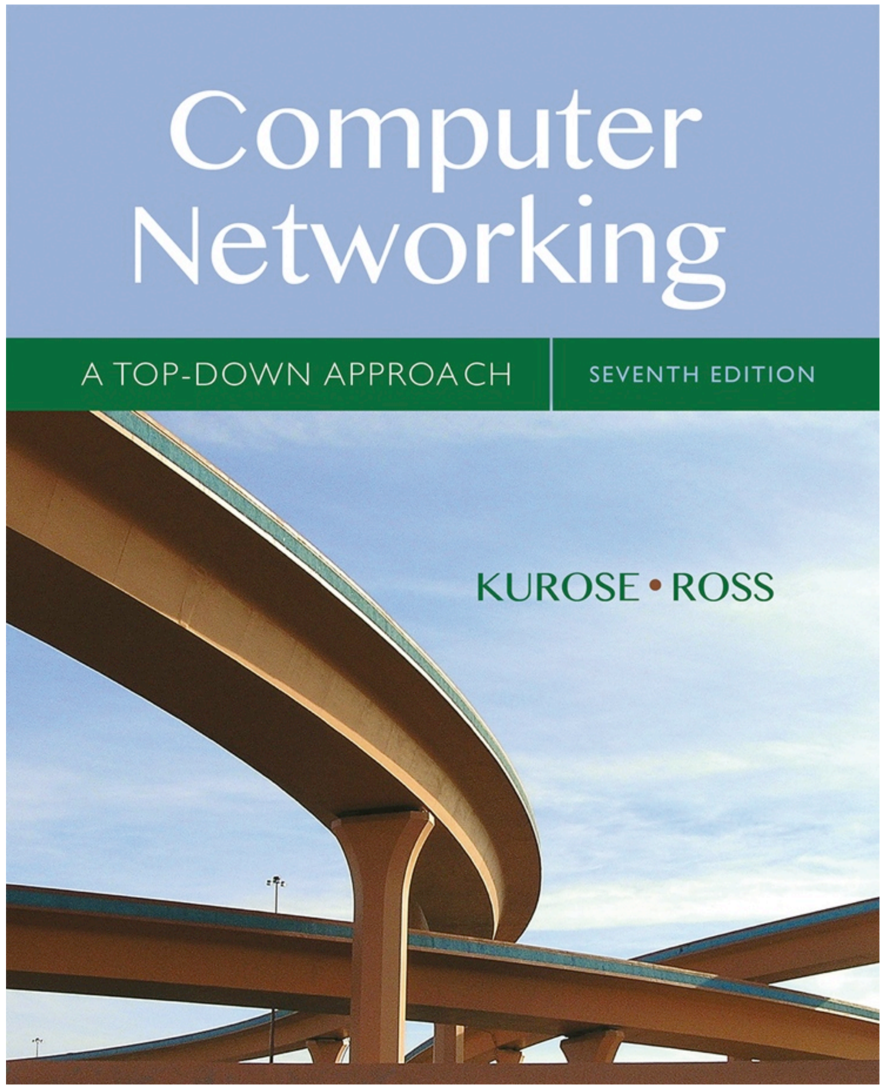
CN-Basic
L18

Electronic Mail

Dr. Ram P Rustagi
rprustagi@ksit.edu.in
http://www.rprustagi.com
https://www.youtube.com/rprustagi

