Network Applications:
Overview, Email

Y. Richard Yang

http://zoo.cs.yale.edu/classes/cs433/

9/13/2018
Outline

- Admin and recap
- Application layer overview
  - Network applications
    - Email
Admin

- Office hours today and tomorrow posted on class home page

- Questions on Assignment One
Recap: Summary of the Taxonomy of Communication Networks

communication network
  └── switched network
      └── packet-switched network
              └── datagram network
      └── circuit-switched network
              └── virtual circuit network
  └── broadcast communication
       └──
Recap: Statistical Multiplexing

A simple model to compare bandwidth efficiency of
- reservation/dedication (aka circuit-switching) and
- no reservation (aka packet switching)

setup
- a single bottleneck link with rate R (L/R to trans. L bits)
- n flows; each flow has an arrival rate of a/n

- no reservation: all arrivals into the single link with rate R, the queueing delay + transmission delay:
  \[
  \frac{L}{R} \frac{1}{1 - \rho}
  \]

- reservation: each flow uses its own reserved (sub)link with rate R/n, the queueing delay + transmission delay:
  \[
  \frac{L}{R} \frac{1}{n} \frac{1}{1 - \rho}
  \]
Recap: Layering

- Why layering
  - reference model
  - modularization

- Concepts
  - service, interface, and protocol
  - physical vs logical communication

- Key layering principle
  - end-to-end arguments to place functions in layers
Recap: Internet Layering

- Five layers
  - highest two layers are implemented in host

- Form an hourglass structure

Diagram:

- Application
- Transport
- Network
- Datalink
- Physical

- Physical medium

- TCP
- UDP
- IP4/6

- Ethernet
- Wireless
- Cable/DSL

Applications:
- Email
- WWW
- FTP
- Telnet
Some Implications of Layered Architecture

- A packet as a stack container
  - \( \ldots \)
  - \( H_{n-1} \)
  - \( H_n \)

- Each layer needs multiplexing and demultiplexing to serve layer above
  - \( L_n \)
  - \( L_{n-1} \)
    - Has a field to indicate which higher layer requires the service
Link Layer (Ethernet)

- Services (to network layer)
  - multiplexing/demultiplexing
    - from/to the network layer
  - error detection
  - multiple access control
    - arbitrate access to shared medium

- Interface
  - send frames to a directly reachable peer
Link Layer: Protocol Header (Ethernet)
Network Layer: IP

- Services (to transport layer)
  - multiplexing/demultiplexing from/to the transport
  - fragmentation and reassembling: partition a fragment into smaller packets
    - removed in IPv6
  - error detection
  - routing: best-effort to send packets from source to destination
  - certain QoS/CoS
  - does not provide reliability or reservation

- Interface:
  - send a packet to a (transport-layer) peer at a specified global destination, with certain QoS/CoS
Network Layer: IPv4 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
<td>Version number of the protocol</td>
</tr>
<tr>
<td>Header length</td>
<td>4</td>
<td>Length of the header in 32-bit words</td>
</tr>
<tr>
<td>Type of Service (TOS)</td>
<td>8</td>
<td>Type of Service forgreges the handling of packets</td>
</tr>
<tr>
<td>Total Length (in bytes)</td>
<td>16</td>
<td>Total length of the packet in bytes</td>
</tr>
<tr>
<td>Identification</td>
<td>16</td>
<td>Identification number of the fragment</td>
</tr>
<tr>
<td>Flag (DF, MF)</td>
<td>13</td>
<td>Flag indicating if the fragment is last (MF) and if it can be fragmented (DF)</td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>8</td>
<td>Time to live, indicating how long the packet can live in the network</td>
</tr>
<tr>
<td>Protocol</td>
<td>8</td>
<td>Protocol number indicating the type of protocol</td>
</tr>
<tr>
<td>Source IP Address</td>
<td>32</td>
<td>IP address of the sender</td>
</tr>
<tr>
<td>Destination IP Address</td>
<td>32</td>
<td>IP address of the receiver</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>16</td>
<td>Checksum to ensure the integrity of the header</td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
<td>Options if any are present</td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td>Data, the actual payload</td>
</tr>
</tbody>
</table>
Transport Layer: UDP

- A connectionless service
- Does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee
  - why is there a UDP?
Transport Services and APIs

- **Multiple services and APIs proposed in history**
  - XTI (X/Open Transport Interface), a slight modification of the Transport Layer Interface (TLI) developed by AT&T.

