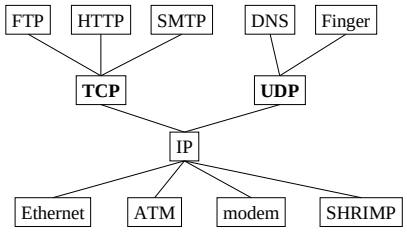
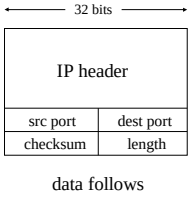
Transport Layer



UDP (User Datagram Protocol)

- unreliable, unordered datagrams
- the main difference from IP:
  - IP sends datagram to a *machine*
  - UDP sends datagram to (*machine, port*) pair
    - » port is 16-bit identifier



UDP and AcmeNet

- an AcmeNet NetAddress is a (machine, port) pair
- AcmeNet NetworkInterfaces communicate via UDP
  - very simple layering
  - look at NetworkInterface source code if you're interested in how this works

TCP

- stands for Transmission Control Protocol
- reliable, bidirectional byte stream
  - hides datagram boundaries
  - handles connection setup and shutdown cleanly
  - acknowledgement, timeout, and retransmission
  - flow control: slows down sender so receiver isn't overwhelmed
  - congestion control: slows down sender so network isn't overwhelmed

Challenges for TCP

- robust connection setup and shutdown
- high variance in round-trip time
- packets can "hide" in the net for a long time
- varying resources available at endpoints
  - must learn each other's needs, and adapt
- robust congestion control

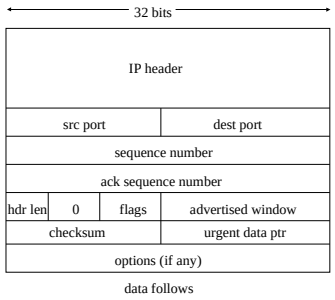
Byte Streams and Segments

- data not divided into chunks; functions like a continuous byte-stream
  - sender buffers data
    - » use largest buffer size that avoids IP fragmentation
  - data sent when buffer fills, or on periodic clock tick, or when explicitly flushed
  - receiver buffers data too
- like buffering in Assignment 2

Sequence Numbers

- each packet carries a sequence number
- seq nums identify *bytes*, not datagrams
- initial sequence numbers chosen "randomly"
- each packet carries an acknowledgement (with sequence number acked)
- no distinction between data/ack packets

TCP Header Format



TCP Header

- src port, dest port
  - same meaning as UDP
- sequence number
  - last byte of data sent, before this packet
- ack sequence number
  - first byte of data not yet received
- hdr len
  - length of TCP header, in 32-bit words

TCP Header

- flags
  - URG: if 1, packet contains urgent data
  - ACK: if 1, "ack seq num" field is valid
  - PSH: if 1, tells receiver not to buffer this packet, or anything ahead of it
  - RST: if 1, denotes serious error
  - SYN: if 1, this is a connection-setup packet
  - FIN: if 1, this packet closes the connection

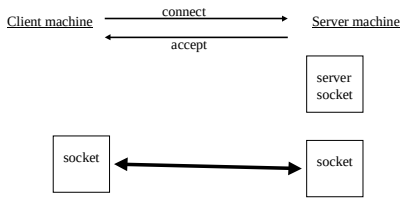
TCP Header

- advertised window
  - number of bytes sender can accept at present
    - » counts forward from ack seq num value
- checksum
  - Internet checksum, covers TCP info + data
- urgent pointer
  - location of end of "urgent data" in packet

TCP Header

- options: examples
  - negotiate packet size
  - window size larger than 64k
  - modified ack/retransmit information

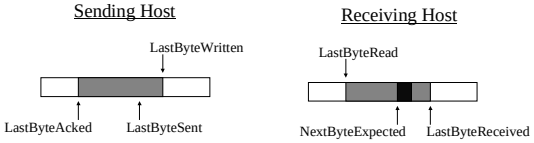
How TCP Uses Ports



Connection Setup

- three-way handshake
  - client says “I want to connect”, gives seq num
    - » SYN flag set
  - server says “I accept”, gives seq num, acks client’s seq num
    - » SYN + ACK flags set
  - client says “OK”, acks server’s seq num
    - » ACK flag set
- careful handling of errors, timeouts, etc.
- complex state transition diagram

Sliding Window Protocol



Goals:
reduce memory requirements;
don’t overrun receiver’s buffers;
keep the pipe full.

Sliding Window Constraints

- receiver decides how much memory to dedicate to this connection (call it M)
- receiver advertises window of M-(LastByteReceived-NextByteRead)
- sender stops sending when LastByteSent-LastByteAcked = Window
- acks from receiver update the window size
- special hack to get out of Window=0 case

Keeping the Pipe Full

- recall: optimum performance requires window size equal to latency-bandwidth product
- receiver tries to allocate enough memory to keep window size large enough
- with fast network, can overflow 16-bit “advertised window” field
  - use TCP header option to get bigger window

Timeout and Retransmission

- use timeout and retransmission
- optimal timeout is just more than round-trip transmission time
  - don’t time out if no packets dropped
  - recover quickly if packets are dropped
- for TCP
  - packet loss rate is nontrivial
  - round-trip time varies widely

Timeout: Original Algorithm

- keep running average of round-trip time
  - interval between sending and arrival of ack
- time out after twice the estimated RTT
- problem: matching problem after retransmission
- problem: bad if variance is high

Timeout: Better Algorithm

- three improvements:
  - don't include retransmitted packets in estimate
    - » avoids matching problem
  - estimate average and standard deviation; timeout after avg + 4 * deviation
  - on retransmission, double timeout (temporarily) for next retransmission

Using TCP in Java

- classes in java.net package
- on the client side:
  - new Socket(serverHostName, serverPort) connects to a server
  - Socket::getInputStream and Socket::getOutputStream get byte streams to use for communicating with server

Using TCP in Java

- on the server side:
  - new ServerSocket(portNum) advertises a connection point
  - ServerSocket::accept accepts a connection from a client; returns a Socket
    - » ServerSocket can accept many connections
  - after getting the Socket, treat it same as the client does

Using TCP in C

- uglier interface
  - form of IP addresses
  - hostname lookup
  - arguments as structs
  - one kind of socket used for server-socket and ordinary sockets
- usually write code by cut-and-paste
- for a good source, see course's Miscellaneous Links page

Well-Known Ports

- many standard services live at the same port on almost all machines
  - http on port 80
  - telnet on port 23
- allows outside clients to find them
- "portmapper" allows a level of indirection