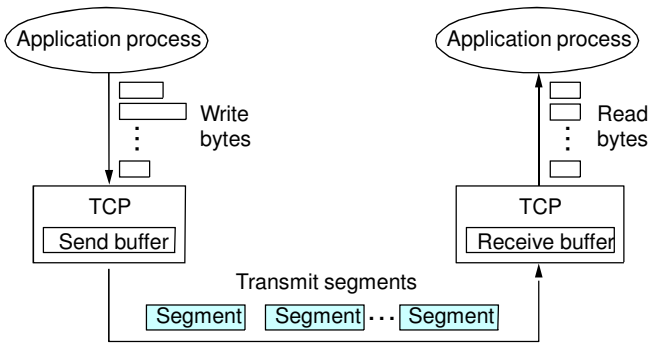
UDP Delivery



Transmission Control Protocol (TCP)

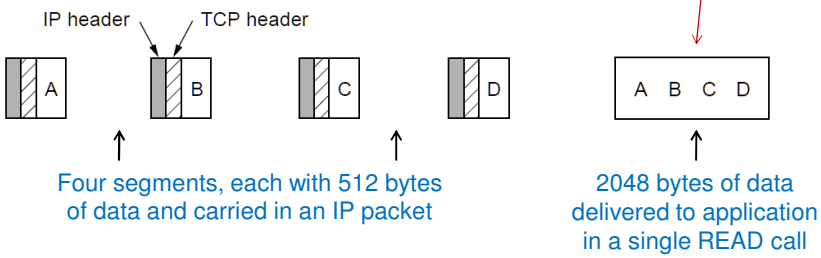
- Reliable bi-directional bytestream between processes
  - Message boundaries are not preserved
- Connections
  - Conversation between endpoints with beginning and end
- Flow control (later)
  - Prevents sender from over-running receiver buffers
- Congestion control (later)
  - Prevents sender from over-running network buffers

TCP Delivery



The TCP Service Model

Applications using TCP see only the byte stream and not the segments sent as separate IP packets

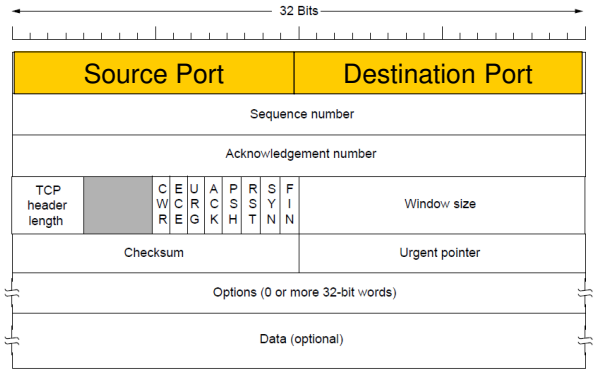


Four segments, each with 512 bytes of data and carried in an IP packet

2048 bytes of data delivered to application in a single READ call

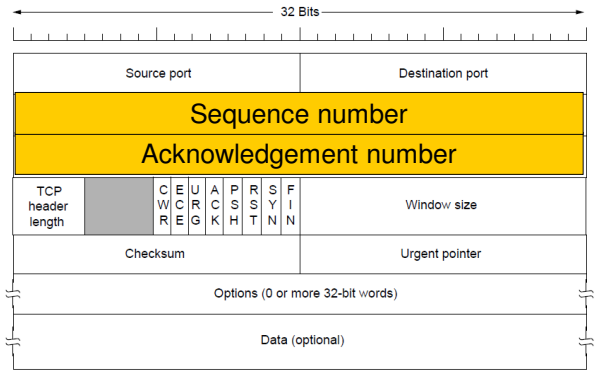
TCP Header Format

- Ports plus IP addresses identify a connection



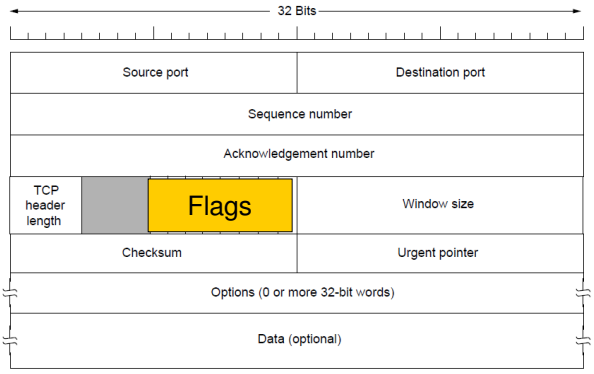
TCP Header Format

- Sequence, Ack numbers used for the sliding window
  - Congestion control works by controlling the window size



