ECE 356/COMPSI 356
Computer Network Architecture

Application Layer Protocols

Monday November 25, 2019

Duke UNIVERSITY

1

Previous Lecture Recap

- Enterprise networking
- Duke University networks

- Readings for this lecture: **PD 9.1.1, 9.1.2**

2

Duke UNIVERSITY

2

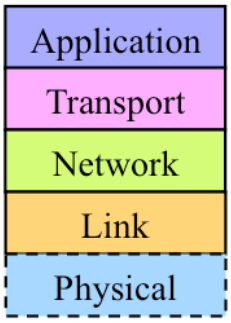
Lecture Outline

- **Application protocols**
- Web and HTTP
- Electronic Mail
  - SMTP
  - Mail access protocols

Application-Layer Protocols (1/2)

- Run above the transport layer
- E.g., DNS is an application-level protocol

- Only some application-layer protocols are in the public domain
  - E.g., Telnet, FTP, HTTP are defined in RFCs
  - Skype is private



Application-Layer Protocols (2/2)

Protocols define:

- Types of messages (e.g., requests and responses)
- Message syntax (e.g., fields, and how to delineate)
- Semantics of the fields (i.e., meaning of the information)
- Rules for when and how a process sends messages
- Platform and programming language independent

5

Duke UNIVERSITY

5

Lecture Outline

- Application protocols
- **Web and HTTP**
  - Cookies
- Electronic Mail
  - SMTP
  - Mail access protocols

6

Duke UNIVERSITY

6

Web and HTTP: An Introduction

- *Web page* consists of *objects*
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of *base HTML-file* which includes *several referenced objects*
- Each object is addressable by a *URL,* e.g.,

`www.someschool.edu`/`someDept/pic.gif`

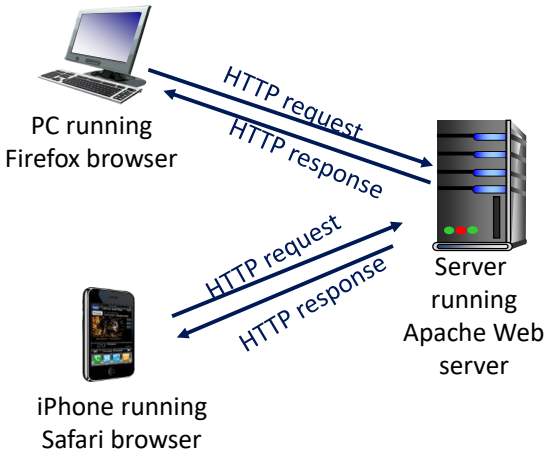
host name path name

7

Duke UNIVERSITY

HTTP Overview (1/3)

- Hypertext transfer protocol
- Web's application layer protocol
- Client/server model
  - *Client:* browser that requests, receives (using HTTP protocol) and displays Web objects
  - *Server:* web server sends (using HTTP protocol) objects in response to requests



8

Duke UNIVERSITY

HTTP Overview (2/3)

- *Uses TCP*
- Process flow:
  - Client initiates TCP connection 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

9

HTTP Overview (3/3)

*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

10

HTTP Connections

Non-persistent HTTP

- At most one object sent over TCP connection
  - Connection then closed
- Downloading multiple objects required multiple connections

