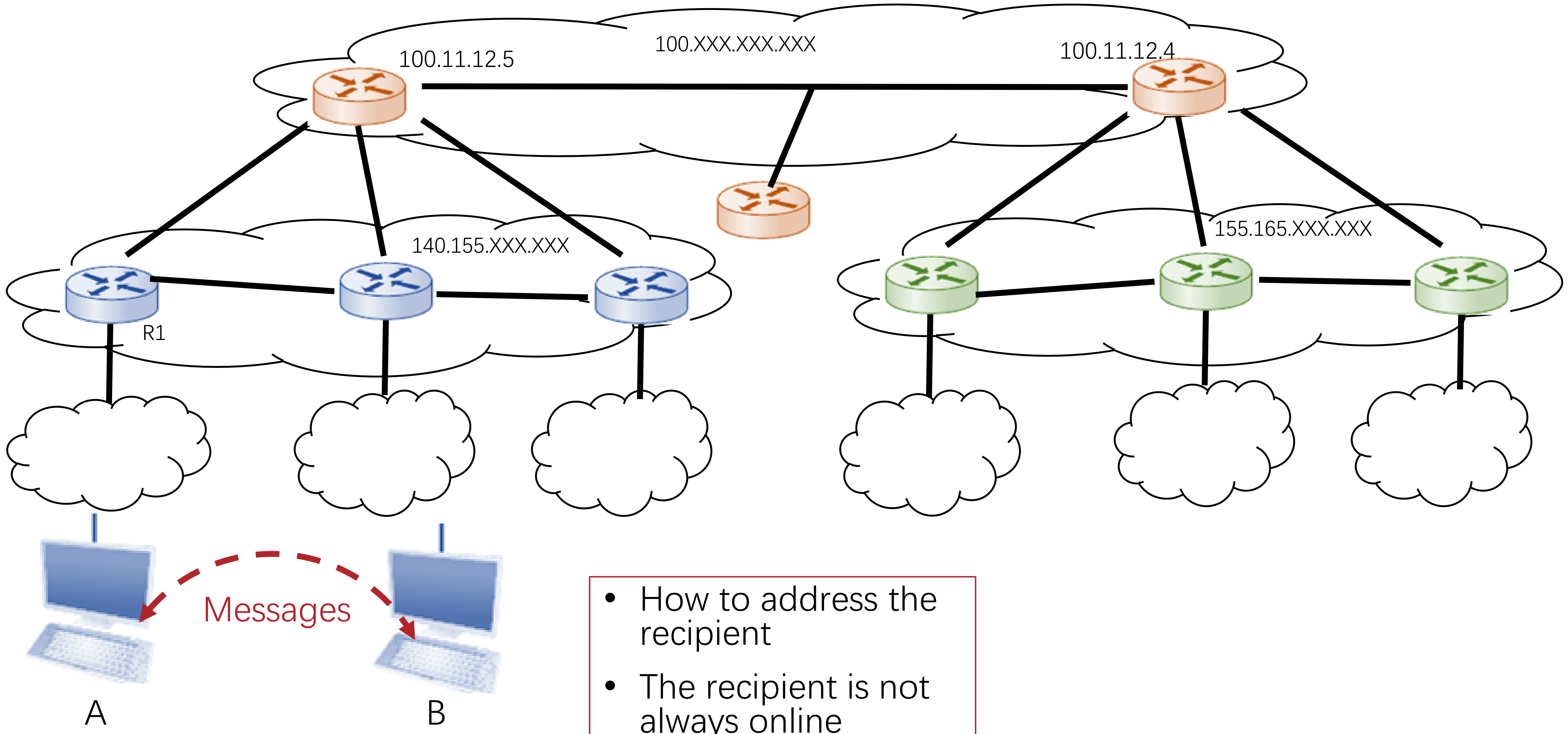
CS120: Computer Networks

Lecture 25. Email & Web

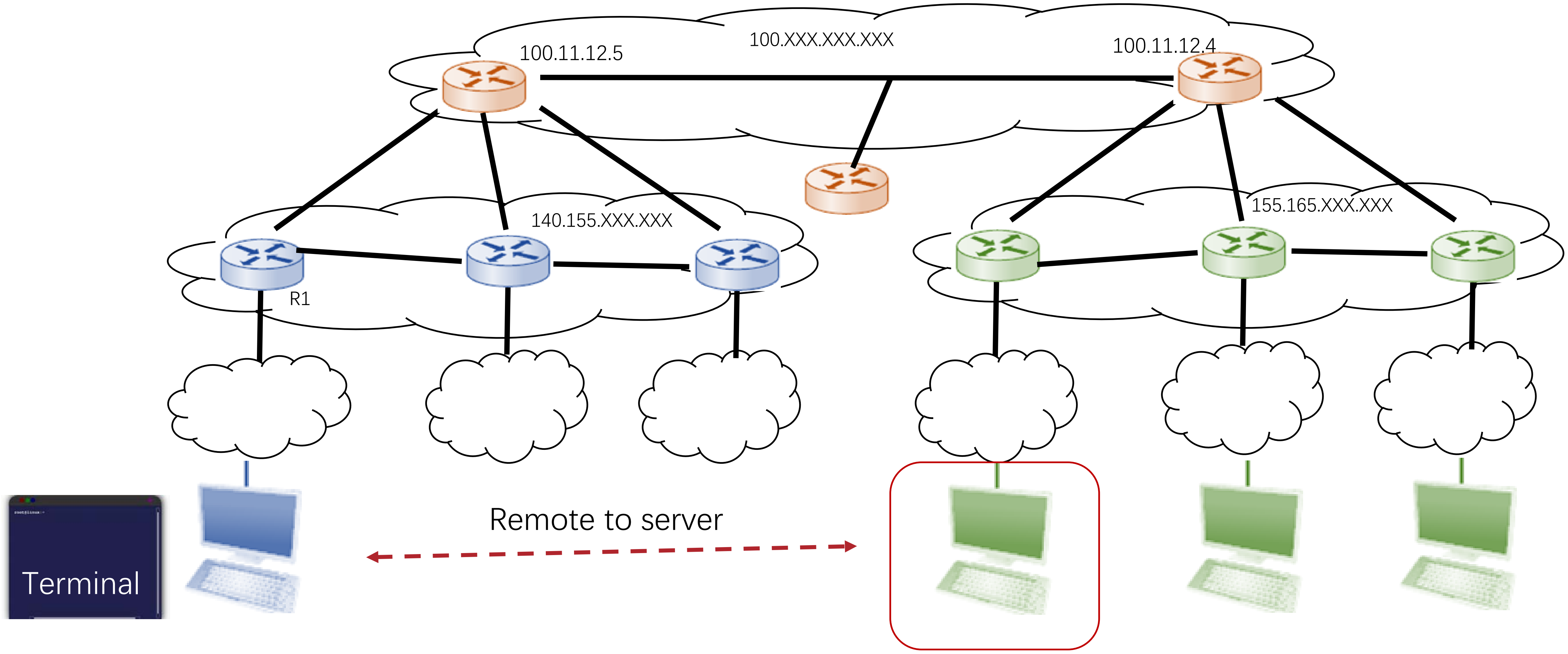
