

ECE 435 – Network Engineering

Lecture 8

Vince Weaver

`http://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

21 September 2017

Announcements

- HW#3 due
- HW#4 will be posted
- Sorry about going over time last class. Let me know if going over.
- Missed things from last time:
 - Common e-mail servers: configuring them, specifically SendMail
 - Common name servers: BIND (named)
 - `gethostbyname()` is how you resolve a hostname,



used in our sockets code



HW#2 Review

- Mostly C issues
 - don't ignore compiler warnings!
 - The biggest issue if your browser isn't displaying things is the wrong Content-length:
If you send less data than you say you will, it will wait forever for it, or else give a "connection reset" if you close the connection.
 - Be sure you read everything the browser is sending (Either big enough buffer, or repeat in loop reading it



all). If you send a response before it is done sending it can confuse things.

- `sizeof()` operator. `char temp[1024];` what is `sizeof(temp)`? hint: it's not the same as `strlen()`
- have to read/write in loop if file sending is bigger than your file buffer
- just don't take addresses with ampersand randomly
- Can you use `strcat()` with binary file? why not?
Can binary files have zeros?
- error checking!
- Many crashed if I requested the README file. Have to



handle unexpected input from user. (in this case, no file extension)

- include proper header file. man can tell you
- Difference between ' ' and " "
- `sprintf()` 300 bytes into 100 byte array (`snprintf?`)
- Only reading a small amount of bytes (100) when your file is bigger.
- Header and time format. There is a particular format, some browsers will ignore it, others not.
- Also depends on browser. For example, if have wrong content size, lynx just runs with it. wget tries forever.



Other browsers give a connection reset error?

- Can use `wget -S` to see the headers you are sending
- If really HTTP/1.1 you should keep connection open, multiple requests on one connection.
- If you use firefox you'll see it also request `favico.ico`?
Why? What should you return (assuming the file doesn't exist?) 404.
- Time wait
- Something Cool



The Transport Layer

	OSI	TCP/IP
7	Application	Application
6	Presentation	
5	Session	
4	Transport	Transport
3	Network	Internet
2	Data Link	Host-to-network
1	Physical	Host-to-network



The Transport Layer

- Responsible for reliable point-to-point data transport independent of whatever lies beneath.
- Provide process-to-process connectivity, and per-segment error control and per-flow reliability, as well as rate control
- Can be more reliable than underlying network
- TCP (Transmission Protocol Layer)
 - connection oriented
 - stateful



- per-flow reliability and rate control
- UDP (User Datagram Protocol)
 - stateless
 - connectionless
- the “socket” is the API from old homework



The Transport Layer

- Terminology: application = process, data-transfer-unit is a segment, traffic is a flow
- addressing – each process needs a unique ID. For internet, this is the “port” number (16-bit)
- Rate control
 - Flow control – between source and destination
 - Congestion control – between source and network
 - None in link layer because only one hop?



Can be done by sender or network

- Real time requirements – things like video and audio need extra info such as timestamp, loss rate, etc. So hard to do with raw TCP/UDP



Unreliable, Connectionless – UDP

- User Datagram Protocol (RFC 768)
- No reliability, no rate control, stateless
 - If you want these things you have to add them at higher layer
- Error control optional
- Why none of those things? All add overhead.
 - Used when want packets to get through quickly.
 - Don't care about re-transmits, better for real-time (VOIP, streaming?)



- Easy to implement, for low-level stuff like bootp/dhcp
- Good for broadcasting
- Provides process-to-process communication and per-segment error control
- Can send UDP packets to a destination without having to set up a connection first



UDP Header

- 16-bits: source port (optional, says where it is coming from in case need to respond, 0 if unused)
- 16-bits: destination port
- 16-bits UDP length (length in bytes, includes the header, minimal size is 8)
- 16-bits checksum (optional, see below)
- data



Port Numbers

- 16-bit, so 64k of them
- Can map to any you want, but there are certain well-known ones. Look in `/etc/services`. For example. WWW is 80/tcp. DNS is 53/udp
- On most operating systems, ports below 1024 require root (why?)
- Source/destination addr + source/destination port + protocol ID (TCP or UDP) is a socket pair (or 5-tuple)



is 104 bits that uniquely identify a flow for IPv4. IPv6 has a specific field for this



UDP checksum

- If set to zero, ignored
- Receiver drops invalid checksums (does not request resend)
- 1s complement of sum all 16-bit words in header and payload
padded with 0s to be multiple of 16-bits
- Also added to the checksum is a 96-bit pseudo header that has source IP, dest IP, protocol, length. Enables receiver to catch problems with there to (delivered to



wrong machine) – why could this be a problem?

- What happens if checksum is 0? entered as 0xffff
What happens if it was 0xffff? Remember in ones complement 0xffff is negative zero.
- Checksum considered mandatory on IPv6 because IPv6 header not checksummed
- Why would you ever leave checksum out? Takes time to compute, might care about latency over errors [video?]



UDP checksum example

- 0x0000: 8875 563d 2a80 0030 18ab 1c39 86dd 6002 .uV=*..0...9..' .
0x0010: 2618 0031 1140 2610 0048 0100 08da 0230 &..1.@&..H....0
0x0020: 18ff feab 1c39 2001 4860 4860 0000 00009..H'H'....
0x0030: 0000 0000 8844

UDP:

e239 0035 0031 9c0e 8657D.9.5.1...W

0x0040: 0120 0001 0000 0000 0001 0377 7777 0465www.e

0x0050: 7370 6e03 636f 6d00 0001 0001 0000 2910 spn.com.....).

0x0060: 0000 0000 0000 00

- 16-bit sum of “virtual header” (two IPv6 addresses, protocol (0x0011) and length of udp packet/header (0x0031)) is 0x29f8c
- 16-bit sum of UDP header leaving off checksum is 0xe29f



- 16-bit sum of UDP data is 0x2e1c0
- Add them get 0x6 63eb
- It's a 16-bit sum, so add $0x6 + 0x63eb = 0x63f1$
ones complement is 0x9c0e, which matches the UDP
checksum field



OS UDP

- When listening on UDP, sets up a queue
- Network stack decodes and gets UDP, finds port, looks to see if any processes listening on that port
- If so, adds to queue
- If not, sends an ICMP “port unreachable” error message
- All UDP messages to that port, no matter who sends them, end up in the same queue.



Writing UDP sockets code

- Use `SOCK_DGRAM` rather than `SOCK_STREAM`
- Can skip the listen/accept state, as no connection is there. Just receive the packets as they come in.



Common UDP Services

- Obsolete: echo/discard/users/daytime/quote/chargen
- Nameserver
- bootp/tftp
- ntp (network time protocol)
- snmp



UDP real-time

- Real-Time Protocol (RFC1889)
- On top of UDP, multiplexes
- data streams
- timestamps

