UDP,
TCP (Part I)

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Overview

- UDP: connectionless, end-to-end service
- UDP Servers
- TCP features, Header format
- Connection Establishment
- Connection Termination
- TCP Server Design
- Ref: Chap 11, 17,18; RFC 793, 1323

User Datagram Protocol (UDP)

- Minimal Transport Service:
  - *Port* addressing: for application multiplexing
  - Error *detection* (Checksum): formerly optional
  - Connectionless end-to-end datagram service
- No flow control. No error recovery (no acks)
- Used by SNMP, DNS, TFTP etc

Source Port	Dest Port	Length	Check-sum	
16	16	16	16	← Size in bits

UDP feature details

- Port number: Used for (de)multiplexing.
  - Client ports are ephemeral (short-lived).
  - Server ports are “well known”.
- UDP checksum:
  - Similar to IP header checksum,
  - Pseudo-header (to help double-check source/destination address validity). Fig 11.3
  - UDP checksum optional, but RFC 1122/23 (host reqts) requires it to be enabled
- Application message is simply encapsulated and sent to IP => can result in fragmentation.

Some UDP Effects

- When UDP datagram fragments at the host, each fragment may generate an ARP request (results in an ARP reply: ***ARP flooding)***
  - RFC 1122/23 limits max ARP rate to 1 request/ sec, and requires the ARP Q to be at least of size one
- Datagram *truncation* possible at destination if dest app not prepared to handle that datagram size ! (note: TCP does not have this problem because it has no message boundaries)
- UDP sources *ignore* source quench messages => can’t respond to packet losses.

UDP Servers

- Most UDP servers are *“iterative”* => a single server process receives and handles incoming requests on a *“well-known”* port.
- Can filter client requests based on incoming IP address, client IP address, incoming port address, or wild card filters
- Port numbers may be reused, but packet is delivered to at most one end-point.
- Queues to hold requests if server busy

TCP: Key features

- *Connection-oriented*
- *Point-to-point*: 2 end-points (no broadcast or multicast)
- Reliable transfer: Data is delivered *in-order*
- *Full-duplex* communication

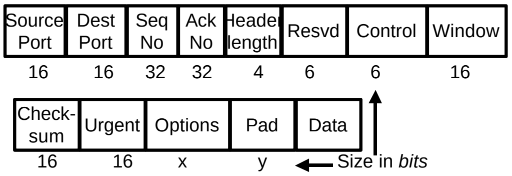
TCP: Key features (Continued)

- *Byte-stream* I/f: sequence of octets
- Reliable *startup*: Data on old connection does not confuse new connections
- Graceful *shutdown*: Data sent before closing a connection is not lost. Reset or immediate shutdown also possible.

Reliability features

- Reliable connection startup: Data on old connection does not confuse new connections
- Graceful connection shutdown: Data sent before closing a connection is not lost.
- Data segmented for transmission and acknowledged by destination. Timeout + Retransmission provided if data unacknowledged
- Checksum provided to catch errors.
- Resequencing of out-of-order data; discarding of duplicate data.
- Window flow control => sender cannot overrun receiver buffers

TCP *Header Format*



Source Port	Dest Port	Seq No	Ack No	Header length	Resvd	Control	Window
16	16	32	32	4	6	6	16

Check-sum	Urgent	Options	Pad	Data
16	16	x	y	

Also see fig: 17.2 in text
Does this header reflect the feature list we saw earlier ?

TCP Header

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

  20 = FTP, 23 = Telnet, 53 = DNS, 80 = HTTP, ...
- 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

TCP Header

- Header length (4 bits): Number of 32-bit words in the header. 4 bits => max header size is 60 bytes
- Reserved (6 bits)
- Control (6 bits)

URG	ACK	PSH	RST	SYN	FIN

- Window (16 bits): Will accept [Ack] to [Ack]+[window]

TCP Header (Continued)

- Checksum (16 bits): covers the segment + pseudo header. Protection from mis-delivery.
- Urgent pointer (16 bits): Points to the byte following urgent data. Lets receiver know how much data it should deliver right away.
- Options (variable):
  Max segment size (does not include TCP header, default 536 bytes), Window scale factor, Selective Ack permitted, Timestamp, No-Op, End-of-options

