

For today...

sit next to your lab partner

Reading Quiz

(graded for correctness)

Put devices away

Note: Reading quiz slides will have an orange border.

End of Reading Quiz!

Where we are

Application: the application (e.g., DNS, Email)

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1's and 0's/bits across a medium
(copper, the air, fiber)

Today

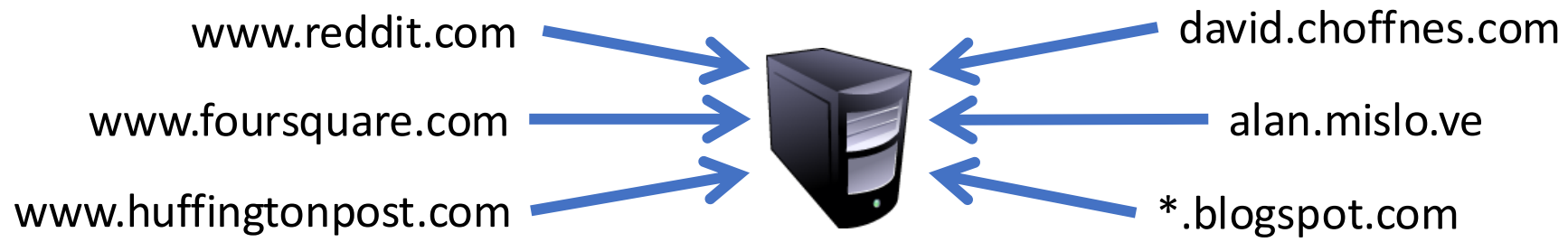
- Finishing up DNS
- Three main parts to email:
 - Mail User Agent
 - Mail Transfer Agent
 - SMTP protocol used to negotiate transfers
- SMTP Protocol
- Mail Access Protocols

DNS as Indirection Service

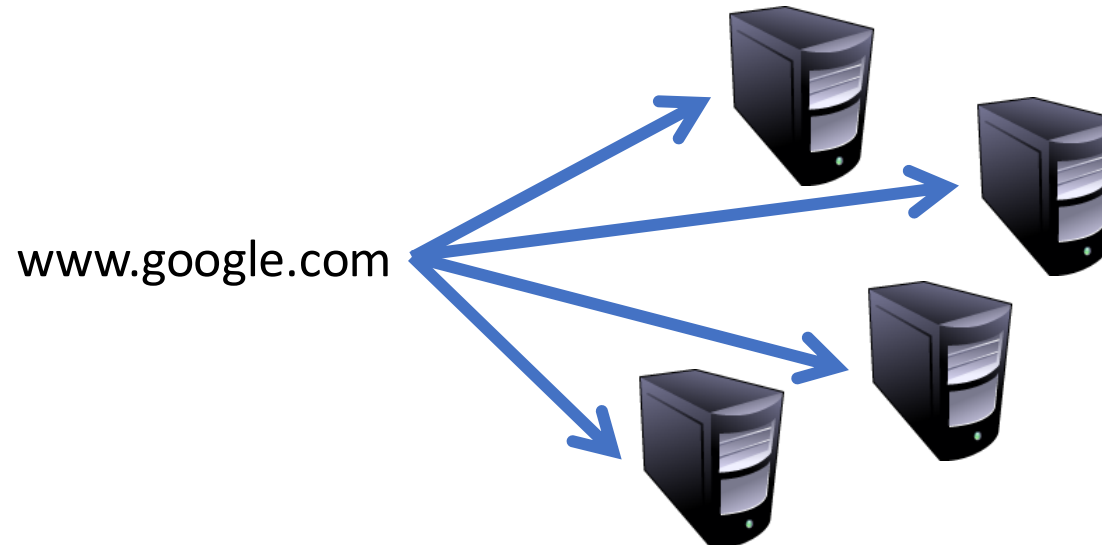
- DNS gives us very powerful capabilities
 - Not only easier for humans to reference machines!
- Changing the IPs of machines becomes trivial
 - e.g. you want to move your web server to a new host
 - Just change the DNS record!

Aliasing and Load Balancing

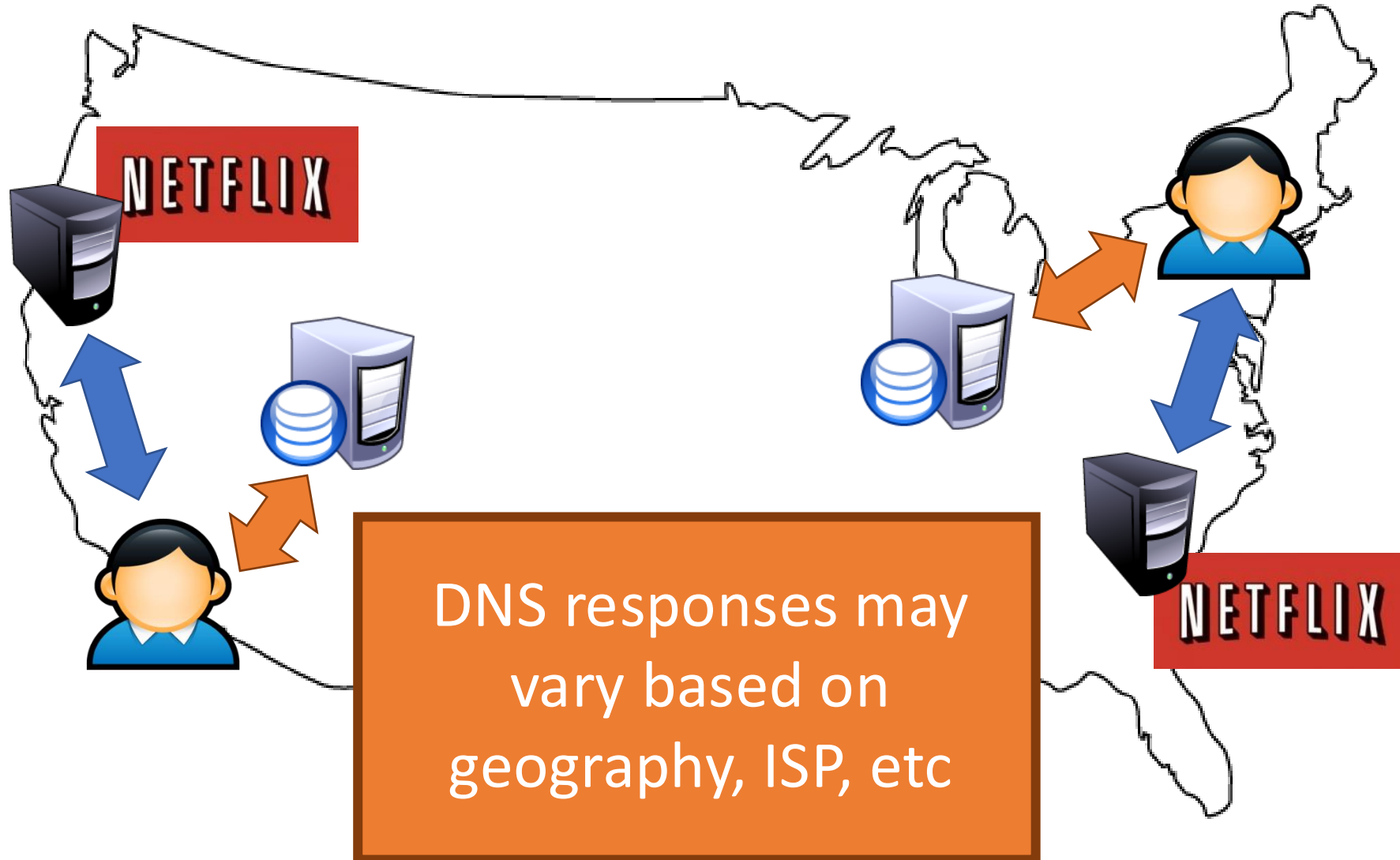
- One machine can have many aliases



- One domain can map to multiple machines



Content Delivery Networks



Inserting (or changing) records

Example: new startup “Network Utopia”

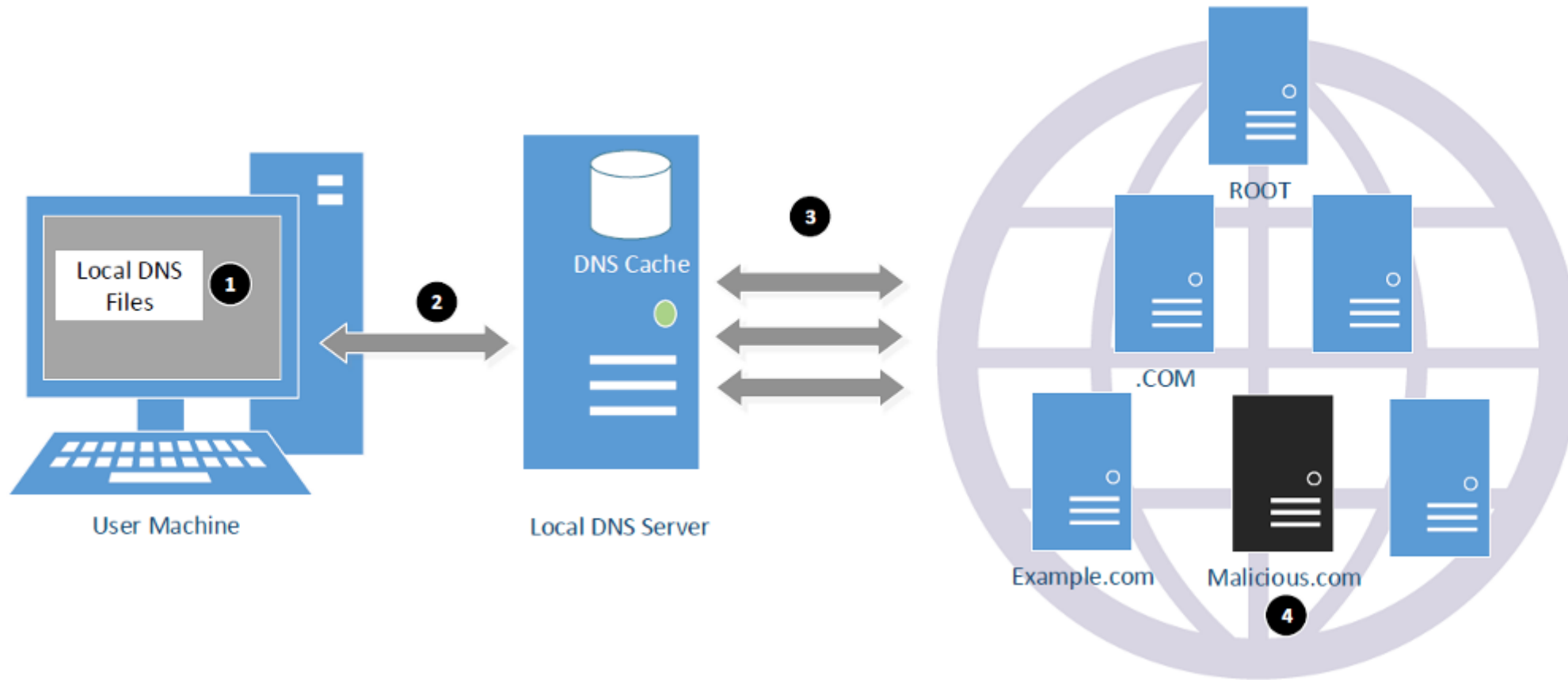
- Step 1: Register networkutopia.com at **DNS registrar**
 - provide names, IP addresses of authoritative name server (primary and secondary)

Inserting (or changing) records

- Example: new startup “Network Utopia”
- Register networkutopia.com at **DNS registrar**
 - provide names, IP addresses of authoritative name server (primary and secondary)
 - registrar inserts two RRs into .com TLD server
 - (networkutopia.com, dns1.networkutopia.com, NS)
 - (dns1.networkutopia.com, 212.212.212.1, A)
- Set up **authoritative server** at that name/address
 - Create records for the services:
 - **type A record** for www.networkutopia.com
 - **type MX record** for @networkutopia.com email