Zhice Yang

Mail Over Network ?



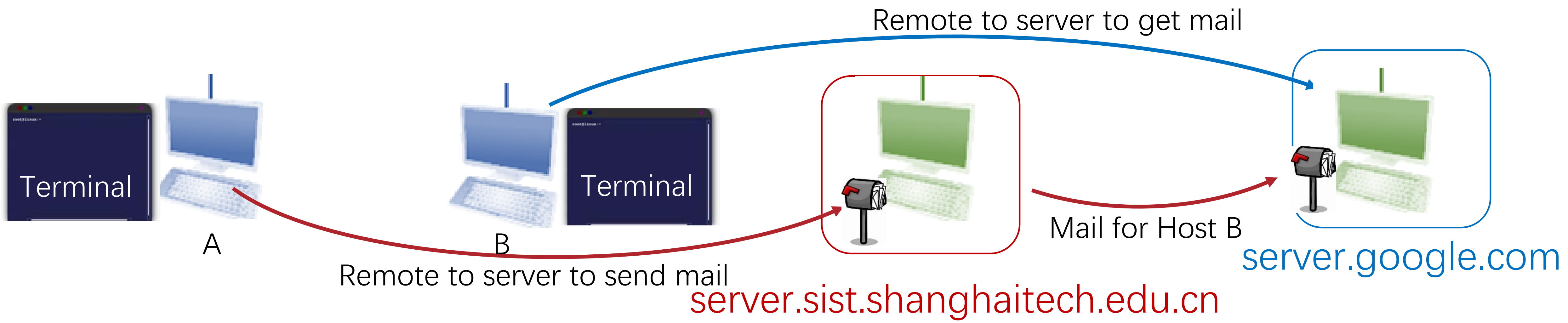
- How to address the recipient
- The recipient is not always online

