The Application Layer: SMTP

CS 352, Lecture 5, Spring 2020

http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Course announcements

- Project 1 will be released on Friday
  - Find a partner if you don’t have one
- Quiz 1 completed yesterday
- Quiz 2 will go up on Friday
  - Due next Tuesday

Review of concepts

- Application-layer protocols: DNS, HTTP
- HyperText Transfer Protocol:
  - Client-server model: requests and responses
- Request methods: GET, POST, HEAD, PUT, DELETE
  - And response codes
- Persistent vs. non-persistent HTTP connection
- Remembering HTTP users via cookies
- Common features of DNS, HTTP:
  - “Plain” text
  - Command line tools to directly speak the protocol
  - Caching

Caching in HTTP

Web caches

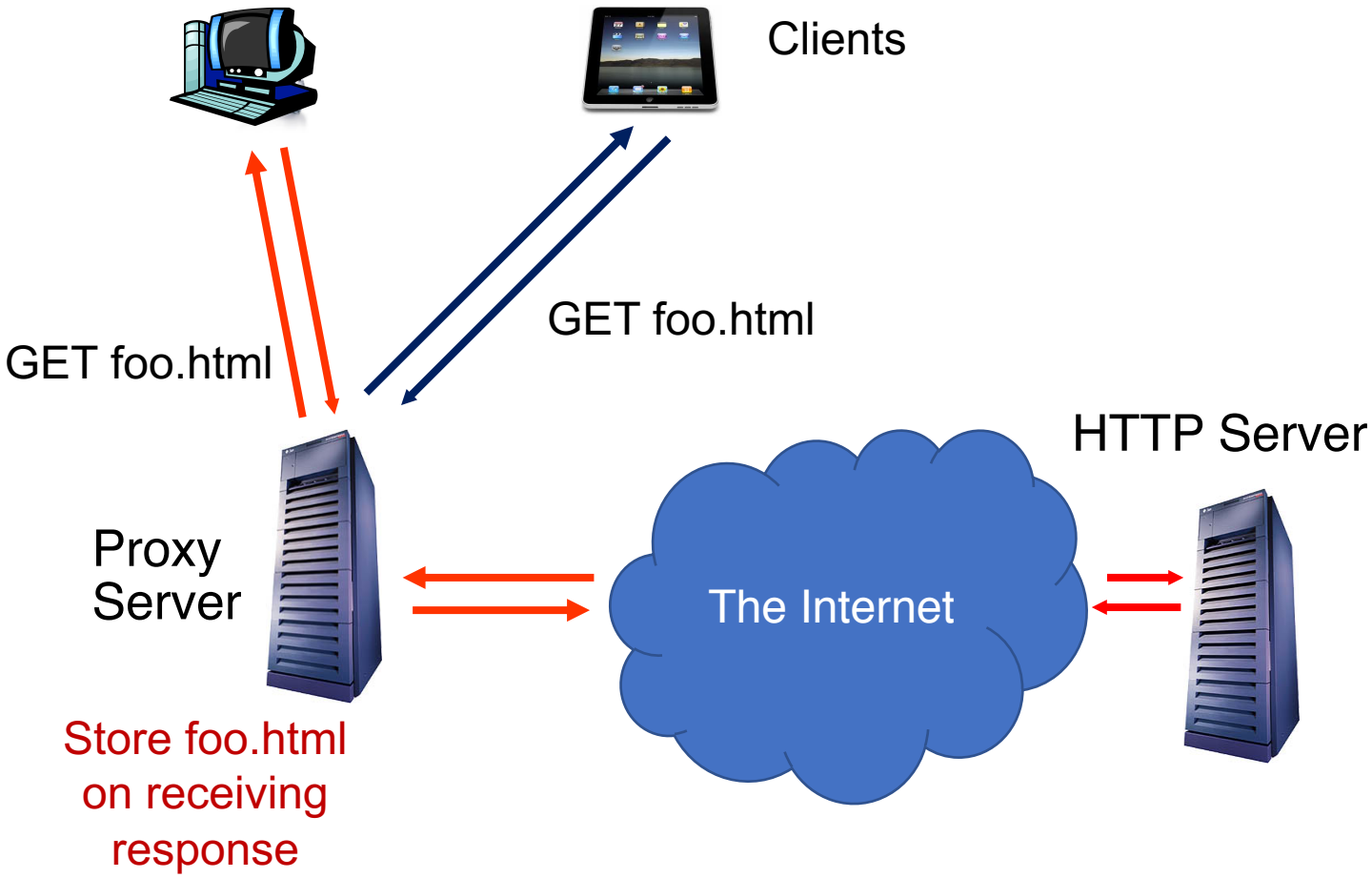
Web caches: Machines that remember web responses for a network

Why cache web responses?

