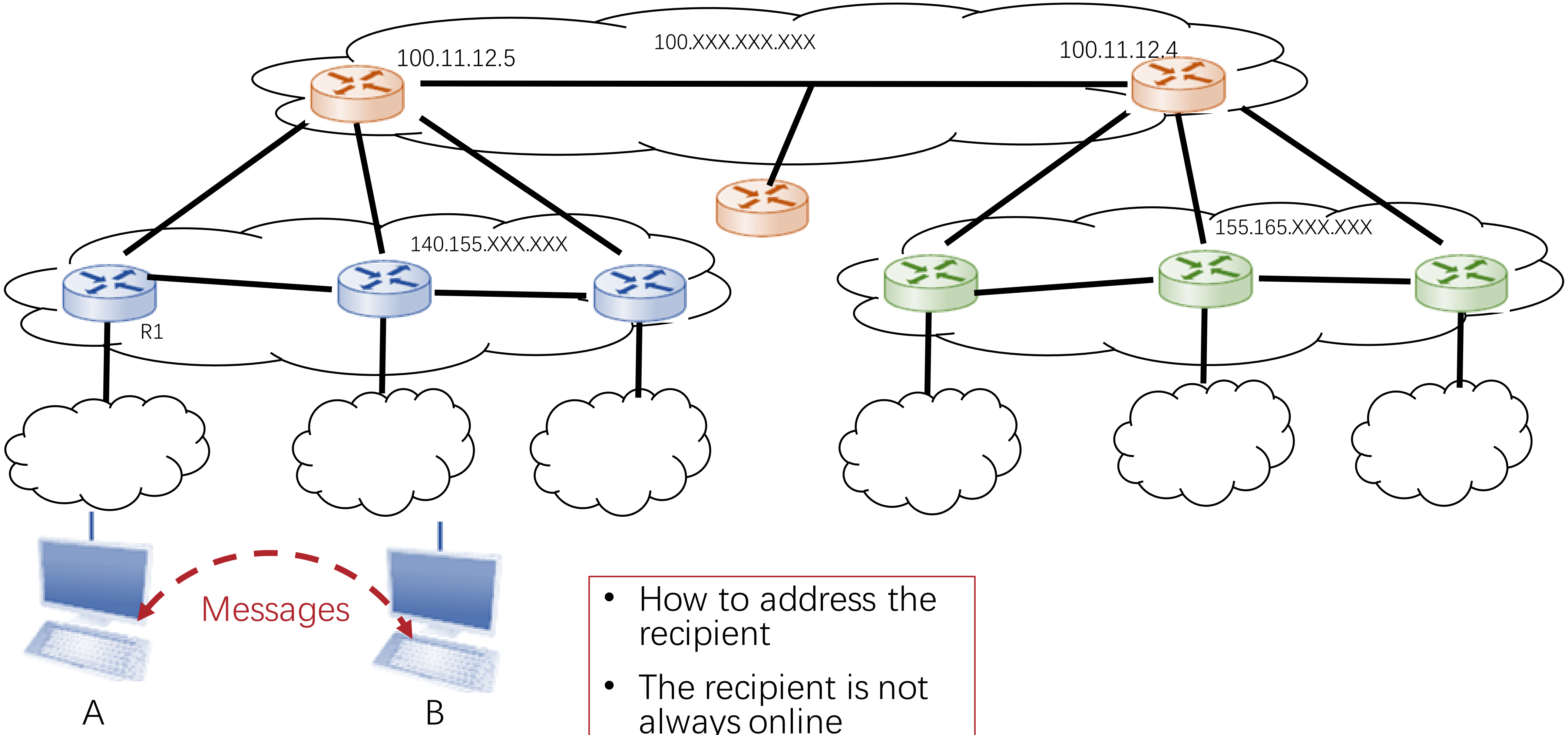
CS120: Computer Networks

Lecture 25. Email & Web

Haoxian Chen

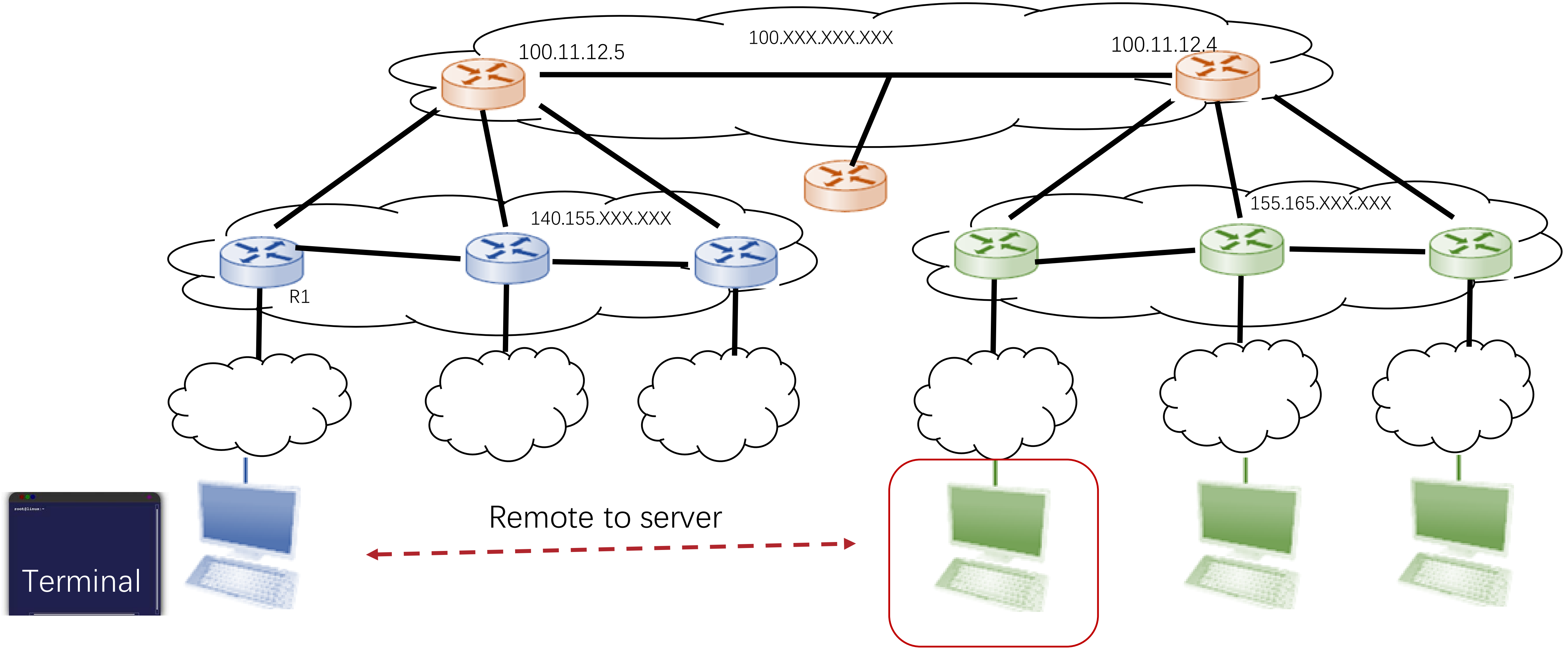
Slides adopted from: Zhice Yang

Mail Over Network ?



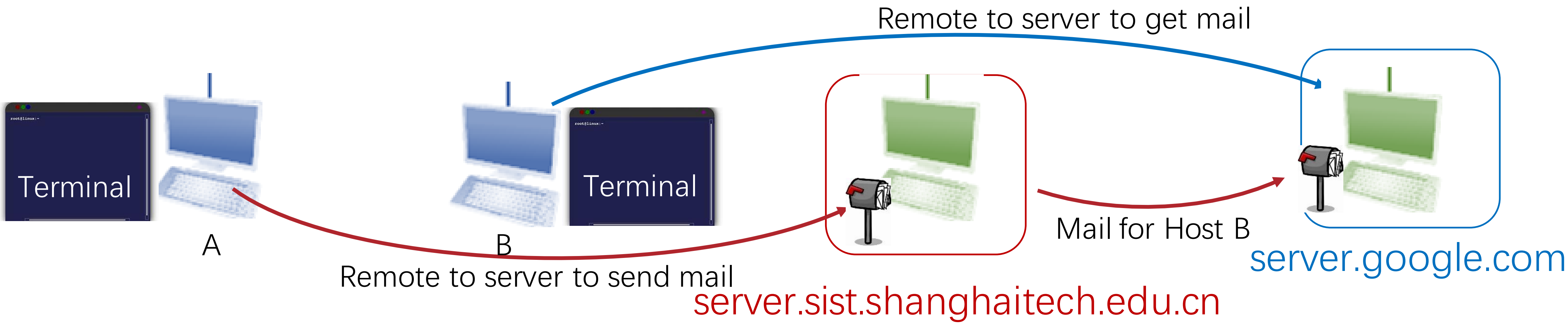
- How to address the recipient
- The recipient is not always online

