CS118 Discussion 1C, Week 3

Zengwen Yuan
Bunche Hall 3156, Friday 2:00—3:50 p.m.
Outline

• Application Layer Protocol: DNS, Proxy, CDN

• Transport Layer Protocol: UDP, principles of reliable transport protocol

• Homework and Project 1
Application Layer: protocols

- DNS:
  - What is the transport layer protocol?
  - How the scalability is achieved?
  - Who will use iterative/recursive query?
  - Why is DNS resolver needed?
Application Layer: protocols

- DNS: convert hostname to IP address (and more)
- A distributed and hierarchical database
  - Root DNS servers (a—m)
  - Top-level domain (TLD) servers
  - Authoritative DNS servers
  - local DNS server (caching resolver, stub resolver)
DNS protocol: exercise

- Assume the caching resolver’s cache is empty initially
- Host A queries www.ucla.edu, how many queries should the caching resolver issue?
- After A’s DNS query, host B queries www.mit.edu, how many queries should the caching resolver issue?
DNS protocol: exercise

• Assume the caching resolver’s cache is empty initially

• Host A queries www.ucla.edu, how many queries should the caching resolver issue?

• After A’s DNS query, host B queries www.mit.edu, how many queries should the caching resolver issue?
DNS protocol: exercise

• Assume the caching resolver’s cache is empty initially

• Host A queries www.ucla.edu, how many queries should the caching resolver issue?

• After A’s DNS query, host B queries www.mit.edu, how many queries should the caching resolver issue?
A fun experiment: DNS query

$ dig google.com
; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44777
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.            IN  A

;; ANSWER SECTION:
google.com. 76 IN A 172.217.2.14

;; AUTHORITY SECTION:
google.com. 85950 IN NS ns1.google.com.
google.com. 85950 IN NS ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com. 59591 IN A 216.239.32.10
ns2.google.com. 50756 IN A 216.239.34.10
ns3.google.com. 40354 IN A 216.239.36.10
ns4.google.com. 36005 IN A 216.239.38.10

;; Query time: 84 msec
;; SERVER: 158.69.209.100#53(158.69.209.100)
;; WHEN: Thu Jan 19 20:37:48 2017
;; MSG SIZE  rcvd: 180

$ dig any mit.edu
$ dig 206.5.217.172.in-addr.arpa
Application Layer: protocols

• P2P: no always-on server, peers are intermittently connected

• Calculate content distribution time

\[ D_{cs} = \max\left(\frac{NF}{\mu_s}, \frac{F}{d_{min}}\right) \]

\[ D_{p2p} = \max\left(\frac{F}{\mu_s}, \frac{F}{d_{min}}, \frac{NF}{\mu_s + \sum_i \mu_i}\right) \]
Web caching: Proxy v.s. CDN

- Proxy acts both as client and server
  - What if cache is stale?
    - HTTP conditional GET
- CDN: Content Distribution Network
  - Globally distributed network of web servers
  - Stores and replicates images, videos and other files
CDN example: Netflix

1. Bob manages Netflix account
2. Bob browses Netflix video
3. Manifest file returned for requested video
4. DASH streaming
More on delay

\[ d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}} \]

- **\(d_{\text{proc}}\): nodal processing**
  - check bit errors
  - determine output link

- **\(d_{\text{trans}}\): transmission delay:**
  - \(L\): packet length (bits)
  - \(R\): link bandwidth (bps)
  - \(d_{\text{trans}} = \frac{L}{R}\)

- **\(d_{\text{queue}}\): queueing delay** [DEMO]
  - time waiting at output link for transmission
  - depends on congestion level of router

- **\(d_{\text{prop}}\): propagation delay:**
  - \(d\): length of physical link
  - \(s\): propagation speed (~2x10^8 m/sec)
  - \(d_{\text{prop}} = \frac{d}{s}\)

HTTP Conditional GET

Request:
GET /sample.html HTTP/1.1
Host: example.com

Response:
HTTP/1.x 200 OK
Via: The-proxy-name
Content-Length: 32859
Expires: Tue, 27 Dec 2005 11:25:11 GMT
Date: Tue, 27 Dec 2005 05:25:11 GMT
Content-Type: text/html; charset=iso-8859-1
Server: Apache/1.3.33 (Unix) PHP/4.3.10
Cache-Control: max-age=21600
Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
Etag: “4135cda4”

Request:
GET /sample.html HTTP/1.1
Host: example.com
If-Modified-Since: Wed, 01 Sep 2004 13:24:52 GMT
If-None-Match: “4135cda4”