- Reduce response time for client requests
- Reduce traffic on an institution's access link

Caches can be implemented in the form of a proxy server

Web caching using a proxy server

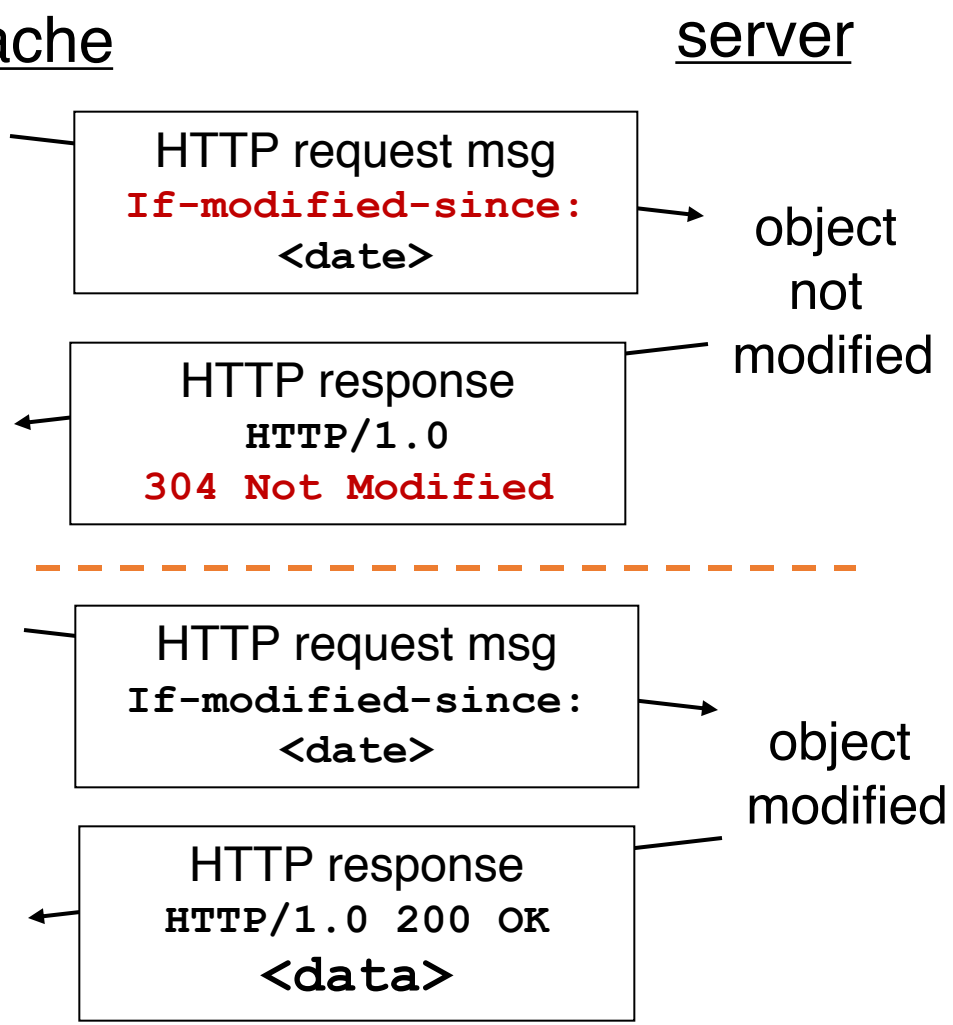


- You can configure a HTTP proxy on your laptop's network settings.
  - If you do, your browser sends all HTTP requests to the proxy (cache).
- Hit: cache returns object
- Miss:
- Proxy requests object from original HTTP server (called origin server)
  - Proxy caches object locally
  - Proxy returns object to client

Web Caches: how does it look on HTTP?

- Conditional GET guarantees cache content is up-to-date while still saves traffic and response time whenever possible

- Date in the cache's request is the last time the server provided in its response header "last modified"



Content Distribution Networks (CDN)

A global network of web caches

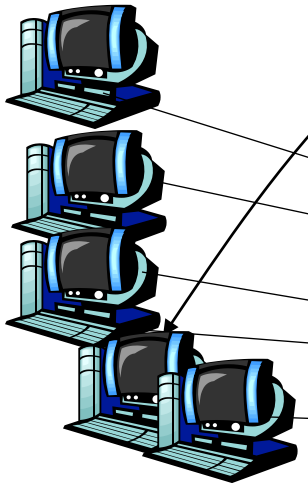
- Provisioned by ISPs and network operators
- Or content providers, like Netflix, Google, …

Uses

- Improve response time to user for a service
- Reduce bandwidth requirements
  - Both on content provider and on a network (e.g., Rutgers)
- Reduce $$ to provision and maintain origin servers

Without CDN

DOMAIN NAME	IP ADDRESS
www.google.com	98.138.253.109
cs.rutgers.edu	128.6.4.2
www.yahoo.com	74.125.225.243
www.princeton.edu	128.112.132.86



Cluster with Google's origin servers

98.138.253.109

- Huge bandwidth requirements
- Large propagation delays to reach users
- So, distribute content to geographically distributed cache servers
- Often, use DNS to redirect request to users to copies of content

