

CS3640

Application Layer (5): Email & SMTP

Prof. Supreeth Shastri

Computer Science

The University of Iowa

Lecture goals

Understand the protocols and mechanics of electronic mail

- *Email infrastructure*
- *SMTP*
- *IMAP*



Chapter 2.3

A brief history of electronic mail

1965: MIT's time sharing system, CTSS, introduces the MAIL command. It allows its users to send messages asynchronously to each other

1971: Ray Tomlinson writes the first mail program for ARPANET. To separate ARPANET users (~50) from their machines (~15), Tomlinson proposes user@host syntax

1976: Jimmy Carter uses emails in his presidential campaign

1992: Emails get the ability to “attach” non-ASCII content

2020: ~246 billion emails are sent everyday!

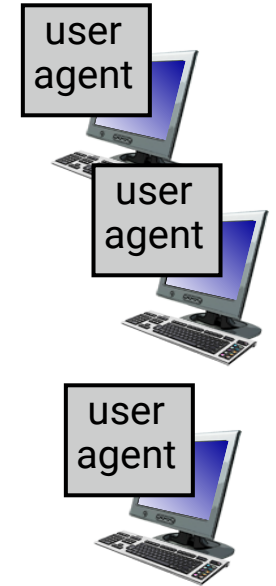
E-mail Infrastructure

Three major components

- user agents
- mail servers
- email protocols

1. User Agents (UA)

- the client app of the email system
- allow users to read, reply to, forward, save and compose emails
- E.g., Outlook, Mail.app, iPhone Mail