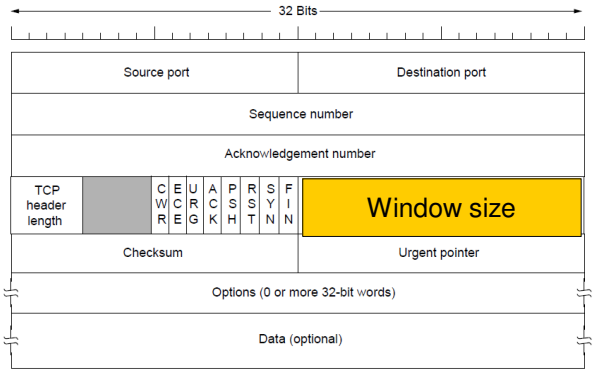
TCP Header Format

- Flags bits may be SYN / FIN / RST / ACK, URG, and ECE / CWR



TCP Header Format

- Advertised window is used for flow control



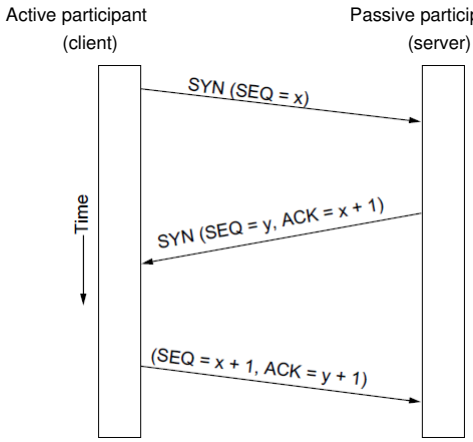
25

Connection Establishment

- Both sender and receiver must be ready before we start to transfer the data
  - Sender and receiver need to agree on a set of parameters
    - e.g., the Maximum Segment Size (MSS)
- This is signaling
  - It sets up state at the endpoints
  - Compare to “dialing” in the telephone network
- In TCP a Three-Way Handshake is used

26

Three-Way Handshake



- Opens both directions for transfer
- This can be viewed as two two-way handshakes
  - improved

27

Some Comments

- We could abbreviate this setup, but it was chosen to be robust, especially against delayed duplicates
  - Three-way handshake from Tomlinson 1975
- Choice of changing initial sequence numbers (ISNs) minimizes the chance of hosts that crash getting confused by a previous incarnation of a connection
- With random ISN it "proves" two hosts can communicate
  - Weak form of authentication
  - "TCP hijacking"

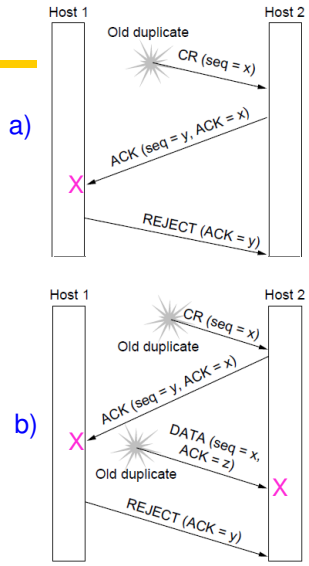
28

Connection Establishment

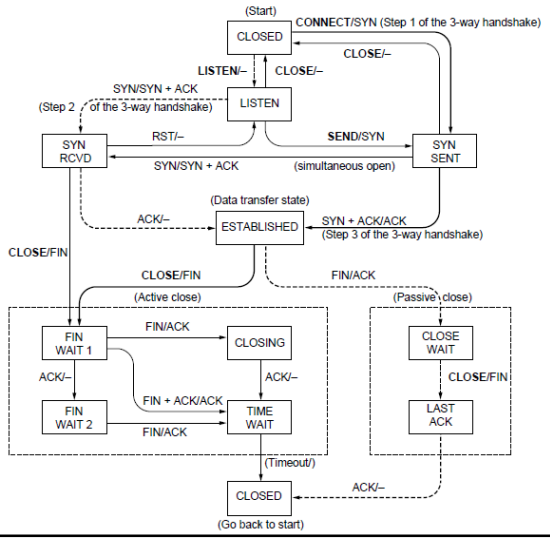
Three-way handshake protects against odd cases:

a) Duplicate CR. Spurious ACK does not connect

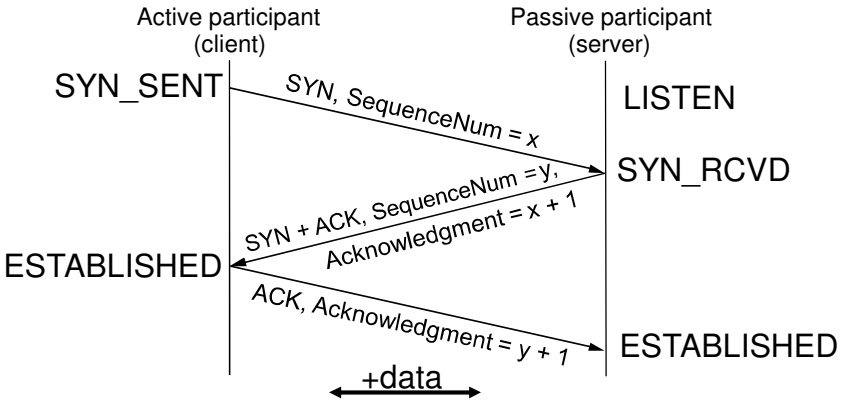
b) Duplicate CR and DATA. Same plus DATA will be rejected (wrong ACK).