Telnet – Remote Command Line Access

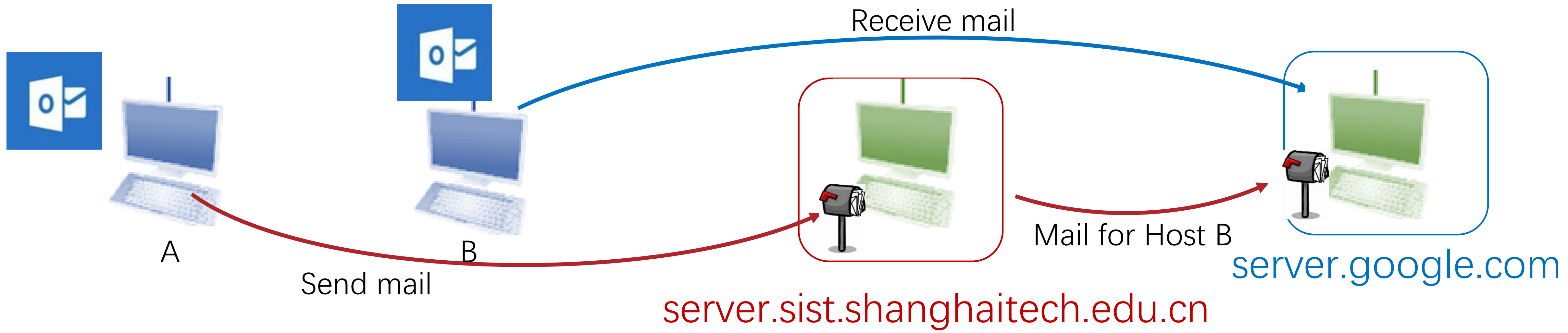


server.sist.shanghaitech.edu.cn

Electronic Mail (Email)