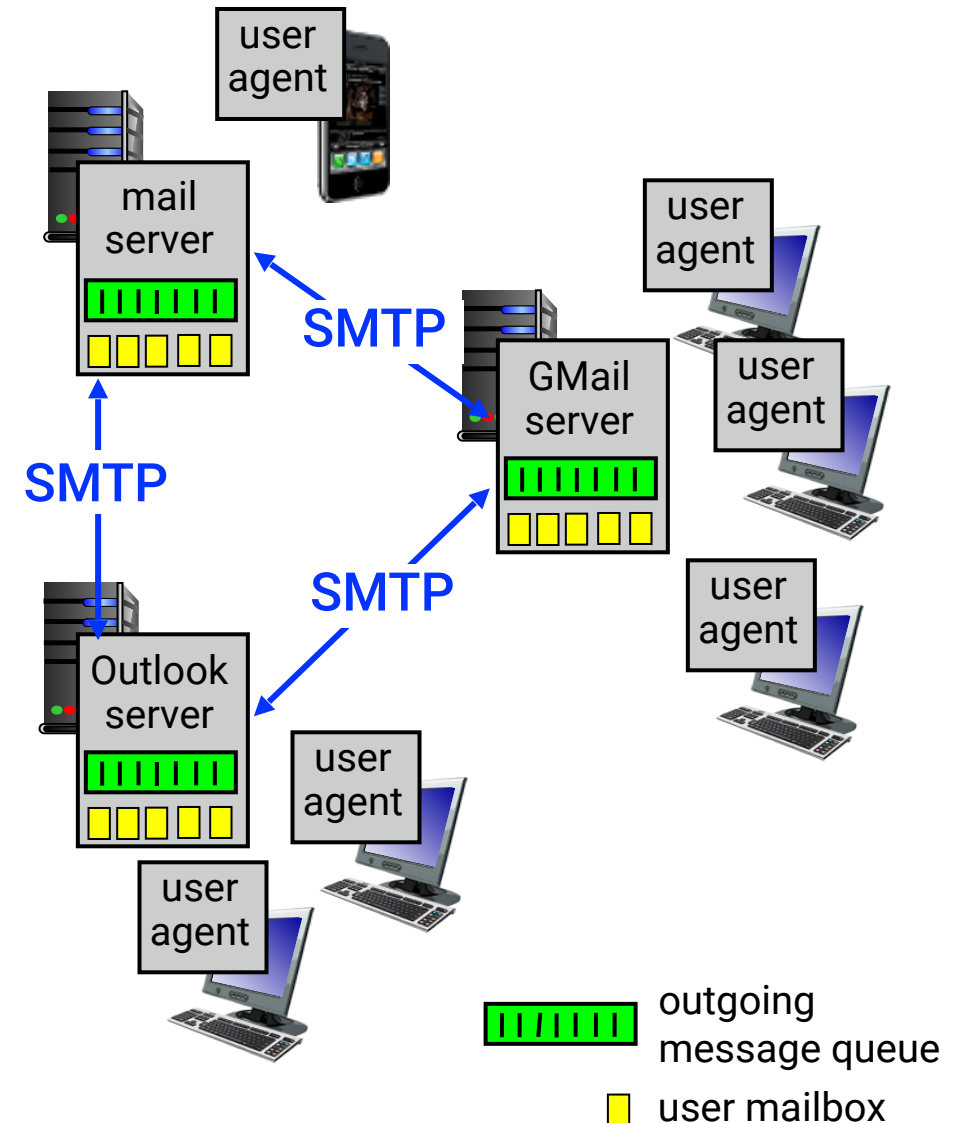
E-mail Infrastructure

2. Mail Servers

- offer email as a service
- creates a **mailbox** for each user, where it stores their incoming mails
- **message queue** of outgoing (to be sent) mail messages

3. Email Protocols

- **SMTP** for sending emails to mail servers
- **IMAP** for retrieving emails from a mail server



Email in action

1

Alice uses her UA to compose an email message to bob@illinois.edu

2

Alice's UA sends message to her uiowa mail server using SMTP; server places it in the message queue

3

uiowa mail server opens a TCP connection with illinois mail server

4

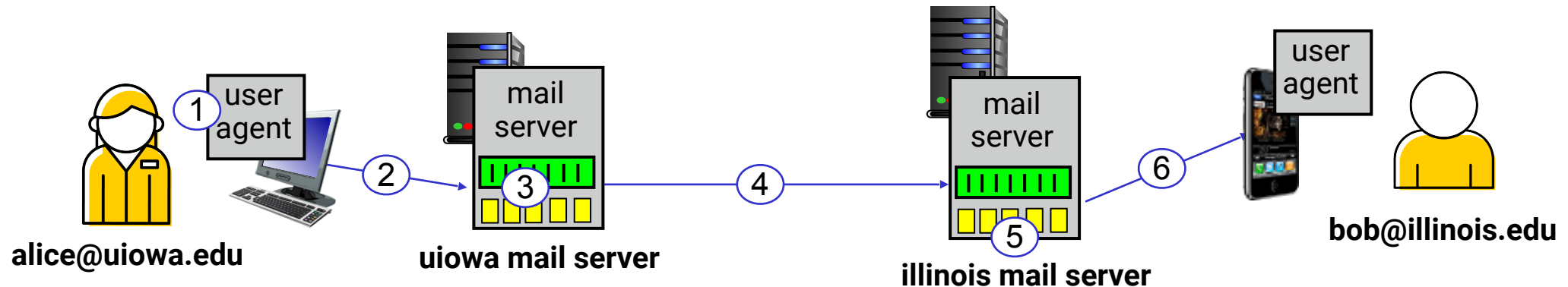
uiowa server (i.e., SMTP client) sends the message to illinois server (SMTP server) over the TCP connection