Telnet – Remote Command Line Access

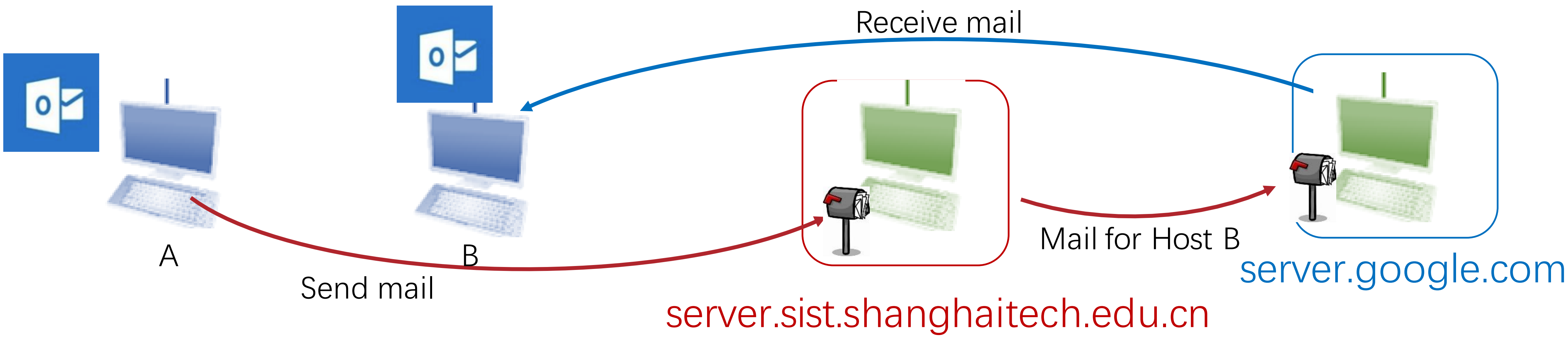


server.sist.shanghaitech.edu.cn

Electronic Mail (Email)

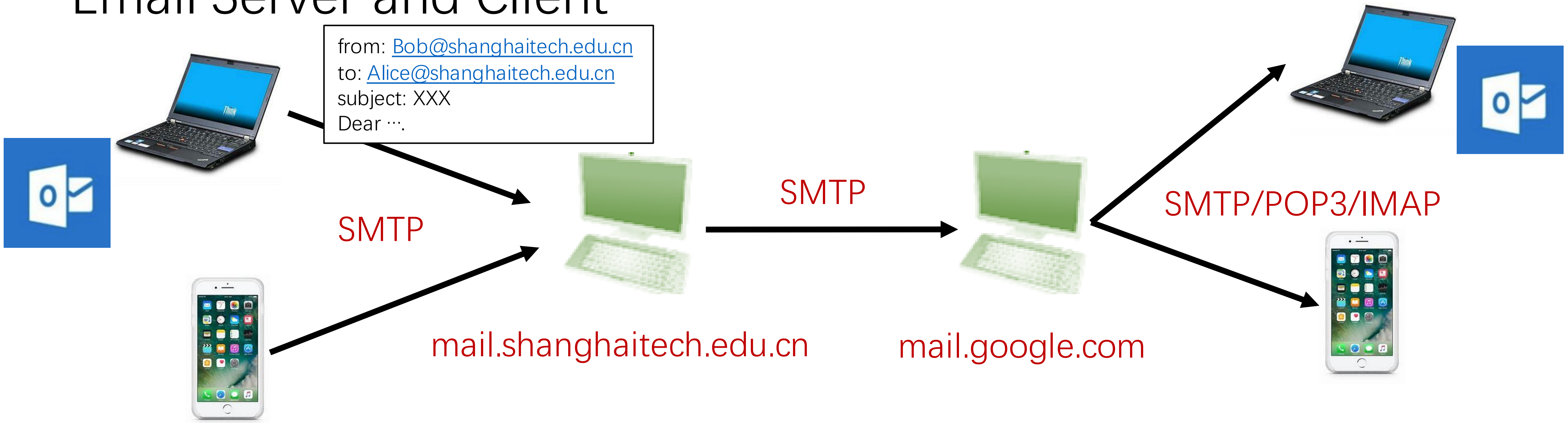


Electronic Mail (Email) via Client Application



Email

- Email Format
  - Multipurpose Internet Mail Extensions (MIME)
- Email Protocols
  - Simple Mail Transfer Protocol (SMTP)
  - Internet Message Access Protocol (IMAP)
- Email Server and Client



Email Format

- Original Email Messages are pure ASCII Text
  - RFC 822
  - Extended by Multipurpose Internet Mail Extensions (MIME)

Email Format

- Header
  - Version, Boundary, FROM, TO, SUBJECT, DATE, etc.
- Body
  - Content Type
    - e.g., image/jpeg, text/plain
  - Content Encoding
    - 7bit ASCII for text
    - Base64 for non-text
      - Map 3-bytes to 4-bytes ASCII
      - To be compatible with old email devices