Worksheet: Q5

Attack Surface Overview



Worksheet: Q6

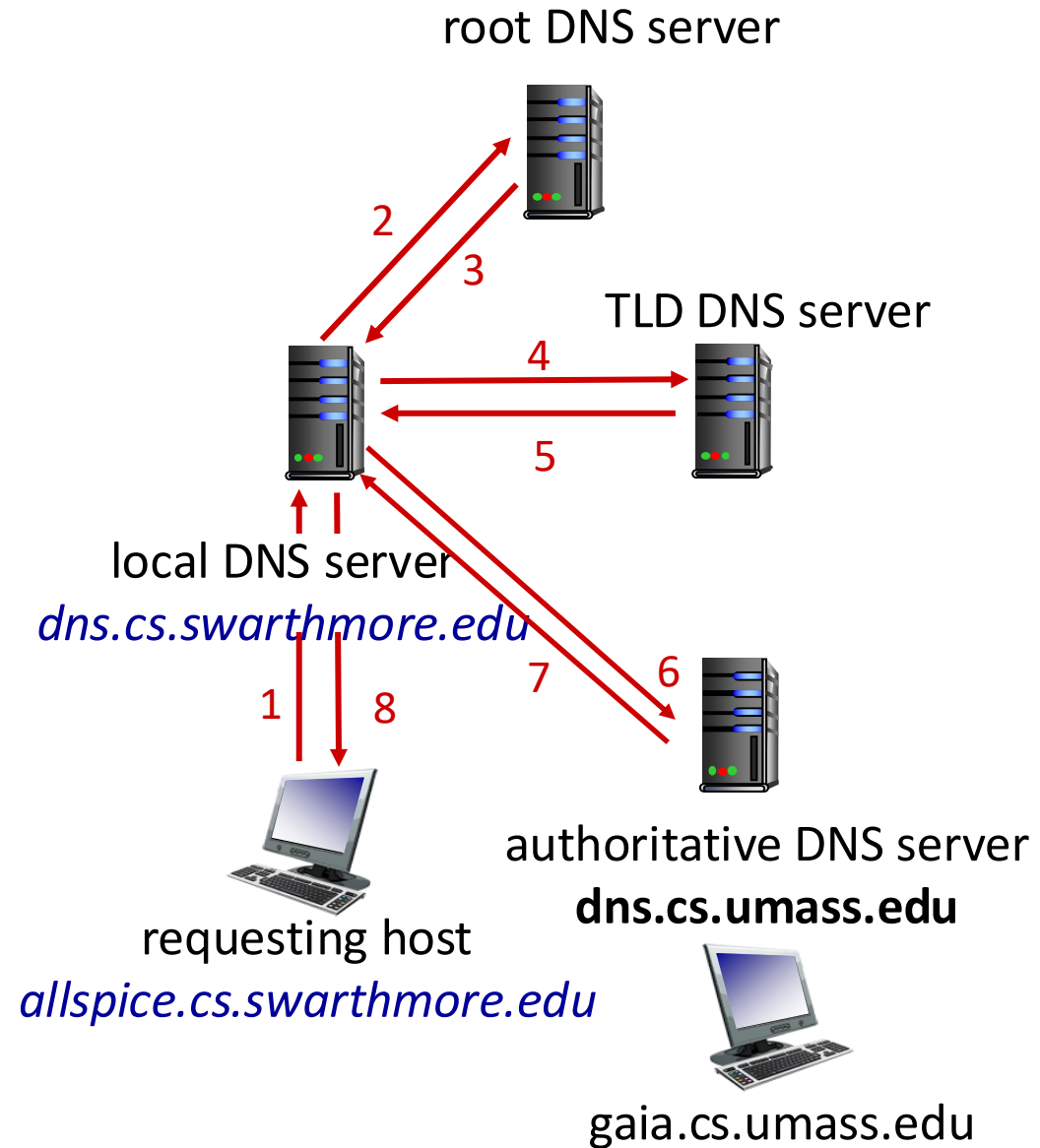
DNS security

DNS Vulnerabilities:

- No authentication
- Connectionless transport layer protocol (UDP)

DNS Attacks:

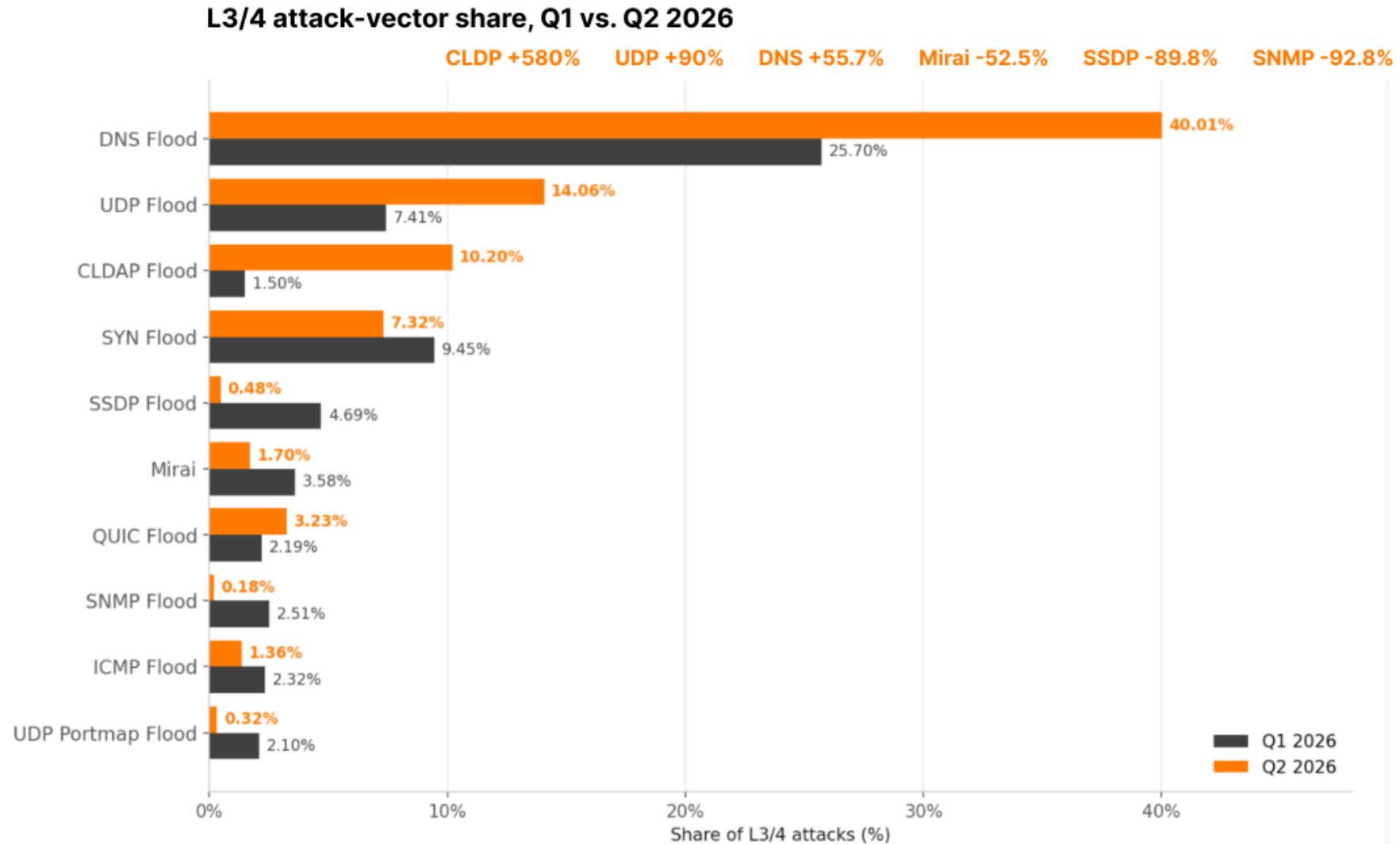
- Amplification Attack
- Cache Poisoning
- Man-in-the-middle
- DNS Redirection
- DDoS
- DNS Injection



Denial Of Service

- Flood DNS servers with requests until they fail
- October 2002: massive DDoS against the root name servers
 - What was the effect?
 - ... users didn't even notice
 - Root zone file is cached almost everywhere
- More targeted attacks can be effective
 - Local DNS server → cannot access DNS
 - Authoritative server → cannot access domain

In 2026, Cloudflare noted most DDoS attacks used DNS.



DNS Hijacking

- Infect their OS or browser with a virus/trojan
 - e.g. Many trojans change entries in /etc/hosts
 - *.bankofamerica.com → evilbank.com
- Man-in-the-middle



- Response Spoofing
 - ▣ Eavesdrop on requests
 - ▣ Outrace the servers response

DNS S

Where is
bankofamerica.com?

123.45.67.89

How do you know that a given
name → IP mapping is correct?

Bank of America

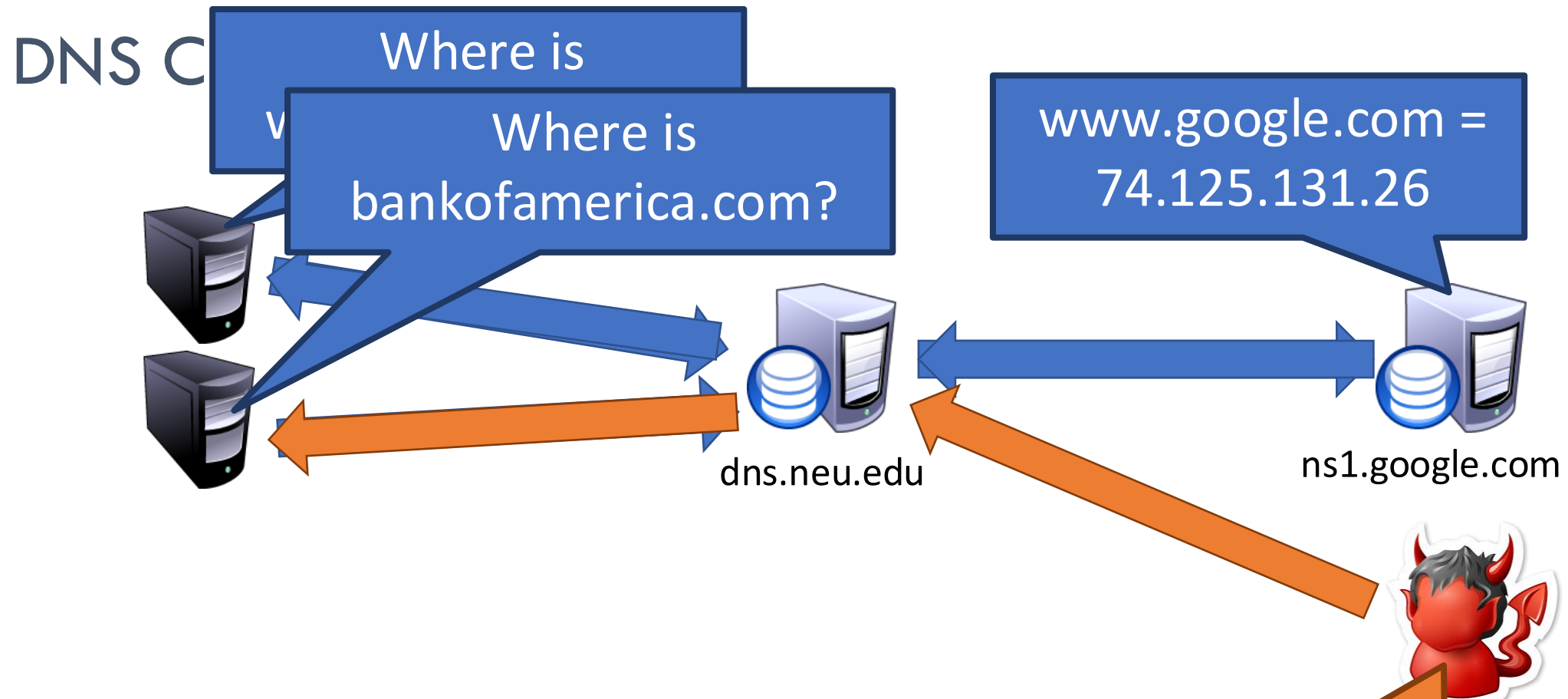
Where is
bankofamerica.com?