Persistent HTTP

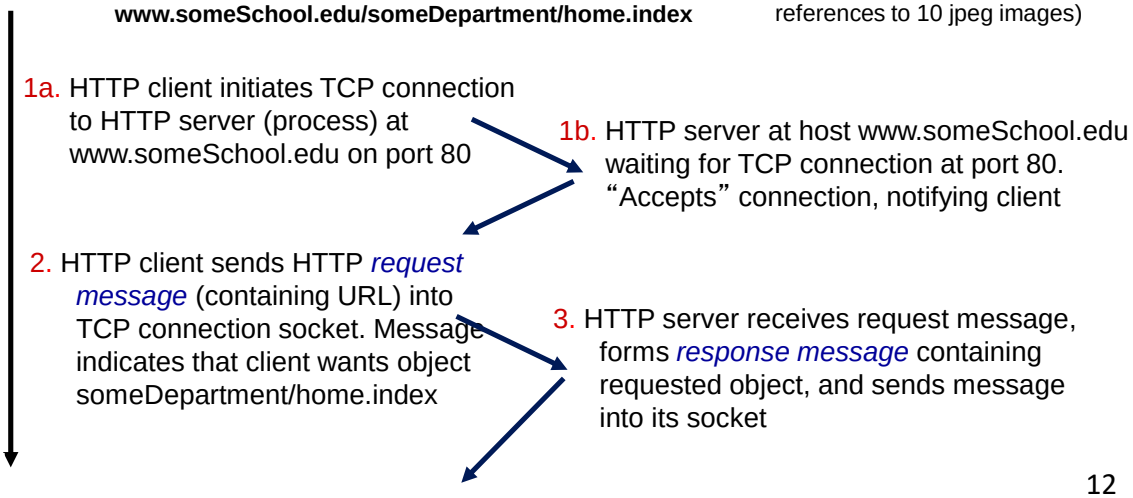
- Multiple objects can be sent over single TCP connection between client, server
- Default option

11

Non-persistent HTTP (1/2)

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

(contains text, references to 10 jpeg images)



12

Non-persistent HTTP (2/2)

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

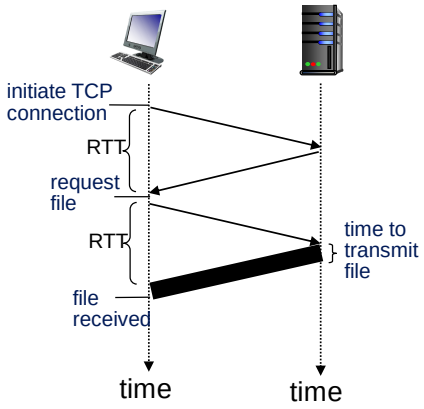
6. Steps 1-5 repeated for each of 10 jpeg objects

time

Non-persistent HTTP: Response Time

- One RTT to initiate TCP connection
- One RTT for HTTP request and first few bytes of HTTP response to return
- File transmission time
- non-persistent HTTP response time =

  2RTT+ file transmission time



Persistent HTTP

Non-persistent HTTP issues:

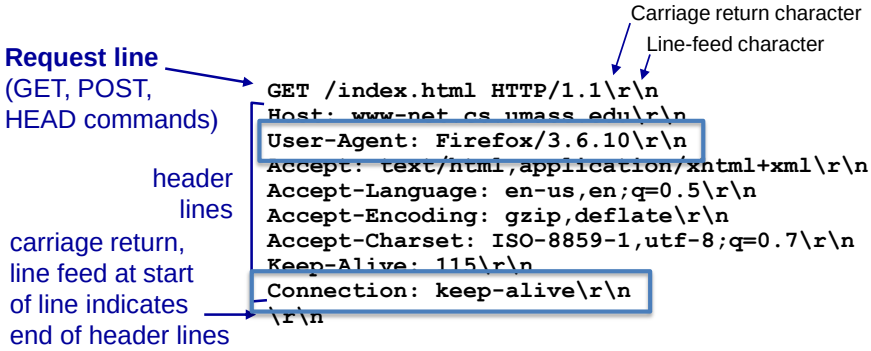
- Requires 2 RTTs per object
- OS overhead for *each* TCP connection
- Browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP:

