

EEC 189Q: Computer Networks

Application Layer:
World Wide Web
Electronic Mail

Reading: Appendix A

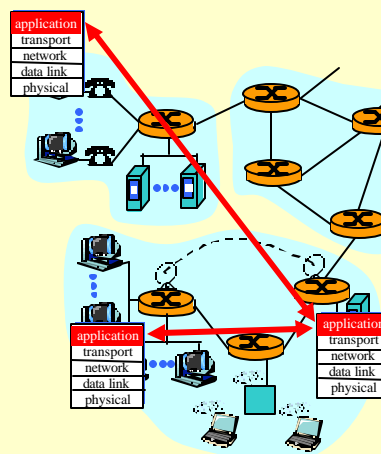
Applications and application-layer protocols

Application: communicating, distributed processes

- Running in network hosts in “user space”
- Exchange messages to implement application
- e.g., email, file transfer, the Web

Application-layer protocols

- One “piece” of an application
- Define messages exchanged by apps and actions taken
- Use services provided by lower layer protocols



Client-server paradigm

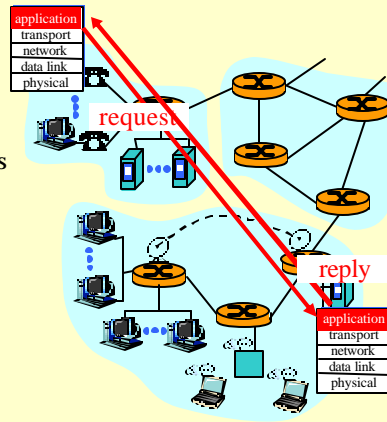
Typical network app has two pieces: *client* and *server*

Client:

- 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:

- Provides requested service to client, e.g., Web server sends requested Web page, mail server delivers e-mail



What transport service does an app need?

Data loss

- Some apps (e.g., audio) can tolerate some loss
- Other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Bandwidth

- Some apps (e.g., multimedia) require minimum amount of bandwidth to be “effective”
- Other apps (“elastic apps”) make use of whatever bandwidth they get

Timing

- Some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

Transport service requirements of common apps

<u>Application</u>	<u>Data loss</u>	<u>Bandwidth</u>	<u>Time Sensitive</u>
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	loss-tolerant	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kb-1Mb video: 10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps up	yes, 100's msec
financial apps	no loss	elastic	yes and no

Chuah, Fall 2004

5

Internet apps: their protocols and transport protocols

<u>Application</u>	<u>Application layer protocol</u>	<u>Underlying transport protocol</u>
e-mail	smtp [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
remote file server	NSF	TCP or UDP
Internet telephony	proprietary (e.g., Vocaltec)	typically UDP

Chuah, Fall 2004

6

The Web: some jargon

- Web page:
 - Consists of “objects”
 - Addressed by a URL
- Most Web pages consist of:
 - Base HTML page, and
 - Several referenced objects.
- URL has two components: host name and path name:
- User agent for Web is called a browser:
 - MS Internet Explorer
 - Netscape Communicator
- Server for Web is called Web server:
 - Apache (public domain)
 - MS Internet Information Server

www.someSchool.edu/someDept/pic.gif

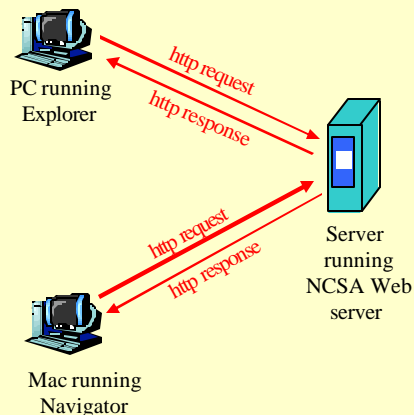
Chuah, Fall 2004

7

The Web: the http protocol

Http: hypertext transfer protocol

- Web's application layer protocol
- Client/server model
 - *client*: browser that requests, receives, “displays” Web objects
 - *server*: Web server sends objects in response to requests
- http1.0: RFC 1945
- http1.1: RFC 2068



Chuah, Fall 2004

8

The http protocol: more

http: TCP transport service:

- Client initiates TCP connection (creates socket) to server, port 80
- Server accepts TCP connection from client
- http messages (application-layer protocol messages) exchanged between browser (http client) and Web server (http server)
- TCP connection closed

http is “stateless”

- Server maintains no information about past client requests

aside

Protocols that maintain “state” are complex!