66.66.66.93

123.45.67.89

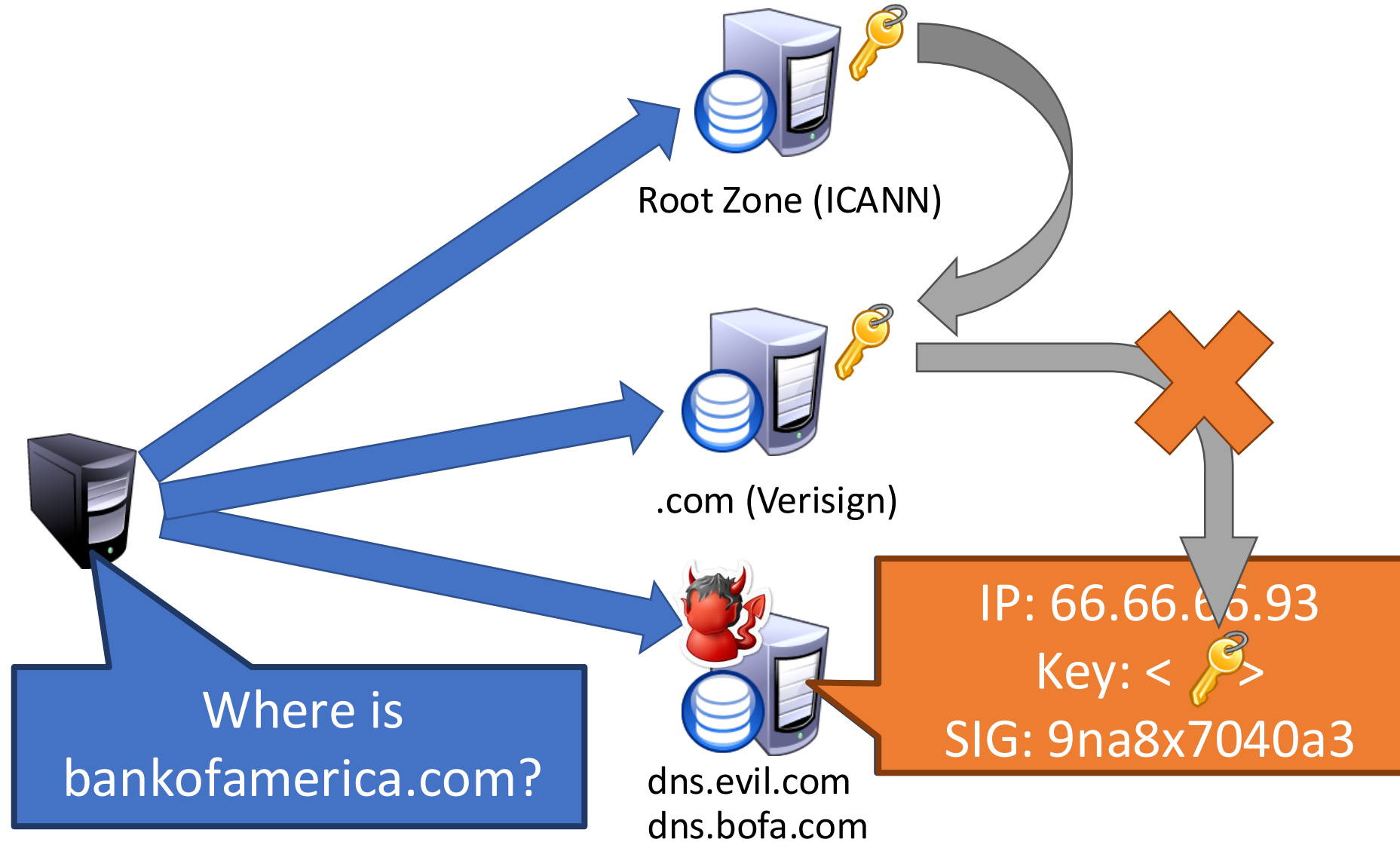
dns.evil.com

66.66.66.93



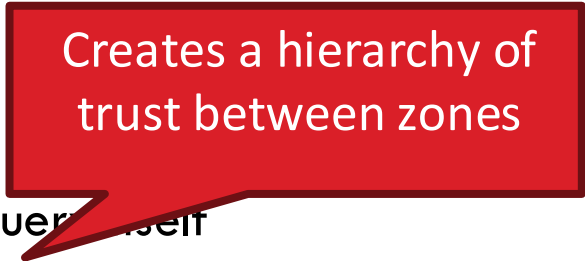
- Until the TTL expires, all queries for BofA to dns.neu.edu will return poisoned re
- Much worse than spoofing/man-in-
 - Whole ISPs can be impacted!

DNSSEC Hierarchy of Trust



Solution: DNSSEC

- Cryptographically sign critical resource records
 - Resolver can verify the cryptographic signature
- Two new resource **types**
 - Type = DNSKEY
 - Name = Zone domain name
 - Value = Public key for the zone
 - Type = RRSIG
 - Name = (type, name) tuple, i.e. the query itself
 - Value = Cryptographic signature of the query results



Creates a hierarchy of trust between zones



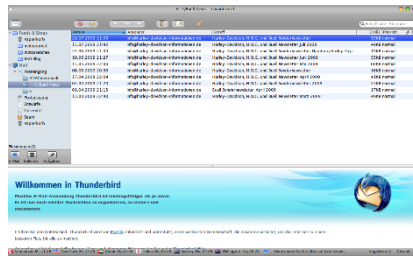
Prevents hijacking and spoofing

Summary

- DNS maps human readable names to IP addresses
- DNS arranged into a hierarchy
 - Scalability / distributed responsibility
 - Autonomous control of local name servers
- Caching crucial for performance
- DNS vulnerable to many kinds of attacks

Electronic Mail – Major Components

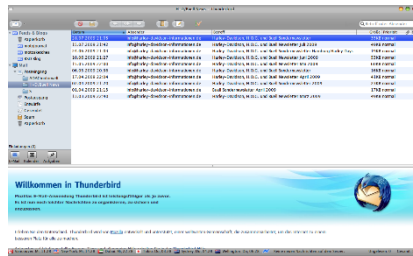
Mail user agent (MUA)
"mail client"



Alice

Electronic Mail – Major Components

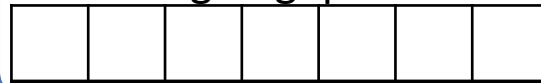
Mail user agent (MUA)
"mail client"



Mail transfer agent (MTA)
"mail server"

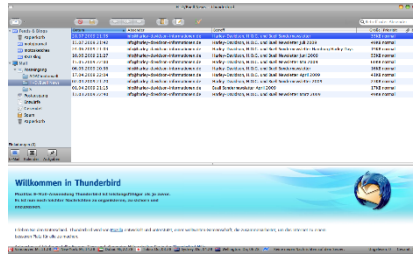


Outgoing queue:

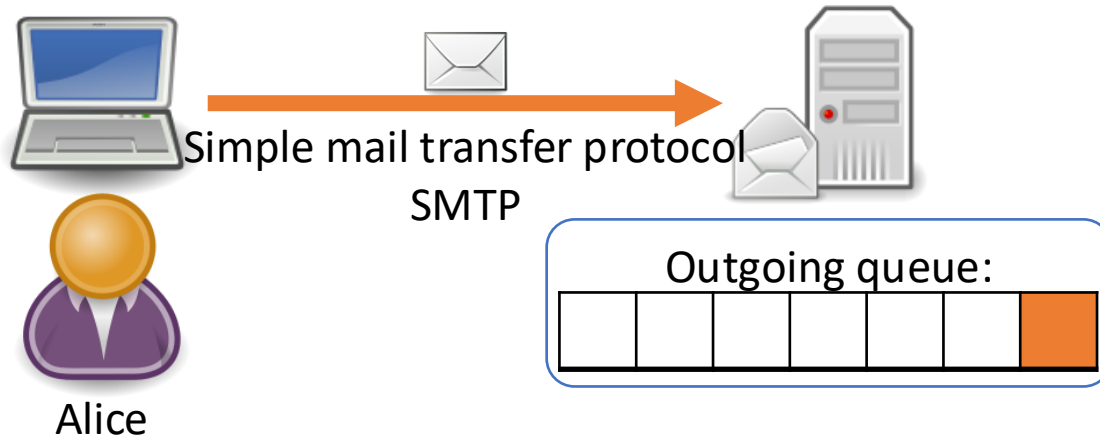


Electronic Mail – Major Components

Mail user agent (MUA)
"mail client"

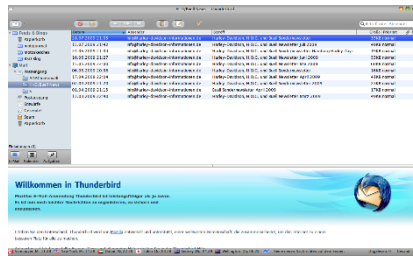


Mail transfer agent (MTA)
"mail server"

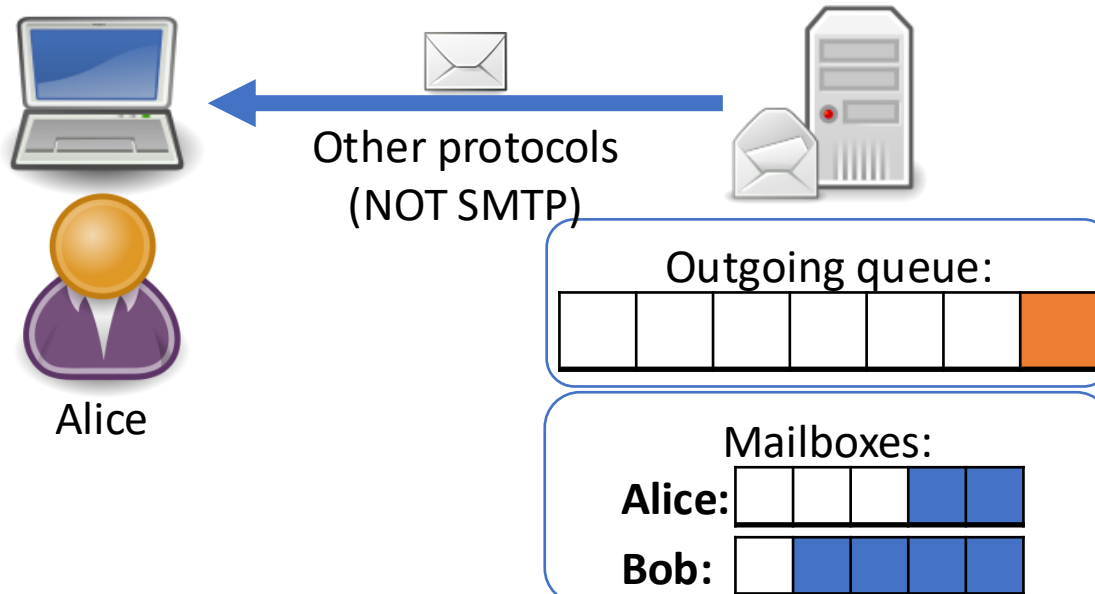


Electronic Mail – Major Components

Mail user agent (MUA)
"mail client"