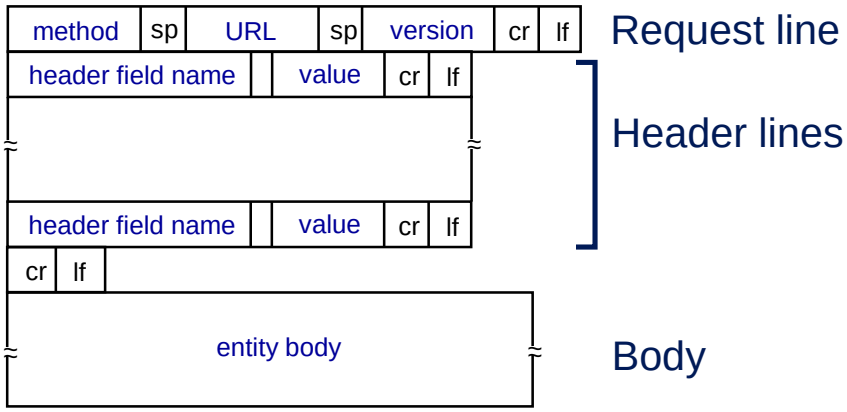
- Server leaves connection open after sending response
- Subsequent HTTP messages between same client/server sent over open connection
- Client sends requests as soon as it encounters a referenced object
- As little as one RTT for all the referenced objects

HTTP Request Message

- Two types of HTTP messages: *request*, *response*
- HTTP request message:
  - ASCII (human-readable format)



HTTP Request Message: General Format



Uploading Form Input

POST method:

- Web page often includes form input
- Input is uploaded to server in entity body

URL method:

- Uses GET method
- Input is uploaded in URL field of request line:

```
www.somesite.com/animalsearch?monkeys&banana
```

Method Types

HTTP/1.0:

- GET
- POST
- HEAD
  - Asks server to leave requested object out of response

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - Uploads file in entity body to path specified in URL field
- DELETE
  - Deletes file specified in the URL field

19

HTTP Response Message

Status line (protocol status code status phrase)