CDN terms

- Origin server
  - Server that holds the authoritative copy of the content
- CDN server
  - A replica server owned by the CDN provider
  - We called this proxy in our earlier example
- CDN name server
  - A DNS server used for redirection
- Client

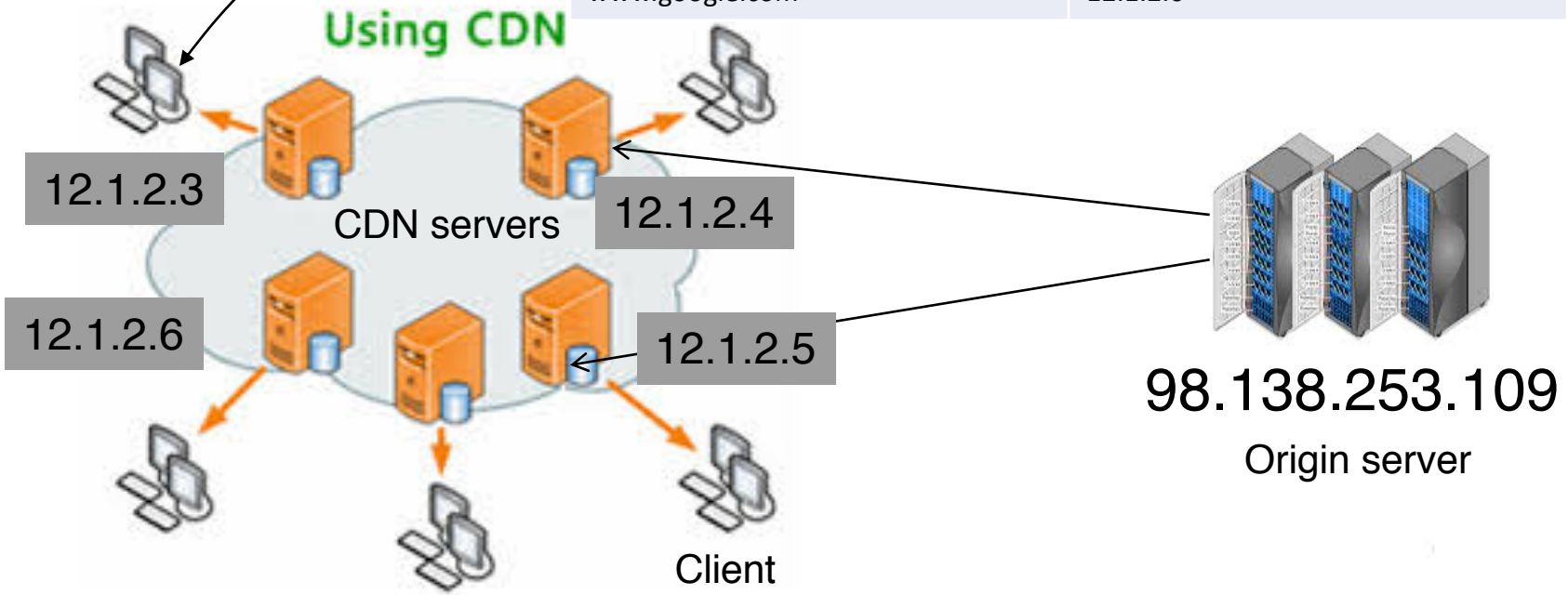
With CDN

Scale a service through indirection to CDN name server.

DOMAIN NAME	IP ADDRESS
www.google.com	124.8.9.8 (NS of CDN)
cs.rutgers.edu	128.6.4.2
www.yahoo.com	74.125.225.243
www.princeton.edu	128.112.132.86

CDN Name Server (124.8.9.8)

DOMAIN NAME	IP ADDRESS
www.google.com	12.1.2.3
www.google.com	12.1.2.4
www.google.com	12.1.2.5
www.google.com	12.1.2.6



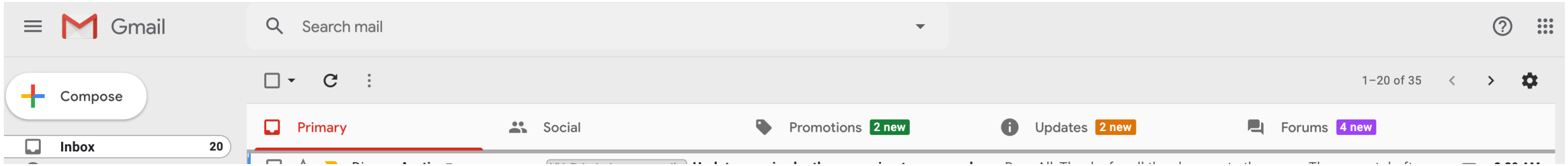
Typically, custom logic to map one domain name to one of many IP addresses

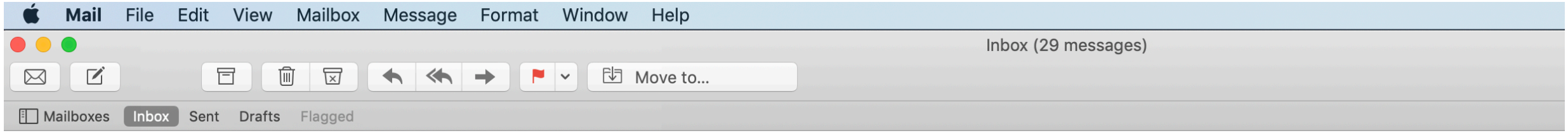
Policies may depend on location of requesting client, load at the different origin servers, apart from other things.