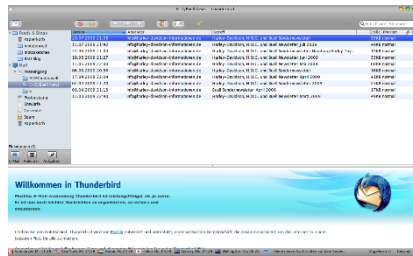


Mail transfer agent (MTA)
"mail server"



Same organization, one server

Mail user agent (MUA)
"mail client"

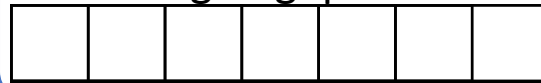


Alice

Mail transfer agent (MTA)
"mail server"



Outgoing queue:



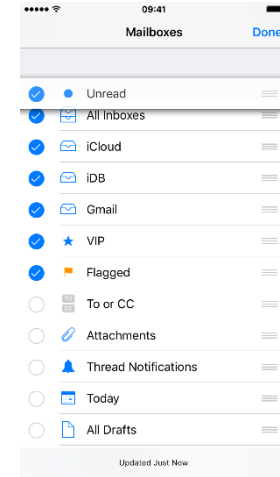
Mailboxes:

Alice:

--	--	--	--	--

Bob:

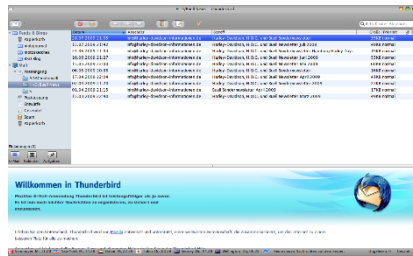
--	--	--	--	--



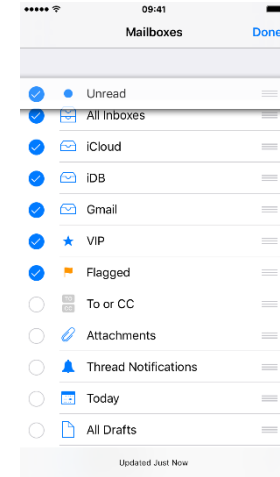
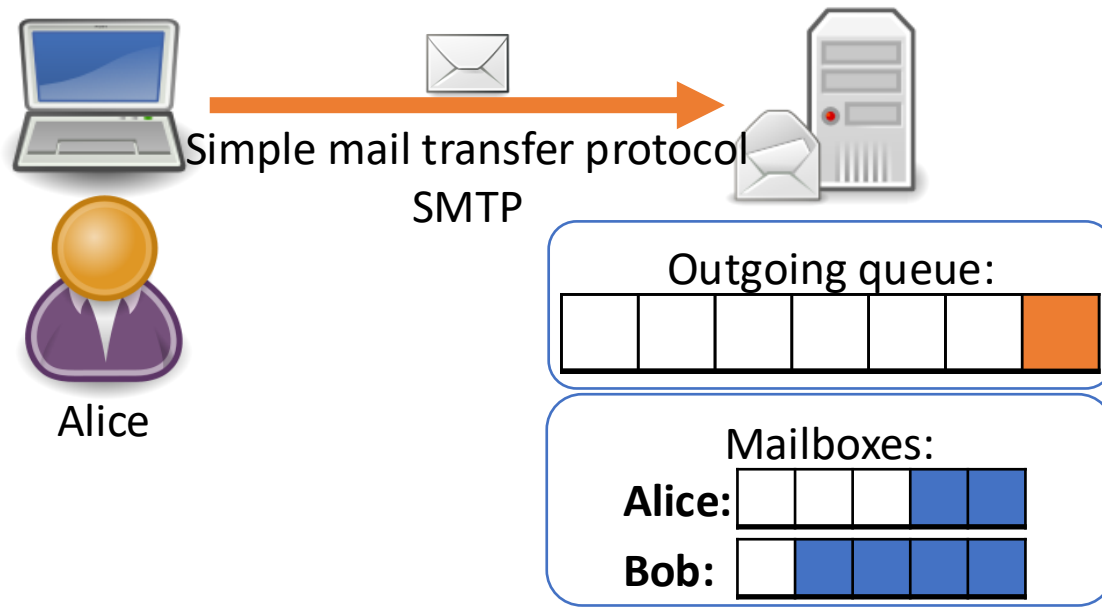
Bob

Same organization, one server

Mail user agent (MUA)
"mail client"

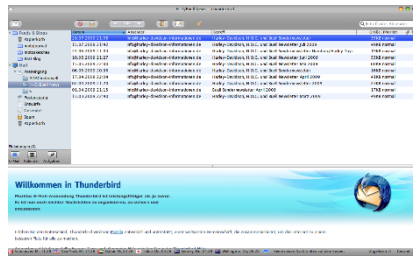


Mail transfer agent (MTA)
"mail server"



Same organization, one server

Mail user agent (MUA)
"mail client"

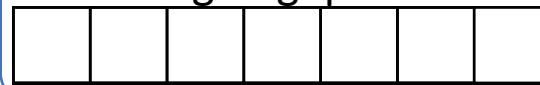


Alice

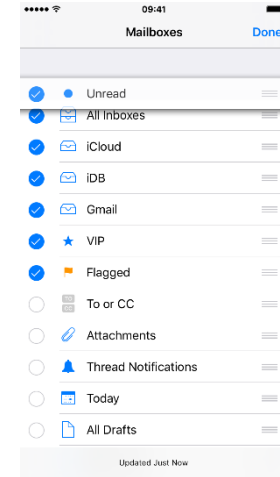
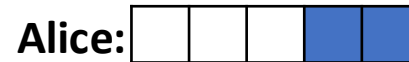
Mail transfer agent (MTA)
"mail server"



Outgoing queue:



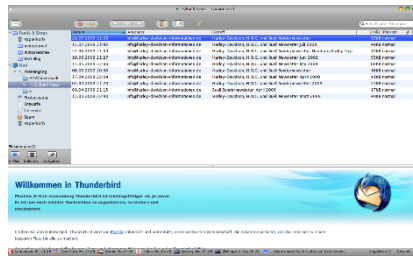
Mailboxes:



Bob

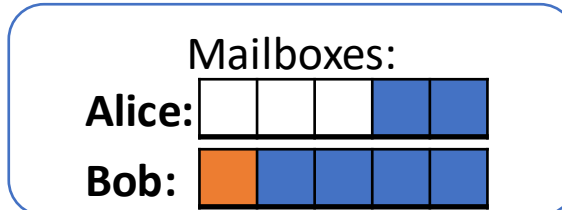
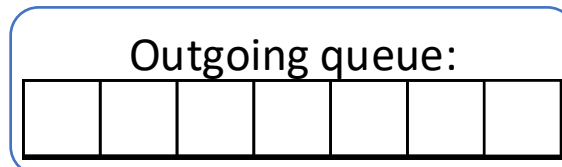
Same organization, one server

Mail user agent (MUA)
"mail client"

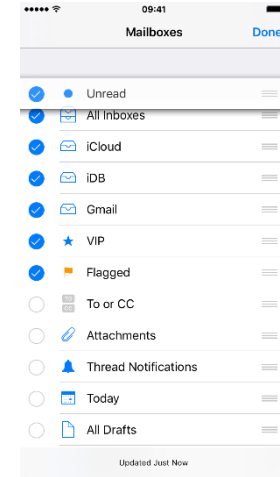


Alice

Mail transfer agent (MTA)
"mail server"



Other protocols
(NOT SMTP)



Bob

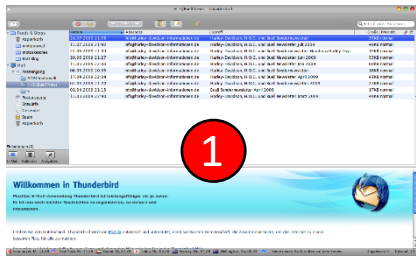
What should happen if Alice sends an email to Bob, but they're in different organizations (different mail servers)?

- A. Her mail **client** sends a message to his mail **server**
- B. Her mail **server** sends a message to his mail **server**
- C. Her mail **server** sends a message to his mail **client**
- D. Her mail **client** sends a message to his mail **client**

Different organizations, two servers

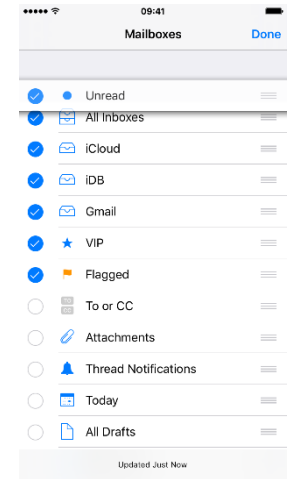
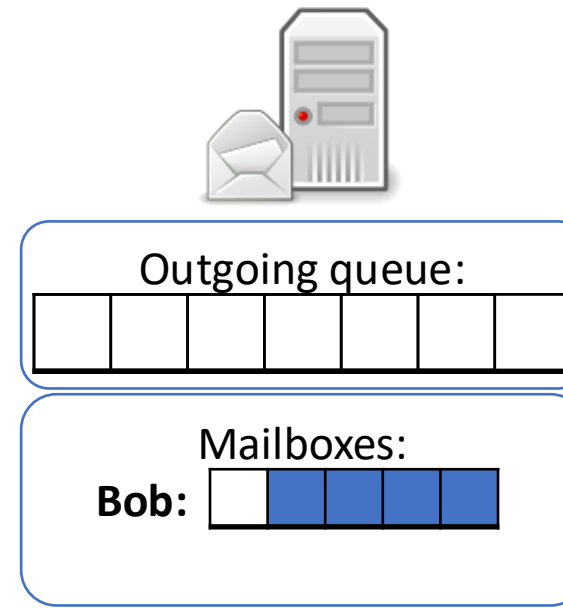
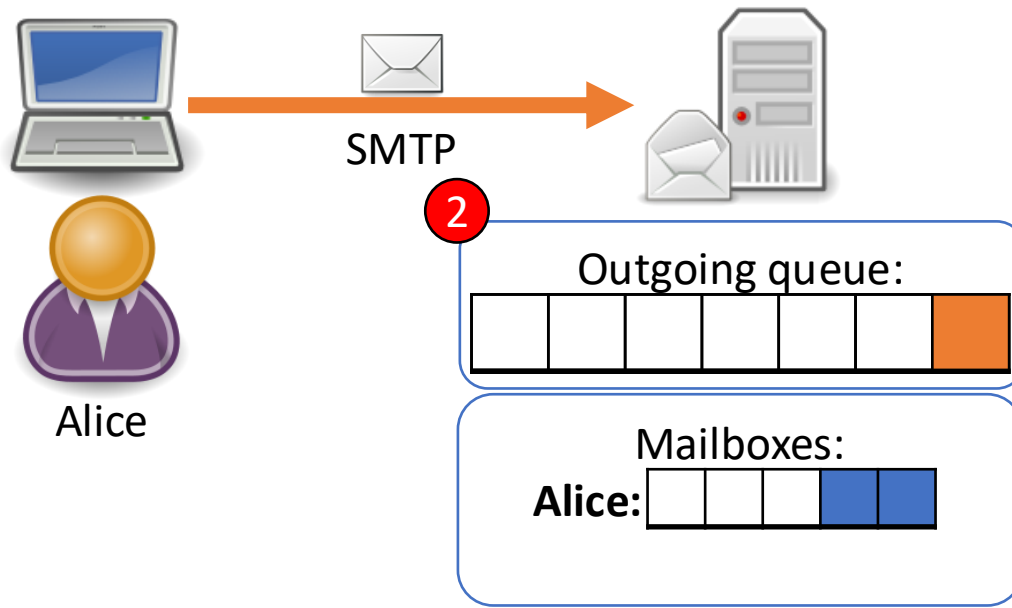
- (1) Alice composes message in mail client.
- (2) Alice's client transfers message to her server, via SMTP, where it goes in a queue.