- **Commonly used transport-layer service model and API: Socket**
  - Sometimes called "Berkeley sockets" acknowledging their heritage from Berkeley Unix
  - A socket has a transport-layer local port number
    - e.g., email (SMTP) port number 25, web port number 80
  - Application can send data into socket, read data out of socket
  - An application process binds to a socket (-a all; -u udp; -n number)
    - %netstat -aun
Socket Service Model and API

Controller by application developer

Controller by operating system

Process

Transport protocol

Internet

Host or server

Socket

buffers, states

Host or server

Socket

buffers, states

Controller by application developer

Controller by operating system
Multiplexing/Demultiplexing

Applications

TCP sockets

TCP ports 1 2 ... 65535

UDP sockets

UDP ports 1 2 ... 65535

TCP

UDP

IP

Socket references

Sockets bound to ports
Transport Layer: UDP Header

- 16-bit source port number
- 16-bit destination port number
- 16-bit UDP length
- 16-bit UDP checksum
- Data (if any)

Protocols:
- TCP
- UDP
- Telnet
- Email
- FTP
- WWW

Networks:
- Ethernet
- Wireless
- Cable/DSL
- FDDI
- Ethernet Wireless
- FDDI
Transport Layer: TCP

- **Services**
  - multiplexing/demultiplexing
  - reliable transport
    - between sending and receiving processes
    - setup required between sender and receiver: a connection-oriented service
  - flow control: sender won’t overwhelm receiver
  - congestion control: throttle sender when network overloaded
  - error detection
  - does not provide timing, minimum bandwidth guarantees

- **Interface:**
  - send a packet to a (app-layer) peer
Transport Layer: TCP Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit source port number</td>
<td></td>
</tr>
<tr>
<td>16-bit destination port number</td>
<td></td>
</tr>
<tr>
<td>32-bit sequence number</td>
<td></td>
</tr>
<tr>
<td>32-bit acknowledgment number</td>
<td></td>
</tr>
<tr>
<td>4-bit header length</td>
<td>reserved (6 bits)</td>
</tr>
<tr>
<td></td>
<td>U R C K A P S T R S Y T N F I N</td>
</tr>
<tr>
<td>16-bit TCP checksum</td>
<td></td>
</tr>
<tr>
<td>16-bit urgent pointer</td>
<td></td>
</tr>
<tr>
<td>options (if any)</td>
<td></td>
</tr>
<tr>
<td>data (if any)</td>
<td></td>
</tr>
</tbody>
</table>

Transport protocols:
- IP
- Ethernet
- Cable/DSL
- UDP
- TCP
- Wirefess
- Telnet
- Email
- FTP
- WWW

Network protocols:
- Ethernet
- Wireless
- FDDI

Transport protocols above Network protocols.
Secure Socket Layer Architecture

%openssl s_client -connect pop.gmail.com:995
### SSL Record-Layer Packet Format

<table>
<thead>
<tr>
<th>Content Type</th>
<th>Major Version</th>
<th>Minor Version</th>
<th>Compressed Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>20: change_cipher</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21: alert</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22: handshake</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23: application</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Encrypted
  - Plaintext (optionally compressed)
  - MAC (0, 16, or 20 bytes)
Summary: The Big Picture of the Internet

- **Hosts and routers:**
  - ~ 1 bil. hosts
  - autonomous systems organized roughly hierarchical
  - backbone links at 100 Gbps

- **Software:**
  - datagram switching with virtual circuit support at backbone
  - layered network architecture
    - use end-to-end arguments to determine the services provided by each layer
    - the hourglass architecture of the Internet
Protocol Formats
Outline

- Admin and recap
  - Application layer overview
Application Layer: Goals

- Conceptual + implementation aspects of network application protocols
  - client server paradigm
  - peer to peer (distributed) paradigm
  - network app. programming

- Learn about applications by examining common applications
  - pop/smtp
  - dns
  - ftp, http (1, 1.1, /2), content distribution
  - freenet, gossiping, BT, consensus
Network Applications vs. Application-layer Protocols

Network application: communicating, distributed processes
❖ a process is a program that is running within a host
  • a user agent is a process serving as an interface to the user
    – web: browser
    – streaming audio/video: media player
❖ processes communicate by an application-layer protocol
  • e.g., email, Web

Application-layer protocols
❖ one “piece” of an app
❖ define messages exchanged by apps and actions taken
❖ implementing services by using the service provided by the lower layer, i.e., the transport layer
### App. Protocols and their Transport Protocols