Resources Acknowledgement

Chapter 2
Application Layer

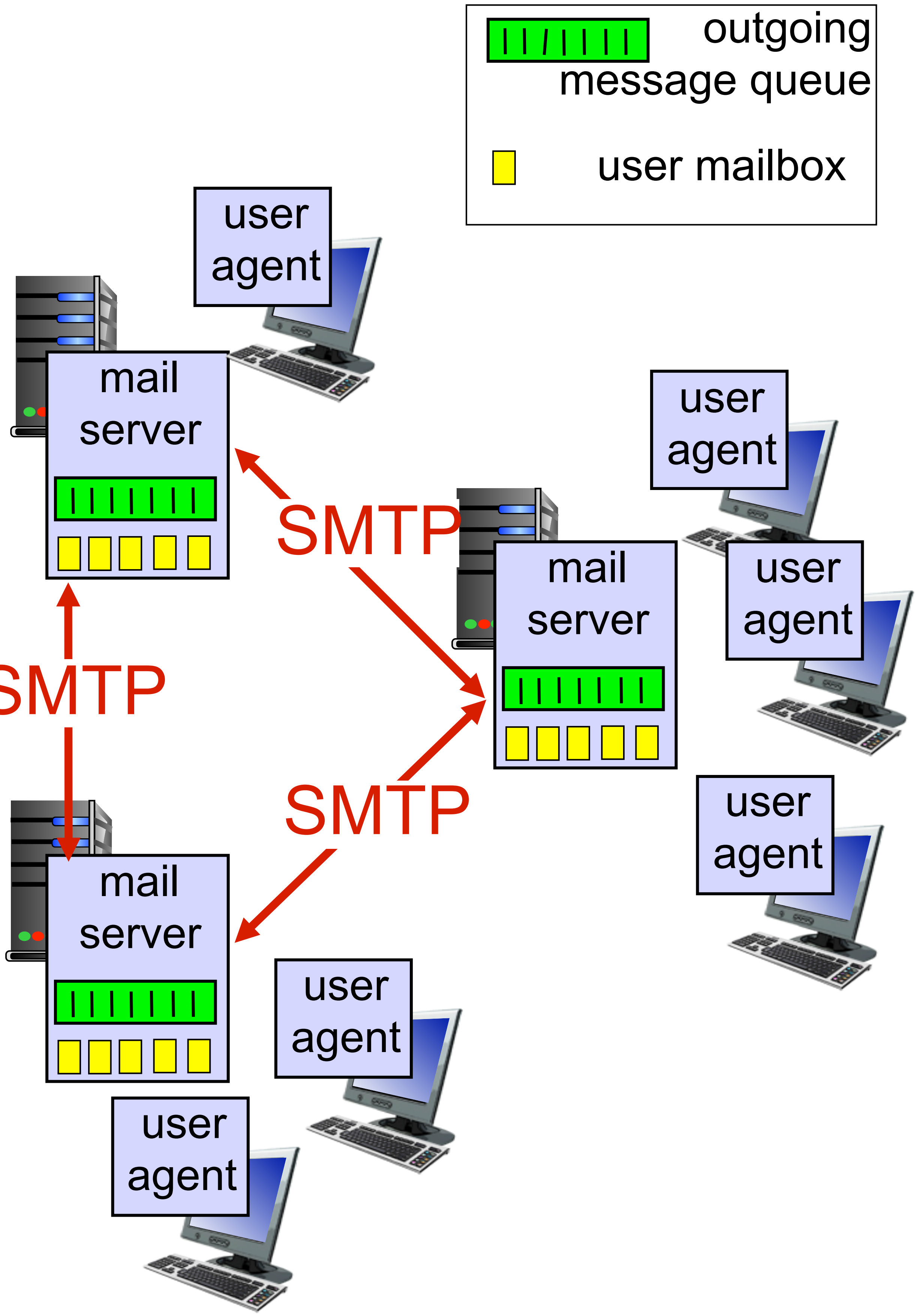


Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

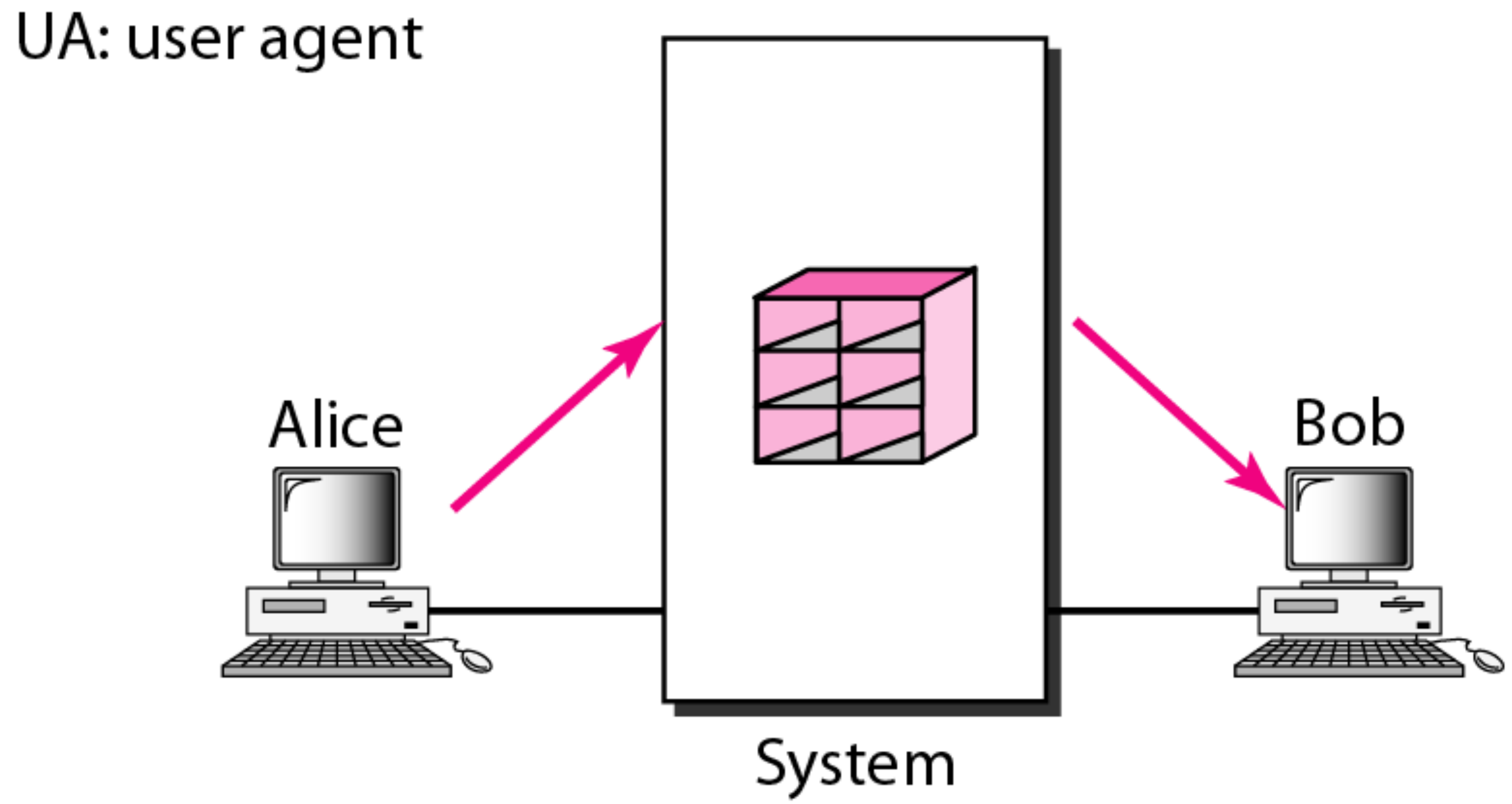
Electronic mail

- *Three major components:*
- user agents
- mail servers
- simple mail transfer protocol: SMTP (RFC 5321)
  - obsoletes RFC 2821
- *User Agent*
- a.k.a. "mail reader"
- composing, editing, reading mail messages
- e.g.: Applications: Outlook, Thunderbird, Mail(Apple)
- Outgoing, incoming messages stored on server