Mail user agent (MUA)
"mail client"



Mail transfer agent (MTA)
"mail server"

Mail transfer agent (MTA)
"mail server"

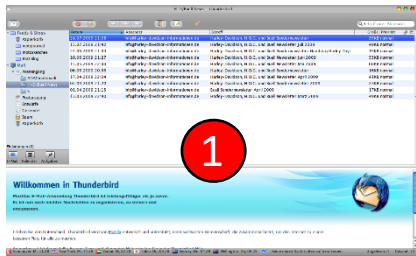


Different organizations, two servers

(3) Alice's server uses SMTP to transmit message to Bob's server.

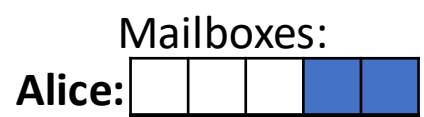
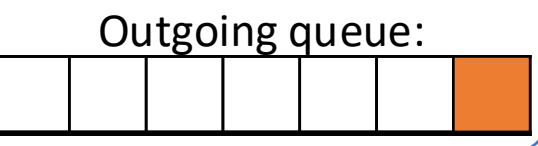
If this fails for some reason, Alice's server will try again (details depend on server configuration).

Mail user agent (MUA)
"mail client"



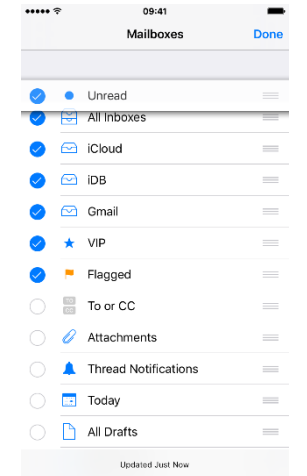
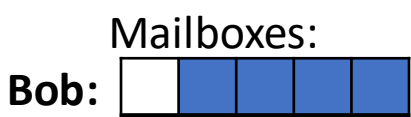
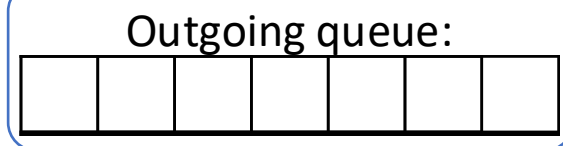
SMTP

2



SMTP

3

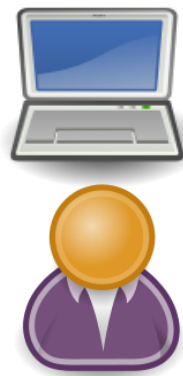
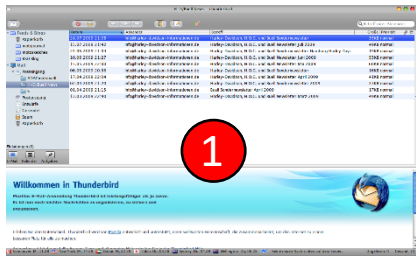


Different organizations, two servers

(4) Bob's server places the message in Bob's mailbox, waiting for him to retrieve it.

(5) Eventually, Bob comes along asking for new messages.

Mail user agent (MUA)
"mail client"

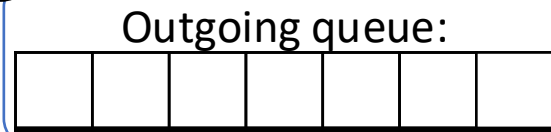


Alice

SMTP

2

Mail transfer agent (MTA)
"mail server"



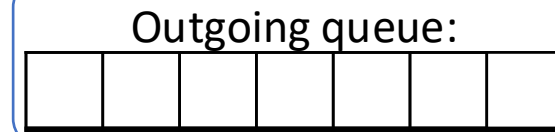
Mailboxes:
Alice:

--	--	--	--	--

SMTP

3

Mail transfer agent (MTA)
"mail server"



Mailboxes:
Bob:

--	--	--	--	--

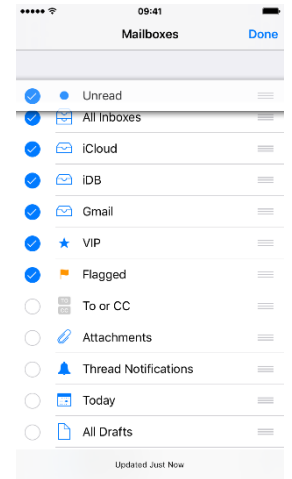
4

Other protocols
(NOT SMTP)

5



Bob



Simple Mail Transfer: SMTP [RFC 2821]

- Uses TCP to reliably transfer email message from client to server, port 25
- Direct transfer: sending server to receiving server (no intermediate servers)
- Three phases of transfer
 - handshaking (greeting)
 - transfer of messages
 - closure
- Command/response interaction (like HTTP, FTP)
 - **commands**: ASCII text
 - **response**: status code and phrase
- Messages must be in 7-bit ASCII

Simple Mail Transfer: SMTP [RFC 2821]

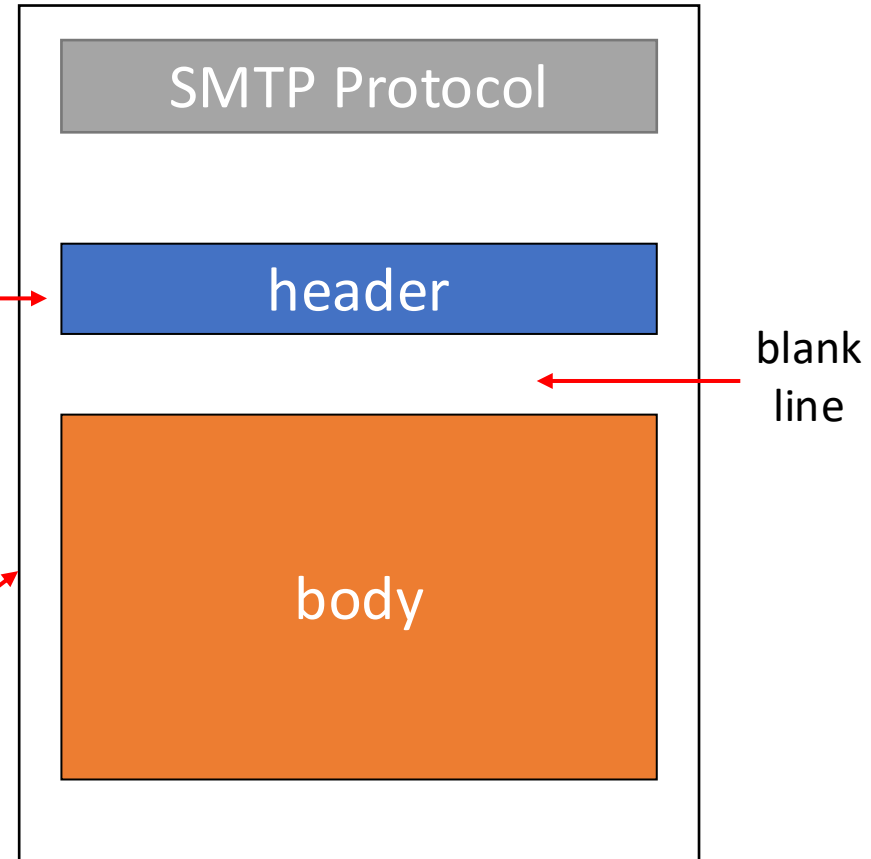
- Uses TCP to reliably transfer email message from client to server, port 25
- Direct transfer: sending server to receiving server (no intermediate servers)
- Three phases of transfer
 - handshaking (greeting)
 - transfer of messages
 - closure
- Command/response interaction (like HTTP, FTP)
 - **commands**: ASCII text
 - **response**: status code and phrase
- **Messages must be in 7-bit ASCII**

Legacy message format before we wanted to transfer binary data.

SMTP Message Format

RFC 822: standard for text message format:

- header lines, e.g.,
 - To:
 - From:
 - Subject:*different from* SMTP MAIL FROM, RCPT TO: commands!
- Body: the “message”
 - ASCII characters only
 - Signal EOM with “\r\n.\r\n”



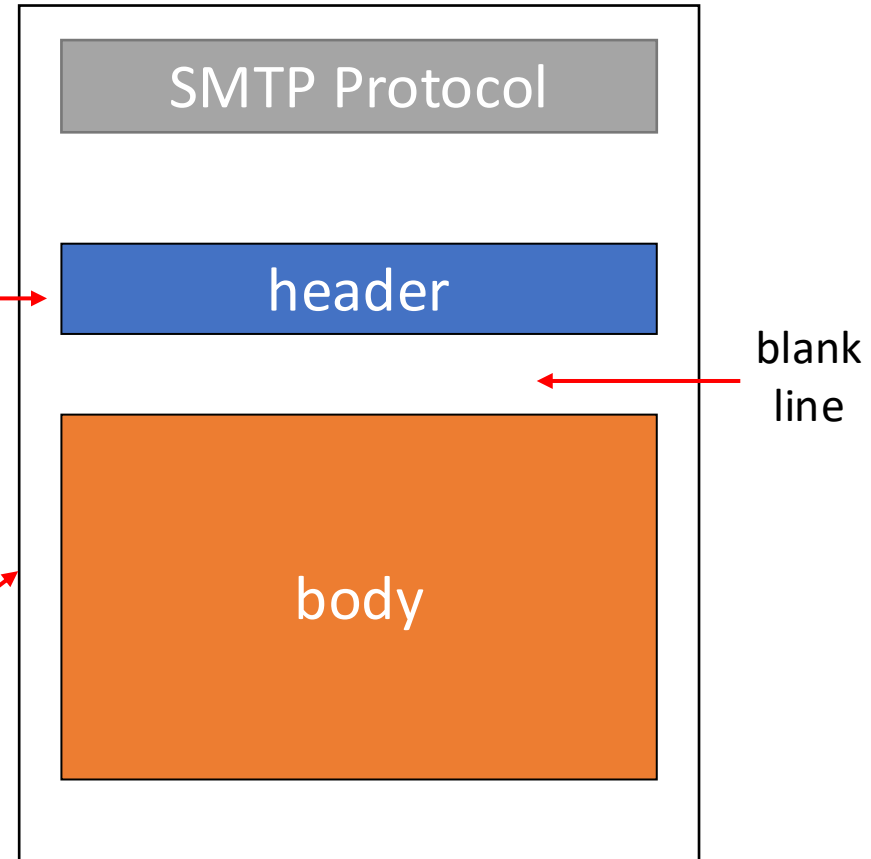
SMTP messages

- HELO <domain> (or)
EHLO <domain>
- MAIL FROM:<address>
- RCPT TO:<address> (can repeat)
- DATA
 - (message goes here)
- CRLF.CRLF
- (greeting)identify the sender's domain name
- Identify sender's address
- Identify recipient address(es)
- End of SMTP phase, message data comes next
- End of message data (done)

SMTP Message Format

RFC 822: standard for text message format:

- header lines, e.g.,
 - To:
 - From:
 - Subject:*different from* SMTP MAIL FROM, RCPT TO: commands!
- Body: the “message”
 - ASCII characters only
 - Signal EOM with “\r\n.\r\n”



Try SMTP interaction for yourself:

- `telnet allspice.cs.swarthmore.edu 25`
- You should see a 220 reply from the server.
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

(lets you send email without using email client (MUA))

Demo

Sample SMTP interaction

```
$ telnet allspice.cs.swarthmore.edu 25
Trying 130.58.68.9...
Connected to allspice.cs.swarthmore.edu
220 allspice.cs.swarthmore.edu ESMTP Postfix
HELO cs.swarthmore.edu
250 allspice.cs.swarthmore.edu
MAIL FROM:<rware@cs.swarthmore.edu>
250 2.1.0 OK
RCPT TO:<rware@cs.swarthmore.edu>
250 2.1.5 OK
DATA
354 End data with <CR><LF>.<CR><LF>
To: Ranysha Ware <rware@cs.swarthmore.edu>
From: Ranysha Ware <rware@cs.swarthmore.edu>
Subject: Telnet test message
```

This is a test message, via telnet, to myself.