Header lines

Data, e.g., requested HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
    GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
    1\r\n
\r\n
data data data data data ...
```

20

HTTP Response Status Codes

- Status code appears in 1st line in server-to-client response message
- Some sample codes:

200 OK
- Request succeeded, requested object later in this msg

400 Bad Request
- Request msg not understood by server

404 Not Found
- Requested document not found on this server

505 HTTP Version Not Supported

Hyper Text Coffee Pot Control Protocol

- April Fool's RFCs:
  - 1998 Hyper Text Coffee Pot Control Protocol (HTCPCP): communication protocol for controlling, monitoring, and diagnosing coffee pots
  - 2014 HTCPCP-TEA to support brewing teas
- Commands: BREW, WHEN
- Status codes:

418 I'm a teapot

- Save418.com

Web and HTTP: Key Points to Remember (1/2)

- HTTP:
  - **Ubiquitous**
  - **Client-server**
  - **Stateless** application-layer protocol
- Client downloads a webpage consisting of multiple objects
  - Multiple objects can be handled in persistent and non-persistent manner

Web and HTTP: Key Points to Remember (2/2)

- HTTP defines *request* and *response* messages
- Request: methods: GET, POST, HEAD, PUT, DELETE
- Reply includes a status code

Lecture Outline

- Application protocols
- Web and HTTP
  - **Cookies**
- Electronic Mail
  - SMTP
  - Mail access protocols

25

User-server State: Cookies (1/2)

- While HTTP is stateless, state information is useful for web transactions
- Many Web sites use **cookies**
- *Four components:*
  1) Cookie header line of HTTP *response* message
  2) Cookie header line in next HTTP *request* message
  3) Cookie file kept on user's host, managed by user's browser
  4) Back-end database at Web site

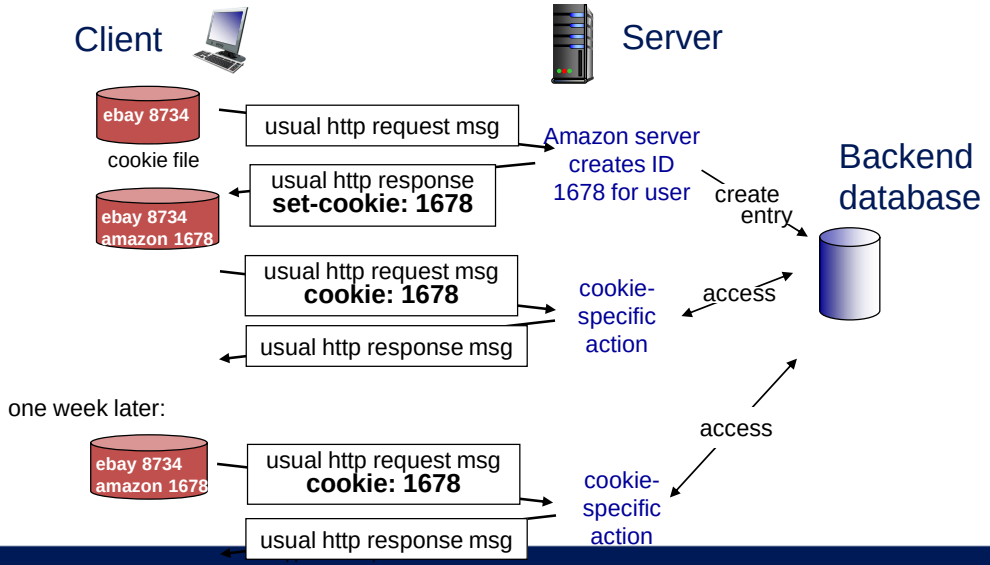
26