Email: First scenario

- Sender and Receiver on same system
- Need only two user agents

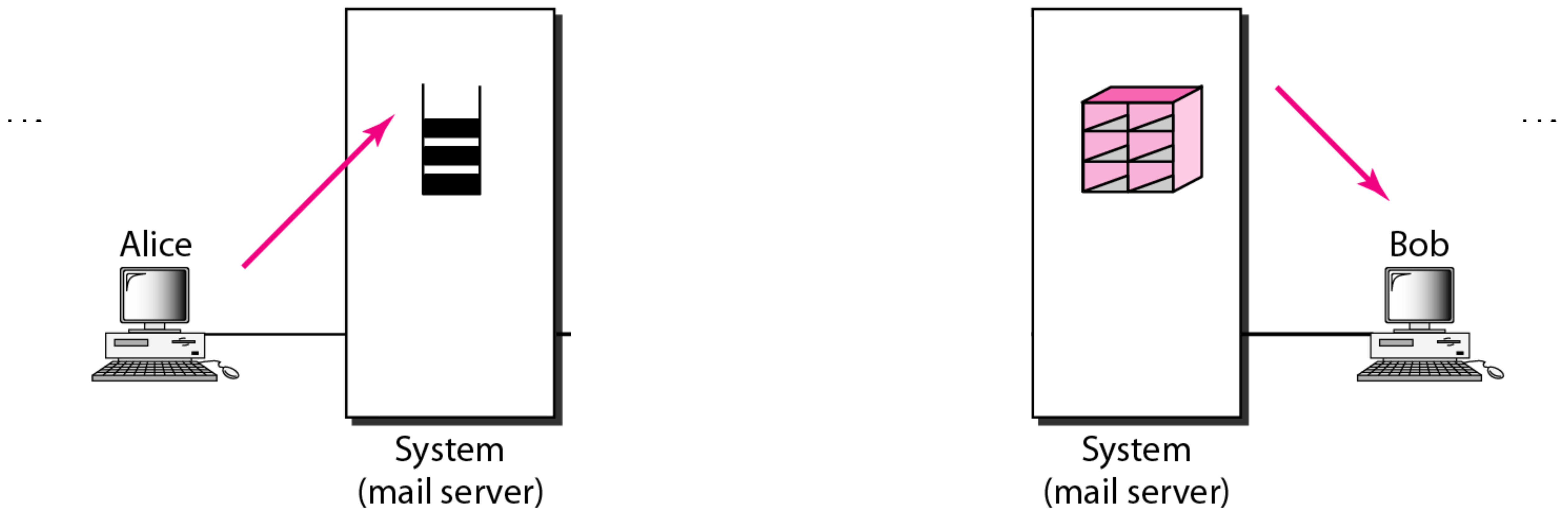


Src(fig): Forouzan - Computer Networking

Email: Second scenario

- Sender and Receiver are on different system
- Need two user agents, one pair of message transfer agent