.

Sample SMTP interaction

```
$ telnet allspice.cs.swarthmore.edu 25
Trying 130.58.68.9...
Connected to allspice.cs.swarthmore.edu
220 allspice.cs.swarthmore.edu ESMTP Postfix
HELO cs.swarthmore.edu
250 allspice.cs.swarthmore.edu
MAIL FROM:<rware@cs.swarthmore.edu>
250 2.1.0 OK
RCPT TO:<rware@cs.swarthmore.edu>
250 2.1.5 OK
DATA
354 End data with <CR><LF>.<CR><LF>
To: Ranysha Ware <rware@cs.swarthmore.edu>
From: Ranysha Ware <rware@cs.swarthmore.edu>
Subject: Telnet test message
```

This is a test message, via telnet, to myself.

End of message:
CRLF (Dot) CRLF



What keeps us from entering a fake information (e.g., FROM address)?

- A. Nothing.
- B. The MTA checks that the FROM is valid.
- C. We enter a name/password logging into the MTA.

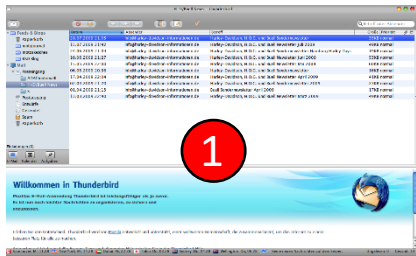
Fun Demo

Wait...why does this work?

(3) Alice's server uses SMTP to transmit message to Bob's server.

If this fails for some reason, Alice's server will try again (details depend on server configuration).

Mail user agent (MUA)
"mail client"



Alice

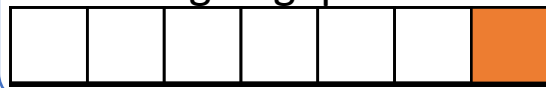
SMTP

2

Mail transfer agent (MTA)
"mail server"



Outgoing queue:



Mailboxes:

Alice: 

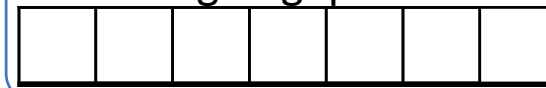
SMTP

3

Mail transfer agent (MTA)
"mail server"

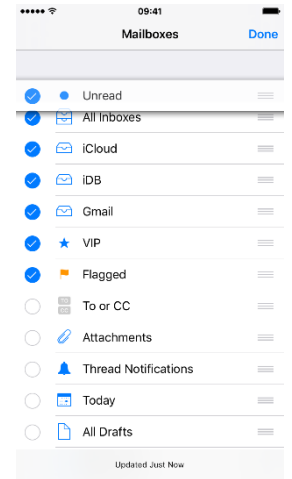


Outgoing queue:



Mailboxes:

Bob: 



Bob

This seems too horrible to be true. Surely
we can prevent header forging?
(How or why not?)

A. Yes

B. No

Message Signing

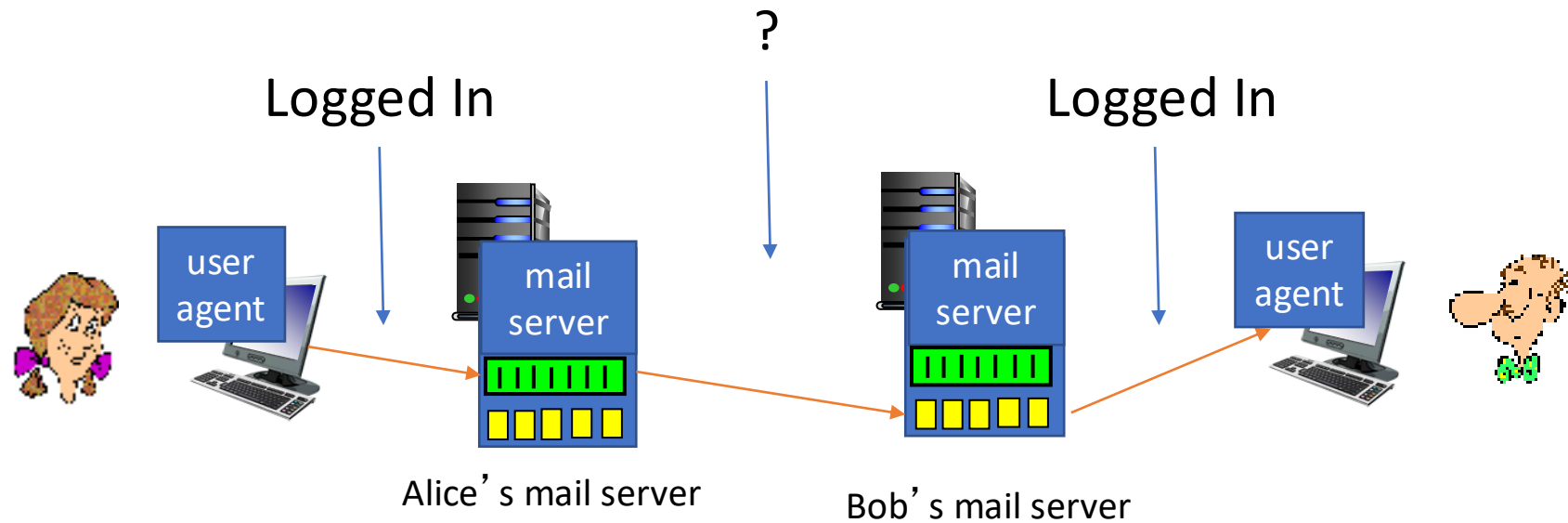
1. Sender creates cryptographic **public/private key pair**, publishes public key to the world.
2. Sender uses private key to sign messages.
3. Receiver can verify*, using published public key, that only the holder of the corresponding private key could have sent the message.

* With very high probability.

Message Signing: Challenges

- Disseminating public keys
 - How do you trust that the published public key isn't also a lie?
- It's more work, can't be bothered...
 - Adoption is very low

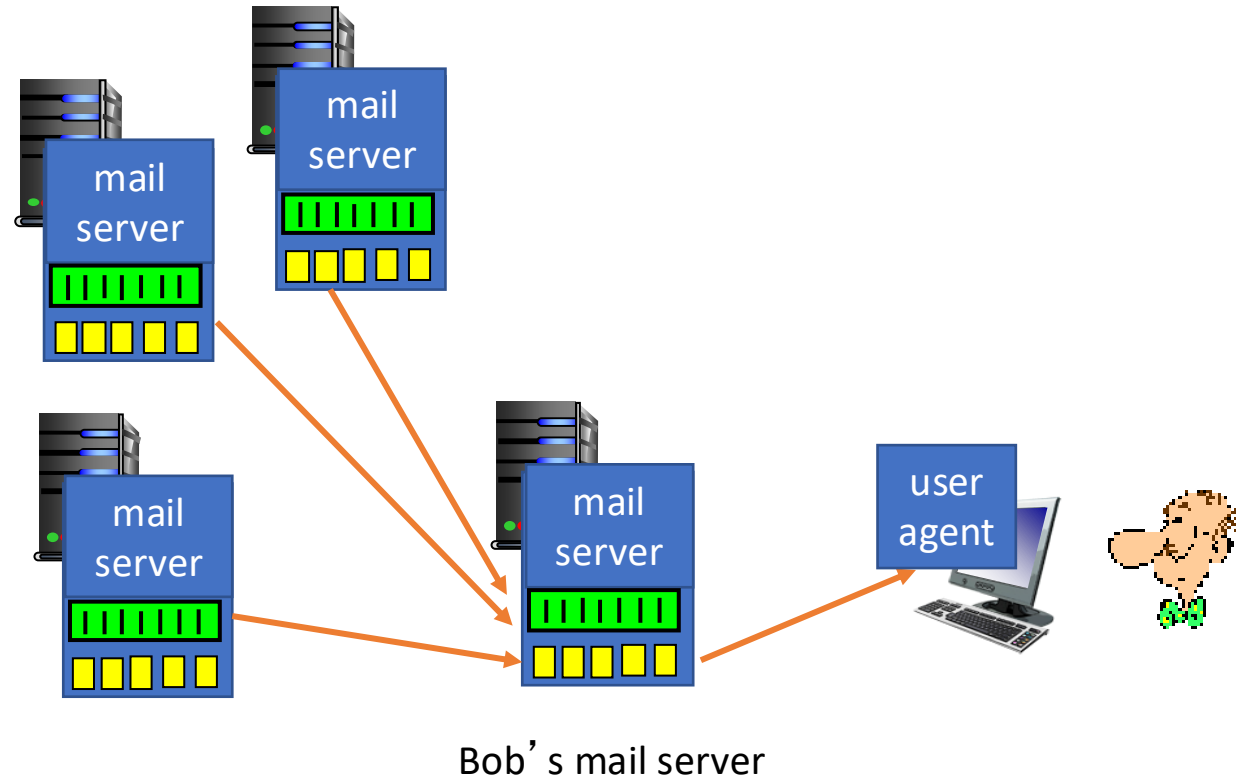
Logging In / Passwords



Logging In / Passwords

Any mail server
may need to send a
message to Bob's.

Tough for them
all to share
credentials...

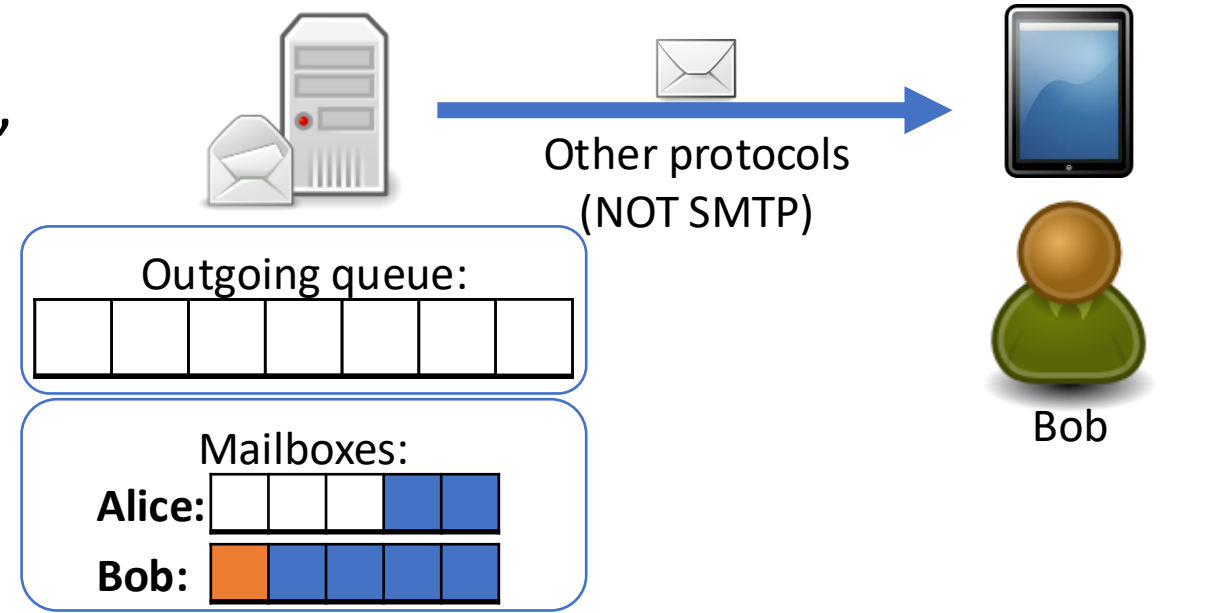


Server to Server Verification

- Published lists of "bad" IP addresses for blocklists
- LOTS of other techniques
 - See: https://en.wikipedia.org/wiki/Anti-spam_techniques
- Big problem: false positives

Mail Access Protocols

- Post Office Protocol: authorization, download
- Internet Mail Access Protocol (IMAP): more features, including manipulation of stored messages on server
- Webmail: gmail, Hotmail, Yahoo! Mail, etc.
- Proprietary protocols (Microsoft Exchange)



SMTP versus HTTP

- HTTP: pull
- SMTP: push
- Both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response message
- SMTP: multiple objects sent in multipart message

SMTP: final words

- SMTP uses persistent connections
 - Can send multiple emails in one session
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses CRLF.CRLF to determine end of message

Reference slides (not covered in lecture)

SMTP messages

- HELO <domain> (or)
EHLO <domain>
- MAIL FROM:<address>
- RCPT TO:<address> (can repeat)
- DATA
 - (message goes here)
- CRLF.CRLF
- (greeting)identify the sender's domain name
- Identify sender's address
- Identify recipient address(es)
- End of SMTP phase, message data comes next
- End of message data (done)

If SMTP only allows 7-bit ASCII, how do we send pictures/videos/files via email?