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="-------417CA6E2DE4ABCAFBC5"
From: Alice Smith <Alice@cisco.com>
To: Bob@cs.Princeton.edu
Subject: promised material
Date: Mon, 07 Sep 1998 19:45:19 -0400
---------417CA6E2DE4ABCAFBC5
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bob,
Here's the jpeg image and draft report I promised.
--Alice
---------417CA6E2DE4ABCAFBC5
Content-Type: image/jpeg
Content-Transfer-Encoding: base64
... unreadable encoding of a jpeg figure
---------417CA6E2DE4ABCAFBC5
Content-Type: application/postscript; name="draft.ps"
Content-Transfer-Encoding: 7bit
... readable encoding of a PostScript document
```

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ref: https://commons.wikimedia.org/wiki/File:ASCII-Table-wide.pdf

Base64

Encode binary into text.

Contains only printable characters: avoid control characters.

Base64 alphabet defined in RFC 4648.

Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/
Padding		=									

ref: https://en.wikipedia.org/wiki/Base64

Encode binary into Base64

Encode every 3 bytes (24 bits) data into 4 6-bit Base64 representation.

Encoding of source characters M, a, n in Base64.

<table>
<tr><th rowspan="2">Source</th><th>Text (ASCII)</th><td colspan="8">M</td><td colspan="8">a</td><td colspan="8">n</td></tr>
<tr><th>Octets</th><td colspan="8">77 (0x4d)</td><td colspan="8">97 (0x61)</td><td colspan="8">110 (0x6e)</td></tr>
<tr><th colspan="2">Bits</th><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><th rowspan="3">Base64 encoded</th><th>Sextets</th><td colspan="6">19</td><td colspan="6">22</td><td colspan="6">5</td><td colspan="6">46</td></tr>
<tr><th>Character</th><td colspan="6">T</td><td colspan="6">W</td><td colspan="6">F</td><td colspan="6">u</td></tr>
<tr><th>Octets</th><td colspan="6">84 (0x54)</td><td colspan="6">87 (0x57)</td><td colspan="6">70 (0x46)</td><td colspan="6">117 (0x75)</td></tr>
</table>

ref: https://en.wikipedia.org/wiki/Base64

Encode binary into Base64

Encode every 3 bytes (24 bits) data into 4 6-bit Base64 representation.

Padding to make the last block contains 4 Base64 codes.

<table>
<tr><th rowspan="2">Source</th><th>Text (ASCII)</th><td colspan="8">M</td><td colspan="8">a</td><td colspan="8" rowspan="2"></td></tr>
<tr><th>Octets</th><td colspan="8">77 (0x4d)</td><td colspan="8">97 (0x61)</td></tr>
<tr><th colspan="2">Bits</th><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><th rowspan="3">Base64 encoded</th><th>Sextets</th><td colspan="6">19</td><td colspan="6">22</td><td colspan="6">4</td><td colspan="6">Padding</td></tr>
<tr><th>Character</th><td colspan="6">T</td><td colspan="6">W</td><td colspan="6">E</td><td colspan="6">=</td></tr>
<tr><th>Octets</th><td colspan="6">84 (0x54)</td><td colspan="6">87 (0x57)</td><td colspan="6">69 (0x45)</td><td colspan="6">61 (0x3D)</td></tr>
</table>

ref: https://en.wikipedia.org/wiki/Base64

Email Format

- Demo
  - Show Email in original/plaintext format
  - Decode Base64 content
    - https://codebeautify.org/base64-to-image-converter#

Email transfer

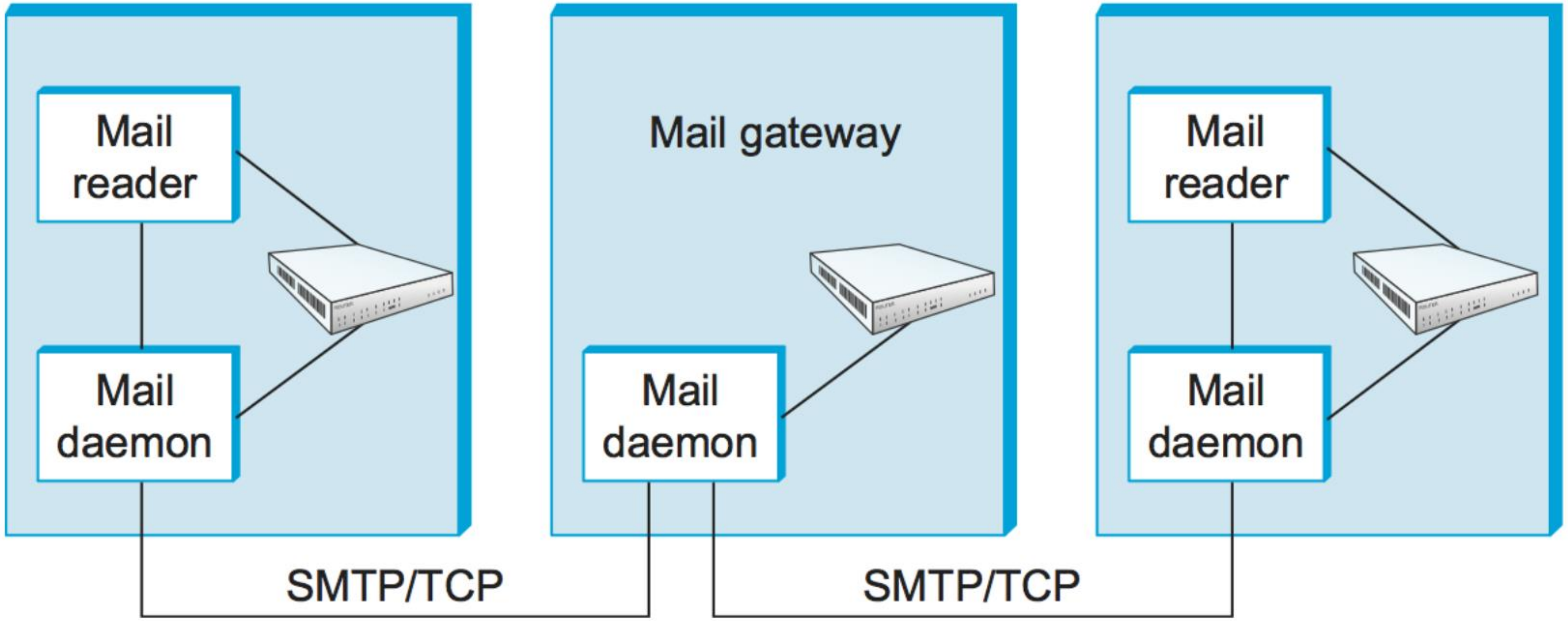


Figure 9.1.: Sequence of mail gateways store and forward email messages.

Why email gateway is necessary?

- Scalability.
  - Large organization has multiple machines to host mailboxes.
- Host may not always up and reachable.
  - Mail gateway holds a message until it can be delivered.
  - May buffer for up to several days.

Email Protocol

hxchen@shanghaitech.edu.cn

$\underbrace{\text{hxchen}}_{\text{Mail Recipient}}$@$\underbrace{\text{shanghaitech.edu.cn}}_{\text{Mail Server}}$

- SMTP
  - Use DNS to find IP of the email server
    - According the domain name after @
  - Use TCP to transfer email messages, port 25
    - Between client and server
    - Between servers
      - Mail server might temporarily store email until the receiver server is ready
      - Mail server supports email relay, i.e., an email usually passes through several email servers
  - Command/response interaction
    - Commands: ASCII text
    - Response: status code and phrase
  - Email Message
    - Format is defined by MIME

Email Protocol

- SMTP Example:
  - Connect email servers through telnet
    - mail.shanghaitech.edu.cn:25

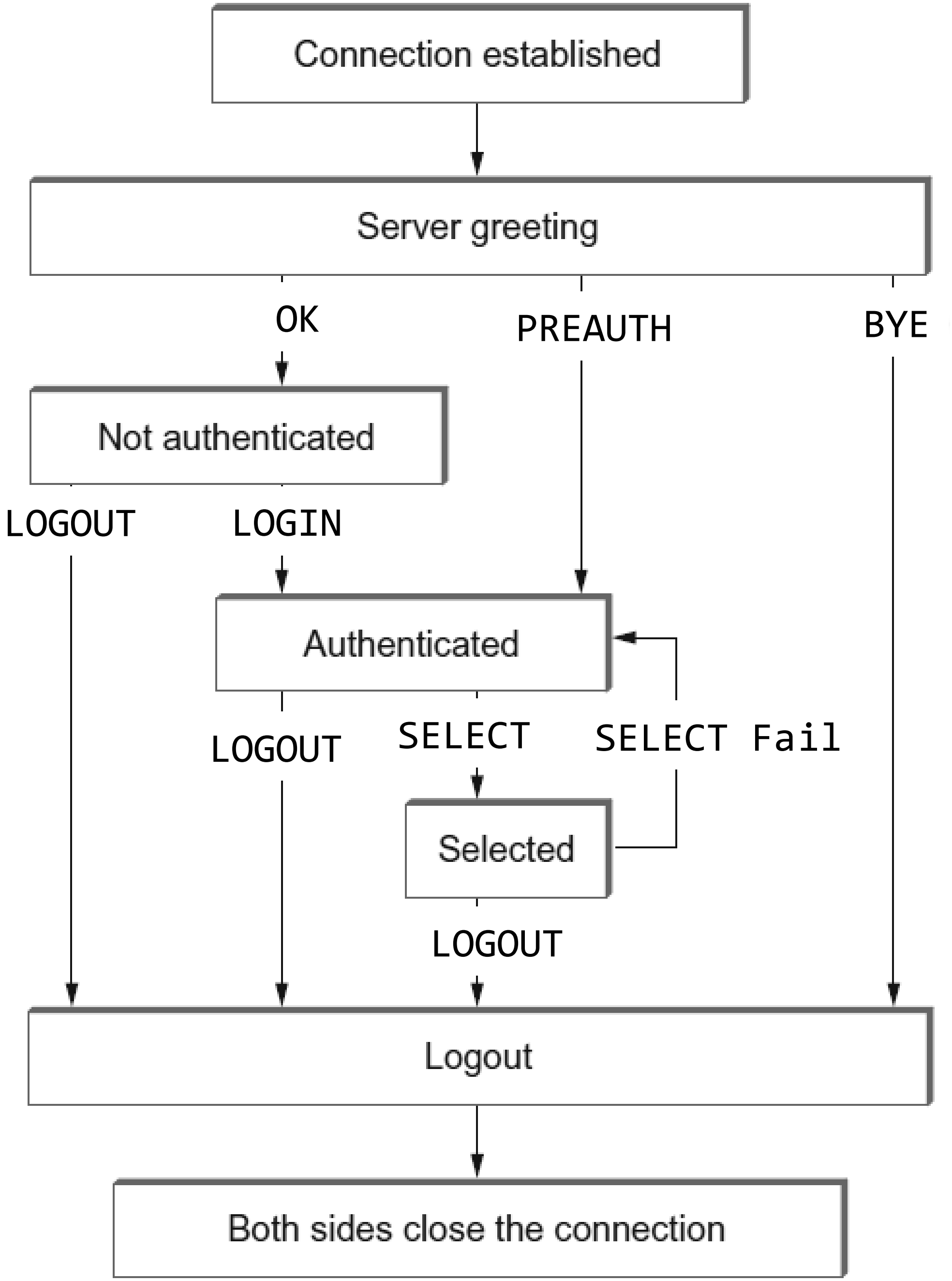
```
HELO cs.princeton.edu
250 Hello daemon@mail.cs.princeton.edu [128.12.169.24]

MAIL FROM:<Bob@cs.princeton.edu>
250 OK

RCPT TO:<Alice@cisco.com>
250 OK

RCPT TO:<Tom@cisco.com>
550 No such user here

DATA
354 Start mail input; end with <CRLF>.<CRLF>
Blah blah blah...
...etc. etc. etc.
<CRLF>.<CRLF>
250 OK