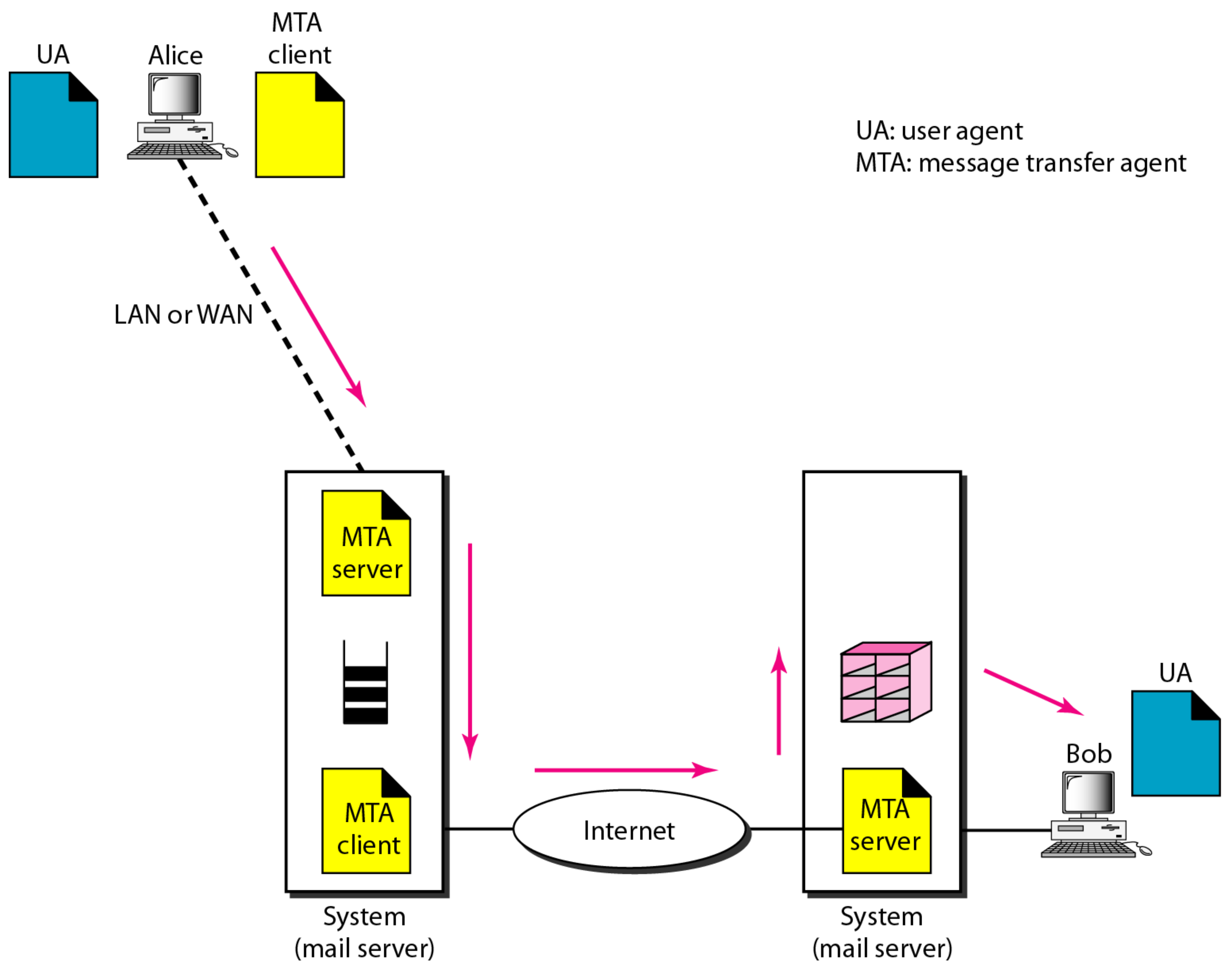
UA: user agent
MTA: message transfer agent



Src (fig): Forouzan - Computer Networking

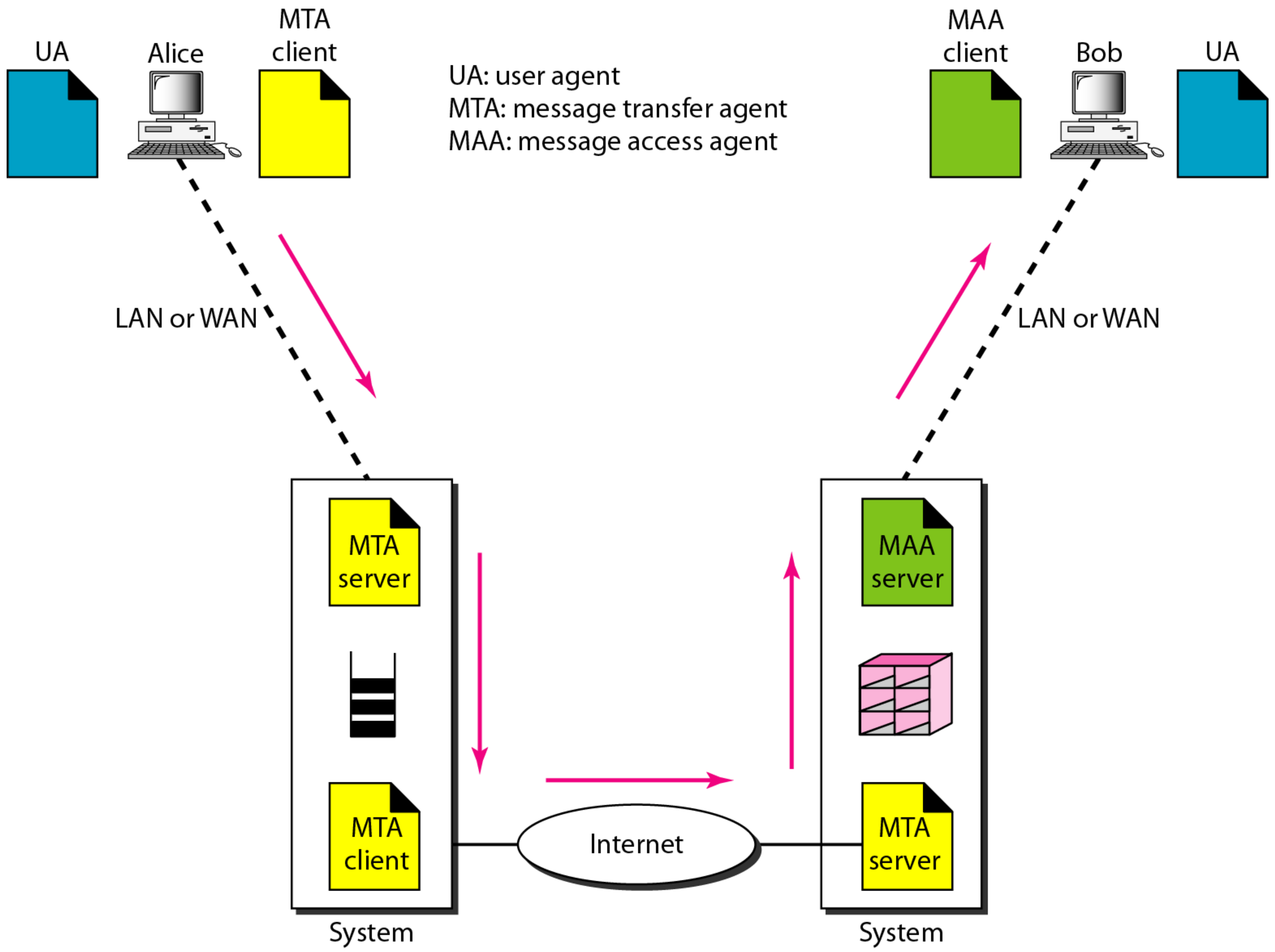
Email: Third scenario

- Sender and Receiver are on different system
  - Sender connected to mail server via LAN
- Need two user agents, two pairs of message transfer agent



