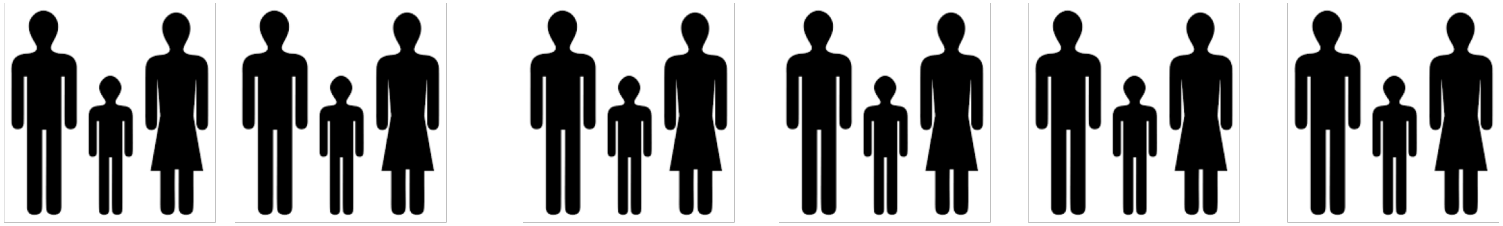
CompSci 356: Computer Network Architectures

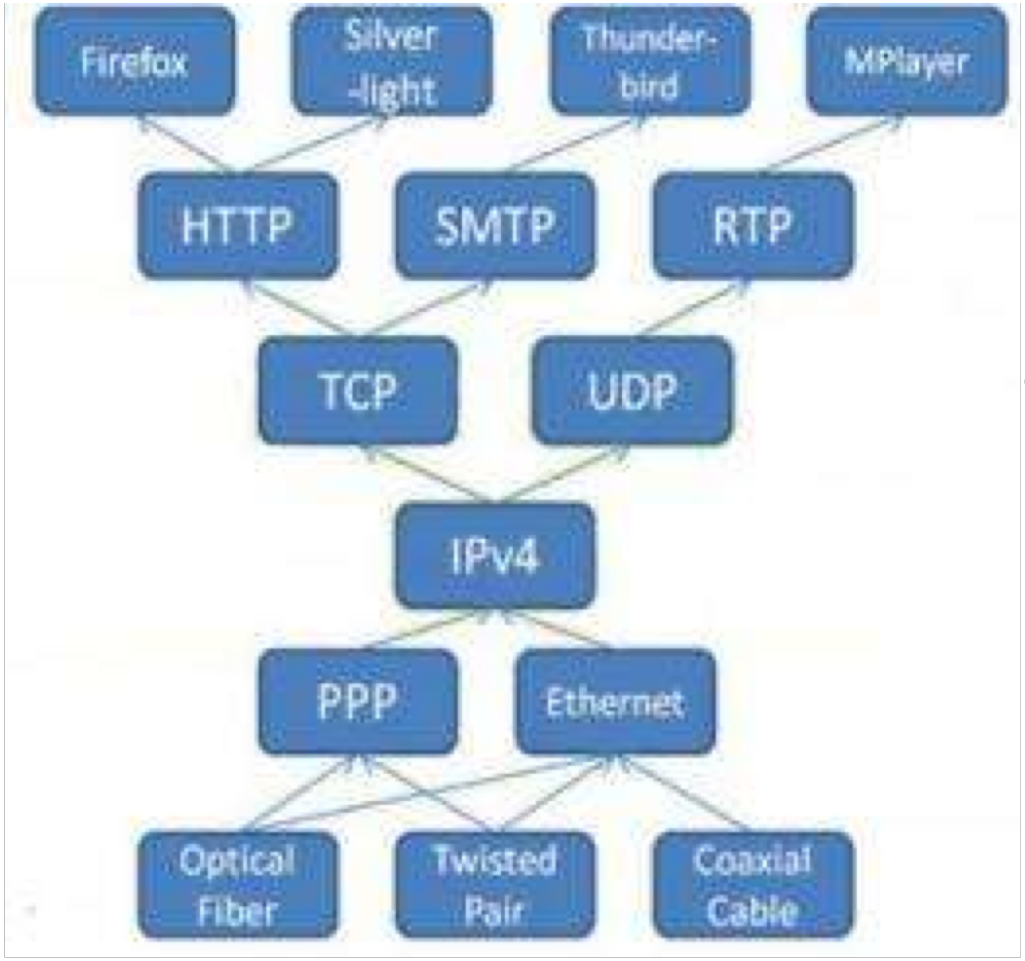
Lecture 25: Application Layer Protocols Chapter 9.1

