Network Applications: Overview, EMail

Y. Richard Yang

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

1/27/2016
Outline

- Admin and recap
  - ISO/OSI Layering and Internet Layering
  - Application layer overview
  - Network applications
    - Email
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 Network
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 $R$
- $n$ 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:
  $L \frac{1}{R \left(1 - \rho\right)}$

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

- Why layering
  - reference model
  - modularization

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

- Key design decision: what functionalities to put in each layer: End-to-end arguments

```
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
```

Host 1 - Layer 5 protocol - Host 2

Layer 4/5 interface
Layer 3/4 interface
Layer 2/3 interface
Layer 1/2 interface

Layer 5 protocol
Layer 4 protocol
Layer 3 protocol
Layer 2 protocol
Layer 1 protocol

Physical medium

End-to-end arguments
Outline

- Recap
  - ISO/OSI Layering and Internet Layering
- Application layer overview
ISO/OSI Reference Model

- Seven layers
  - lower three layers are hop-by-hop
  - next four layers are end-to-end (host-to-host)
Internet Layering

- Lower three layers are hop-by-hop
- Next two layers are end-to-end
Internet Protocol Layers

- **Five layers**
  - **Application**: specific network applications
    - ftp, smtp, http, p2p, IP telephony, ...
  - **Transport**: host-host data transfer
    - tcp (reliable), udp (not reliable)
  - **Network**: routing of datagram from source to destination
    - ipv4, ipv6
  - **Link**: data transfer between neighboring network elements
    - ethernet, 802.11, cable, DSL, ...
  - **Physical**: bits “on the wire”
    - cable, wireless, optical fiber
The Hourglass Architecture of the Internet
Link Layer (Ethernet)

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

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

- **Services**
  - routing: best-effort to send packets from source to destination
  - multiplexing/demultiplexing from/to the transport
  - fragmentation and reassembling: partition a fragment into smaller packets
    - removed in IPv6
  - error detection
  - 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>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
</tr>
<tr>
<td>Header length</td>
<td>4</td>
</tr>
<tr>
<td>Type of Service (TOS)</td>
<td>8</td>
</tr>
<tr>
<td>Total length</td>
<td>16</td>
</tr>
<tr>
<td>Identification</td>
<td>16</td>
</tr>
<tr>
<td>Fragment offset</td>
<td>13</td>
</tr>
<tr>
<td>Time to live (TTL)</td>
<td>8</td>
</tr>
<tr>
<td>Protocol</td>
<td>8</td>
</tr>
<tr>
<td>Header checksum</td>
<td>16</td>
</tr>
<tr>
<td>Source IP address</td>
<td>32</td>
</tr>
<tr>
<td>Destination IP address</td>
<td>32</td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
Services Provided by UDP

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

- 16-bit source port number
- 16-bit destination port number
- 32-bit sequence number
- 32-bit acknowledgment number
- 4-bit header length
- 16-bit TCP checksum
- 16-bit window size
- 16-bit urgent pointer
- Options (if any)
- Data (if any)

Diagram showing the structure of the TCP header with fields for source and destination ports, sequence numbers, acknowledgment numbers, header length, checksum, window size, urgent pointer, options, and data.
Secure Socket Layer Architecture

- SSL Handshake Protocol
- SSL Change Cipher Spec Protocol
- SSL Alert Protocol
- SSL Record Protocol
- TCP
- IP

HTTP
POP3
### 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>

Plaintext (optionally compressed)

MAC (0, 16, or 20 bytes)
Summary: The Big Picture of the Internet

- Hosts and routers:
  - ~ 1 bil. hosts (July 2015)
  - 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

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

- 4-bit version
- 4-bit header length
- 8-bit type of service (TOS)
- 16-bit total length (in bytes)
- 16-bit identification
- 13-bit fragment offset
- 8-bit time to live (TTL)
- 8-bit protocol
- 16-bit header checksum

- 32-bit source IP address
- 32-bit destination IP address
- options (if any)

- data

- DA | SA | Type | Data | CRC
- 6 | 6 | 2 | 46-1500 | 4

Ethernet frame
Minimum size = 64 bytes
Outline

- Recap
- ISO/OSI Layering and Internet Layering
  - Application layer overview
Application Layer: Goals

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

- Learn about applications by examining common applications
  - smtp/pop
  - dns
  - http
  - content distribution
How does an Application Access the Transport Service?