Response:
HTTP/1.x 304 Not Modified
Via: The-proxy-server
Expires: Tue, 27 Dec 2005 11:25:19 GMT
Date: Tue, 27 Dec 2005 05:25:19 GMT
Server: Apache/1.3.33 (Unix) PHP/4.3.10
Keep-Alive: timeout=2, max=99
Etag: “4135cda4”
Cache-Control: max-age=21600

Last-Modified & If-Modified-Since
Transport Layer
Transport Layer V.S. Network Layer

- Network layer: logical communication between **hosts**
  - **IP address** is used for identifying a host

- Transport layer: logical communication between **processes**
  - **IP address and port number** are used for identifying a process
Multiplexing and De-multiplexing

- Multiplexing at send host: gather data from multiple sockets
- De-multiplexing at receiving host: deliver received segments to the right socket
- **Five tuples** (src_ip, src_port, dst_ip, dst_port, protocol) are used for multiplexing/demultiplexing
  - How to identify a TCP/UDP socket? **Isf -i**
  - Can TCP and UDP share the same port numbers? **Yes!** e.g. DNS
UDP

- No connection establishment
- No connection state
- Small packet overhead (8 byte)

- How to calculate checksum?
  - **Pseudo header** + **UDP header** + data
  - Also applicable to TCP
  - Why pseudo header?
Principles of Reliable Data Transfer

• How to deal with bit errors?
  • Error detection (e.g. checksum)
  • Receiver feedback
  • Retransmission
• Why not error correction?
  • How to deal with duplicate packets due to retransmission? Sequence number
  • How can the sender detect that ACK or data is lost? Timer
Stop and Wait Protocol

- **Main Issue:** limited performance

- Consider two hosts that are directly connected by a 50 Kbit/sec satellite link that has a 250 milliseconds propagation delay. If these hosts send 1000 bits segments, what is the maximum throughput in stop-and-wait protocol if we ignore the transmission time of ACK?
Stop and Wait Protocol

- **Main Issue:** limited performance

- Consider two hosts that are directly connected by a 50 Kbit/sec satellite link that has a 250 milliseconds propagation delay. If these hosts send 1000 bits segments, what is the maximum throughput in stop-and-wait protocol if we ignore the transmission time of ACK?

  - Ans: \( \frac{1000}{(1000/50 + 250 + 250)} = 2 \text{ Kbit/sec} \)
Pipelined Protocols

• Go-back-N: receiver only sends cumulative ACKs
  • Drop out-of-order segments
  • reACK packet with highest in-order sequence number
  • Timer for oldest unACKed packet only, retransmit all unACKed packets

• Selective repeat: receiver ACKs individual packets
  • Buffer out of order segments
  • Timer for each individual unACKed packet, retransmit any unACKed packet
Serve Multiple TCP Connections Simultaneously

- Problem: `accept()` works under **blocking mode** by default
  - Unless a new connection request arrives, `accept()` will not return
- Perquisite: `listen()` allows multiple TCP connection
- Three approaches
  - `fork()`: each connection is served by a new process
    - Easy to write, but expensive and hard to share data between processes
  - POSIX pthread: each connection is served by a new thread
    - Can be hard to maintain
  - **Non-blocking mode**: use `select()`
What is `select()`?

- A monitor for multiple sockets (or fd, file descriptors)
  - Given a set of sockets, if any of them were ready to receive/send, `select()` would exit

- `int select (int numfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);`

  - `numfds`: the highest file descriptor plus one
  - `readfds`, `writefds`, `exceptfds`: set of sockets
  - `timeout`: timer for select to exit without any changes
  - return when some sockets are ready, or timeout
  - return value: the number of sockets that are ready
What is `fd_set`?

- A set of sockets (or file descriptors) that will be monitored by `select()`

- Macros of set operation
  - `FD_SET(int fd, fd_set *set);` // add fd to the set
  - `FD_CLR(int fd, fd_set *set);` // remove fd from the set
  - `FD_ISSET(int fd, fd_set *set);` // return if fd is in the set
  - `FD_ZERO(fd_set *set);` // clear all entries in the set
How to use `select()`?