Xiaowei Yang

xwy@cs.duke.edu

The Internet Architecture





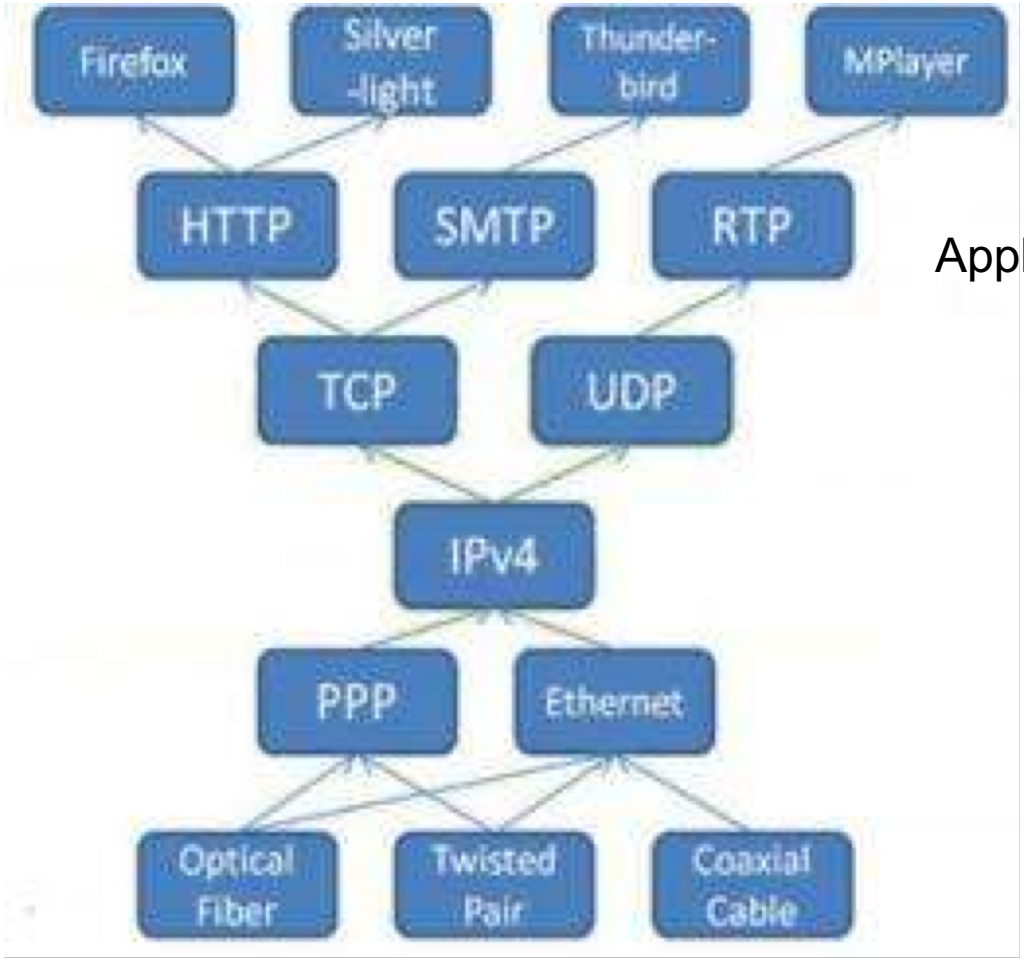
Application layer

Transport layer / Layer 4

Network layer

Link layer

Physical layer



Applications

Application layer protocol

Application vs application protocol

- Application protocol: ”specs” of a particular application.
- Applications implementing the same protocol can interact with each other despite different implementations
- Not all network applications have open network protocols