SMTP

Simple Mail Transfer Protocol

What we’re familiar with

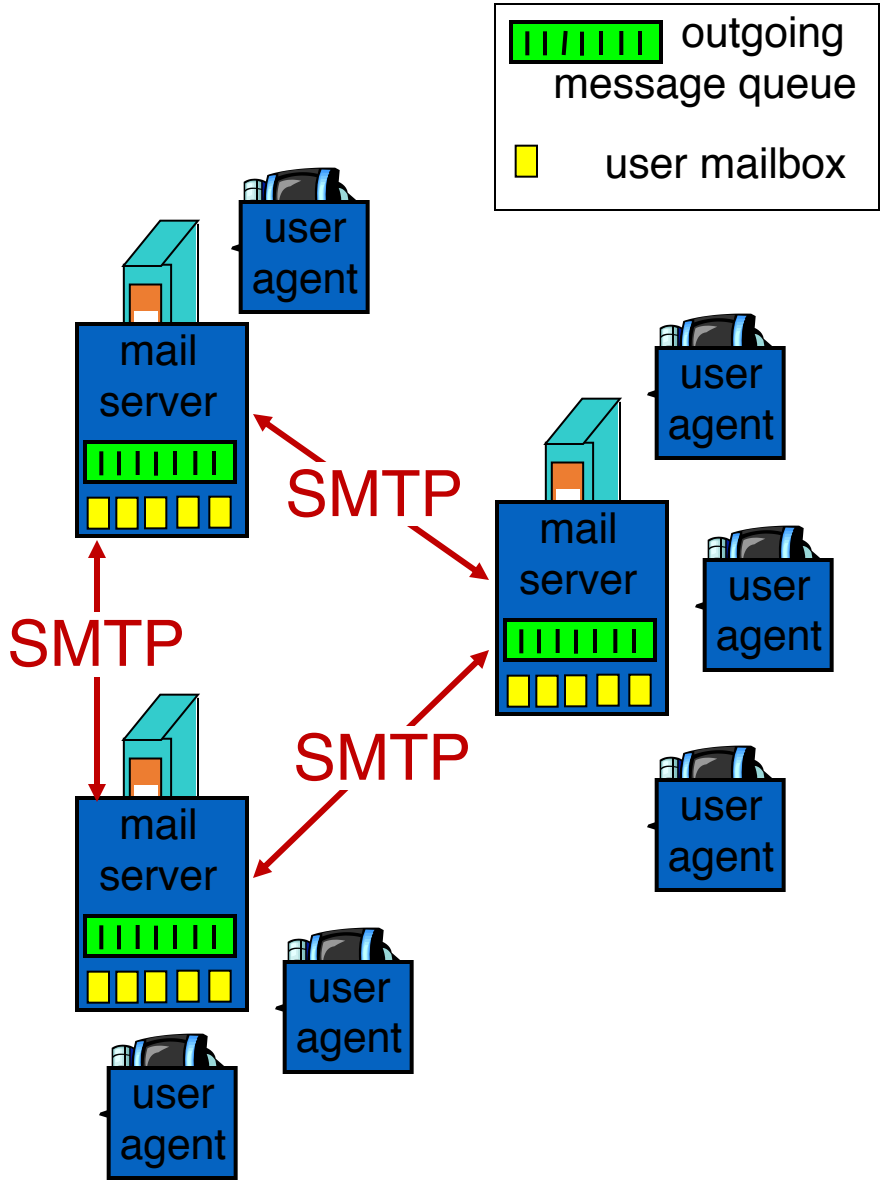




Electronic Mail

Three major components:

1. User agents
   - a.k.a. “mail reader”
   - e.g., Applemail, Outlook
   - Web-based user agents (ex: gmail)



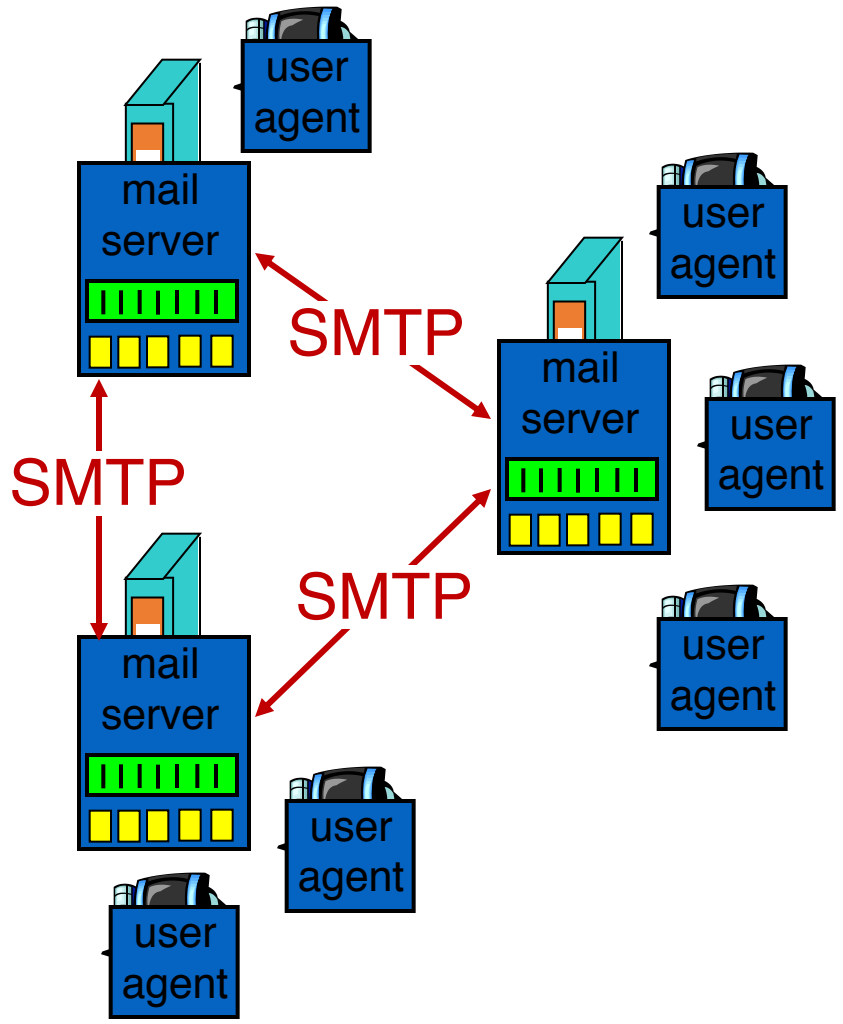
Electronic Mail: Mail servers

2. Mail Servers

- Mailbox contains incoming messages for user
- Message queue of outgoing (to be sent) mail messages
- Sender mail server makes connection to Receiver mail server
  - IP address, port 25

3. SMTP protocol

- Used to send messages
- Client: sending user agent or sending mail server
- server: receiving mail server



Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and "to" `bob@someschool.edu`

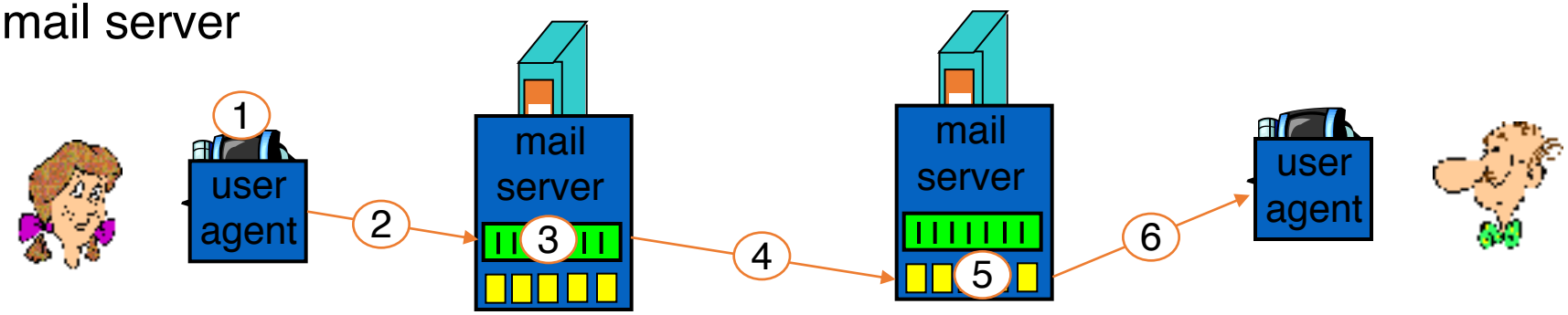
2) Alice's UA sends message to her mail server; message placed in outgoing message queue

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 incoming mailbox

6) Bob invokes his user agent to read message



Sample SMTP interaction

220 hill.com SMTP service ready

HELO town.com

250 hill.com Hello town.com, pleased to meet you

MAIL FROM: <jack@town.com>

250 <jack@town.com>… Sender ok

RCPT TO: <jill@hill.com>

250 <jill@hill.com>… Recipient ok

DATA

354 Enter mail, end with "." on a line by itself

Jill, I'm not feeling up to hiking today. Will you please fetch me a pail of water?

.