- An application needs to choose the transport protocol

<table>
<thead>
<tr>
<th>Application</th>
<th>Application layer protocol</th>
<th>Underlying transport protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-mail</td>
<td>smtp [RFC 821]</td>
<td>TCP/SSL</td>
</tr>
<tr>
<td>remote terminal access</td>
<td>telnet [RFC 854]</td>
<td>TCP</td>
</tr>
<tr>
<td>Web</td>
<td>http [RFC 2068]</td>
<td>TCP/SSL</td>
</tr>
<tr>
<td>file transfer</td>
<td>ftp [RFC 959]</td>
<td>TCP</td>
</tr>
<tr>
<td>Internet telephony</td>
<td>proprietary (e.g., Vocaltec)</td>
<td>typically UDP</td>
</tr>
<tr>
<td>remote file server</td>
<td>NFS</td>
<td>TCP or UDP</td>
</tr>
<tr>
<td>streaming multimedia</td>
<td>proprietary</td>
<td>typically UDP but moving to http</td>
</tr>
</tbody>
</table>
Client-Server Paradigm

Typical network app has two pieces: **client** and **server**

**Client (C):**
- initiates contact with server ("speaks first")
- typically requests service from server
- for Web, client is implemented in browser; for e-mail, in mail reader

**Server (S):**
- provides requested service to client
- e.g., Web server sends requested Web page; mail server delivers e-mail
Client-Server Paradigm: Key Questions

Key questions to ask about a C-S application design

- Is the design extensible?
- Is the design scalable?
- Is the design robust (e.g., server crashes)?
- Is the design secure (e.g., what kinds of security issues does it have/address)?
Outline

- Admin and recap
- Application layer overview
- Network applications
  - Email
Electronic Mail

- Still active
  - 80B emails/day
  - 3.9B active email boxes
- As simple as email, its app structure has complexities
  - A highly recommended reading: a history of Email development
    - linked on the Schedule page
Electronic Mail: Components

Two subsystems
- Message handling system (MHS)
- User agents

Two types of protocols
- Mail access protocols
  - POP3: Post Office Protocol [RFC 1939]
  - IMAP: Internet Mail Access Protocol [RFC 1730]
- Mail transport protocol
  - SMTP: Simple Mail Transport Protocol [RFC 5321]
Email Transport Architecture

MUA: User Agent
Mediator: User-level Relay
MHS: Mail Handling (transit) Service
MSA: Submission
MTA: Transfer
MDA: Delivery
Bounce: Returns

POP3 Protocol: Mail Retrieval (a Basic C-S Protocol)

Authorization phase

- client commands:
  - **user**: declare username
  - **pass**: password

- server responses
  - **+OK**
  - **-ERR**

Transaction phase, client:

- **list**: list message numbers
- **retr**: retrieve message by number
- **dele**: delete
- **quit**

% openssl s_client -connect pop.gmail.com:995

```plaintext
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged on

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```
Exercise

- Send email to yalecs433533
- Retrieve using pop3

See pop3-trace.txt
Some Observations

- An application protocol can be designed on top of different transport protocols (TCP, SSL)

- An application C-S protocol typically has state (e.g., first user before pass)—a sequence of commands forming a transaction
Evaluation of POP

Key questions to ask about a C-S application

- extensible?
- scalable?
- robust?
- security?
SMTP: Simple Mail Transport Protocol

The envelope and message define a mail transport transaction, specifying one client request.
Mail Message

SMTP: protocol for exchanging email msgs
RFC 822: standard for text message format:
- Header lines, e.g.,
  - To:
  - From:
  - Subject:
- Body
  - the “message”, ASCII characters only
SMTP: Simple Mail Transport Protocol

Envelope (meta info)

Message

Discussions

- Aren’t the fields in envelope duplicate of those in message headers?
- What is a main architecture benefit of the envelope-message separation design?
- SMTP is derived from MTP, which has command as MAIL <from@host1> TO <remote@host2>. Which implementation experience might motivate the separation?
- Historically SMTP often produced duplicates [RFC1047]. Why?
Exercise

- Look at structure of Mail message format

See pop3-trace.txt
Mail Message Extension

- MIME: multimedia mail extension, RFC 2045, 2056
  - Additional lines in msg header declare MIME content type

From: yry@cs.yale.edu
To: cs433@cs.yale.edu
Subject: Network map.
MIME-Version: 1.0
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

base64 encoded data ......
............................
......base64 encoded data

Benefit of MIME type: self-describing data type, adding extensibility.
Multipart Type: How Attachment Works

From: yry@cs.yale.edu
To: cs433@cs.yale.edu
Subject: Network map.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Hi,
Attached is network topology map.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
..........................
......base64 encoded data
--98766789--
Evaluation of SMTP

Key questions to ask about a C-S application

- extensible?
  separat. of envelope and msg; self-describing message; ehlo negotiation

- scalable?
  have not seen mechanism yet

- robust?
  have not seen mechanism yet

- security?
  authentication/authorization (spoof, spam) are major issues of mail transport