- Past history (state) must be maintained
- If server/client crashes, their views of “state” may be inconsistent, must be reconciled

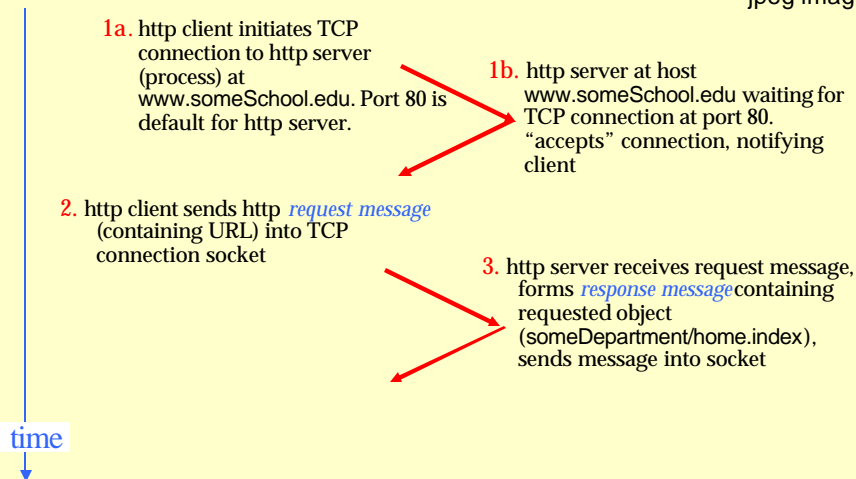
Chuah, Fall 2004

9

http example

Suppose user enters URL
www.someSchool.edu/someDepartment/home.index


(contains text,
references to 10
jpeg images)



Chuah, Fall 2004

10

http example (cont.)

- 
4. http server closes TCP connection.
 5. http client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects
 6. Steps 1-5 repeated for each of 10 jpeg objects

Non-persistent and persistent connections

Non-persistent

- HTTP/1.0
- Server parses request, responds, and closes TCP connection
- 2 RTTs to fetch each object
- Each object transfer suffers from slow start

Persistent

- Default for HTTP/1.1
- On same TCP connection: server, parses request, responds, parses new request,...
- Client sends requests for all referenced objects as soon as it receives base HTML.
- Fewer RTTs and less slow start.

But most 1.0 browsers use parallel TCP connections.

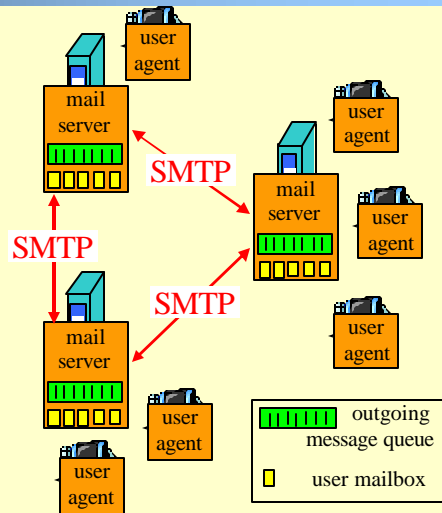
Electronic Mail

Three major components

- User agents
- Mail servers
- Simple mail transfer protocol: smtp

User Agent

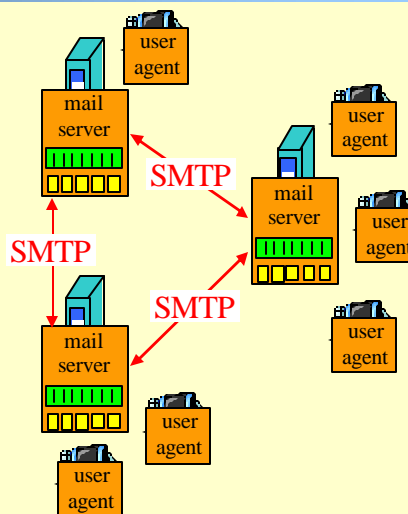
- a.k.a. “mail reader”
- Composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, Netscape Messenger
- Outgoing, incoming messages stored on server



Electronic Mail: mail servers

Mail Servers

- **Mailbox** contains incoming messages (yet to be read) for user
- **Message** queue of outgoing (to be sent) mail messages
- **smtp protocol** between mail servers to send email messages
 - Client: sending mail server
 - “Server”: receiving mail server