User-server State: Cookies (2/2)

Example:

- Susan always access Internet from PC
- Visits specific e-commerce site for first time
- When initial HTTP requests arrives at site, site creates:
  - Unique ID
  - Entry in backend database for ID

Cookies: An Example



Cookies: Usage and Concerns

What cookies can be used for:

- Authorization
- Shopping carts
- Recommendations
- User session state (Web e-mail)

aside

Cookies and privacy:

- Cookies permit sites to learn a lot about you
- You may supply name and e-mail to sites

How to keep "state":

- Protocol endpoints: maintain state at sender/receiver over multiple transactions
- Cookies: HTTP messages carry state

29

Where Cookies Get Creepy

- (Arguably) OK for a service provider to help orient you on their website: these are so-called "first-party cookies"
- "Third-party cookies" are another matter
- Ad servers, Facebook can track your behavior across *multiple disjoint websites*
  - A page with a Facebook "like" button generates a Facebook cookie
  - *You do not have to engage with the "like" button to be tracked*: only need to visit the website

30

Lecture Outline

- Application protocols
- Web and HTTP
- **Electronic Mail**
  - SMTP
  - Mail access protocols

31

Duke UNIVERSITY

31

Electronic Mail in the Internet

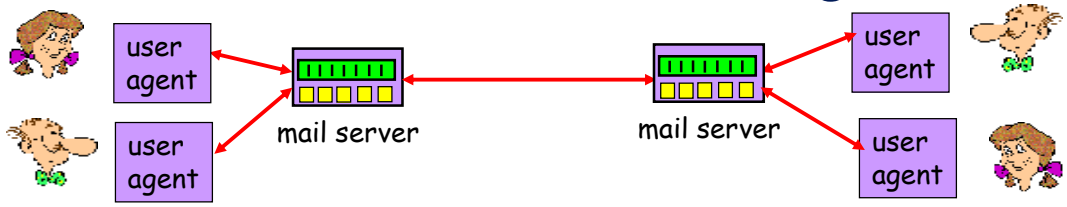
- Has been around since the beginning of the Internet
- Was the most popular application when Internet was in its infancy
- *Asynchronous* communication medium
  - People send and read messages when it is convenient for them
- A suite of protocols:
  - Simple Mail Transfer Protocol (**SMTP**)
  - Post Office Protocol (**POP3**)
  - Internet Message Access Protocol (**IMAP**)

32

Duke UNIVERSITY

32

Electronic Mail: Mail Servers and User Agents

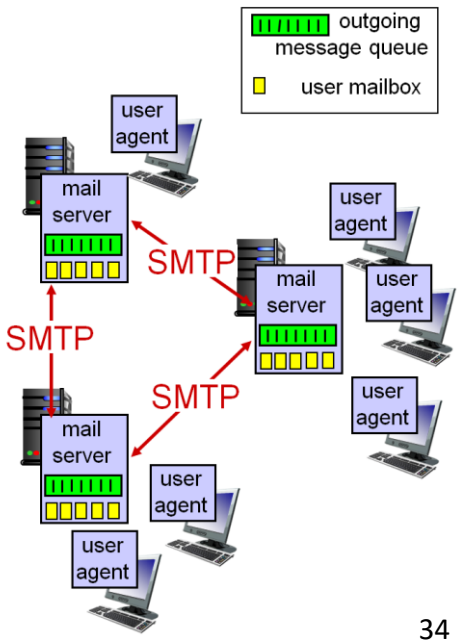


- Mail servers