QUIT
221 Closing connection
```

Email Protocol

- Mail Access Protocol: retrieve email from server
  - POP: Post Office Protocol [RFC 1939]
  - IMAP: Internet Mail Access Protocol [RFC 1730]
  - HTTP: Access Mail Server via Webpage.

Email Protocol

- IMAP
  - Use TCP to transfer email from server to client, port 143
  - Similar to SMTP
    - Command/response Interaction
    - Additional Commands:
      - LOGIN, AUTHENTICATE, FETCH, DELETE, etc.



Email Implementation

- Mail Client
  - Composing, Editing, Reading mail messages
  - Outlook, iPhone mail client,

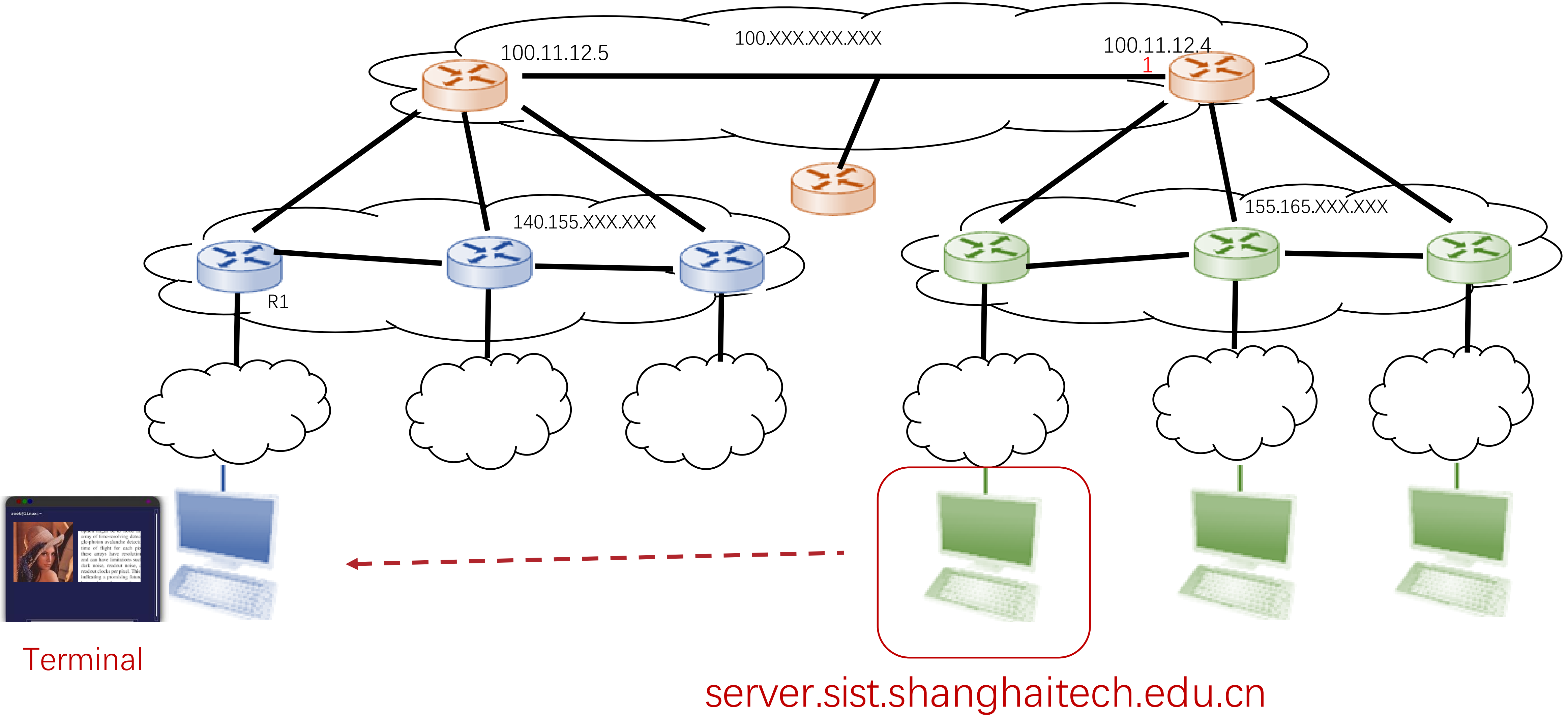
Email Implementation

- Mail Server (Mail Daemon)
  - Receive and store emails for client
  - Send email to other email servers
  - Implementation
    - e.g.: sendmail, postfix, and a lot more.

mail.shanghaitech.edu.cn

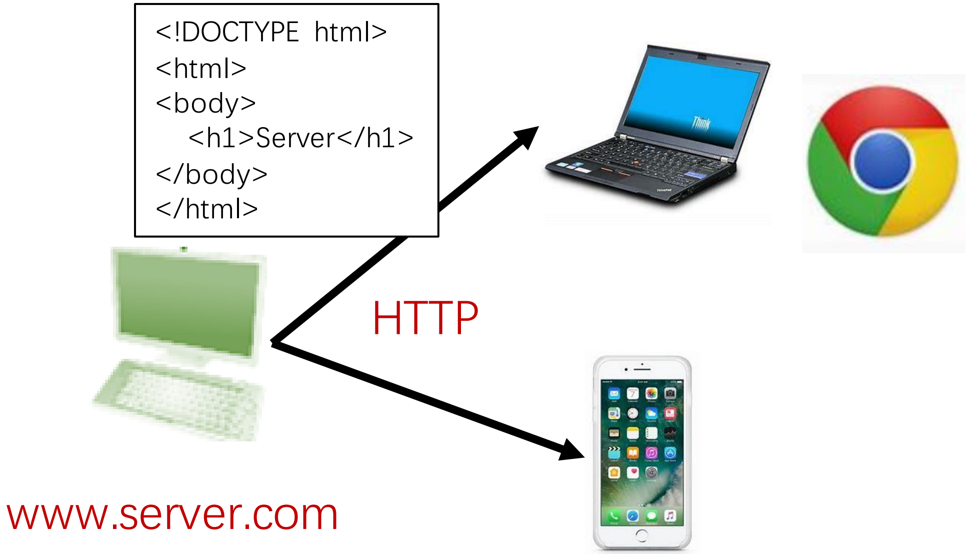
mail.google.com

Transmit Information beyond Text ?



World Wide Web (WWW)

- Web Page Format
  - Hypertext Markup Language (HTML)
- Web Server and Browser
- Web Protocol
  - HyperText Transfer Protocol (HTTP)



Web Page Format

ssist.shanghaitech.edu.cn/upload/image/xxx.png

Hostname | Resource Path

- Web page is more than text
  - “Hypertext”
  - Web page consists of objects
    - Object can be HTML file, JPEG image, Java applet, audio file,…
      - e.g.,: index.html, XXX.png, etc.
    - The HTML-file describes the referenced objects
    - Each object is addressable by a Uniform Resource Locator (URL)

HTML File

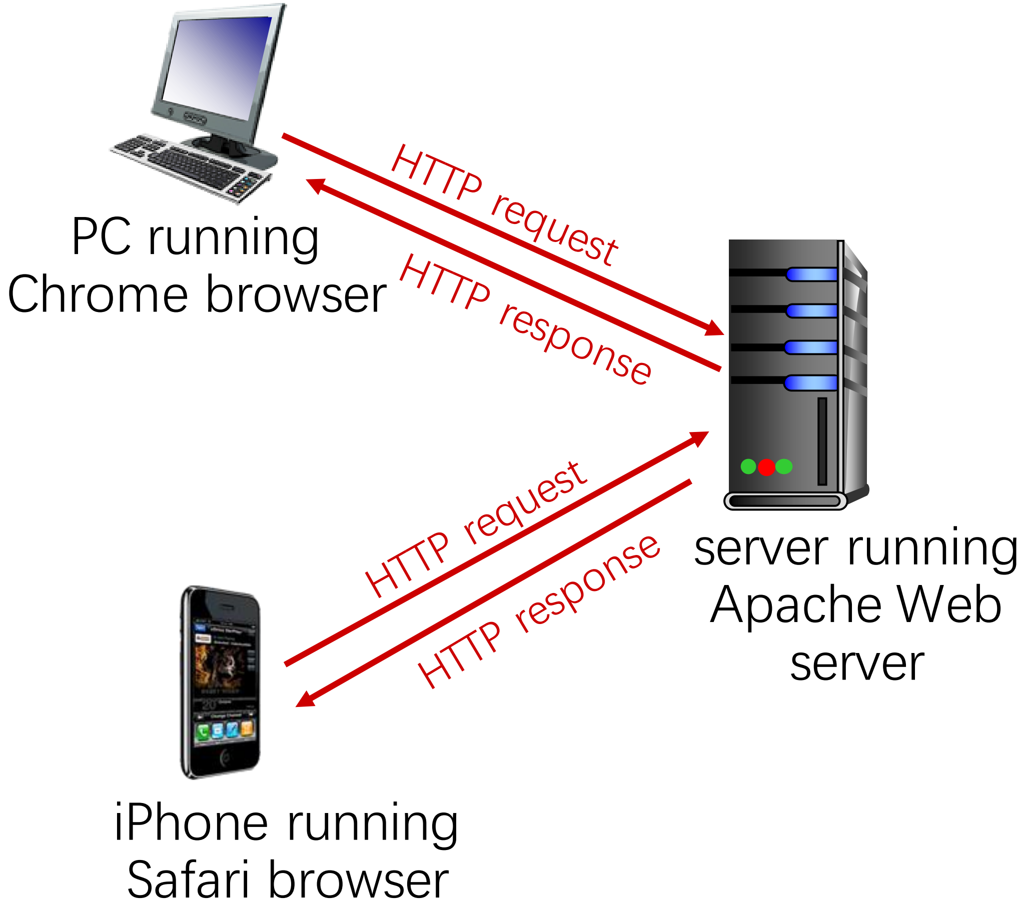
```html
<!doctype html>
<html itemscope itemtype="http://schema.org/WebPage" lang="en-DE">
▶<head>…</head>
▼<body class="hp vasq" id="gsr">
    <meta content="Happy Holidays!" property="twitter:title">
    <meta content="Happy Holidays #GoogleDoodle" property="twitter:description">
    <meta content="Happy Holidays #GoogleDoodle" property="og:description">
    <meta content="summary_large_image" property="twitter:card">
    <meta content="@GoogleDoodles" property="twitter:site">
    <meta content="https://www.google.com/logos/doodles/2018/holidays-2018-northern-