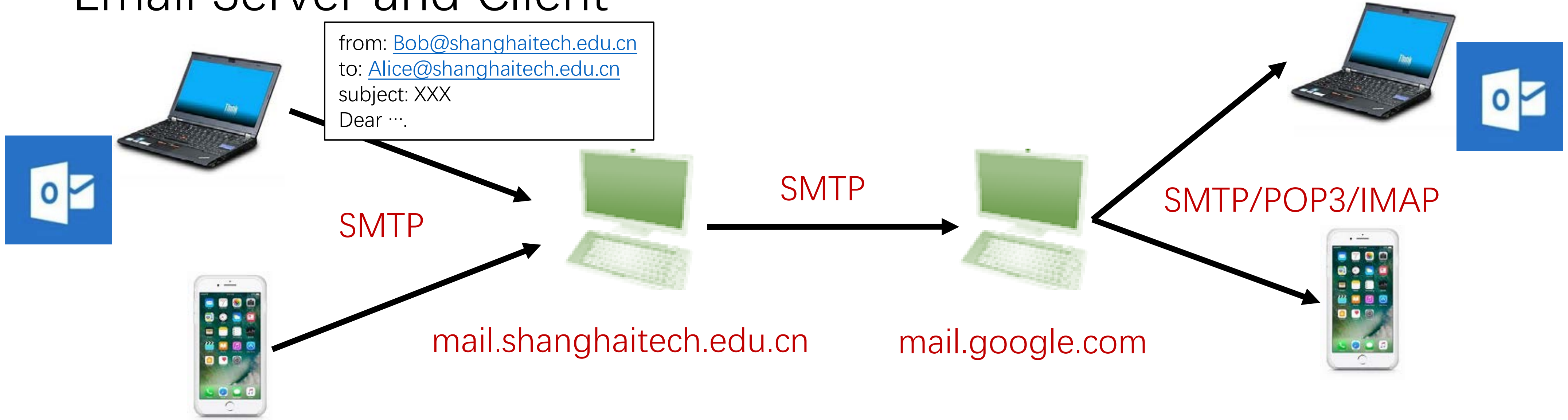

Remote to server to get mail
Terminal
Terminal
A
B
Remote to server to send mail
Mail for Host B
server.google.com
server.sist.shanghaitech.edu.cn

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


from: Bob@shanghaitech.edu.cn
to: Alice@shanghaitech.edu.cn
subject: XXX
Dear ….

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
```

Email Format

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

Email Protocol

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

- 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 rely, 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

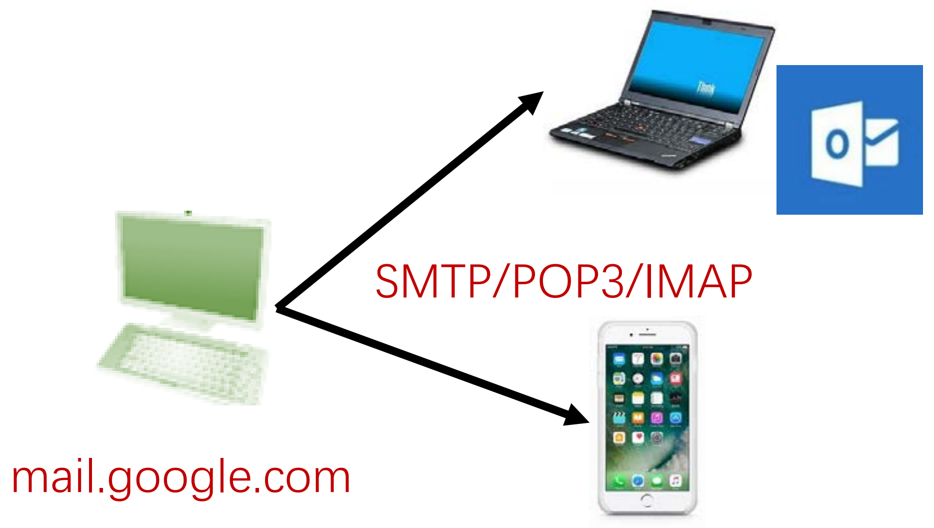
Email Protocol

- 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
```