- A. We encode these objects as 7-bit ASCII
- B. We use a different protocol instead of SMTP
- C. We're really sending links to the objects, rather than the objects themselves

Base 64

- Designed to be an efficient way to send binary data as a string
- Uses A-Z, a-z, 0-9, “+” and “/” as digits
- A number with digits $d_n d_{n-1} \dots d_1 d_0 = 64^n * d_n + 64^{n-1} * d_{n-1} + \dots + 64^1 * d_1 + d_0$
- Recall from CS 31: Other non-base-10 number systems (binary, octal, hex).

Multipurpose Internet Mail Extensions (MIME)

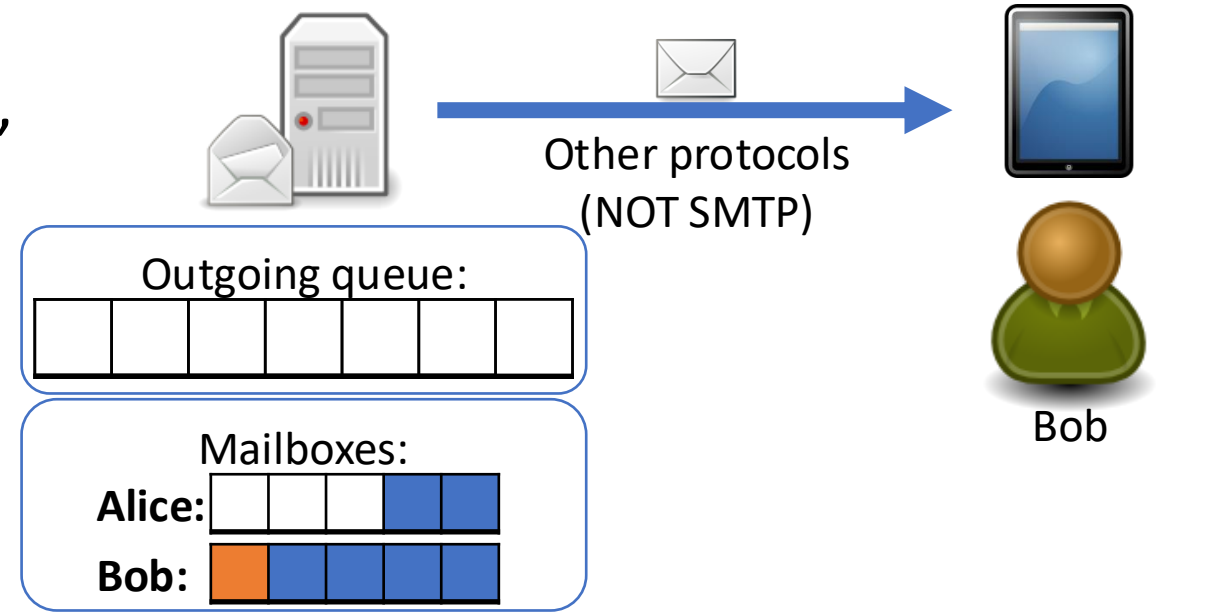
- Special formatting instructions
- Indicated in the header portion of message (not SMTP)
 - SMTP does *not* care, just looks like message data
- Supports
 - Text in character sets other than ASCII
 - Non-text attachments
 - Message bodies with multiple parts
 - Header information in non-ASCII character sets

MIME

- Adds optional headers
 - Designed to be compatible with non-MIME email clients
 - Both clients must understand it to make sense of it
- Specifies content type, other necessary information
- Designates a boundary between email text and attachments

Mail Access Protocols

- Post Office Protocol: authorization, download
- Internet Mail Access Protocol (IMAP): more features, including manipulation of stored messages on server
- Webmail: gmail, Hotmail, Yahoo! Mail, etc.
- Proprietary protocols (Microsoft Exchange)