Anatomy of an Application protocol

- 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

Anatomy of a content-rich application-layer protocol

1. A companion protocol that specifies the data format

2. A protocol that describes the interactions between two end systems

- Examples
  - HTTP and HTML
  - SMTP and RFC 2822 (Internet Text Messages), and Multipurpose Internet Mail Extensions (MIME)

- Simpler protocols may specify both data format and interactions in one protocol
  - Eg. DNS
    - Request/response
    - The record has a simple format
  - Telnet
  - FTP

Sample application protocols: HTTP

- Client sends a request with method, URL, and meta-data
- Server applies the request to the resource and returns the response, including meta-data
- Single TCP connection for control and data

Telnet

- Client simply relays user keystrokes to the server
- Server simply relays any output to the client
- TCP connection persists for duration of the login session
- Network Virtual Terminal (NVT) format for transmitting ASCII data, and control information (e.g., End-of-Line delimiter)

FTP

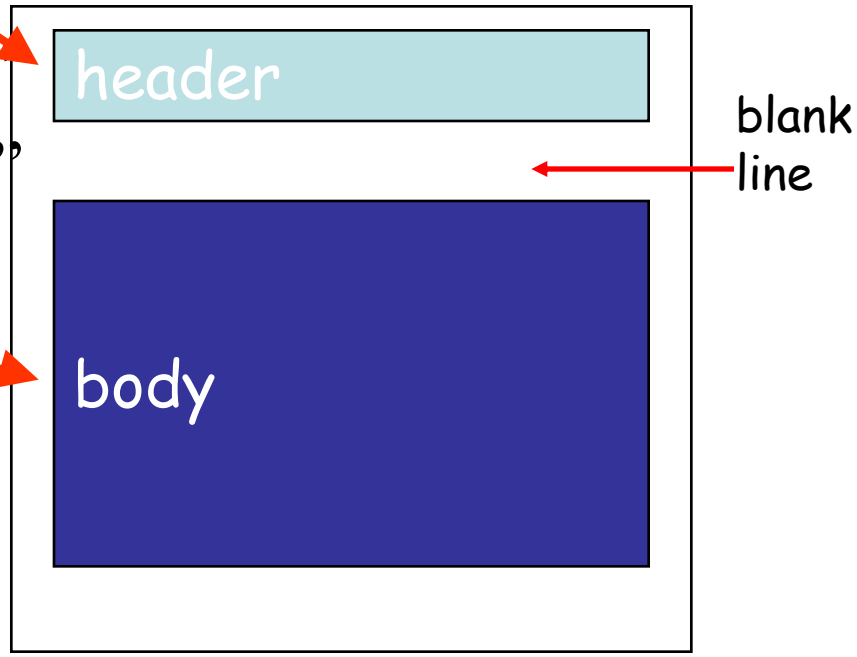
- Client connects to remote machine, “logs in”, and issues commands for transferring files to/from the account
- Server responds to commands and transfers files
- Separate TCP connections for control and data
- Control connection uses the same NVT format as Telnet

Electronic Mail

A case study to show the various complicated matters involved in designing an application protocol

E-Mail Message (RFC 822)

- 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
- Header
  - Lines with "type: value"
  - "To: xwy@cs.duke.edu"
  - "Subject: Hello!"
- Body
  - The text message
  - No particular structure or meaning



Limitation: Sending Non-Text Data

- E-mail body is 7-bit U.S. ASCII
  - What about non-English text?
  - What about binary files (e.g., images and executables)?