  - Always on and always accessible
  - Transferring e-mail to and from other servers
- User agents
  - Sometimes on and sometimes accessible
  - Intuitive interface for the user

33

Electronic Mail: 3 Major Components

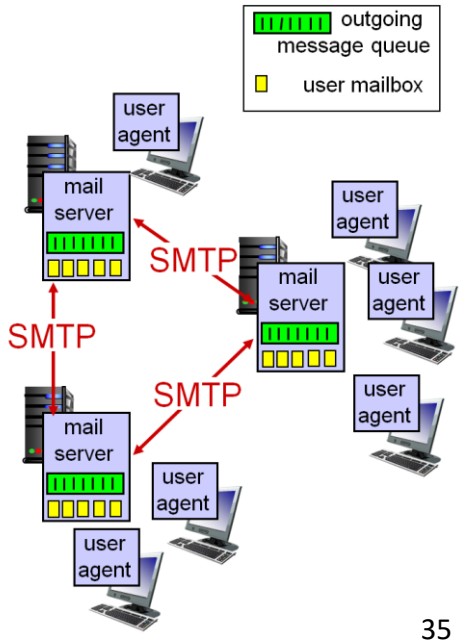
- User agents
- Mail servers
- Simple mail transfer protocol: SMTP



34

Electronic Mail: User Agent

- A.k.a. “mail reader”
- Composing, editing, reading mail messages
- E.g., Outlook, Thunderbird, iPhone mail client
- Outgoing, incoming messages stored on server



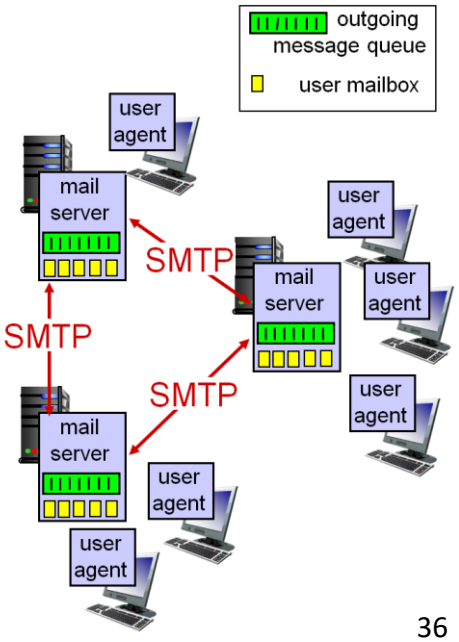
35

Duke UNIVERSITY

35

Electronic Mail: Mail Server

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



36

Duke UNIVERSITY

36

SMTP (1/2)

- Uses TCP to reliably transfer email message from client to server
  - Port 25
- Three phases of transfer
  - Handshaking (greeting)
  - Transfer of messages
  - Closure

37

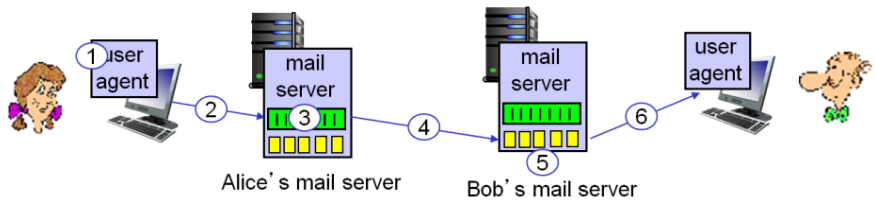
SMTP (2/2)

- Command/response interaction (like HTTP)
  - Commands: ASCII text
  - Response: status code and phrase
- Messages must be in 7-bit ASCII

38

Scenario: Alice Sends Message to Bob (1/2)

1) Alice uses UA to compose message to bob@someschool.edu
2) Alice's UA sends message to her mail server; message placed in message queue
3) Client side of SMTP opens TCP connection with Bob's mail server



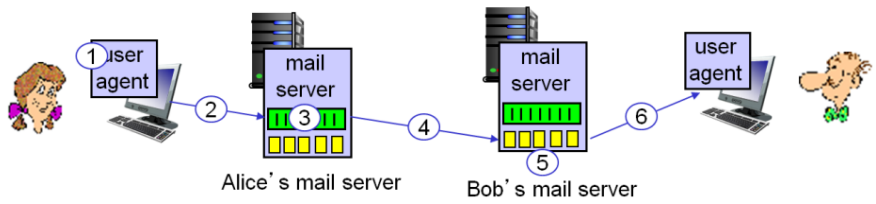
39

Duke UNIVERSITY

39

Scenario: Alice Sends Message to Bob (2/2)

4) SMTP client sends Alice's message over the TCP connection
5) Bob's mail server places the message in Bob's mailbox
6) Bob invokes his user agent to read message



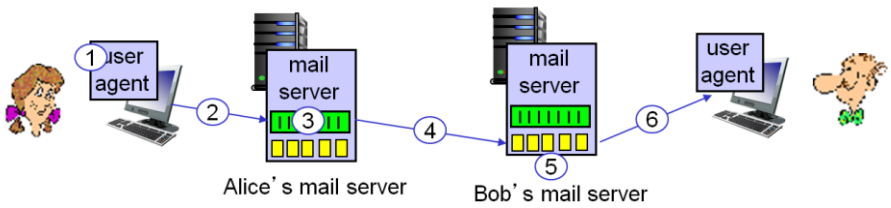
40

Duke UNIVERSITY

40

Question: Why Do We Need Mail Servers?

Why not let user agents do the work of the servers?



41

Duke UNIVERSITY

41

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
```