5

illinois mail server places the received message in Bob's mailbox

6

Bob uses his UA to retrieve the message at a later time



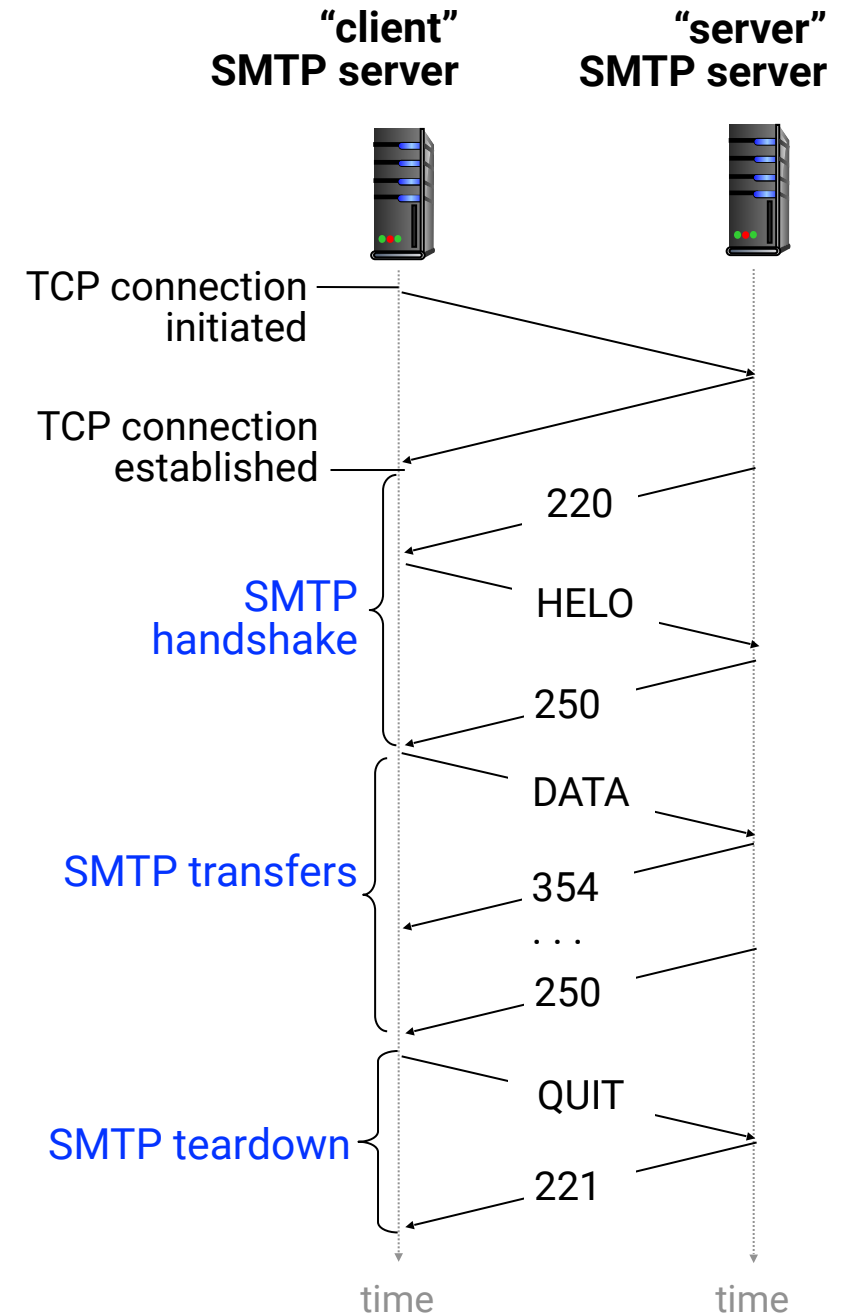
SMTP

SMTP

- Protocol for pushing email messages to a mail server
- Defined in RFC 5321 (*original RFC 821 created in 1982*)
- Uses client-server model and ASCII syntax
- Uses TCP for reliable transfer
- SMTP servers listen on port 25