    hemisphere-day-2-5676669204430848-2xa.gif" property="twitter:image">
```

Web Page Format

- Try Simple HTML
  - https://www.w3schools.com/html/html_examples.asp
- View HTML Source in Browser
  - F12

Web Server and Browser

- Web Browser: request, receive, and "displays" web objects according to the received HTML file
  - Edge, Firefox, Chrome, etc.
- Server: Send objects in response to requests
  - Apache, Nginx, etc.



HyperText Transfer Protocol (HTTP)

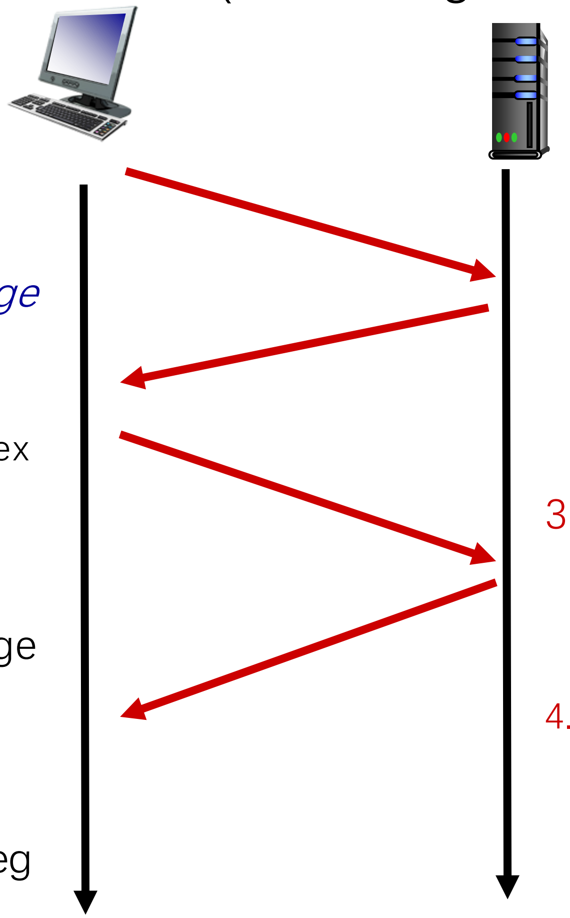
- Client/Server Model
  - Similar to SMTP
- Use TCP, Port 80
  - Client initiates TCP connection to server
  - Server accepts TCP connection from client
  - Exchange HTTP messages
  - Close TCP connection

HTTP Evolution

- Non-persistent HTTP
  - One Object on TCP connection
- Persistent HTTP (HTTP 1.1)
  - Multiple objects Single Connection
- HTTP/2
  - Transmission order of requested objects based on client-specified object priority
- HTTP/3
  - i.e., QUIC

Non-persistent HTTP Example

User enters URL: **www.someSchool.edu/someDepartment/home.index**
(containing text, references to 10 jpeg images)



1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80 “accepts” connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

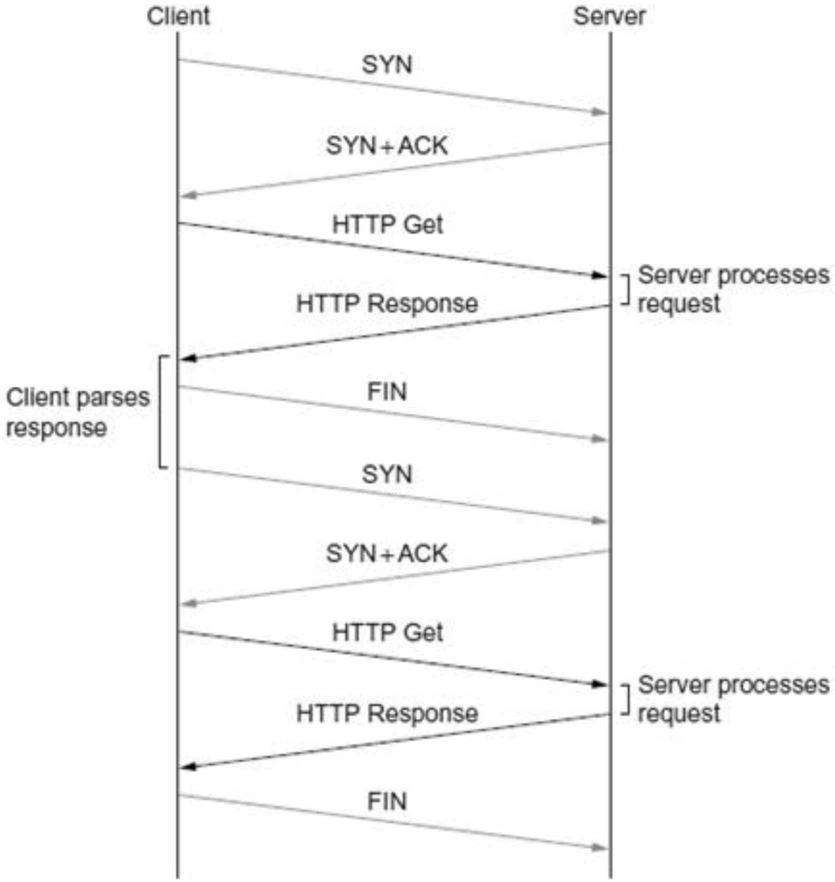
6. Steps 1-5 repeated for each of 10 jpeg objects

Persistent HTTP (HTTP 1.1)

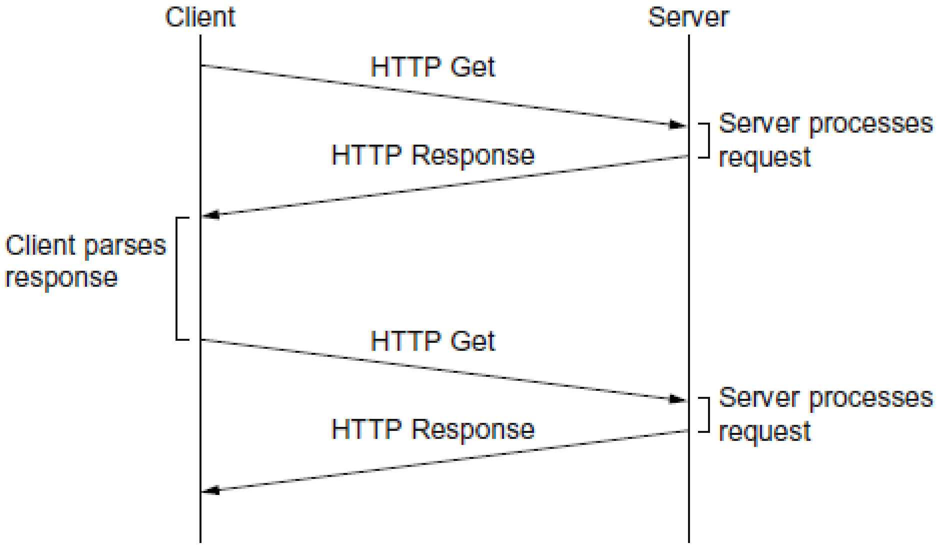
- Non-persistent HTTP issues:
  - Requires 2 RTTs per object
  - OS overhead for each TCP connection
  - Browsers often open multiple parallel TCP connections to fetch referenced objects in parallel
- Persistent HTTP (HTTP1.1):
  - Server leaves connection open after sending response
  - Subsequent HTTP messages between same client/server sent over open connection
  - Client sends requests as soon as it encounters a referenced object
  - As little as one RTT for all the referenced objects (cutting response time in half)

Persistent HTTP (HTTP 1.1)

- Non-persistent HTTP



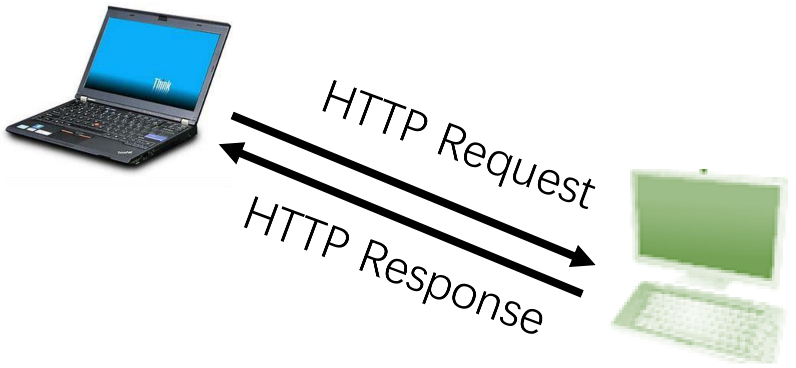
- Persistent HTTP



HTTP Messages

- Like SMTP, HTTP is Text-oriented
- Two types of HTTP messages
  - Request
  - Response
- Message Format