42

Duke UNIVERSITY

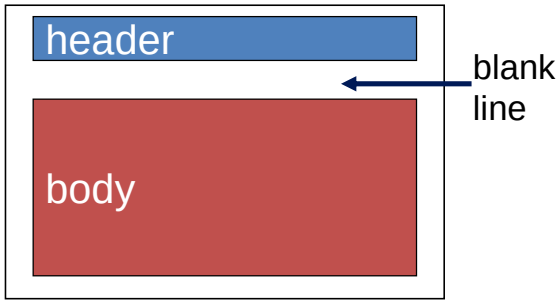
42

SMTP vs. HTTP

- HTTP: pull
  - TCP connection initiated by the machine that wants to *receive* a file
- SMTP: **push protocol**
  - TCP connection initiated by the machine that wants to *send* a file
- Both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response message
- SMTP: multiple objects sent in multipart message
  - Persistent connections

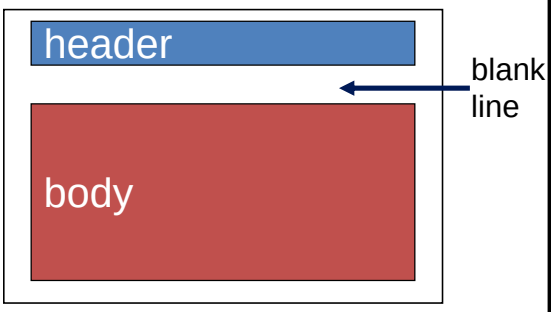
E-Mail Message Format (1/2)

- E-mail messages have two parts
  - A header, in 7-bit U.S. ASCII text
  - A body, also represented in 7-bit U.S. ASCII text
- A blank line between the two



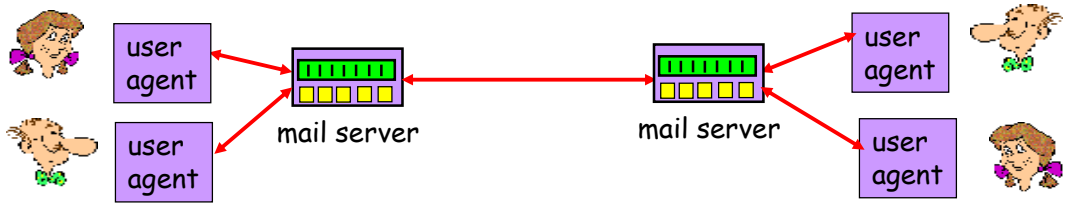
E-Mail Message Format (2/2)

- Header
  - *To*, *from*, *subject* lines with "type: value"
  - "To: xyz@cs.duke.edu"
  - "Subject: Hello!"
- Body
  - The text message
  - No particular structure or meaning



45

Electronic Mail and SMTP: Key Points To Remember



- SMTP protocol between mail servers to send email messages
- Client/server architecture:
  - Client: sending mail server
  - "Server": receiving mail server
- Push protocol

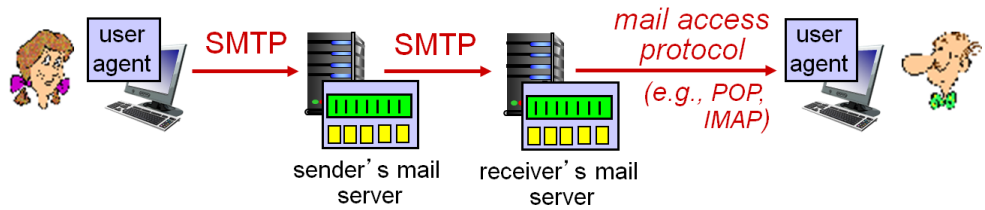
46

Lecture Outline

- Application protocols
- Web and HTTP
- Electronic Mail
  - SMTP
  - **Mail access protocols**

47

Mail Access Protocols



- **SMTP:** delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol: authorization, download
  - IMAP: Internet Mail Access Protocol: more features, including manipulation of stored messages on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

48

Retrieving E-Mail From the Server

- Server stores incoming e-mail by mailbox
  - Based on the "From" field in the message
- Users need to retrieve e-mail
  - *Asynchronous access*
  - Need to view, reply, organize, and store messages
- In the pre-PC days…
  - User logged on to the machine where mail was delivered
  - Users received e-mail on their main work machine

49

Influence of PCs on E-Mail Retrieval

- Separate machine for personal use
  - Users did not want to log in to remote machines
- Resource limitations
  - Most PCs did not have enough resources to act as a full-fledged e-mail server
- Intermittent connectivity
  - PCs only sporadically connected to the network
  - … due to dial-up connections, and shutting down of PC
  - Too unwieldy to have sending server keep trying
- Led to the creation of Post Office Protocol (POP)

50

Post Office Protocol (POP) (1/3)

- Simple protocol with limited capabilities
- POP goals
  - Allow users retrieve e-mail messages when connected
  - … and view/manipulate messages when disconnected
- *User agent still uses SMTP to send messages*