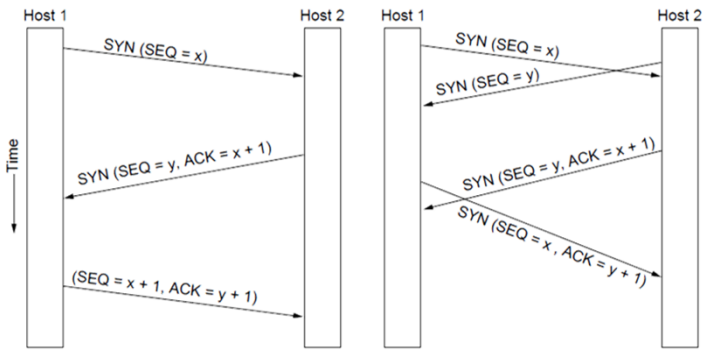
TCP State Transitions



Again, with States



Simultaneous Connect



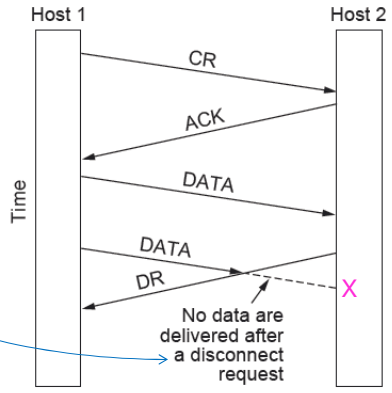
Connection Teardown

- Orderly release by sender and receiver when done
  - Delivers all pending data and “hangs up”
- Cleans up state in sender and receiver
- TCP provides a “symmetric” close
  - Each side shuts down independently
    - “I won’t be sending any more data”

33

Connection Release

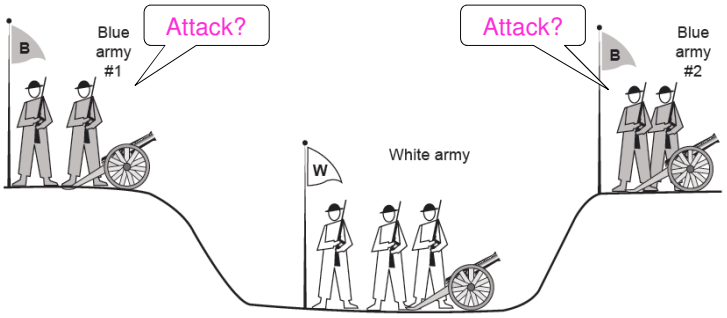
- Key problem is to ensure reliability while releasing
- Asymmetric release (when one side breaks connection) is abrupt and may lose data



Connection Release

Symmetric release (both sides agree to release) can't be handled solely by the transport layer

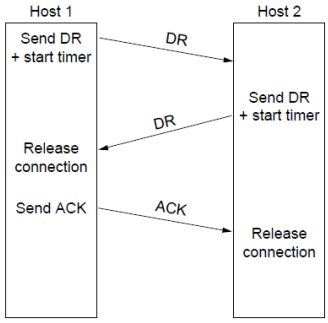
- Two-army problem shows pitfall of agreement



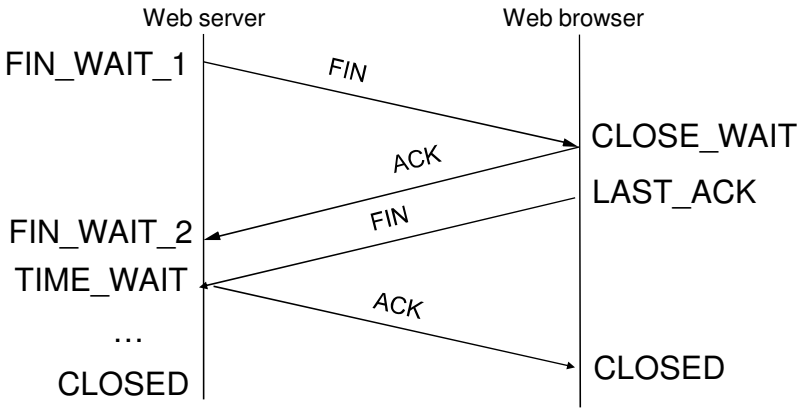
Connection Release

Normal release sequence, initiated by transport user on Host 1

- DR=Disconnect Request
- Both DRs are ACKed by the other side



TCP Connection Teardown



37

The TIME_WAIT State

- We wait 2MSL (two times the maximum segment lifetime of 60 seconds) before completing the close
- Why?
  - ACK might have been lost and so FIN will be resent
  - Could interfere with a subsequent connection

38