- Solution: convert non-ASCII data to ASCII
  - Base64 encoding: map each group of three bytes into four printable U.S.-ASCII characters
  - Uuencode (Unix-to-Unix Encoding) was widely used
    - Output an encoded text file
    - Begin and ending line shows the encoding algorithm

```
begin-base64 644 cat.txt
#AC/0

====
```

```
begin 644 cat.txt
MT,\X

end
```

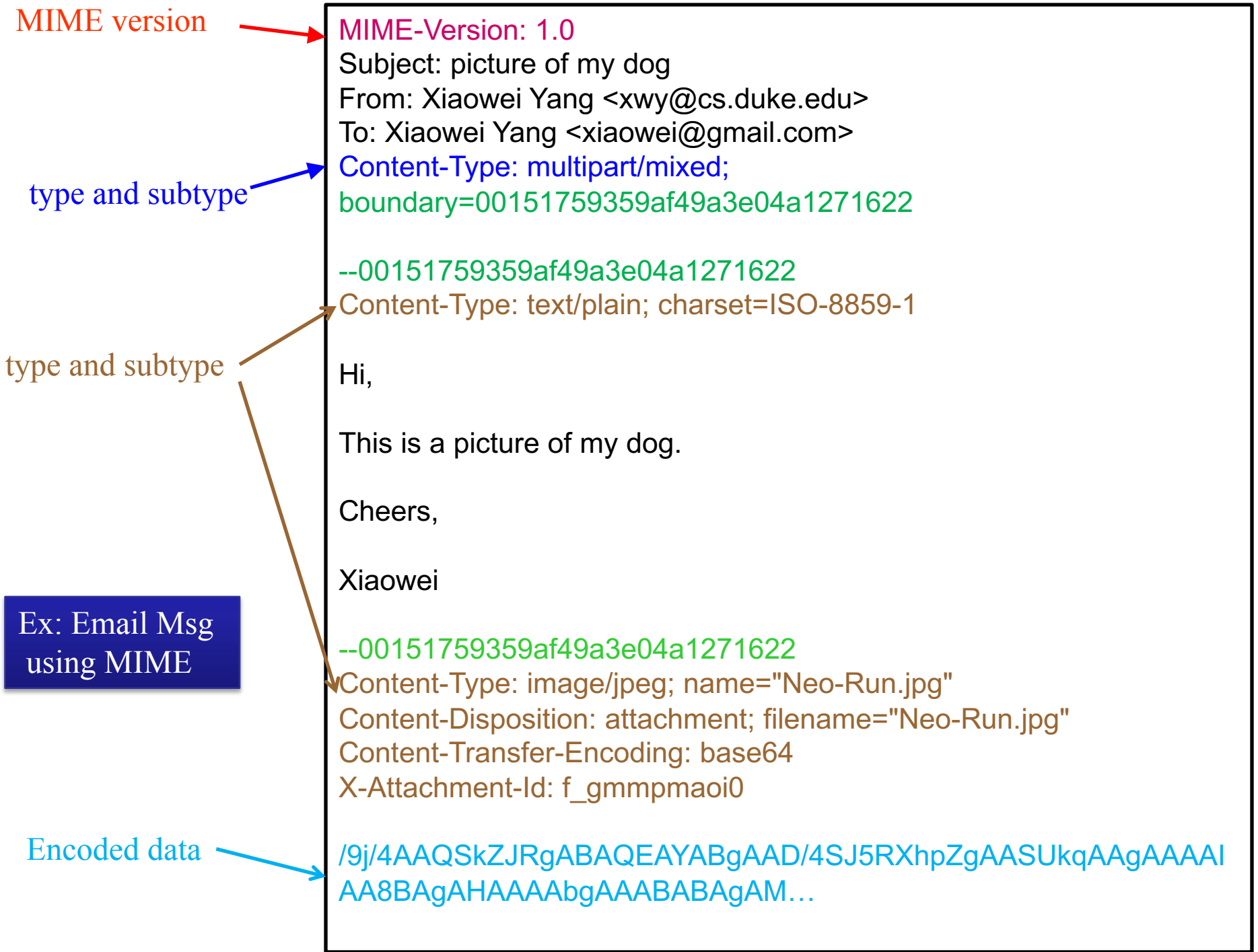
  - 1st: uuencode –m cat.txt < lecture.ppt; -m: MIME Base64
  - 2nd: uuencode cat.txt < lecture.ppt.; historical algorithm
  - Uudecode produces lecture.ppt