```
START_LINE <CRLF>
MESSAGE_HEADER <CRLF>
<CRLF>
MESSAGE_BODY <CRLF>
```



HTTP Request Format

START_LINE

```
GET /2018/ HTTP/1.1\r\n
Host: sist-admission.shanghaitech.edu.cn\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWe
Accept: text/html,application/xhtml+xml,application/xml;q=0.9
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9\r\n
```

MESSAGE_HEADER

MESSAGE_BODY is normally empty for request

Table 9.1 HTTP Request Operations

Operation	Description
OPTIONS	Request information about available options
GET	Retrieve document identified in URL
HEAD	Retrieve metainformation about document identified in URL
POST	Give information (e.g., annotation) to server
PUT	Store document under specified URL
DELETE	Delete specified URL
TRACE	Loopback request message
CONNECT	For use by proxies

Some Other HTTP Request Messages

- POST:
  - web page often includes form input
  - Include user data in message body
- GET(for sending data to server):
  - Include user data in URL field of HTTP GET request message (following a ‘?’)
  - e.g., www.somesite.com/animalsearch?monkey
- HEAD:
  - Requests headers (only)
  - For checking header without retrieving the content
- PUT:
  - Uploads new file (object) to server
  - Replaces file that exists

HTTP Response

START_LINE