Src (fig): Forouzan - Computer Networking

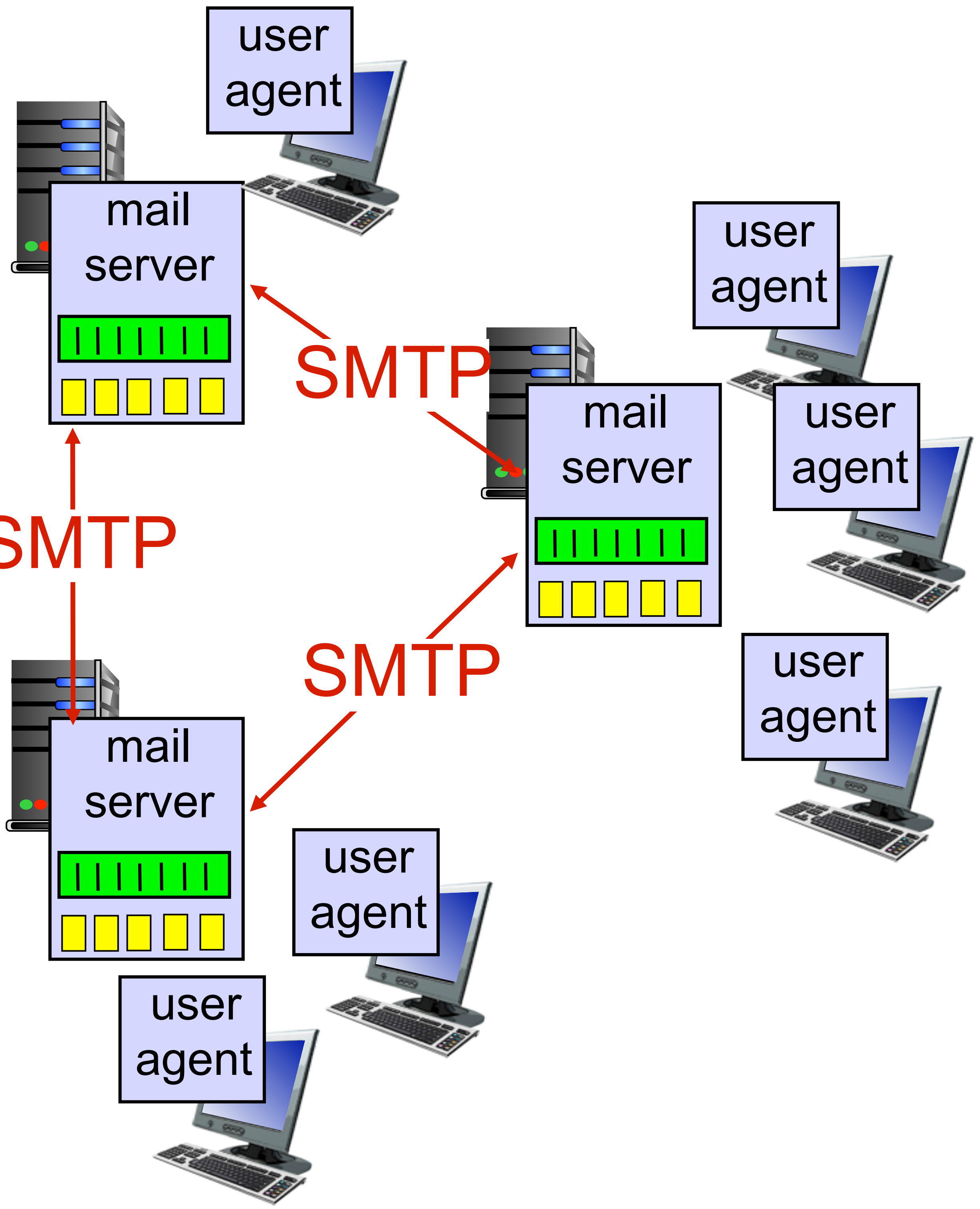
Email: Fourth scenario



Src (fig): Forouzan - Computer Networking

Electronic 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



Email - User Agent

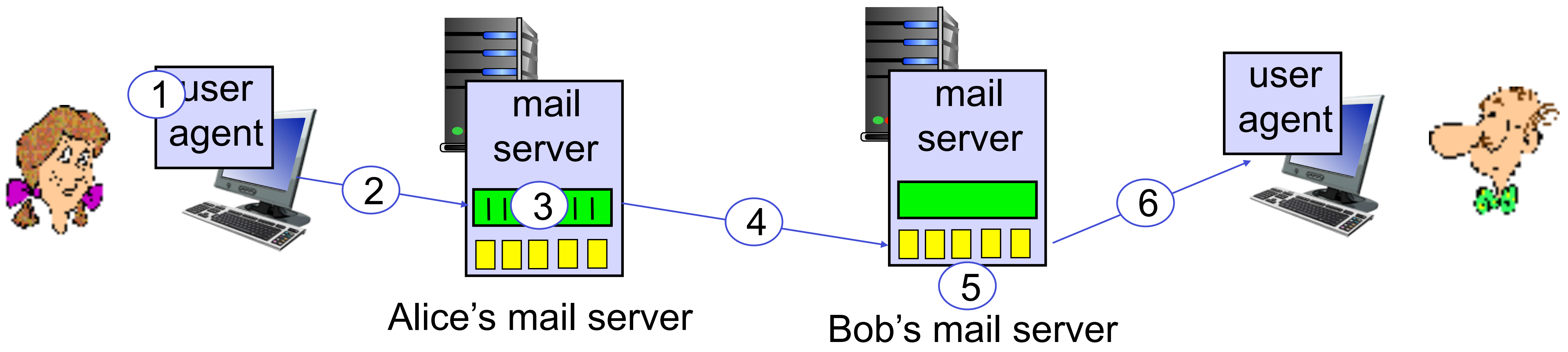
- User Agent functions
  - Compose
    - *send*
  - Read
  - Reply
  - Forward
  - Delete
  - Handling Mailboxes
- Command line UAs
  - Mail, mailx, pine,elm etc
- GUI UAs
  - iMail, Eudora, Outlook,Thunderbird etc

Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server, port 25 (unsecure)
  - secure ports (465, 587)
- direct transfer: sending server to receiving server
- 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

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

```
$ telnet 10.211.55.10 25 (#or use nc)
   Trying 10.211.55.10...
   Connected to 10.211.55.10.
   Escape character is '^]'.
   220 ubuntu.localdomain ESMTP Postfix (Ubuntu)
HELO ksit.edu.in
   250 ubuntu.localdomain