Limitation: Sending Multiple Items

- Users often want to send multiple pieces of data
  - Multiple images, powerpoint files, or e-mail messages
  - Yet, e-mail body is a single, uninterpreted data chunk
- Example: e-mail digests
  - Encapsulating several e-mail messages into one aggregate message (i.e., a digest)
  - Commonly used on high-volume mailing lists
- Conventions arose for how to delimit the parts
  - E.g., well-known separator strings between the parts
  - Yet, having a standard way to handle this is better

Multipurpose Internet Mail Extensions

- Additional headers to describe the message body
  - MIME-Version: the version of MIME being used
  - Content-Type: the type of data contained in the message
  - Content-Transfer-Encoding: how the data are encoded
- Definitions for a set of content types and subtypes
  - E.g., image with subtypes gif and jpeg
  - E.g., text with subtypes plain, html, and richtext
  - E.g., application with subtypes postscript and msword
  - E.g., multipart for messages with multiple data types
- A way to encode the data in ASCII format
  - Base64 encoding, as in uuencode/uudecode


MIME version
type and subtype
type and subtype
Ex: Email Msg using MIME
Encoded data
MIME-Version: 1.0
Subject: picture of my dog
From: Xiaowei Yang <xwy@cs.duke.edu>
To: Xiaowei Yang <xiaowei@gmail.com>
Content-Type: multipart/mixed;
boundary=00151759359af49a3e04a1271622
--00151759359af49a3e04a1271622
Content-Type: text/plain; charset=ISO-8859-1
Hi,
This is a picture of my dog.
Cheers,
Xiaowei
--00151759359af49a3e04a1271622
Content-Type: image/jpeg; name="Neo-Run.jpg"
Content-Disposition: attachment; filename="Neo-Run.jpg"
Content-Transfer-Encoding: base64
X-Attachment-Id: f_gmmpmaoi0
/9j/4AAQSkZJRgABAQEAYABgAAD/4SJ5RXhpZgAASUkqAAgAAAAI
AA8BAgAHAAAAbgAAABABAgAM…