```
HTTP/1.1 200 OK\r\n
Date: Tue, 29 May 2018 17:38:51 GMT\r\n
Server: Apache/2.4.7 (Ubuntu)\r\n
X-Powered-By: PHP/5.5.9-1ubuntu4.20\r\n
Cache-Control: max-age=0,must-revalidate,private\r\n
Vary: Accept-Encoding\r\n
Content-Encoding: gzip\r\n
Content-Length: 3076\r\n
Keep-Alive: timeout=5, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=UTF-8\r\n
\r\n
[HTTP response 1/2]
[Time since request: 0.019359000 seconds]
[Request in frame: 726]
[Next request in frame: 770]
[Next response in frame: 771]
Content-encoded entity body (gzip): 3076 bytes -> 7204 bytes
File Data: 7204 bytes
Line-based text data: text/html
<!DOCTYPE html>\n
<html lang="en">\n
```

MESSAGE_HEADER

MESSAGE_BODY

Table 9.2 Five Types of HTTP Result Codes

Code	Type	Example Reasons
1xx	Informational	request received, continuing process
2xx	Success	action successfully received, understood, and accepted
3xx	Redirection	further action must be taken to complete the request
4xx	Client Error	request contains bad syntax or cannot be fulfilled
5xx	Server Error	server failed to fulfill an apparently valid request

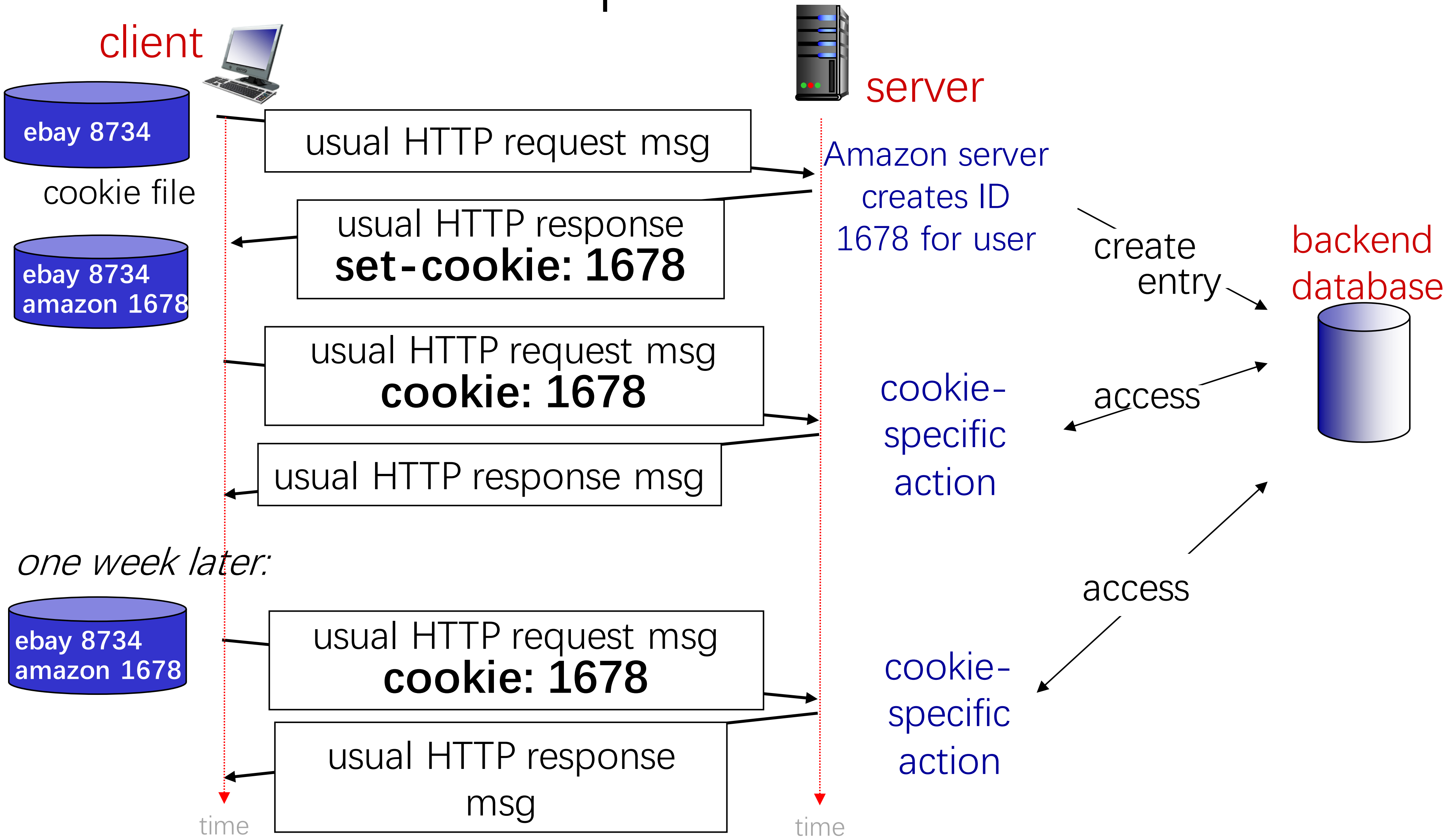
Demo

- HTTP Protocol
  - http://sist-admission.shanghaitech.edu.cn/
  - Wireshark
  - Telnet to host: sist-admission.shanghaitech.edu.cn:80
    - Copy and paste GET /

Web Cookies

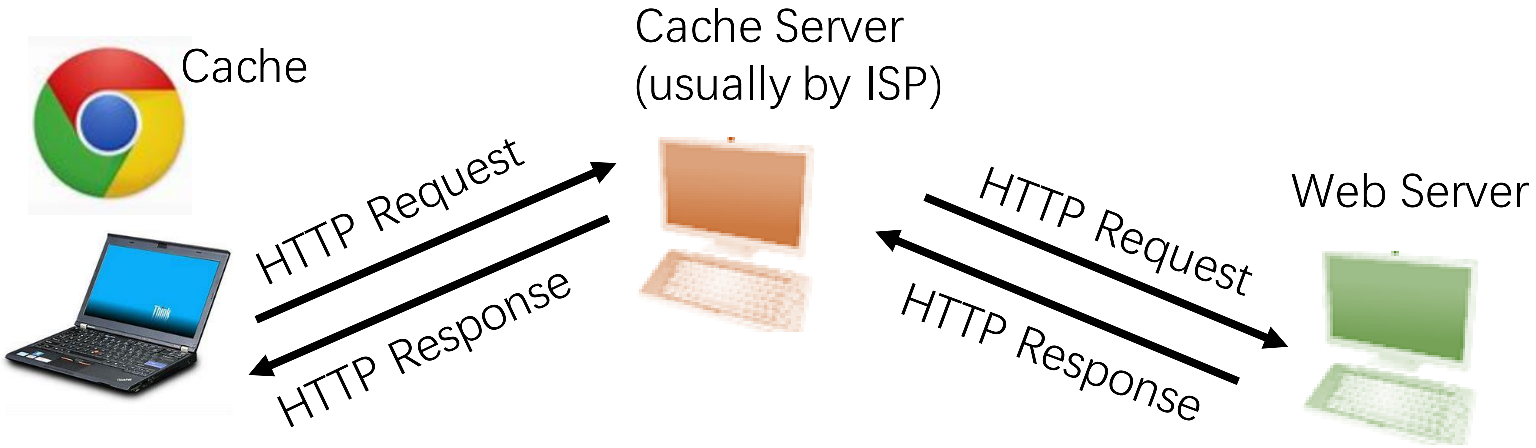
- Why
  - HTTP is stateless, server does not recode the states of HTTP client
  - Some applications need client state, e.g., online shopping
- How
  - Web server and client browser use cookies to maintain some state between transactions
  - Four components:
    - cookie header line of HTTP response message
    - cookie header line in next HTTP request message
    - cookie file kept on user's host, managed by user's browser
      - identified by server origin
    - back-end database at web server

Web Cookies Example



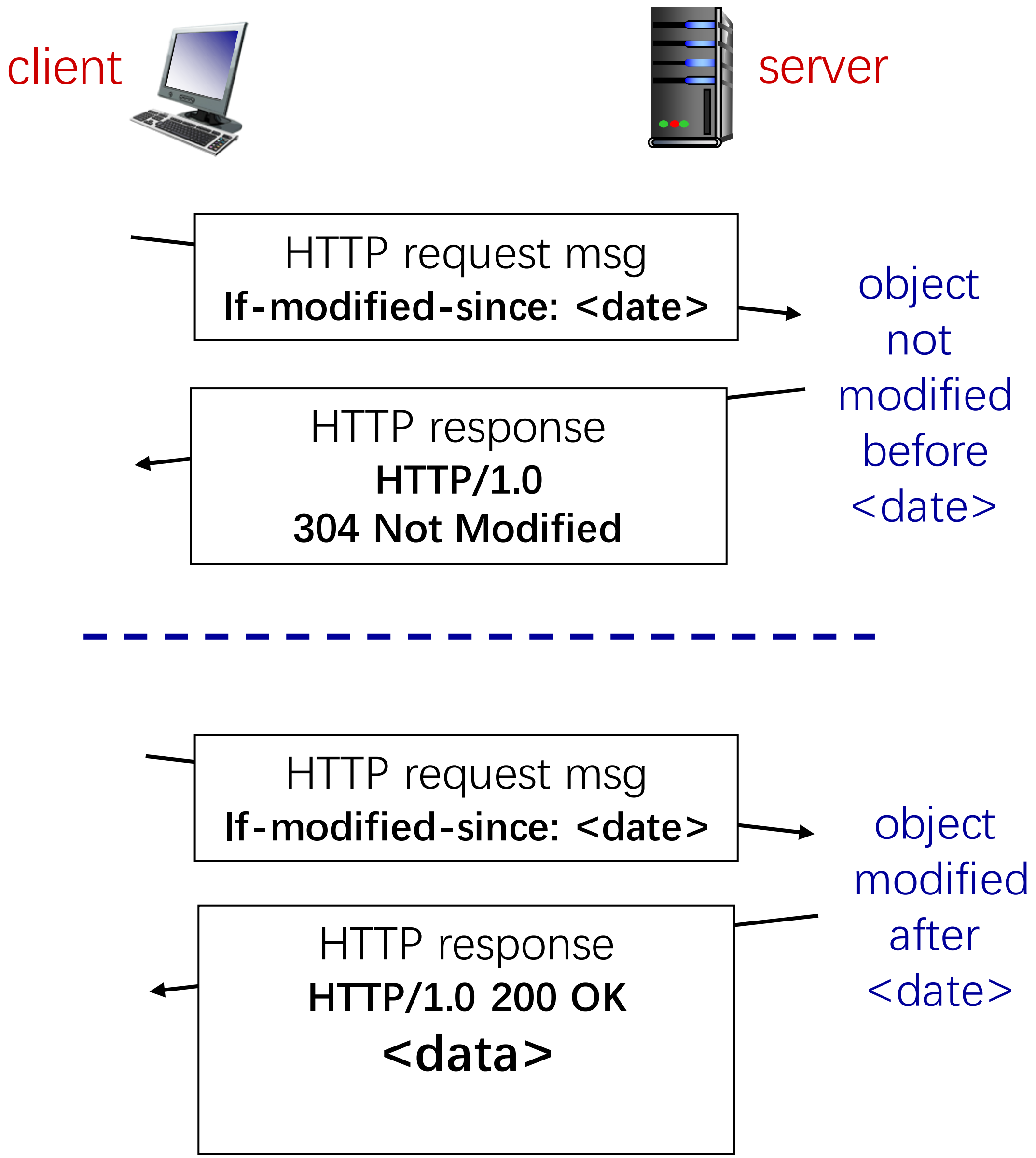
Web Caching

- Why
  - Reduce response time for client request
  - Reduce traffic
- How
  - Browser sends all HTTP requests to cache
    - Object in cache: cache returns object
    - Else cache requests object from original server, then returns object to client



Conditional GET

- Goal: Don't send object if cache has up-to-date cached version