API: application programming interface

- Defines interface between application and transport layer

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

- Commonly used: Socket API
  - Sometimes called "Berkeley sockets" acknowledging their heritage from Berkeley Unix
  - A socket has a network-layer host IP address and a transport-layer local port number
    - E.g., email (SMTP) port number 25, web port number 80
  - An application process binds to a socket
    - %netstat or lsof
  - Two processes communicate by sending data into socket, reading data out of socket
Socket API

- Host or server
- Process
- Socket
- Buffers, states
- Controlled by application developer
- Controlled by operating system

- Transport protocol
- Internet
- Host or server
- Process
- Socket
- Buffers, states
- Controlled by application developer
- Controlled by operating system
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>
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
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
Key questions to ask about a C-S application:

- Is the application extensible?
- Is the application scalable?
- How does the application handle server failures (being robust)?
- How does the application provide security?
Electronic Mail

- Still active
  - 80B emails/day
  - 3.9B active email boxes

Three major components:
- User agents
- Mail servers
- Protocols
  - Outgoing email
    - SMTP
  - Retrieving email
    - POP3: Post Office Protocol [RFC 1939]
    - IMAP: Internet Mail Access Protocol [RFC 1730]
SMTP: Outgoing Email as a Client-Server Application

S: 220 mr1.its.yale.edu
C: HELO cyndra.yale.edu
S: 250 Hello cyndra.cs.yale.edu, pleased to meet you
C: MAIL FROM: <spoof@cs.yale.edu>
S: 250 spoof@cs.yale.edu... Sender ok
C: RCPT TO: <yry@yale.edu>
S: 250 yry@yale.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Date: Wed, 23 Jan 2008 11:20:27 -0500 (EST)
C: From: "Y. R. Yang" <yry@cs.yale.edu>
C: To: "Y. R. Yang" <yry@cs.yale.edu>
C: Subject: This is subject
C:
C: This is the message body!
C: Please don’t spoof!
C:
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 mr1.its.yale.edu closing connection
Email Transport Architecture

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

Mail Message Data Format

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 (any problem?)
Message Format: Multimedia Extensions

- 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
```
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--
Design Review

S: 220 mr1.its.yale.edu
C: HELO cyndra.yale.edu
S: 250 Hello cyndra.cs.yale.edu, pleased to meet you
C: MAIL FROM: <spoof@cs.yale.edu>
S: 250 spoof@cs.yale.edu... Sender ok
C: RCPT TO: <yry@yale.edu>
S: 250 yry@yale.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: From: yry@cs.yale.edu
C: To: cs433@cs.yale.edu
C: Subject: Network map.
C: MIME-Version: 1.0
C: Content-Type: image/jpeg
C: Content-Transfer-Encoding: base64
C:
C: base64 encoded data ..... 
C: ........................
C: ......base64 encoded data
C:
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 mr1.its.yale.edu closing connection

Why not make the msg headers smtp headers?
POP3 Protocol: Mail Retrieval

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

```plaintext
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
```

```plaintext
% openssl s_client -connect pop.gmail.com:995
```
Evaluation of SMTP/POP/IMAP

Key questions to ask about a C-S application
- extensible?
- scalable?
- robust?
- security?
Email: Positive

- Some nice design features we can learn from the design of the email
  - separate protocols for different functions
    - email retrieval (e.g., POP3, IMAP)
    - email transmission (SMTP)
  - simple/basic requests to implement basic control; fine-grain control through ASCII header and message body
    - make the protocol easy to read/debug/extend (analogy with end-to-end layered design?)
  - status code in response makes message easy to parse
Email: Challenge

- Spam (Google)

https://mail.google.com/intl/en/mail/help/fightspam/spamexplained.html
Email: Challenge

- A large percentage of spam/phish

Global spam volume as percentage of total e-mail traffic from 2007 to 2014

Email: Challenge

Leading countries of origin for unsolicited spam emails as of 2nd quarter 2015, by share of worldwide spam volume

- United States: 14.59%
- Russia: 7.82%
- China: 7.14%
- Vietnam: 5.04%
- Germany: 4.13%
- Ukraine: 3.9%
- India: 3.41%
- France: 3.22%
- Argentina: 2.69%
- Great Britain: 2.61%

Discussion: How May Email Spams Be Detected?
Detection Methods Used by GMail

- Known phishing scams
- Message from unconfirmed sender identity
- Message you sent to Spam/similarity to suspicious messages
- Administrator-set policies
- Empty message content

https://support.google.com/mail/answer/1366858?hl=en