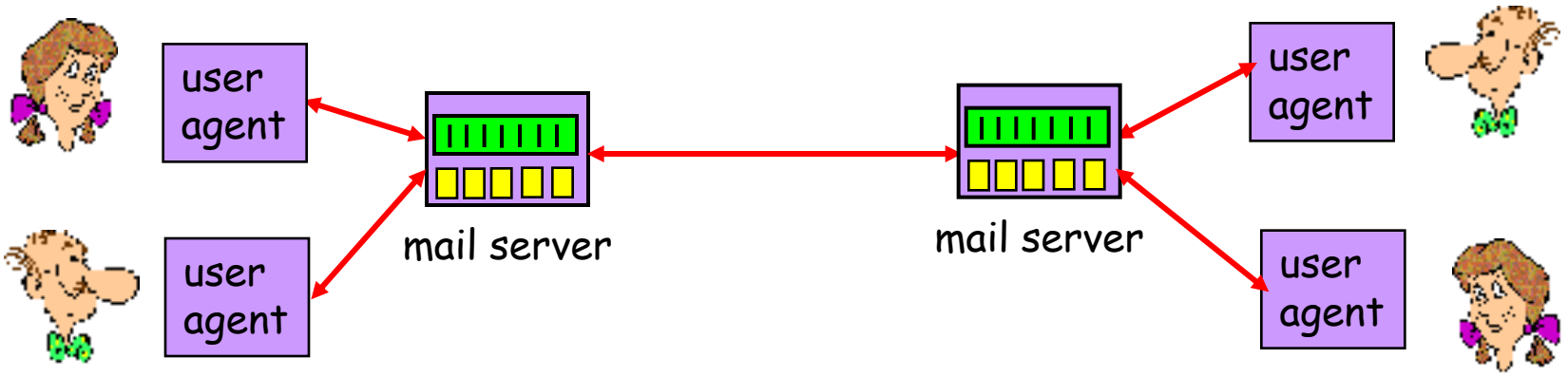
E-Mail Addresses

- Components of an e-mail address
  - Local mailbox (e.g., xwy or bob.flower)
  - Domain name (e.g., cs.duke.edu)

- Domain name is not necessarily the mail server
  - Mail server may have longer/cryptic name
    - E.g., cs.duke.edu vs. one.cs.duke.edu
  - Multiple servers may exist to tolerate failures
    - E.g., duke.edu vs. authdns{1,2,3,4}.netcom.duke.edu

- Identifying the mail server for a domain
  - DNS query asking for MX records (Mail eXchange)
    - E.g., nslookup –q=mx duke.edu, dig MX duke.edu
  - Then, a regular DNS query to learn the IP address

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

SMTP Store-and-Forward Protocol

- Messages sent through a series of servers
  - A server stores incoming messages in a queue
  - … to await attempts to transmit them to the next hop
- If the next hop is not reachable
  - The server stores the message and tries again later
- Each hop adds its identity to the message
  - By adding a "Received" header with its identity
  - Helpful for diagnosing problems with e-mail

Multiple Server Hops

- Typically at least two mail servers
  - Sending and receiving sides
- May be more
  - Separate servers for key functions
    - Spam filtering
    - Virus scanning
  - Servers that redirect the message
    - From xwy@cs.duke.edu to xiaowei@gmail.com
    - Messages to cs.duke.edu go through extra hops
  - Electronic mailing lists
    - Mail delivered to the mailing list's server
    - … and then the list is expanded to each recipient

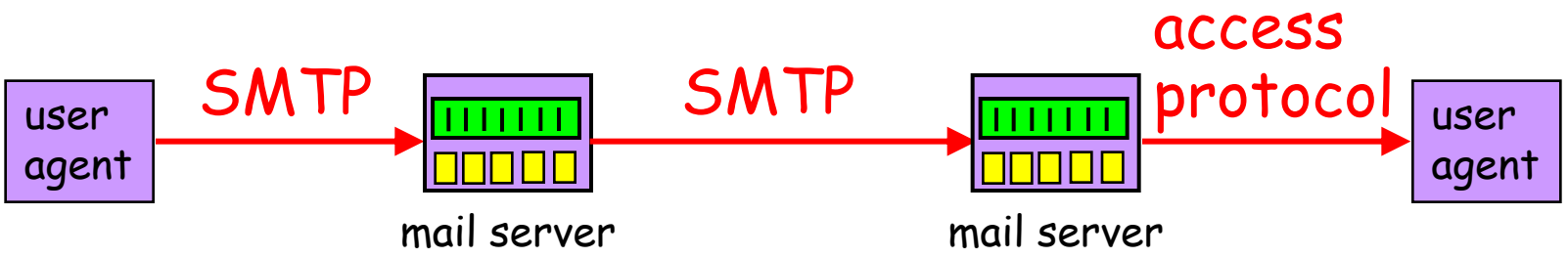
Sample Email

- Show one from personal mail box
- Anti-spoofing methods
  - SPF: Sender Policy Framework
  - DKIM: DomainKeys Identified Mail