250 message accepted

QUIT

221 hill.com closing connection

MAIL command response codes

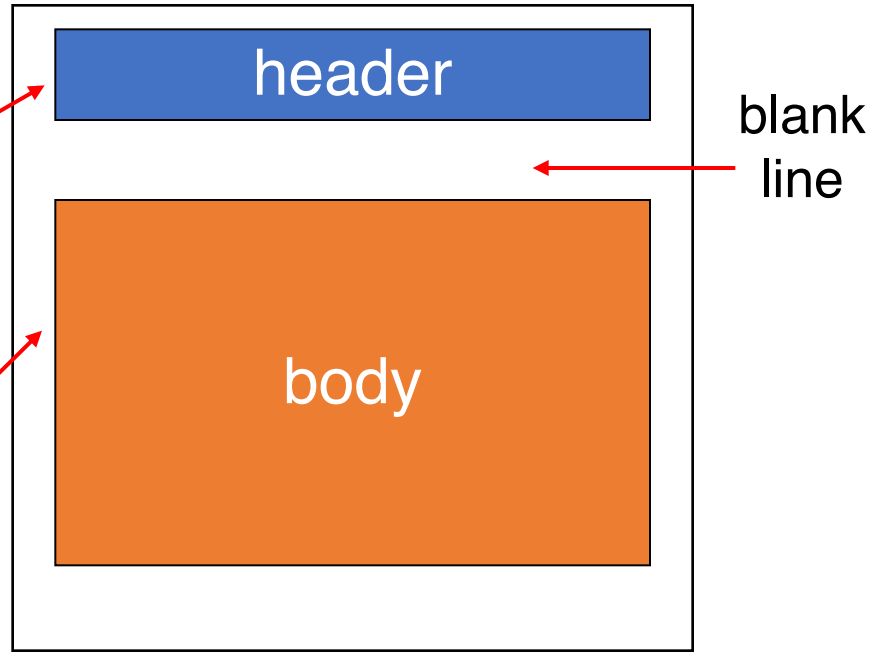
Table 23.2 *Responses*

Code	*Description*
Positive Completion Reply	
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
Positive Intermediate Reply	
354	Start mail input
Transient Negative Completion Reply	
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
Permanent Negative Completion Reply	
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Mail message (stored on server) format

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:



- header lines, e.g.,
  - To:
  - From:
  - Subject:

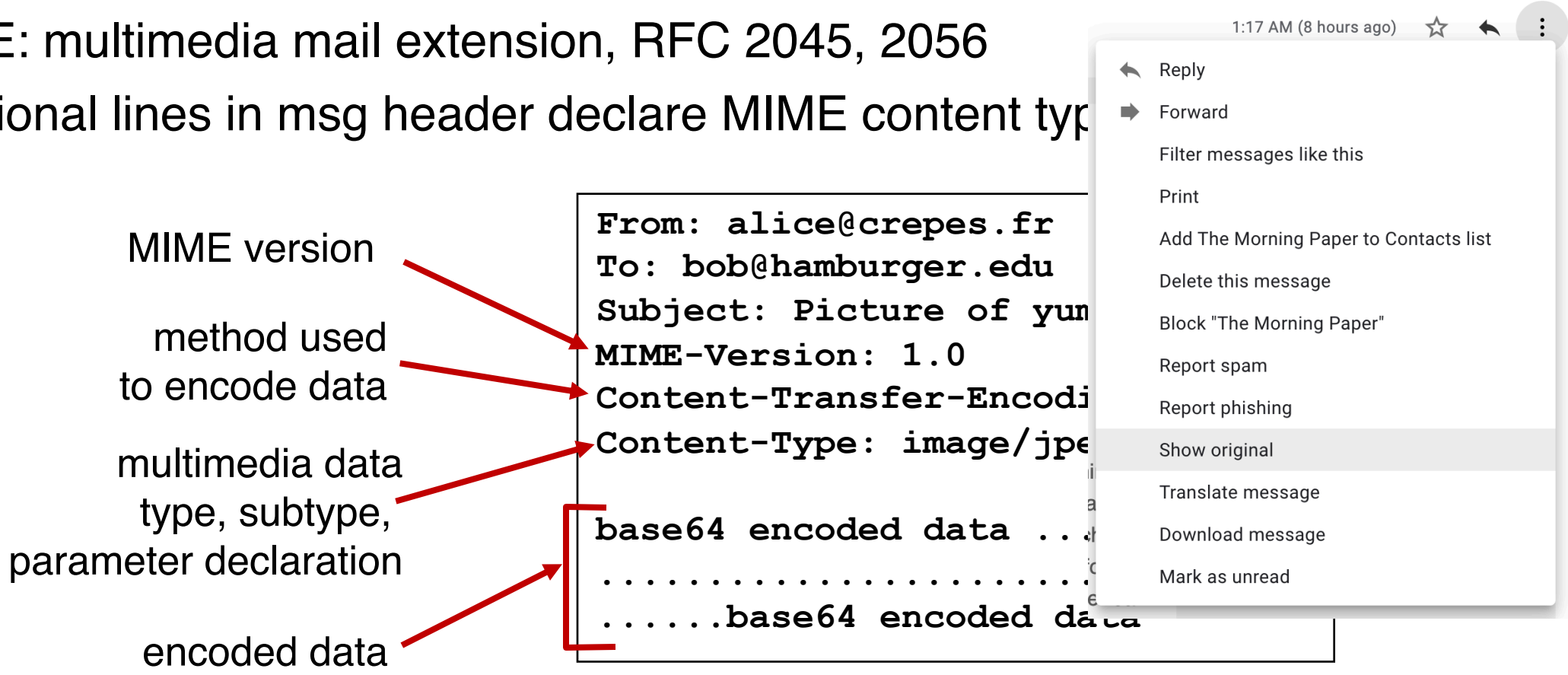
  different from SMTP commands!