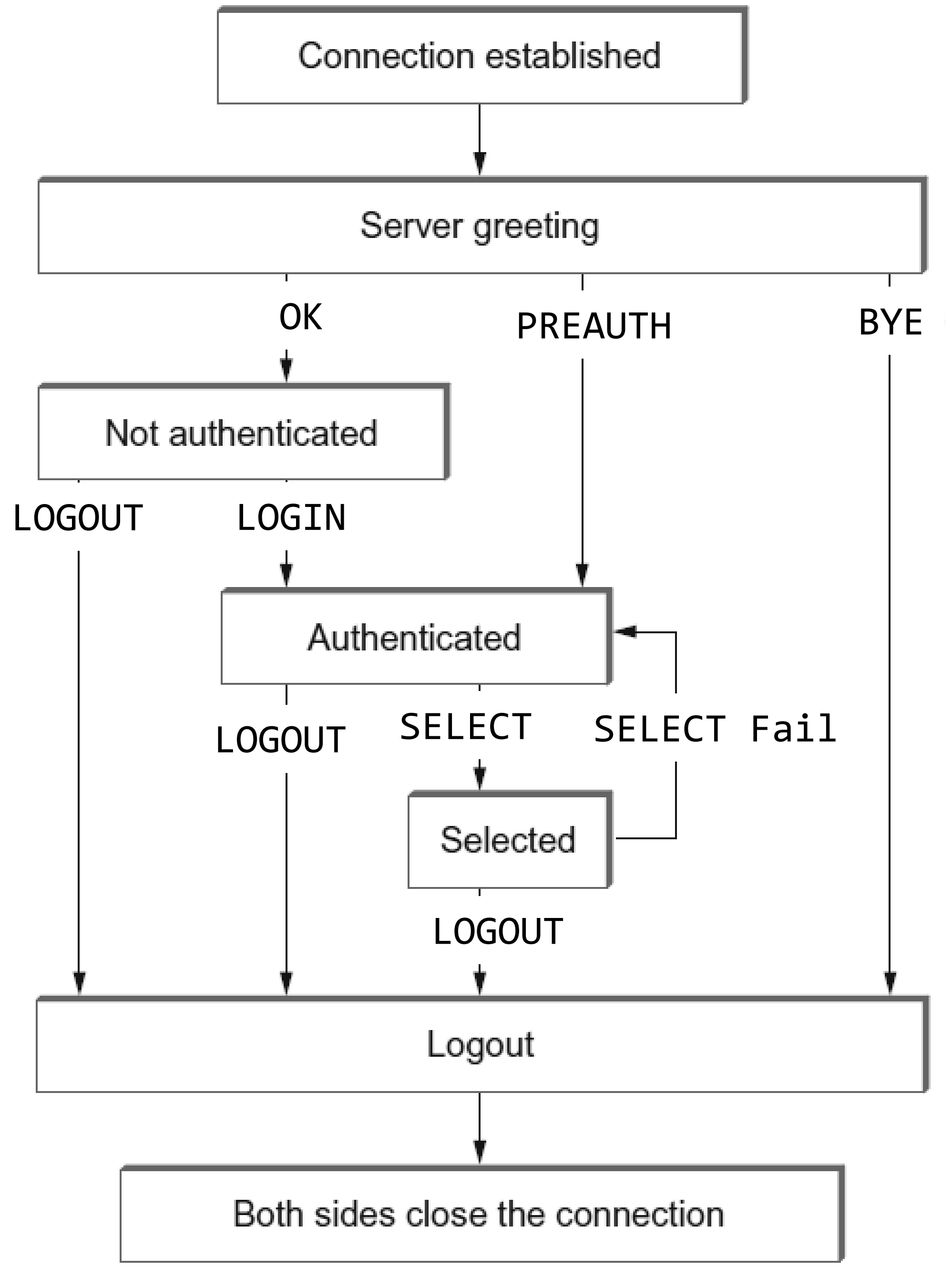
Email Protocol

- 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,

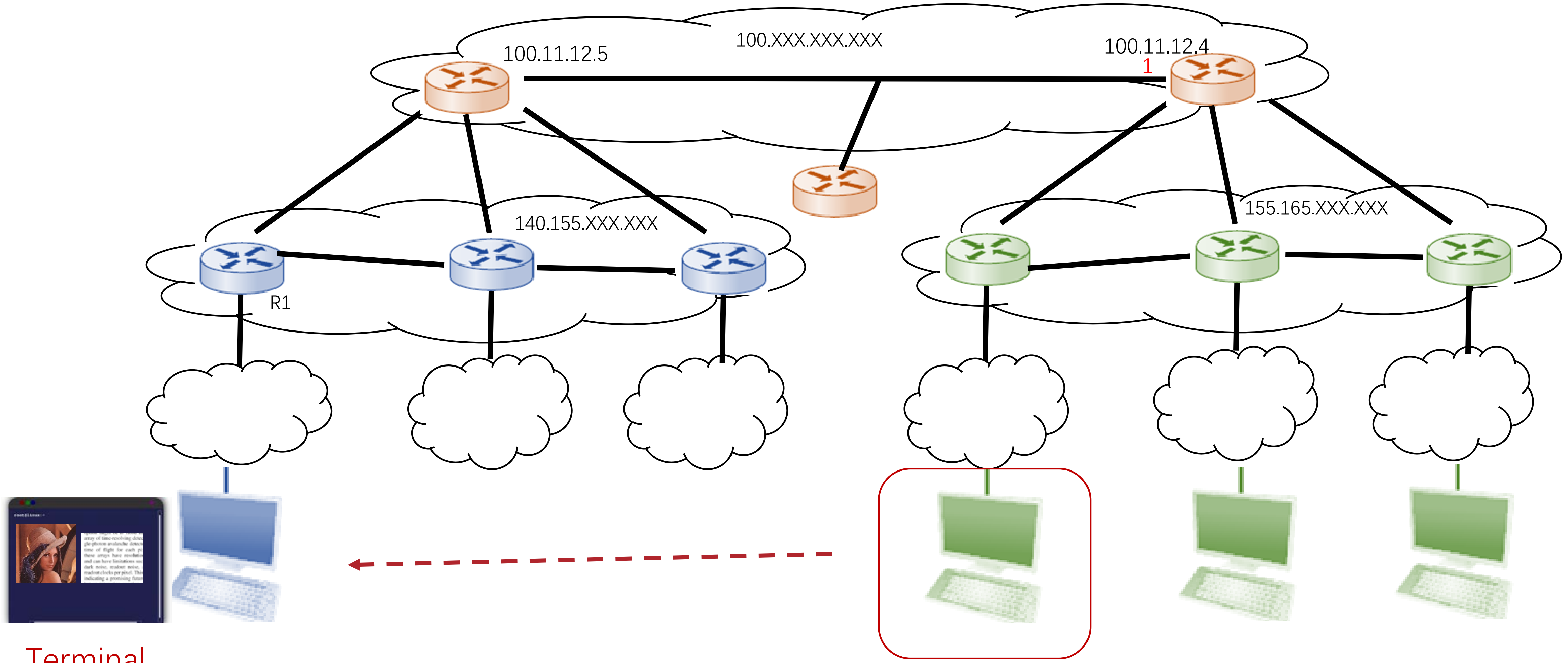
Email Implementation