Electronic Mailing Lists

- Community of users reachable by one address
  - Allows groups of people to receive the messages
- Exploders
  - Explode a single e-mail message into multiple messages
  - One copy of the message per recipient
- Handling bounced messages
  - Mail may bounce for several reasons
  - E.g., recipient mailbox does not exist; resource limits
- E-mail digests
  - Sending a group of mailing-list messages at once
  - Messages delimited by boundary strings
  - … or transmitted using multiple/digest format

Simple Mail Transfer Protocol



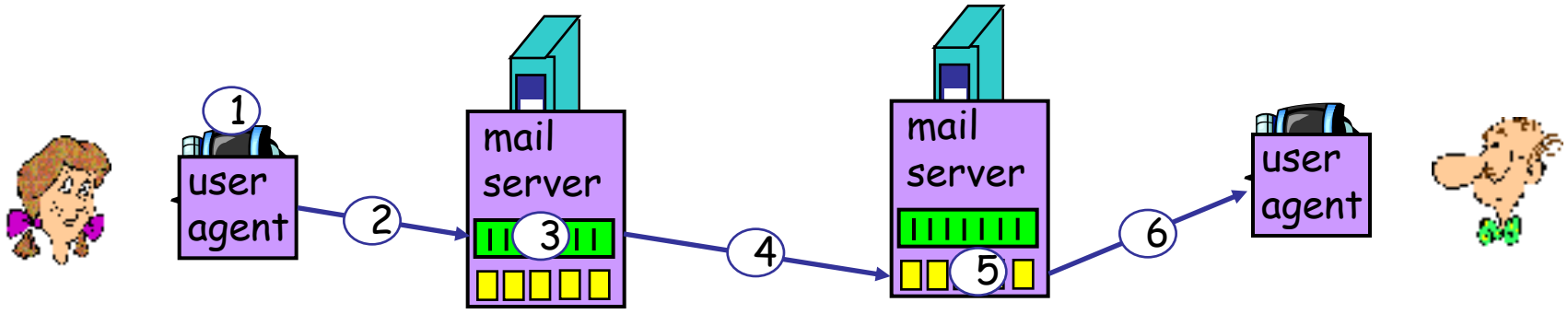
- Client-server protocol
  - Client is the sending mail server
  - Server is the receiving mail server
- Reliable data transfer
  - Built on top of TCP (on port 25)
- Push protocol
  - Sending server pushes the file to the receiving server
  - … rather than waiting for the receiver to request it

Simple Mail Transfer Protocol (Cont.)

- Command/response interaction
  - Commands: ASCII text
  - Response: three-digit status code and phrase
- Synchronous
  - Sender awaits response from a command
  - … before issuing the next command
  - Though pipelining of commands was added later
- Three phases of transfer
  - Handshaking (greeting)
  - Transfer of messages
  - Closure

Scenario: Alice Sends Message to Bob

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
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



Sample SMTP interaction

- Show an example via telnet
- Use dig to find the mail server
- telnet smtp.duke.edu 25

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 For Yourself

- Running SMTP
  - Run “telnet servername 25” at UNIX prompt
  - See 220 reply from server
  - Enter HELO, MAIL FROM, RCPT TO, DATA commands
- In the old days one can spoof
  - Very easy
  - Just forge the argument of the “FROM” command
  - … leading to all sorts of problems with spam
- Spammers can be even more clever
  - E.g., using open SMTP servers to send e-mail
  - E.g., forging the “Received” header

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 from when the message was sent
  - With a way to view the message and reply
  - With a way to organize and store the messages
- In the old days…
  - User logged on to the machine where mail was delivered
  - Users received e-mail on their main work machine

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)

Post Office Protocol (POP)

- POP goals
  - Support users with intermittent network connectivity
  - Allow them to retrieve e-mail messages when connected
  - … and view/manipulate messages when disconnected
- Typical user-agent interaction with a POP server
  - Connect to the server
  - 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
- User agent still uses SMTP to send messages