  (these would still be under “DATA”)

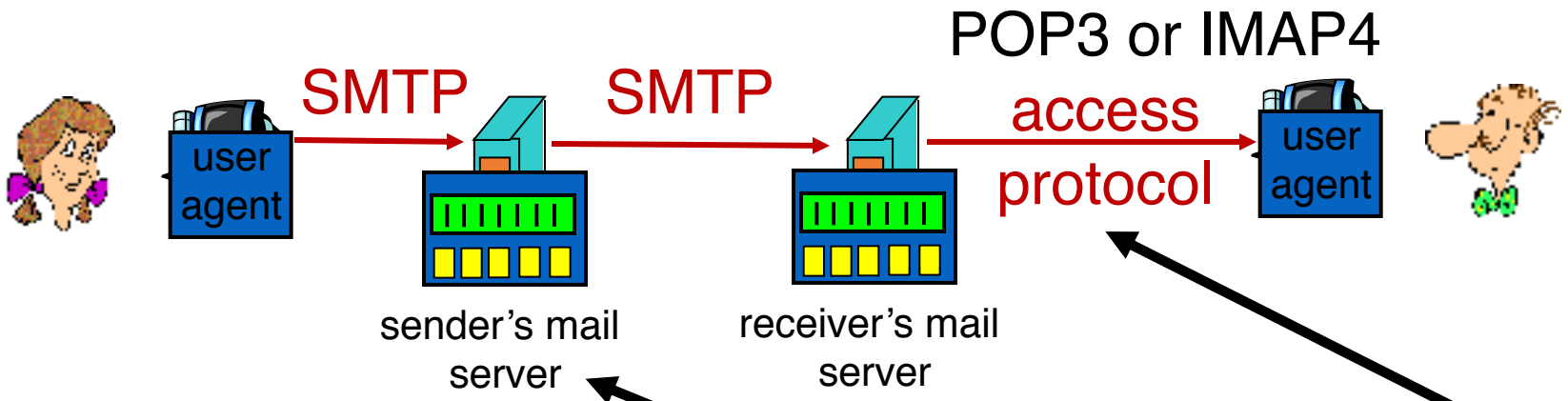
- body
  - the “message”, ASCII characters only

Message format: multimedia extensions

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



Mail access protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - Client connects to POP3 server on TCP port 110
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Client connects to TCP port 143
  - HTTP: gmail, Yahoo! Mail, etc.

Why not use SMTP here?

Why do we need a sender side mail server?

POP vs IMAP

- POP3
- Stateless server
- UA-heavy processing
- UA retrieves email from server, then typically deleted from server
- Latest changes are at the UA
- Simple protocol (list, retr, del within a POP session)