MAIL FROM: rprustagi@ksit.edu.in
   250 2.1.0 Ok
RCPT TO: dummy@ksit.local
   250 2.1.5 Ok
RCPT TO: dummyuser@ksit.local
   250 2.1.5 Ok
DATA
   354 End data with <CR><LF>.<CR><LF>
Subject: mail trial with postfix
This is a test mail to try that POP and IMAP works
with mail delivery.
   250 2.0.0 Ok: queued as 89715362D67
quit
221 2.0.0 Bye
```

Try SMTP interaction for yourself:

- **`telnet servername 25`**
- see `220` reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands
  - Current EHLO is used in place of HELO

- above lets you send email without using email client (reader)
- Sending multiple mails in a single connection
- Start with MAIL FROM after .(period)

Protocol Demonstrations

- **SMTP delivery**
- POP retrieval
- IMAP Retrieval

SMTP: and HTTP

- 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

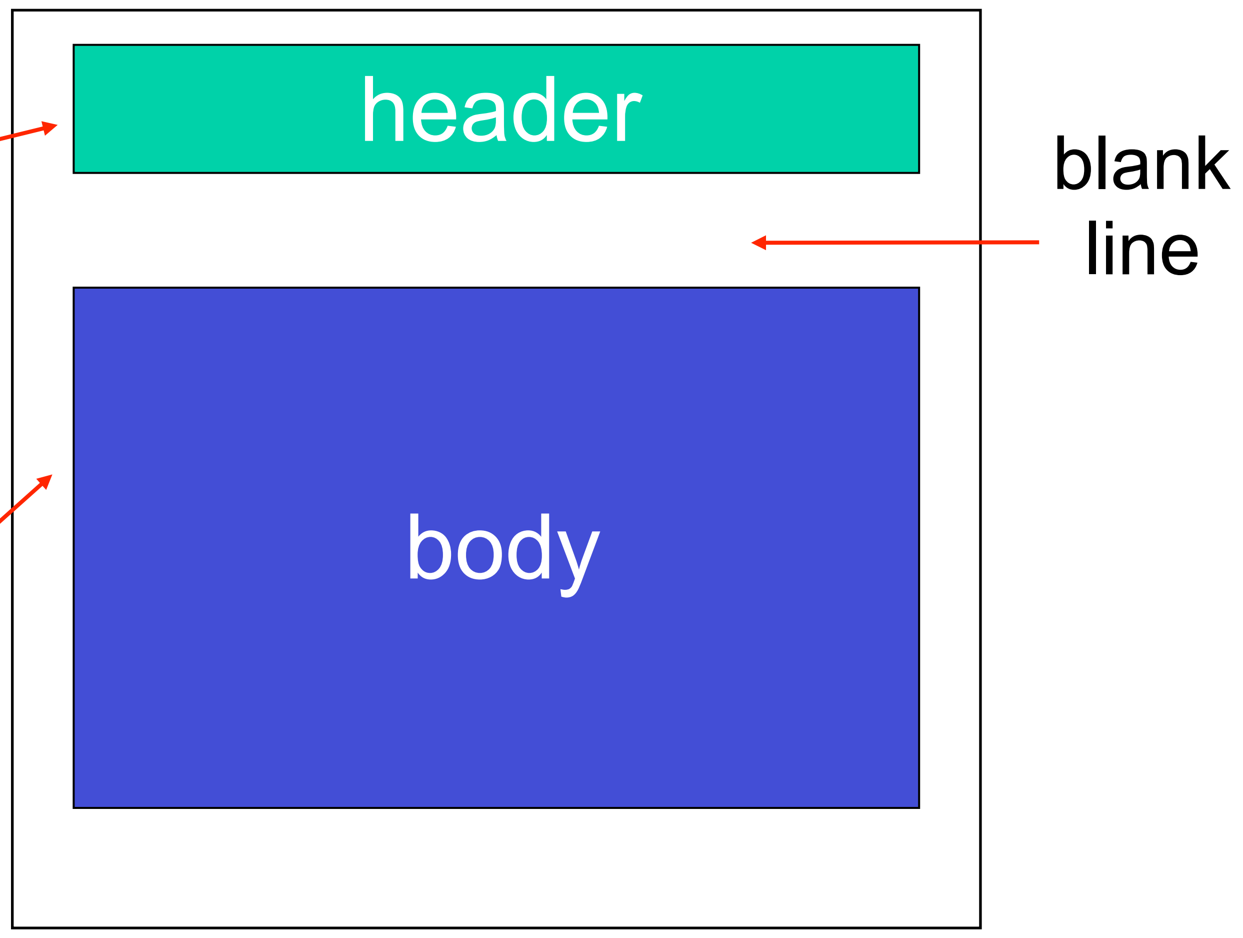
- *comparison with HTTP:*
- HTTP: pull
- SMTP: push
- both have ASCII command/ response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