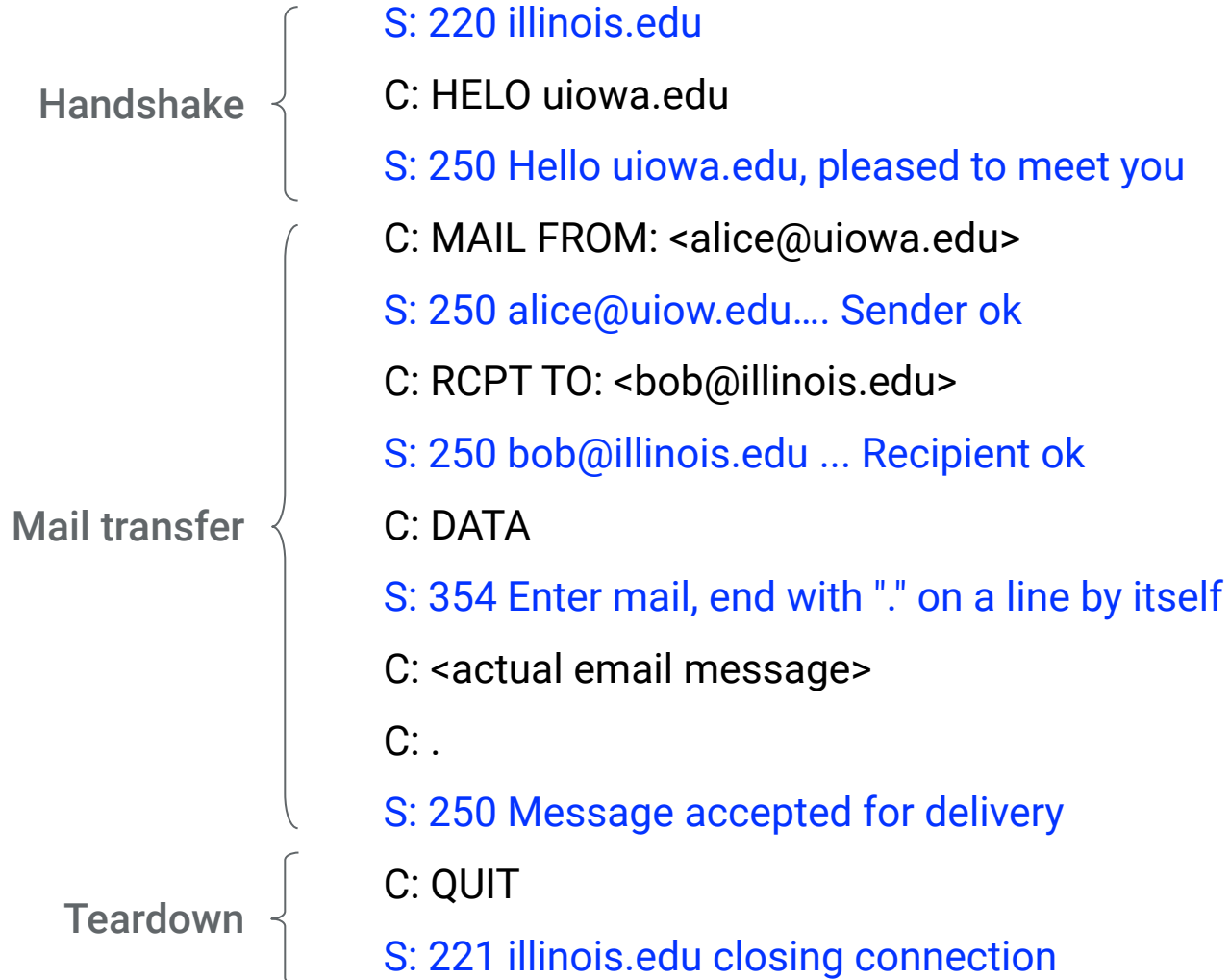
Three phases of SMTP dialog

- SMTP handshake
- SMTP transfer
- SMTP closure



SMTP interaction

— Server
— Client



SMTP response codes

2XX Positive completion

- 211** *System status, or system help reply*
- 214** *Help message*
- 220** *Service ready*
- 221** *Goodbye*
- 235** *Authentication succeeded*
- 250** *Requested mail action completed*
- 251** *User not local; will forward*
- 252** *Cannot verify the user, but it will try to deliver*

3XX Positive intermediate

- 334** *Server challenge*
- 354** *Start mail input*

4XX Negative transient

- 421** *Service not available,*
- 432** *A password transition is needed*
- 450** *Mailbox unavailable*
- 451** *Requested action aborted: local error*
- 451** *IMAP server unavailable*
- 452** *Requested action not taken: insufficient storage*
- 454** *Temporary authentication failure*
- 455** *Server unable to accommodate parameters*

5XX Negative permanent

- 500** *Syntax error, command unrecognized*
- 502** *Command not implemented*
- 503** *Bad sequence of commands*
- 504** *Command parameter is not implemented*
- 521** *Server does not accept mail*
- 523** *Encryption Needed*
- 530** *Authentication required*
- 534** *Authentication mechanism is too weak*
- 535** *Authentication credentials invalid*
- 554** *Message too big for system*
- 556** *Domain does not accept mail*