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

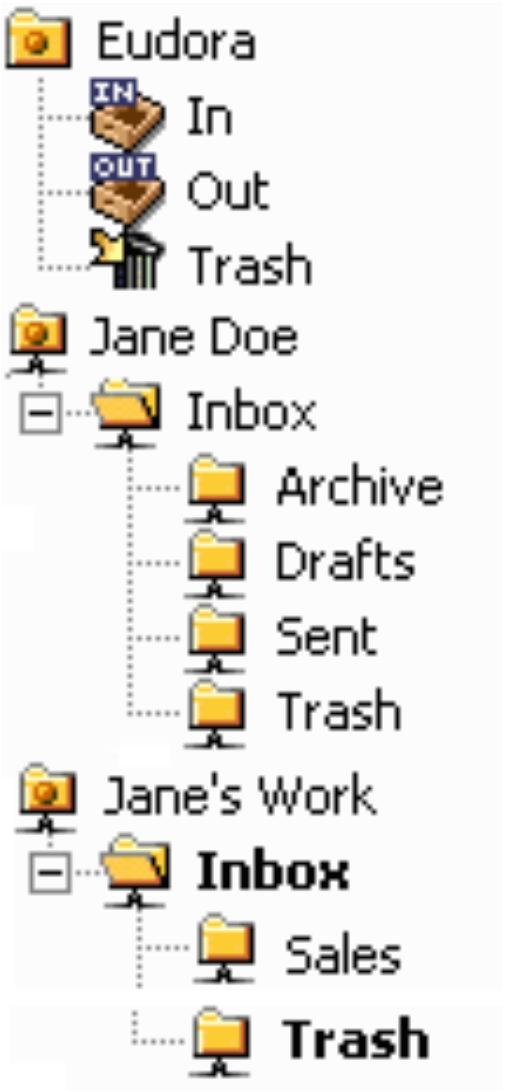
```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

Limitations of POP

- Does not handle multiple mailboxes easily
  - Designed to put user's incoming e-mail in one folder
- Not designed to keep messages on the server
  - Instead, designed to download messages to the client
- Poor handling of multiple-client access to mailbox
  - Increasingly important as users have home PC, work PC, laptop, cyber café computer, friend's machine, etc.
- High network bandwidth overhead
  - Transfers all of the e-mail messages, often well before they are read (and they might not be read at all!)

Interactive Mail Access Protocol (IMAP)

- Supports connected and disconnected operation
  - Users can download message contents on demand
- Multiple clients can connect to mailbox at once
  - Detects changes made to the mailbox by other clients
  - Server keeps state about message (e.g., read, replied to)
- Access to MIME parts of messages & partial fetch
  - Clients can retrieve individual parts separately
  - E.g., text of a message without downloading attachments
- Multiple mailboxes on the server
  - Client can create, rename, and delete mailboxes
  - Client can move messages from one folder to another
- Server-side searches
  - Search on server before downloading messages


Eudora
IN
In
OUT
Out
Trash
Jane Doe
Inbox
Archive
Drafts
Sent
Trash
Jane's Work
Inbox
Sales
Trash

Web-Based E-Mail

- User agent is an ordinary Web browser
  - User communicates with server via HTTP
  - E.g., Gmail, Yahoo mail, and Hotmail
- 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
- 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
- Easy to send anonymous e-mail (e.g., spam)

Conclusion

- Application-layer protocols
  - Applications vs. application-layer protocols
  - Tailoring the protocol to the application

- Electronic mail
  - E-mail messages, and multipurpose Internet Mail extensions (MIME)
  - E-mail addresses, and role of DNS
  - E-mail servers and user agents

- Electronic mail protocols
  - Transferring e-mail messages between servers
    - Simple Mail Transfer Protocol (SMTP)
  - Retrieving e-mail messages (POP, IMAP, and HTTP)
    - Client-side protocols