SMTP : Miscellaneous details

- Secure SMTP: RFC 4954
  - Uses AUTH command
    - E.g AUTH DIGEST-MD5
    - Uses SASL (RFC 4422)
      - Simple Authentication and Secure Layer
  - Plaintext password is not to be used
- Mail Exchange (MX) Records
  - DNS provides this information
- Sendmail daemon
  - On unix systems
- Internationalized Email: RFC 6351
  - vCard

Mail message format

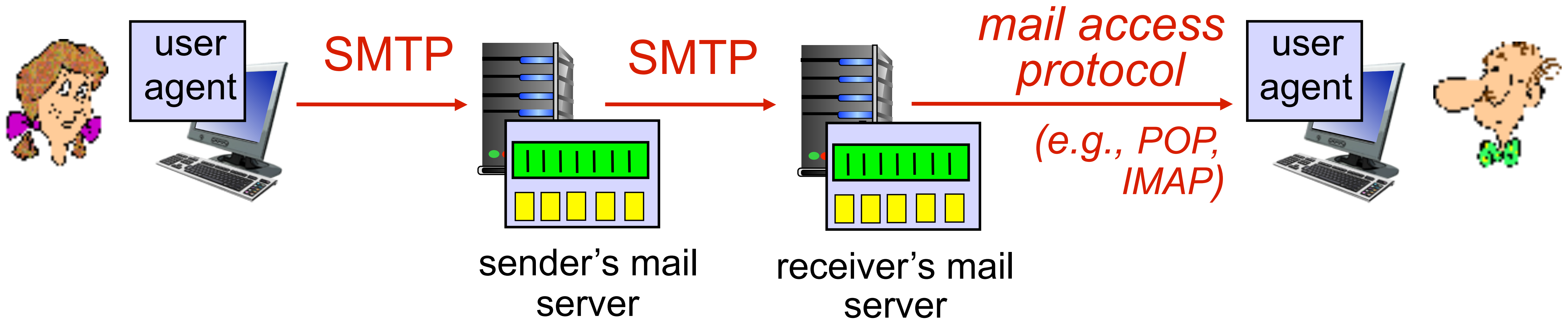
- SMTP: protocol for exchanging email msgs
- RFC 822: standard for text message format:
- header lines, e.g.,
  - To:
  - From:
  - Subject:
    - *different from* SMTP
    - MAIL FROM, RCPT TO: commands!
- Body: the "message"
  - ASCII characters only



Email

- RFC 822 (obsoleted by 2822) initially defined message structure
    - Header and body
    - Header separated by body with a blank line
    - Header is a series of lines terminated by <CR><LF>
    - Header format is **type: value**
        - To:, Subject:, Date:, Received: etc
        - More details, see RFC 822 (or 2822)
- RFC 822 extended in 1993/96
    - To carry different (than text) types of data
        - Audio, video, word docs, images,
        - MIME (Multipurpose Internet Mail Extension) format

Mail access protocols



- SMTP: delivery/storage to receiver's server
- mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 3501]: more features, including manipulation of stored msgs on server
  - HTTP: Gmail, Hotmail, Yahoo! Mail, etc.

POP3

- RFC 1939
- Basic operation
  - Server listens on port 110 (Default)
  - Client connects to the POP3 server
  - Client issues commands (case insensitive) with args
    - Typically 4 (or 3) character long keywords
  - Response consists of
    - status code, keyword, additional information
    - Two status indicators: "+OK", "-ERR"
  - Session goes thru
    - Authorization state
    - Transaction state
    - Update and goodbye