Email Format

- Formatting of the email is described in RFC 2822
- Think of this as what HTML is for HTTP

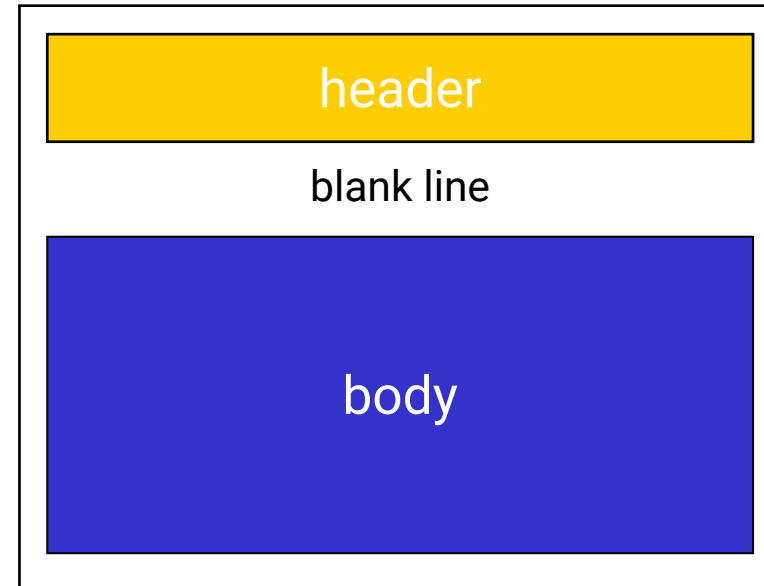
Header

- **From:** alice@uiowa.edu
- **To:** bob@illinois.edu
- **Subject:** Catch up at B1G this week?

These lines, within the body of the email message are different from SMTP's *MAIL FROM:*, *RCPT TO:* commands

Body

Actual message in ASCII format



Observations

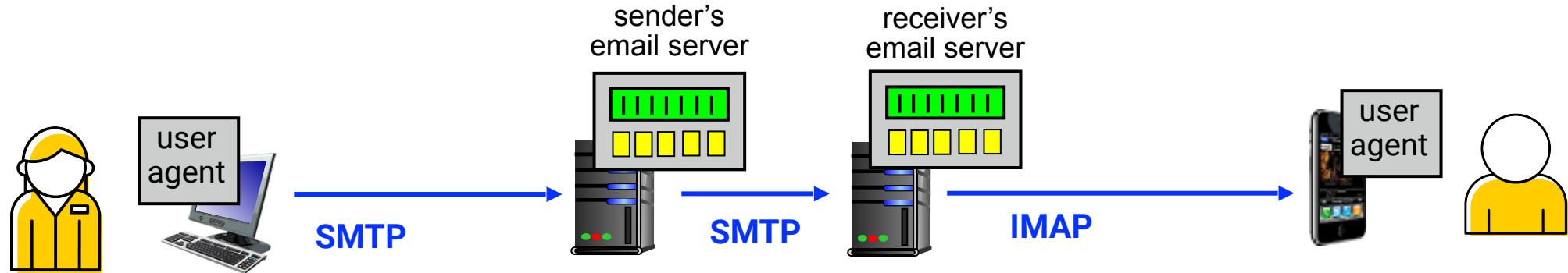
SMTP

HTTP/1.0

client pushes the content	client pulls the content
uses a persistent connection for sending multiple emails	initiates separate connections per object
requires content to be in ASCII	allows arbitrary coded content
follows a client-server model	follows a client-server model
stateless across invocations	stateless across invocations
human readable commands, headers, and status codes	human readable commands, headers, and status codes

IMAP

Retrieving Emails



SMTP is used to push e-mail messages from sender's server to receiver's server

How about sending emails to your mail server?

