Network Applications: Overview, EMail

Y. Richard Yang

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

9/10/2013

Outline

- Admin and recap
- ISO/OSI Layering and Internet Layering
- Application layer overview
- Network applications
  - Email

Admin

- Questions on Assignment One

Recap: Summary of the Taxonomy of Communication Networks

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:
  \[
  L \frac{1}{R} \frac{1}{1 - \rho}
  \]

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

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
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
    - ip
  - 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 reassembly: 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

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

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

Secure Socket Layer Architecture

SSL Record-Layer Packet Format

Summary: The Big Picture of the Internet

Outline

Application Layer: Goals

Recap

ISO/OSI Layering and Internet Layering

Application layer overview

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

Hosts and routers:

- ~ 1 bil. hosts (July 2013)
- 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

Learn about applications by examining common applications:

- SMTP/POP
- DNS
- HTTP
- Content distribution
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

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.
- Our focus: 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
    - "Socket get" two processes communicate by sending data into socket, reading data out of socket

Socket API


- 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
- 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

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

Email Transport Architecture

Mail Message Data Format

Message Format: Multimedia Extensions

Multipart Type: How Attachment Works
**Design Review**

S: 220 mail.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: base64 encoded data ....
C: ................................
C: ......base64 encoded data
C: ..
S: 250 Message accepted for delivery
C: QUIT
S: 221 mail.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

---

**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 (Yale)

[Graph showing spam trends]

---

Spam (Google)

[Graph showing spam trends]

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

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

Optional Slides

Services Provided by Transport

- User data protocol (UDP) - multiplexing/demultiplexing
- Transmission control protocol (TCP) - multiplexing/demultiplexing, reliable data transfer, rate control: flow control and congestion control

Secure Socket Layer: Services

- server authentication
  - authentication through trusted certificate authority (CA): server obtains a certificate from one of the trusted CAs
- data encryption and integrity
- client authentication (optional)
# Details of the Seven ISO/OSI Layers

## Physical Layer (1)
- **Service:** moves information between two systems connected by a physical link
- **Interface:** specifies how to send a bit
- **Protocol:** coding scheme used to represent a bit, voltage levels, duration of a bit
- **Examples:** coaxial cable, optical fiber links; transmitters, receivers

## Datalink Layer (2)
- **Service:**
  - framing, i.e., attach frames separator
  - send data frames between peers
  - others:
    - arbitrates the access to common physical media
    - ensures reliable transmission
    - provides flow control
- **Interface:** sends a data unit (packet) to a machine connected to the same physical media
- **Protocol:** layer addresses, implement Medium Access Control (MAC) (e.g., CSMA/CD)...

## Network Layer (3)
- **Service:**
  - delivers a packet to a specified destination
  - performs fragmentation/reassembly of packets
  - others:
    - packet scheduling
    - buffer management
- **Interface:** sends a packet to a specified destination
- **Protocol:** defines global unique addresses; constructs routing tables; implement packet forwarding; fragments/reassembles packets

## Data and Control Planes
- **Data plane:** concerned with
  - packet forwarding
  - buffer management
  - packet scheduling
- **Control Plane:** concerned with installing and maintaining the states for the data plane

## Transport Layer (4)
- **Service:**
  - provides an in-order, error-free, and flow and congestion controlled end-to-end connection
  - multiplex/demultiplex packets
- **Interface:** sends a packet to a destination
- **Protocol:** implements reliability, as well as flow and congestion control
- **Examples:** TCP and UDP
  - TCP: in-order, error free, flow and congestion control
Session Layer (5)
- Service:
  - full-duplex
  - access management, e.g., token control
  - synchronization, e.g., provide check points for long transfers
- Interface: depends on service
- Protocols: token management; insert checkpoints, implement roll-back functions

Presentation Layer (6)
- Service: converts data between various representations
- Interface: depends on service
- Protocol: defines data formats and rules to convert from one format to another

Application Layer (7)
- Service: any service provided to end users
- Interface: depends on the application
- Protocol: depends on the application
- Examples: FTP, Telnet, WWW

What Transport Service Does an App Need?
- Data loss
  - some apps can tolerate some packet losses
  - other apps require 100% reliable data transfer
- Bandwidth
  - some apps require minimum amount of bandwidth to be "effective"
  - other apps make use of whatever bandwidth they get
- Timing
  - some apps require low delay to be "effective"