- Assume the server has created a socket `sock`, which is bound with server’s IP address and port number

```c
fd_set active_fd_set;  // set for holding sockets
int new_sock;     // socket representing client
/* Initialize the set of active sockets */
FD_ZERO(&active_fd_set);
/* put sock to the set, s.t. we can monitor whether a new connection request arrives */
FD_SET(sock, &active_fd_set);

while (1) {
    /* Block until some sockets are active. Let N is #sockets+1 in active_fd_set */
    if (select(sock + 1, &active_fd_set, NULL, NULL, NULL) < 0) {
        exit(-1); // error
    }
    /* Now some sockets are active */
    if (FD_ISSET(sock, &active_fd_set)) { // new connection request
        new_sock = accept(sock, (struct sockaddr*) &client_addr, sizeof(client_addr));
        FD_SET(new_sock, &active_fd_set);
    }
    /* Decide whether client has sent data */
    if (FD_ISSET(new_sock, &active_fd_set)) {
        /* receive and process data here */
    }
} // end of while
```
Exercises
Q1

Suppose Host A wants to send a large file to Host B. The path from Host A to Host B has three links, of rates $R_1 = 500$ Kbps, $R_2 = 2$ Mbps, and $R_3 = 1$ Mbps.

(a) Assuming no other traffic in the network, what is the throughput for the file transfer?

(b) Suppose the file is 4 million bytes. Dividing the file size by the throughput, roughly how long will it take to transfer the file to Host B?

(c) Repeat (a) and (b), but now with $R_2$ reduce to 100 Kbps.
(a) Assuming no other traffic in the network, what is the throughput for the file transfer?

• **Ans: 500 Kbps**

(b) Suppose the file is 4 million bytes. Dividing the file size by the throughput, roughly how long will it take to transfer the file to Host B?

• **Ans: \( \frac{(4 \times 10^6) \times 8}{(500 \times 10^3)} = 64 \text{ seconds} \)**

(c) Repeat (a) and (b), but now with R\(_2\) reduce to 100 Kbps

• **Ans: 100 Kbps, 320 seconds**
Q2

(a) How long does it take a packet of length 1000 bytes to propagate over a link of distance 2500 km, propagation speed $2.5 \times 10^8$ m/s, and transmission rate 2 Mbps?

(b) More generally, how long does it take a packet of length $L$ to propagate over a link of distance $d$, propagation speed $s$, and transmission rate $R$ bps?

(c) Does this delay depend on packet length?

(d) Does this delay depend on transmission rate?
(a) How long does it take a packet of length 1000 bytes to propagate over a link of distance 2500 km, propagation speed 2.5x10^8 m/s, and transmission rate 2 Mbps?

\[ \text{Ans: } \frac{2500 \times 10^3}{2.5 \times 10^8} = 0.01 \text{s} = 10 \text{ ms} \]

(b) More generally, how long does it take a packet of length L to propagate over a link of distance d, propagation speed s, and transmission rate R bps?

\[ \text{Ans: } \frac{d}{s} \]

(c) Does this delay depend on packet length? \textbf{Ans: No}

(d) Does this delay depend on transmission rate? \textbf{Ans: No}
Q3

Consider an HTTP client that wants to retrieve a Web document at a given URL. The IP address of the HTTP server is initially unknown. What transport and application layer protocols besides HTTP are needed in this scenario?
Consider an HTTP client that wants to retrieve a Web document at a given URL. The IP address of the HTTP server is initially unknown. What transport and application layer protocols besides HTTP are needed in this scenario?

- Application layer protocols: DNS and HTTP
- Transport layer protocols: UDP for DNS; TCP for HTTP
Problem 3

Review the car-caravan analogy in lecture #1 slides (for Chapter 1). Assume a propagation speed of 100 km/h.

(a) Suppose the caravan (10 cars) travels 100 km, beginning in front of one tollbooth, passing through a second tollbooth, and finishing just after a third tollbooth. The distance between two tollbooths is 50 km. Each car takes 12 sec to serve. What is the end-to-end delay?

(b) Repeat (a), now assuming that there are 8 cars in the caravan instead of 10.

(a) Since the caravan travels 100 km and passes three tollbooths, the distance between adjacent tollbooths is 50 km. So, propagation delay between each tollbooth is

\[ d_{prop} = \frac{50 \text{ km}}{100 \text{ km/h}} = 30 \text{ min}; \]

Each car taking 12 sec to serve, so the processing delay at each tollbooth is:

\[ d_{proc} = 10 \times 12 \text{ sec} = 2 \text{ min}; \]

Therefore, the total delay equals:

\[ d_{total} = 2 \times d_{prop} + 3 \times d_{proc} = 66 \text{ min}. \]

(b) Change car number to eight, the propagation delay is still the same; the new processing delay changes to

\[ d'_{proc} = 8 \times 12 \text{ sec} = 96 \text{ sec}; \]

Therefore the new total delay is:

\[ d'_{total} = 2 \times d_{prop} + 3 \times d'_{proc} = 64 \text{ min 48 sec.}(64.8 \text{ min}) \]