- Client: specify date of cached copy in HTTP request
  - If-modified-since: <date>
- Server: response contains no object if cached copy is up-to-date:
  - HTTP/1.0 304 Not Modified



Demo

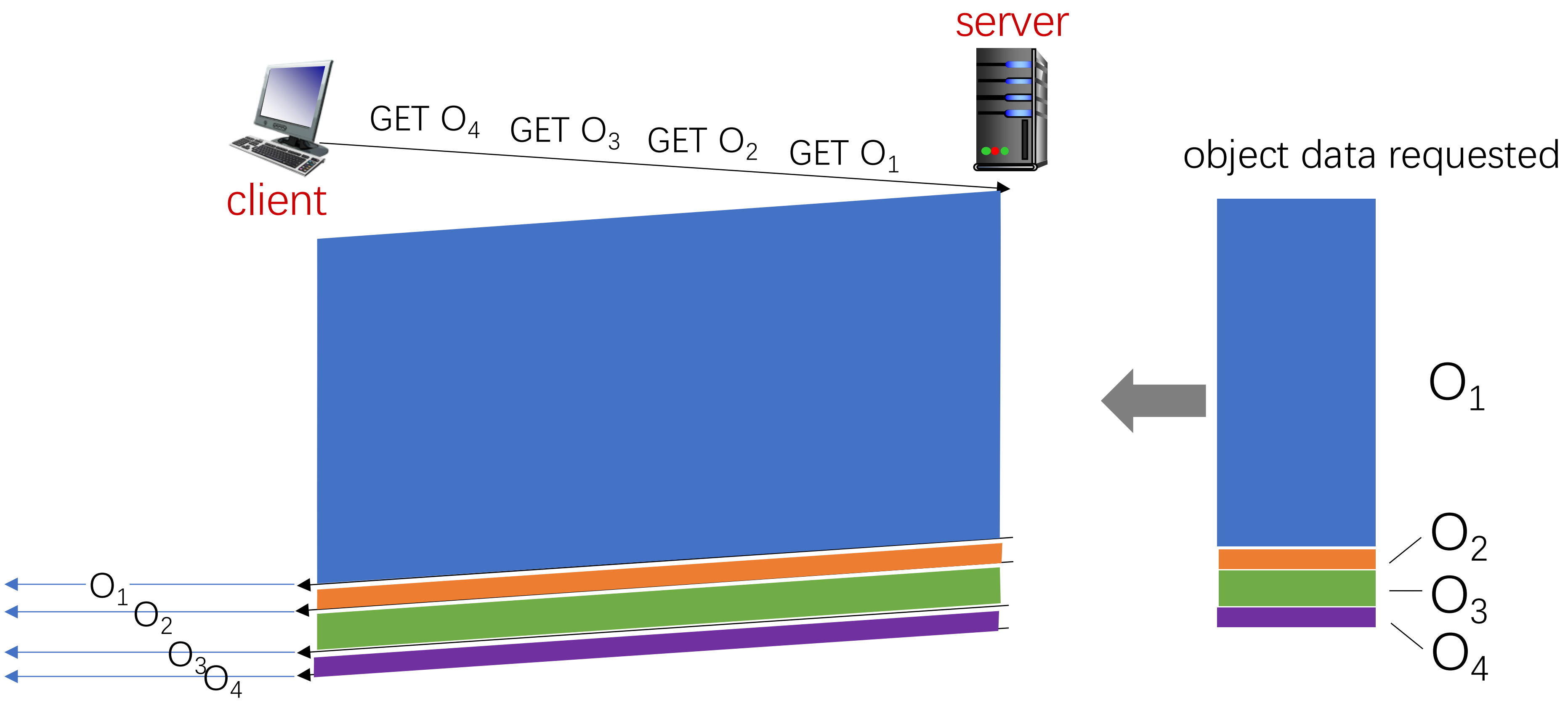
- Web caching for shanghaitech.edu.cn

HTTP/2

- Goal: decreased delay in multi-object HTTP requests
- Problems in HTTP 1.1
  - HTTP1.1 uses multiple, pipelined GETs over single TCP connection
  - Server responds in-order (FCFS: first-come-first-served scheduling) to GET requests
  - With FCFS, small object may have to wait for transmission (head-of-line (HOL) blocking) behind large objects
  - Loss recovery (retransmitting lost TCP segments) stalls object transmission

HOL Blocking

HTTP 1.1: client requests 1 large object (e.g., video file) and 3 smaller objects

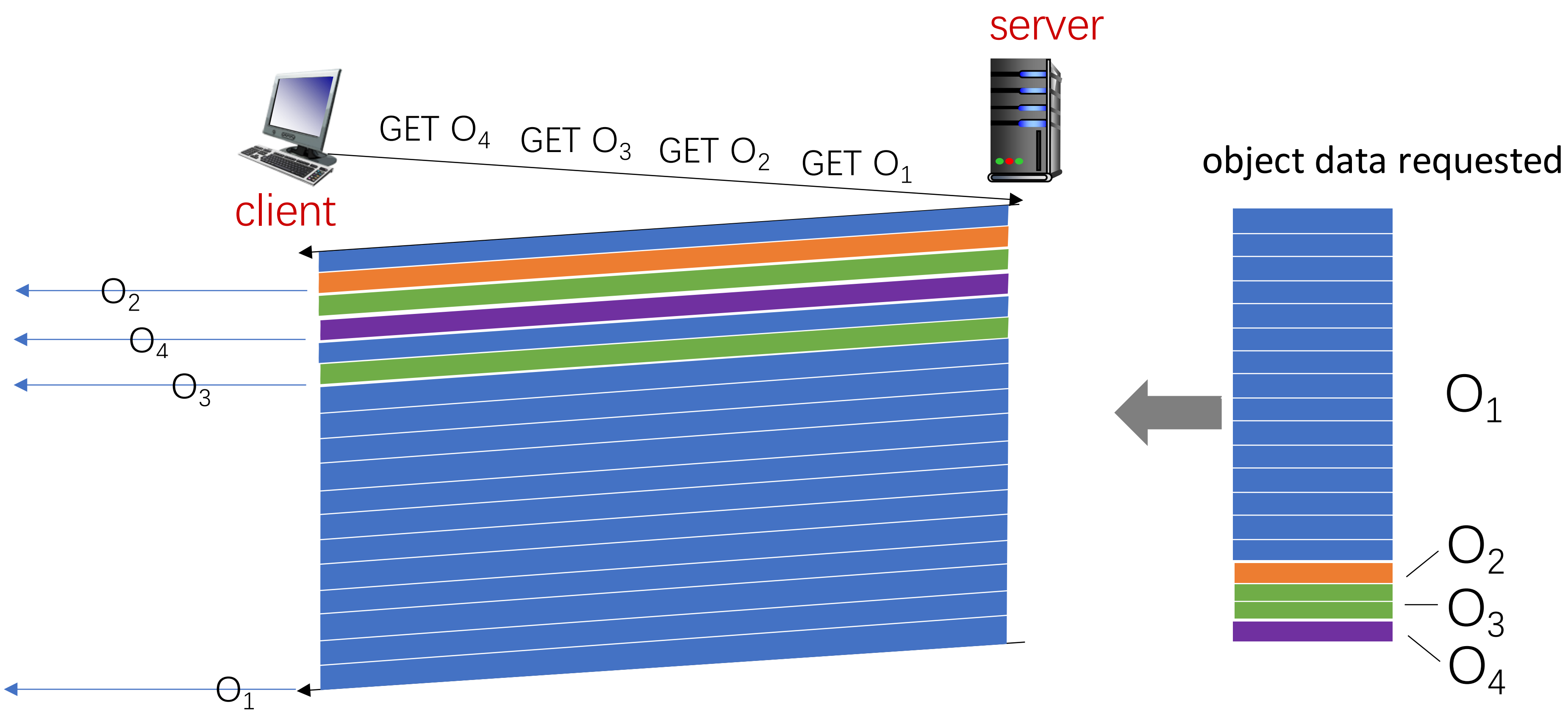


HTTP/2

- Goal: decreased delay in multi-object HTTP requests
- Problems in HTTP 1.1
  - HTTP1.1 uses multiple, pipelined GETs over single TCP connection
  - Server responds in-order (FCFS: first-come-first-served scheduling) to GET requests
  - With FCFS, small object may have to wait for transmission (head-of-line (HOL) blocking) behind large objects
  - Loss recovery (retransmitting lost TCP segments) stalls object transmission
- HTTP/2 key designs
  - Transmission order of requested objects based on client-specified object priority (not necessarily FCFS)
  - Divide objects into frames, schedule frames to mitigate HOL blocking
  - Push unrequested objects to client (server push)

HTTP/2 Example

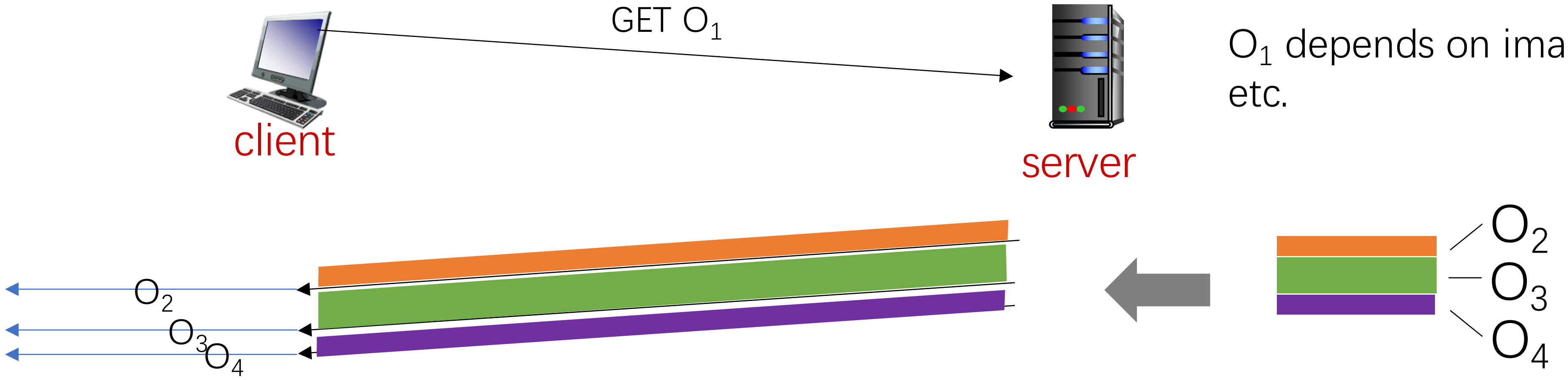
(1) Concurrent object transmissions: objects divided into frames, frame transmission interleaved



O_2, O_3, O_4 delivered quickly, O_1 slightly delayed

HTTP/2

(2) Server proactively push dependent elements to clients



O_1 depends on images, files, etc.

Instead of waiting clients to request dependent objects, server actively push to clients.

Bundled transmissions enable better compression.

HTTP/2 to HTTP/3

- Problems in HTTP/2
  - Recovery from packet loss still stalls all object transmissions
  - Needs another security layer
    - More latency
- See lecture on QUIC

Reference

- Textbook 9.1
- Some slides are adapted from http://www-net.cs.umass.edu/kurose_ross/ppt.htm by Kurose Ross