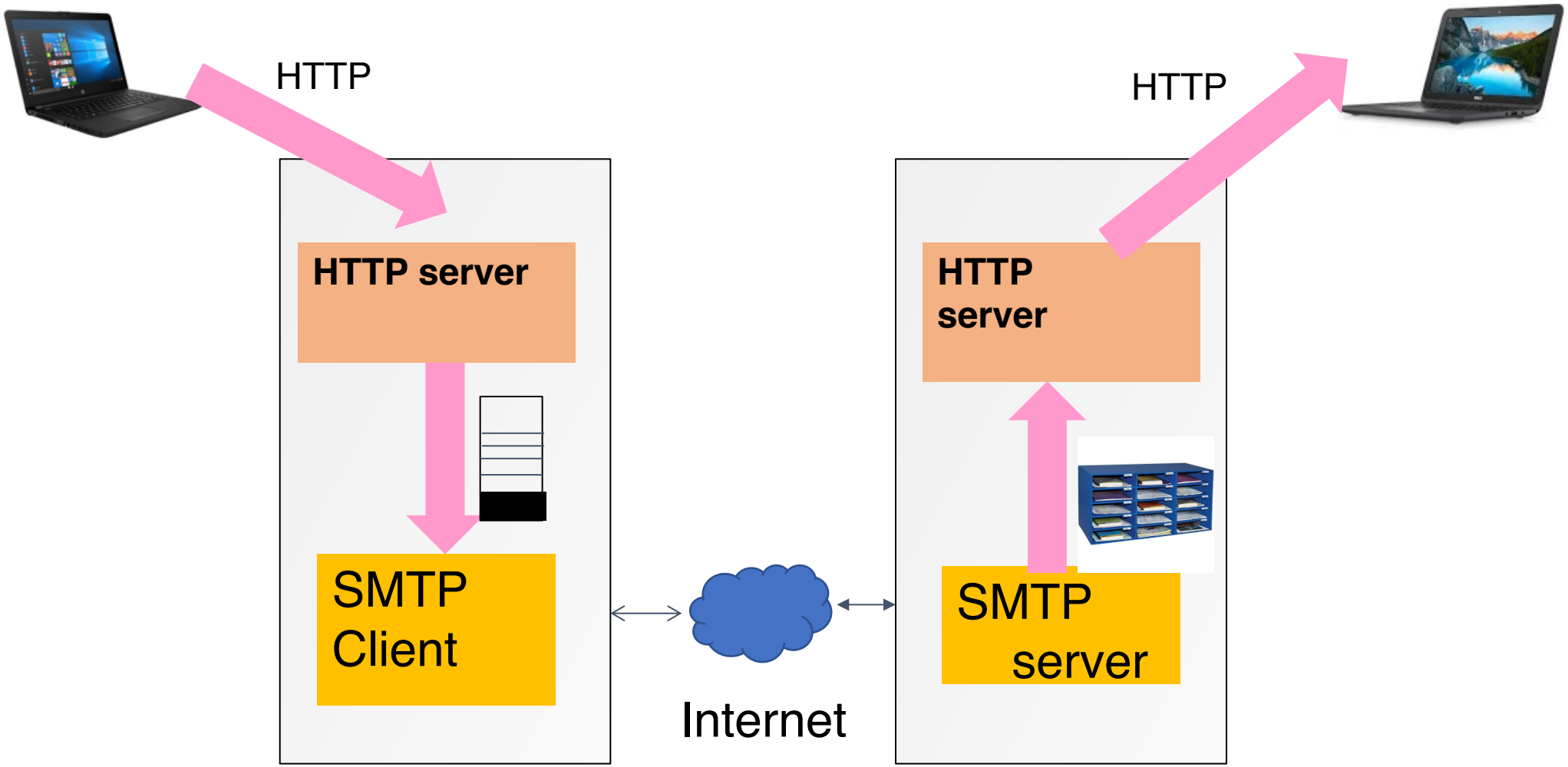
- IMAP4
- Stateful server
- UA and server processing
- Server sees folders, etc. which are visible to UAs
- Changes visible at the server
- Complex protocol

What about web-based email?

- Connect to mail servers via web browser
  - Ex: gmail, outlook, etc.

- Browsers speak HTTP
- Email servers speak SMTP
- Need a bridge to retrieve email using HTTP

Web based email



Comparing SMTP with HTTP

- HTTP: pull
- SMTP: push

- both have ASCII command/response interaction, status codes

- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

- HTTP: can put non-ASCII data directly in response
- SMTP: need ASCII-based encoding

Try an SMTP interaction

```
ngsrinivas@ubuntu18-vbox:~$ nslookup
> set type=MX
> rutgers.edu
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
rutgers.edu     mail exchanger = 10 mx.rutgers.edu.

Authoritative answers can be found from:
>
ngsrinivas@ubuntu18-vbox:~$ telnet mx.rutgers.edu 25
Trying 128.6.68.142...
Connected to mx.rutgers.edu.
Escape character is '^]'.
220 mx.rutgers.edu ESMTP
HELO cs.rutgers.edu
250 annwn11.rutgers.edu
MAIL FROM: <sn624@cs.rutgers.edu>
250 2.1.0 Ok
RCPT TO: <srinivas.narayana@rutgers.edu>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Hello, world!
Goodbye, cruel world.
.
250 2.0.0 Ok: queued as 2B2E5460035
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
ngsrinivas@ubuntu18-vbox:~$ ^C
```

```
[flow:~]$  telnet mx.rutgers.edu 25
Trying 128.6.68.142...
Connected to mx.rutgers.edu.
Escape character is '^]'.
220 mx.rutgers.edu ESMTP
HELO cs.rutgers.edu
250 annwn12.rutgers.edu
MAIL FROM: <sn624@cs.rutgers.edu>
250 2.1.0 Ok
RCPT TO: <srinivas.narayana@rutgers.edu>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: sn624@cs.rutgers.edu
To: srinivas.narayana@rutgers.edu
Subject: A test message

Hello. Bleh bleh bleh.
.
250 2.0.0 Ok: queued as 904AA634015
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

More themes from app-layer protocols

- Separation of concerns. Examples:
  - Content rendering for users (browser, UA) separate from protocol operations (mail server)
  - Reliable mail sending and receiving: mail UA doesn't need to be "always on" to send or receive email reliably
- In-band vs. out-of-band control:
  - In-band: headers determine the actions of all the parties of the protocol
  - There are protocols with out-of-band control, e.g., FTP
- Keep it simple until you really need complexity
  - ASCII-based design; stateless servers. Then introduce:
  - Cookies for HTTP state
  - IMAP for email organization
  - Security extensions
  - Different methods to set up and use underlying connections, etc.