POP3 protocol

- *authorization phase*
- client commands:
  - **`user`**: declare username
  - **`pass`**: password
  - Alternatively APOP command
- 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
```

Protocol Demonstrations

- SMTP delivery
- **POP retrieval**
- IMAP Retrieval

MIME Based Email

- MIME based message structure
  - Collection of Header lines
    - Augment the original RFC822 headers
    - Describe the ways in which data is carried
  - Content type and sub-types
    - Defines how the content is to be interpreted
  - Encoding of various data types
    - Enables sending of data as an ASCII email msg
    - Intermediate gateways require
      - That email content be ASCII
    - Base64:
      - 3 bytes of data represented in 4 bytes

MIME Based Email

- Email Header
  - MIME-Version: 1.1
  - Content-Type: <type/subtype>
  - Content-Transfer-Encoding: <encoding type>
    - 7-bit, 8-bit, Base64, Binary, Quoted printable
  - Content-Id: <message id>
  - Content-Description: textual explanation
- Examples:
  - `Ch02-SMTP-attachment.pcap`
  - Look at MIME headers

Base64 encoding

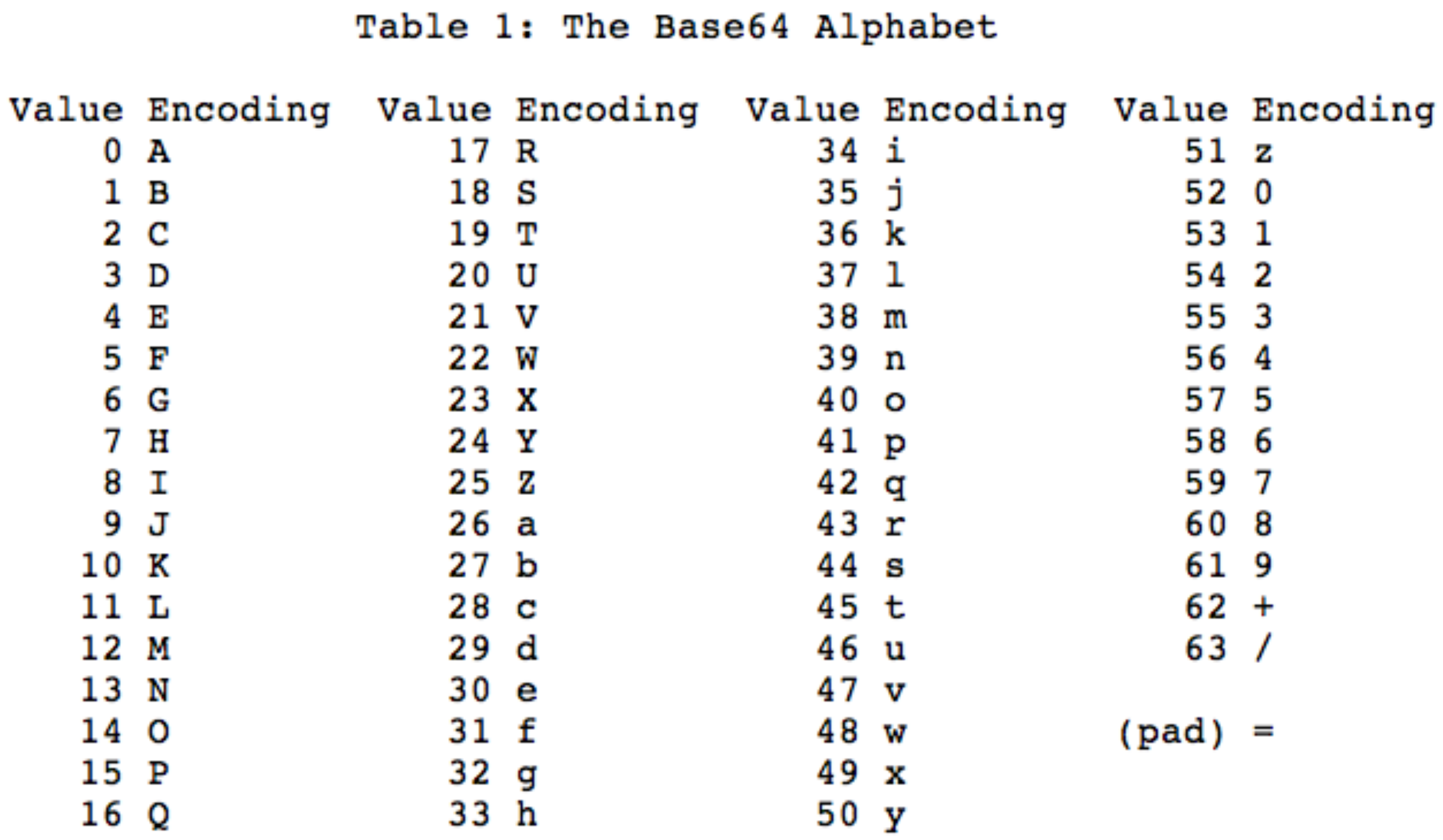


Table 1: The Base64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Src: RFC 2045

Installing Mail programs

- On Ubuntu
  - https://help.ubuntu.com/community/PostfixBasicSetupHowto

  - `sudo apt-get install mailutils`
  - `sudo apt-get install courier-pop`
  - `sudo apt-get install courier-imap`

POP3 protocol

- Examples
    - `Ch02-POP-user-passed.pcap`
    - `Ch02-POP3-APOP-NoMail.pcap`
    - `Ch02-POP-APOP-Retr-onemail.pcap`
    - `Ch02-POP-No-auth-CAPA.pcap`

POP3 (more) and IMAP

- *more about POP3*
- 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

- *IMAP (RFC 3501)*
- 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

POP3 vs IMAP4

- POP3 deficiencies
  - Does not allow user to organize mails on server
    - User can not have different folders on server
    - Use can create folders on local system
  - Does not allow partial email content check
- IMAP4 features
  - User can check email header prior to download
  - User can search the contents prior to download
  - Allows partial download (MIME multipart)
  - User can create, delete, rename mailboxes
    - can create a hierarchy of mailboxes
- Web based email
  - Using HTTP with a email website

Summary

- User Agent
- Mail agent
- Mail transfer protocol
  - SMTP, POP3, IMAP4
- Mail authorization
- ASCII text
- MIME email
  - uses Base64 encoding
  -