- 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 ?

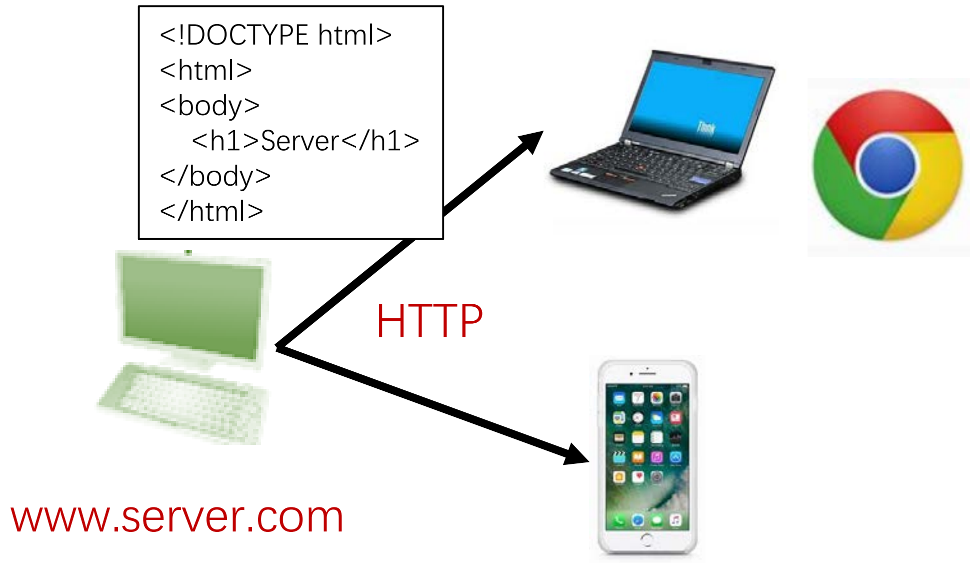


Terminal

server.sist.shanghaitech.edu.cn

World Wide Web (WWW)