Post Office Protocol (POP) (2/3)

- Typical user agent interaction with a POP server
  - Open a TCP connection to the mail server
    - Port 110
  - Retrieve all e-mail messages
  - Store messages on the user's PCs as new messages
  - Delete the messages from the server
  - Disconnect from the server

Post Office Protocol (POP) (3/3)

- 3 operational phases: **authorization**, **transaction**, **update**
  - Authorization: username, password sent in the clear
  - Transaction: user agent retrieves messages, marks messages for deletion
  - Update: after the *quit* command, mail server deletes messages

53

Duke UNIVERSITY

POP3 Protocol (1/2)

Authorization phase →

- Client commands:
  - **`user:`** declare username
  - **`pass:`** password
- Server responses
  - **`+OK`**
  - **`-ERR`**

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

54

Duke UNIVERSITY

POP3 Protocol (2/2)

Transaction phase, client:

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

```
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
```


POP: Comments and Limitations

- Previous example uses POP3 “download and delete” mode
  - ➢ Bob cannot re-read e-mail if he changes client
- POP3 “download-and-keep”: copies of messages on different clients
- POP3 is stateless across sessions
  - ➢ Simple implementation, limited capability

Interactive Mail Access Protocol (IMAP)

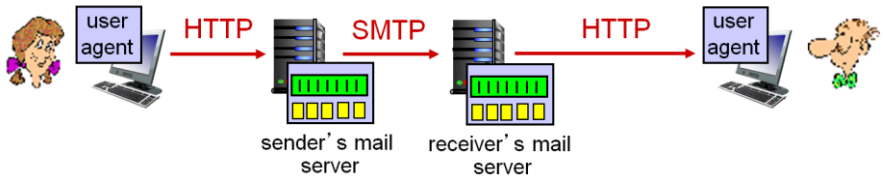
- POP: protocol for retrieving content of a mailbox
- IMAP: remote access mailbox protocol
  - Keeps all messages in one place: at server
- Allows user to organize messages in folders
- Keeps user state across sessions:
  - Names of folders and mappings between message IDs and folder name

Web-Based E-Mail (1/2)

- Introduced by Hotmail in mid 1990s
- User agent is an ordinary Web browser
  - User communicates with server via HTTP
  - Gmail, Yahoo mail, …
- Reading e-mail
  - Web pages display the contents of folders
  - … and allow users to download and view messages
  - "GET" request to *retrieve* the various Web pages

Web-Based E-Mail (2/2)

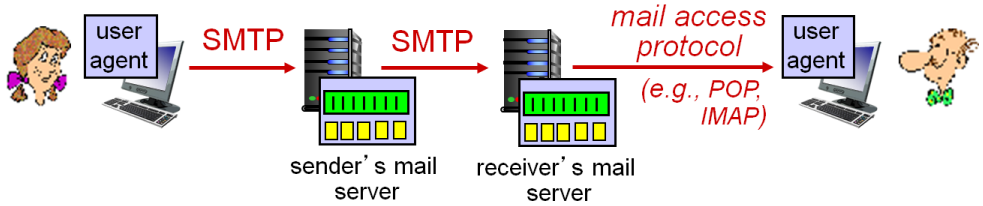
- Sending e-mail
  - User types the text into a form and submits to the server
  - "POST" request to *upload* data to the server
- ***Server uses SMTP to deliver message to other servers***



59

Duke UNIVERSITY

Mail Access Protocols: Key Points To Remember



- POP: Post Office Protocol: authorization, download
- IMAP: Internet Mail Access Protocol: more features, including manipulation of stored messages on server
- HTTP: gmail, Hotmail, Yahoo! Mail, etc.

60

Duke UNIVERSITY

Lecture Summary

- Application protocols
- Web and HTTP
- Electronic Mail
  - SMTP
  - Mail access protocols

61

Duke UNIVERSITY

61

62

Duke UNIVERSITY

62

Next Lecture

- Network security