- Yes, SMTP could be used to push the emails

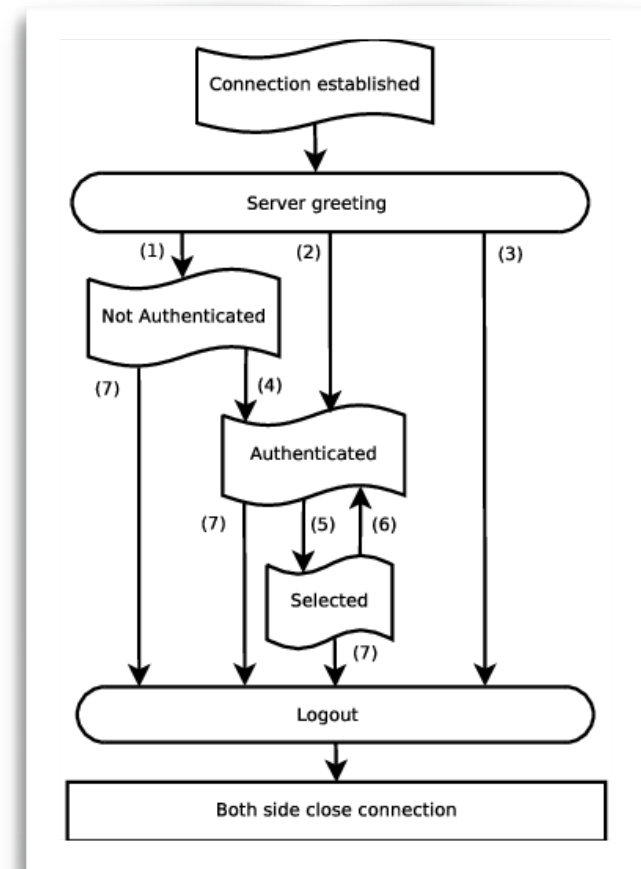
How about retrieving emails from the server?

- SMTP cannot be used \because it requires the UA to be available all the time
- Solution: **Internet Mail Access Protocol (RFC 3501)**
- Example, *Apple Mail client* or *Microsoft Outlook client*

IMAP

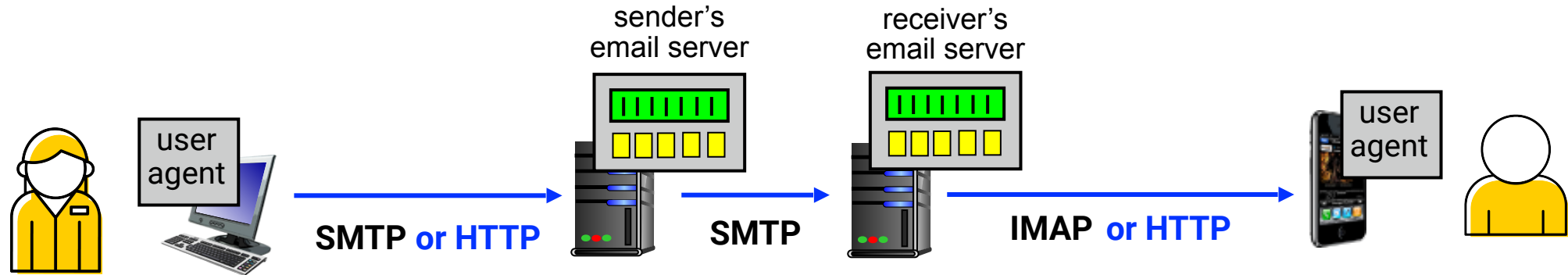
- Defined in RFC 3501 (*original RFC 1064 created in 1988*)
- **Goal:** permit a complete management of an email inbox
- Supports simultaneous access to an inbox by multiple clients. E.g., from home and office
- Uses unique mail-ids, flags to keep track of email state. E.g., whether an email has been read, replied to, or deleted
- Allows features such as server-side searches, storage management etc.
- Clients maintain persistent open TCP connections to be notified of new incoming mails

IMAPv4 state transition diagram



courtesy: Cesarini et.al., ERLANG 2008

Retrieving Emails



There is another way to retrieve emails from the server!

- Web-based Emails
- Email server also acts as an HTTP server and talks to UA, which is now an HTTP client
- But wait... can't we use HTTP interface to send mails as well?
- Sure, this is what web-based email services like *Gmail* and *Outlook365* do

Next lecture

*how to build client-server applications
that communicate using sockets*

- *Socket programming*
- *TCP sockets*
- *UDP sockets*



Chapter 2.7

Spot Quiz (ICON)