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

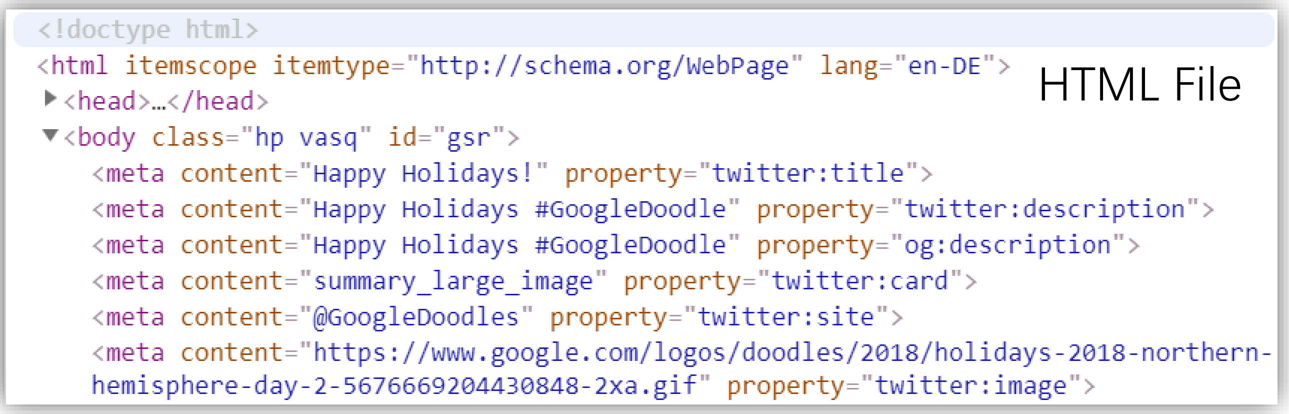


```
<!DOCTYPE html>
<html>
<body>
   <h1>Server</h1>
</body>
</html>
```

Web Page Format

$$\underbrace{\text{ssist.shanghaitech.edu.cn}}_{\text{Hostname}}/\underbrace{\text{upload/image/xxx.png}}_{\text{Resource Path}}$$