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
```

More about POP3

- Previous example uses “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**
- Limitations:
 - Can’t retrieve just the headers
 - Can’t impose structure on messagesa

IMAP

- 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
- Can **request pieces of a message** (e.g., text parts without large attachments)

Webmail

- Uses a web browser
- Sends emails using HTTP rather than POP3 or IMAP
- Mail is stored on the 3rd party webmail company's servers

Summary

- Three main parts to email:
 - Mail User Agent (mail client): read / write for humans
 - Mail Transfer Agent: server that accepts / sends messages
 - SMTP protocol used to negotiate transfers
- No SMTP support for fraud detection
- Extensions (MIME) and encodings (Base64) for sending non-text data

More Reference Slides (from K&R textbook)

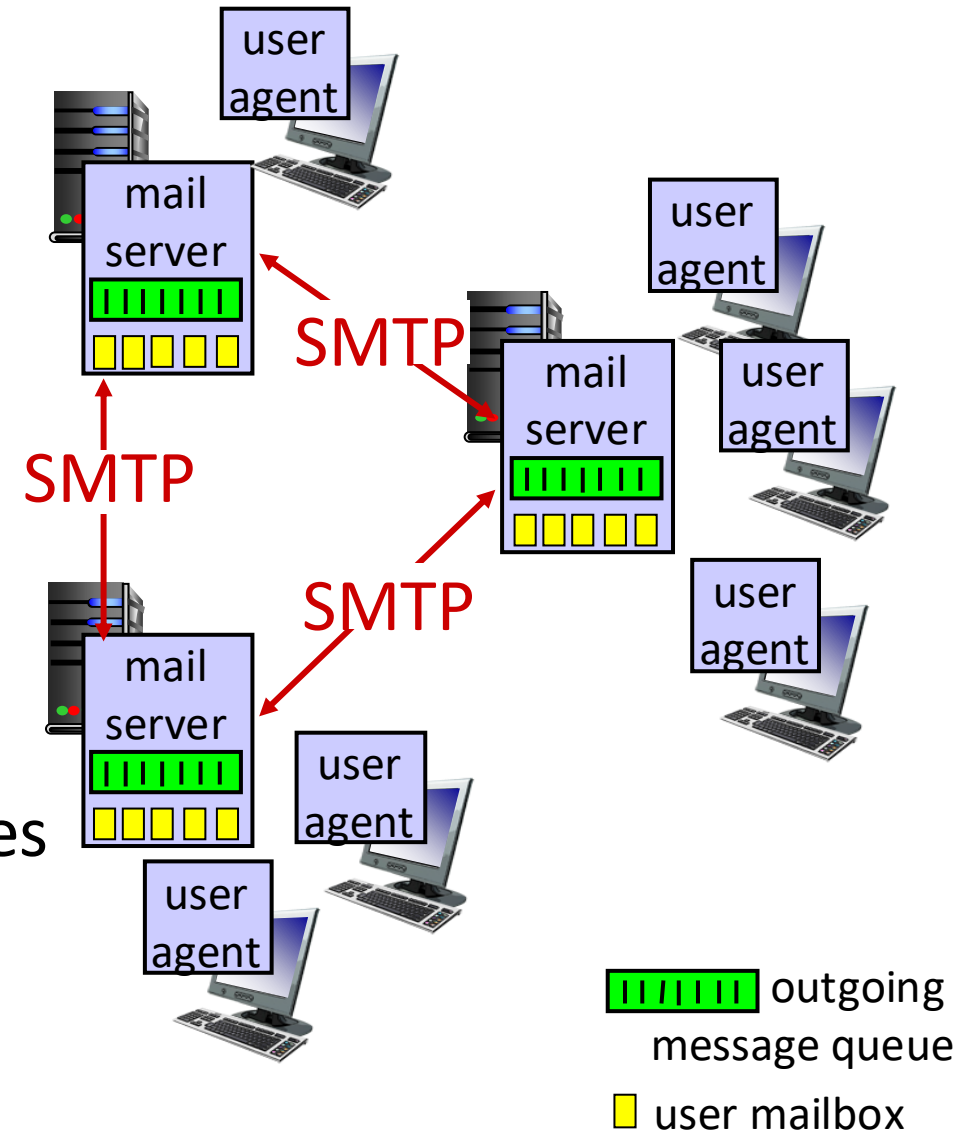
E-mail

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., Outlook, iPhone mail client
- outgoing, incoming messages stored on server



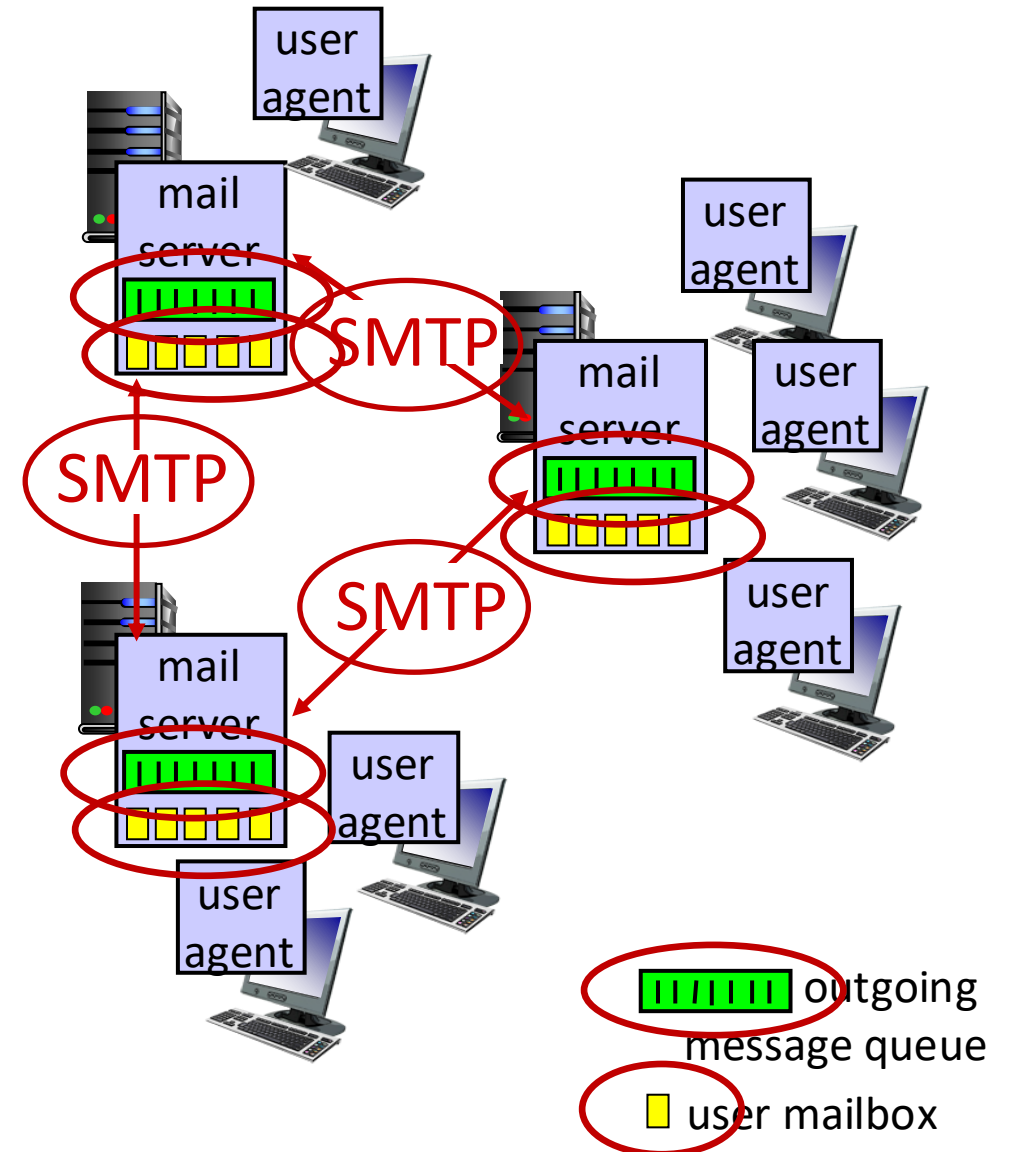
E-mail: mail servers

mail servers:

- *mailbox* contains incoming messages for user
- *message queue* of outgoing (to be sent) mail messages

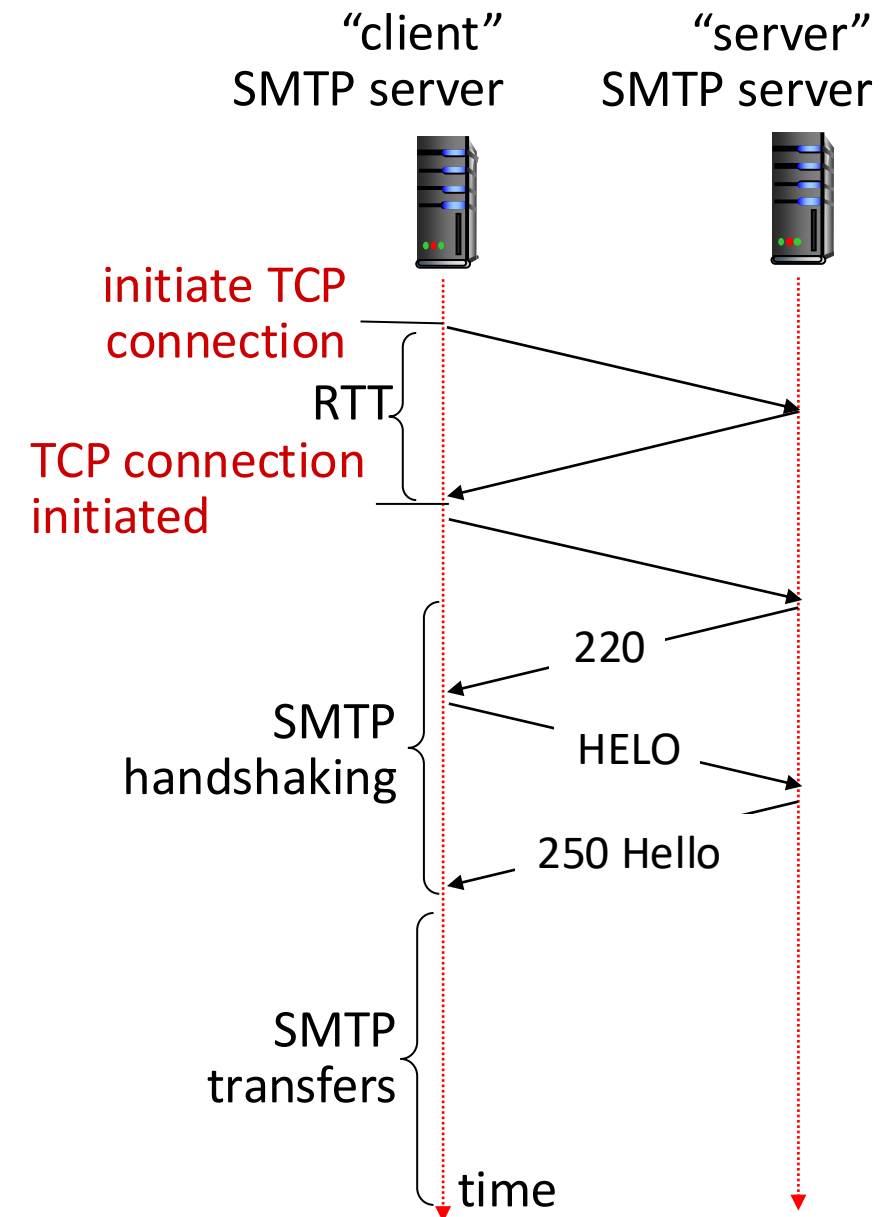
SMTP protocol between mail servers to send email messages

- **client**: sending mail server
- **“server”**: receiving mail server



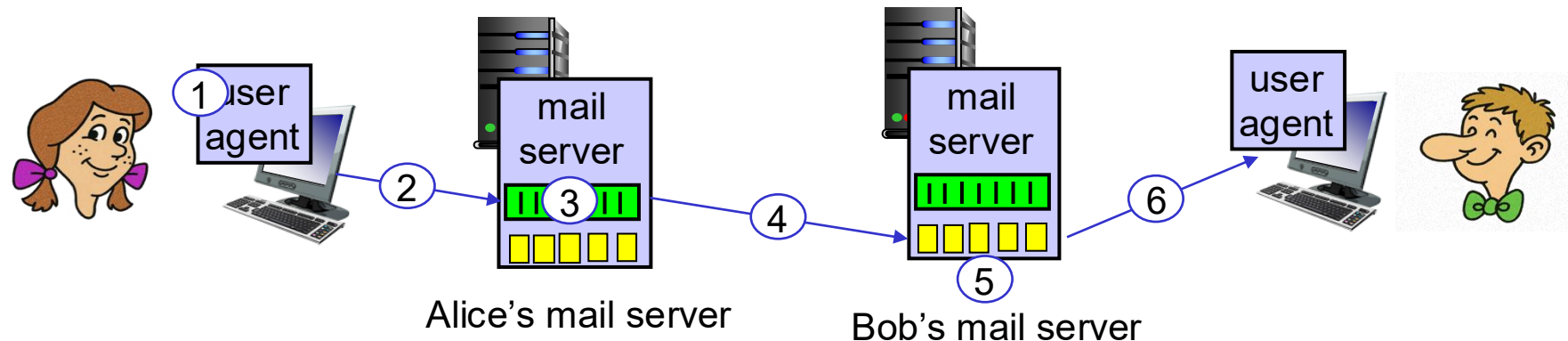
SMTP RFC (5321)

- uses TCP to reliably transfer email message from client (mail server initiating connection) to server, port 25
 - direct transfer: sending server (acting like client) to receiving server
- three phases of transfer
 - SMTP handshaking (greeting)
 - SMTP transfer of messages
 - SMTP closure
- command/response interaction (like HTTP)
 - **commands**: ASCII text
 - **response**: status code and phrase



Scenario: Alice sends e-mail to Bob

- 1) Alice uses UA to compose e-mail message "to" bob@some school.edu
- 2) Alice's UA sends message to her mail server using SMTP; message placed in message queue
- 3) client side of SMTP at mail server 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

```
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
```

SMTP: observations

comparison with HTTP:

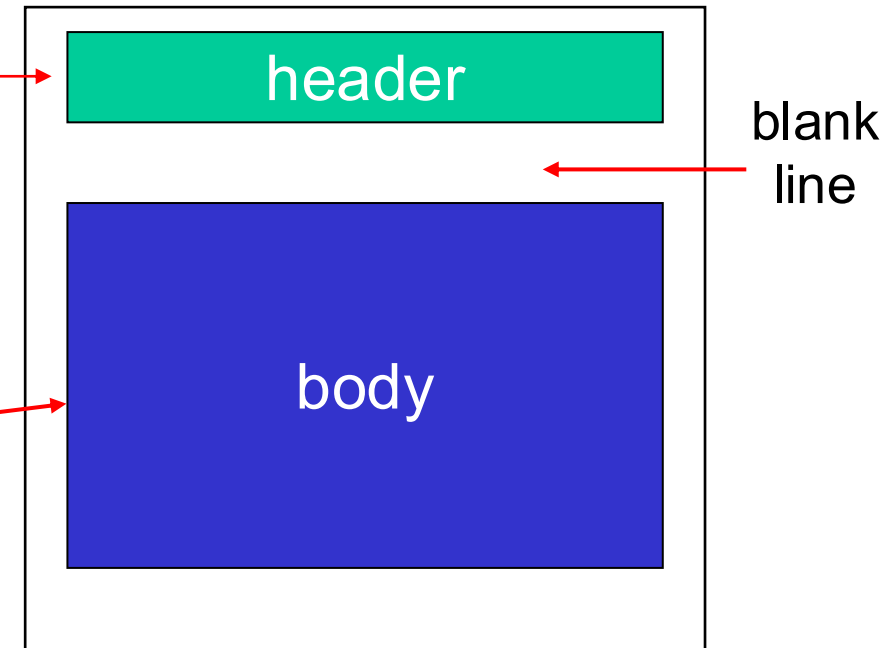
- HTTP: client pull
- SMTP: client push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response message
- SMTP: multiple objects sent in multipart message
- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses CRLF.CRLF to determine end of message

Mail message format

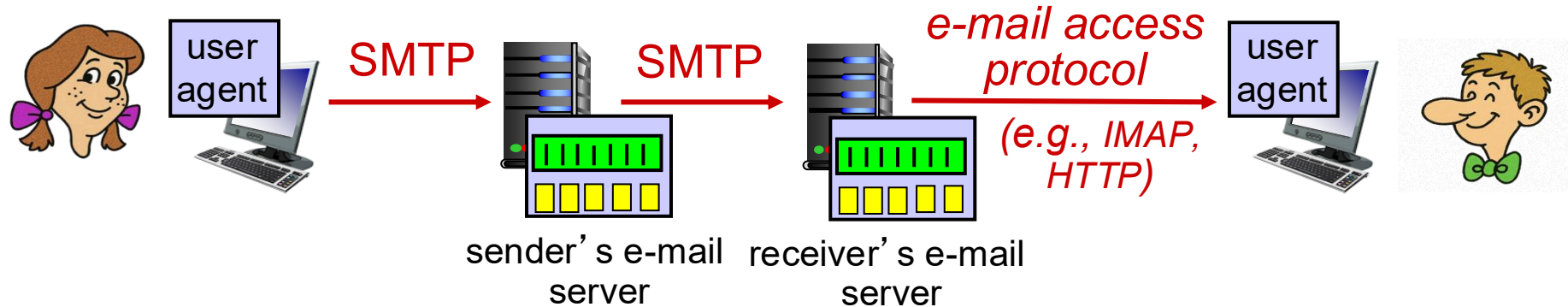
SMTP: protocol for exchanging e-mail messages, defined in RFC 5321 (like RFC 7231 defines HTTP)

RFC 2822 defines *syntax* for e-mail message itself (like HTML defines syntax for web documents)

- header lines, e.g.,
 - To:
 - From:
 - Subject:these lines, within the body of the email message area different from SMTP MAIL FROM:, RCPT TO: commands!
- Body: the “message” , ASCII characters only



Retrieving email: mail access protocols



- **SMTP**: delivery/storage of e-mail messages to receiver's server
- mail access protocol: retrieval from server
 - **IMAP**: Internet Mail Access Protocol [RFC 3501]: messages stored on server, IMAP provides retrieval, deletion, folders of stored messages on server
- **HTTP**: gmail, Hotmail, Yahoo!Mail, etc. provides web-based interface on top of SMTP (to send), IMAP (or POP) to retrieve e-mail messages