TCP Checksum

- Checksum is the *16-bit one's complement of the one's complement sum of a pseudo header*,
  - The TCP header, and data, (padded with zero octets at the end if necessary to make a multiple of two octets.)
  - Checksum field filled with zeros initially
- Pseudo header (similar to UDP)

Source Adr	Dest. Adr	Zeros	Protocol	TCP Length
32	32	8	8	16

TCP Header TCP data

Connection Establishment

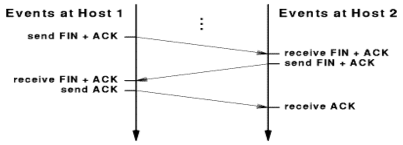
- Fig 18.3
- Client sends SYN, with an initial sequence number (ISN) and a Max Segment Size (MSS). Called ***"active open".***
- Server acks the SYN (for the *forward connection*), and also sets the SYN bit, with its own ISN (for the *reverse connection*). Called ***"passive open".***
- Client acks the reverse direction SYN.
- 3 segments transmitted.

Connection Termination

- Fig 18.3 again, also fig 18.5
- Client sends FIN. Server acks this and notifies its application. However it can keep its half-connection open. Each connection closed separately.
- Server app issues a "close" and server sends FIN to client. Client acks this.
- 4 segments transmitted.

Three-Way Handshake

- 3-way handshake: necessary and sufficient for unambiguous setup/teardown even under conditions of loss, duplication, and delay



More Connection Establishment

- **Socket**: BSD term to denote an IP address + a port number.
  - *A connection is fully specified by a* ***socket pair*** i.e. the source IP address, source port, destination IP address, destination port.
- Initial Sequence Number (ISN): counter maintained in OS.
  - BSD increments it by 64000 every 500ms or new connection setup => time to wrap around < 9.5 hours.

Connection Establishment (Contd)

- SYN pkt lost => retransmitted.
  - Exponential timeout backoff (6, 12, 24 s etc)
  - Connection timeout is 75 s.
- Timer granularity is 500 ms => first timeout between 5.5 and 6s. See Fig. 18.7

MSS

- Maximum Segment Size (MSS)
- Largest "chunk" sent between TCPs.
  - Default = 536 bytes. Not negotiated.
  - Announced in connection establishment.
  - Different MSS possible for forward/reverse paths.
  - Does not include TCP header
  - Many BSD systems restrict MSS to be multiples of 512 bytes: inefficient.
  - Path MTU restricts size of MSS further.

TCP State Transition Diagram

- Figure 18.12: client (dark line) , server (dashed line) transitions.
- **2MSL wait:** wait for final segment to be transmitted before releasing connection (typically 2 min)
  - Socket ***pair*** cannot be reused during 2MSL
  - Delayed segments dropped
- Establishment: SYN_SENT, SYN_RCVD, ESTABLISHED, LISTEN
- Close: FIN_WAIT_1, FIN_WAIT_2, CLOSING, TIME_WAIT, CLOSE_WAIT, LAST_ACK

Effect of 2MSL wait

- Can't kill server & restart immediately to use the same well known port (1-4 min!)
- Reason: TCP cannot reallocate the socket pair (i.e. the connection) till 2MSL.
- Kill client and restart => it will get a different port
- 2MSL wait protects against delayed segments from the previous "incarnation" of the connection.
- If server crashes and reboots within 2 MSL wait, it is still safe because RFC 793 prevents having connections for 1 MSL after reboot.

TCP Servers

- Most TCP servers are *concurrent* i.e. separate process to handle each client - for ease of connection management
- Server listens to well-known port.
  - Socket pair distinguishes connections
  - A separate "*endpoint*" in the ESTABLISHED state is associated with each connection
  - One endpoint is used to listen (LISTEN state) for new connections

TCP Servers (Continued)

  - Endpoints in the ESTABLISHED state cannot receive SYN packets
- Possible to wildcard or select specific interfaces (local IP addresses) to listen to.
- Multiple connection requests => backlog queue of connections established but new process not yet created by server to handle it.
- Queue full => send RESET to new connection requests

Summary

- ❑ UDP is connectionless and simple. No flow/error control.
- ❑ TCP provides reliable full-duplex connections.
- ❑ TCP state diagram, 3-way handshake, Options
- ❑ UDP and TCP servers