Electronic Mail: smtp [RFC 821]

- Uses tcp to reliably transfer email msg from client to server, port 25
- Direct transfer: sending server to receiving server
- Three phases of transfer
 - Handshaking (greeting)
 - Transfer of messages
 - Closure
- Command/response interaction
 - **Commands**: ASCII text
 - **Response**: status code and phrase
- Messages must be in 7-bit ASCII

Sample smtp interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C:   How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Try smtp interaction for yourself:

- `telnet servername 25`
 - see 220 reply from server
 - enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands
- above lets you send email without using email client (reader)

smtp: final words

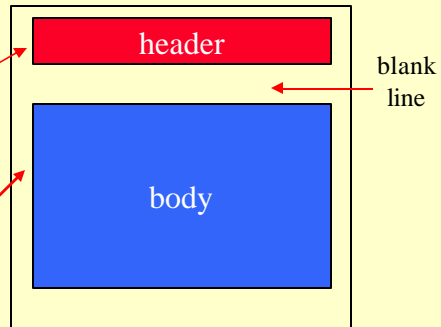
- Smtplib uses persistent connections
 - Smtplib requires that message (header & body) be in 7-bit ascii
 - Certain character strings are not permitted in message (e.g., CRLF . CRLF). Thus message has to be encoded (usually into either base-64 or quoted printable)
 - Smtplib server uses CRLF . CRLF to determine end of message
- Comparison with http**
- http: pull
 - Email: push
 - Both have ASCII command/response interaction, status codes
 - http: each object is encapsulated in its own response message
 - smtp: multiple objects message sent in a multipart message

Mail message format

smtp: protocol for exchanging email msgs

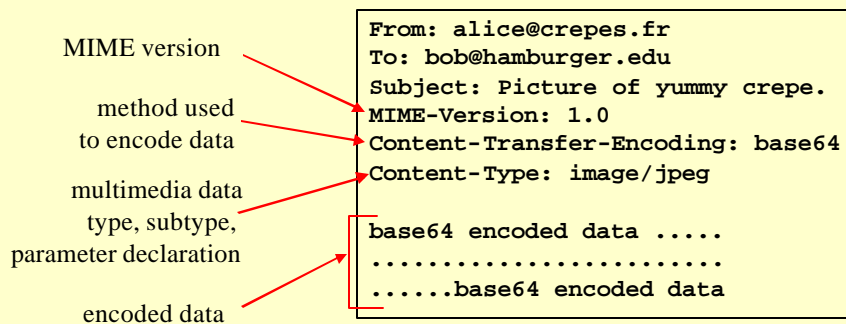
RFC 822: standard for text message format:

- Header lines, e.g.,
 - To:
 - From:
 - Subject:*different from smtp commands*
- Body
 - the "message", ASCII characters only



Message format: multimedia extensions

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



MIME types

Content-Type: type/subtype; parameters

Text

- Example subtypes: `plain`, `html`

Image

- Example subtypes: `jpeg`, `gif`

Audio

- Example subtypes: `basic` (8-bit mu-law encoded), `32kadpcm` (**32 kbps coding**)

Video

- Example subtypes: `mpeg`, `quicktime`

Application

- Other data that must be processed by reader before “viewable”
- Example subtypes: `msword`, `octet-stream`

Multipart Type

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789
```

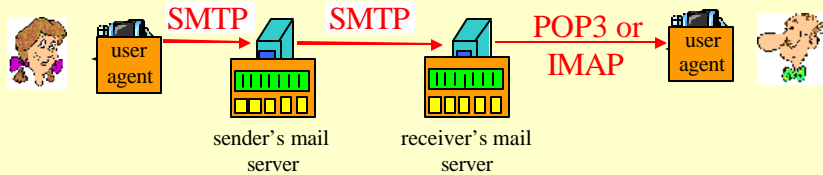
```
--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain
```

```
Dear Bob,
Please find a picture of a crepe.
```

```
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
```

```
base64 encoded data .....
.....base64 encoded data
--98766789--
```

Mail access protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
 - POP: Post Office Protocol [RFC 1939]
 - Authorization (agent <-->server) and download
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - More features (more complex)
 - Manipulation of stored msgs on server
 - HTTP: Hotmail , Yahoo! Mail, etc.

POP3 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**

```
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
```