- 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)



```
<!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
  - IE, Firefox, Chrome, etc.
- Server: Send objects in response to requests
  - Apache, Nginx, etc.

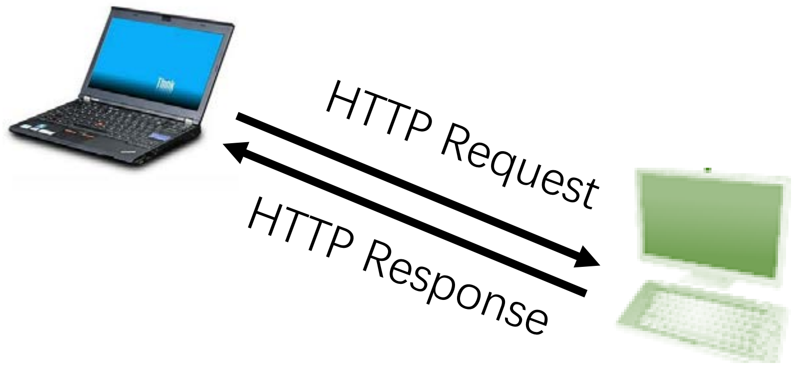
HyperText Transfer Protocol (HTTP)

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

HTTP Messages

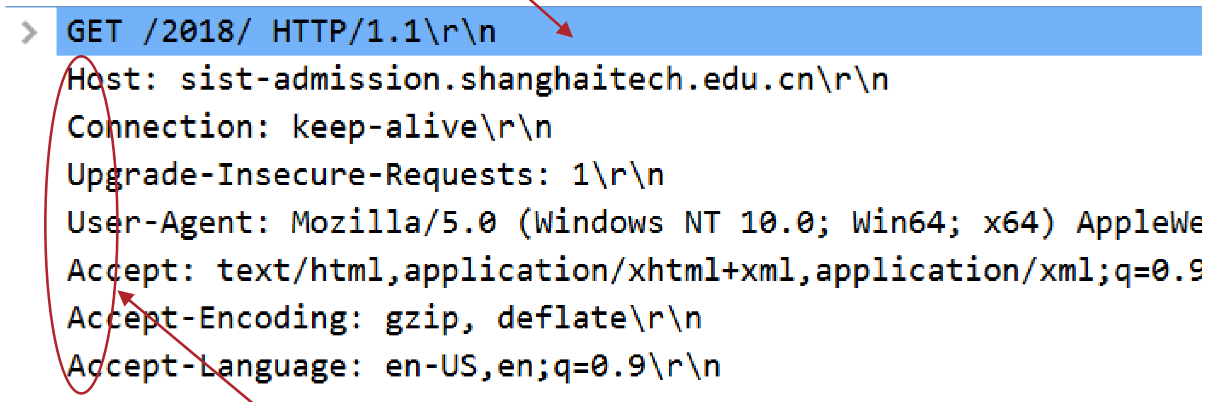
- Like SMTP, HTTP is Text-oriented
- Two types of HTTP messages
  - Request
  - Response
- Message Format

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



HTTP Request

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.
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

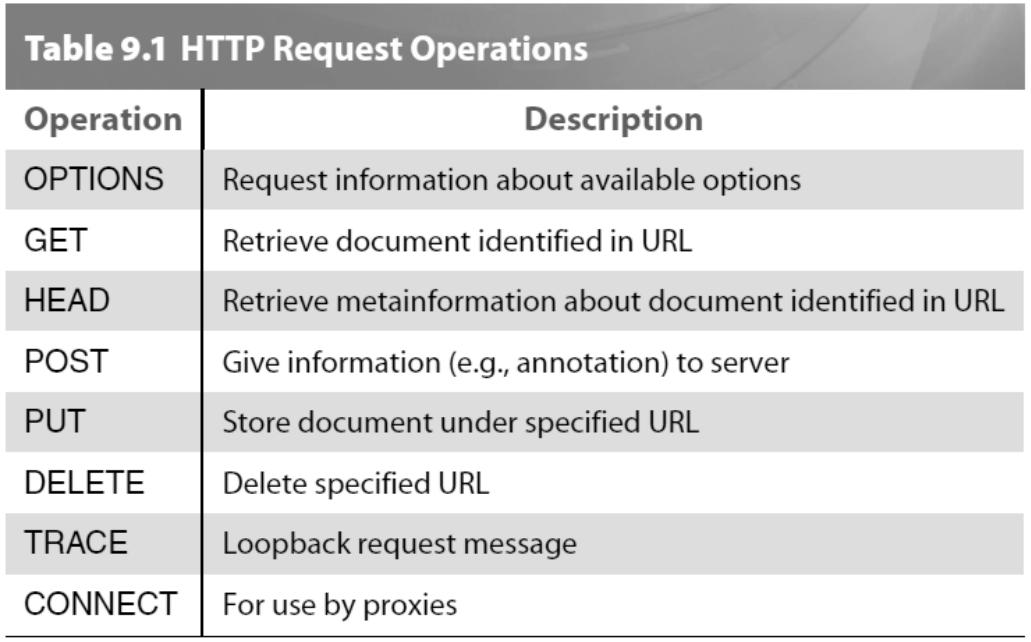


Table 9.1 HTTP Request Operations

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

HTTP Response

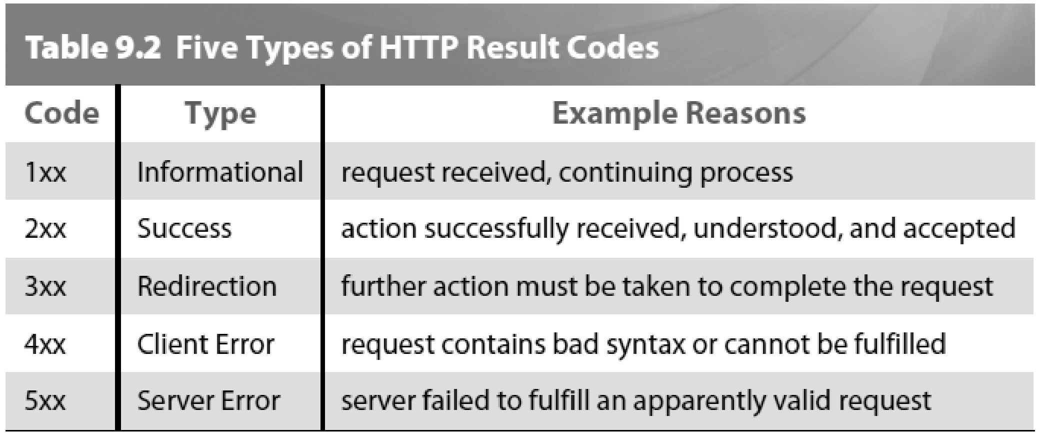
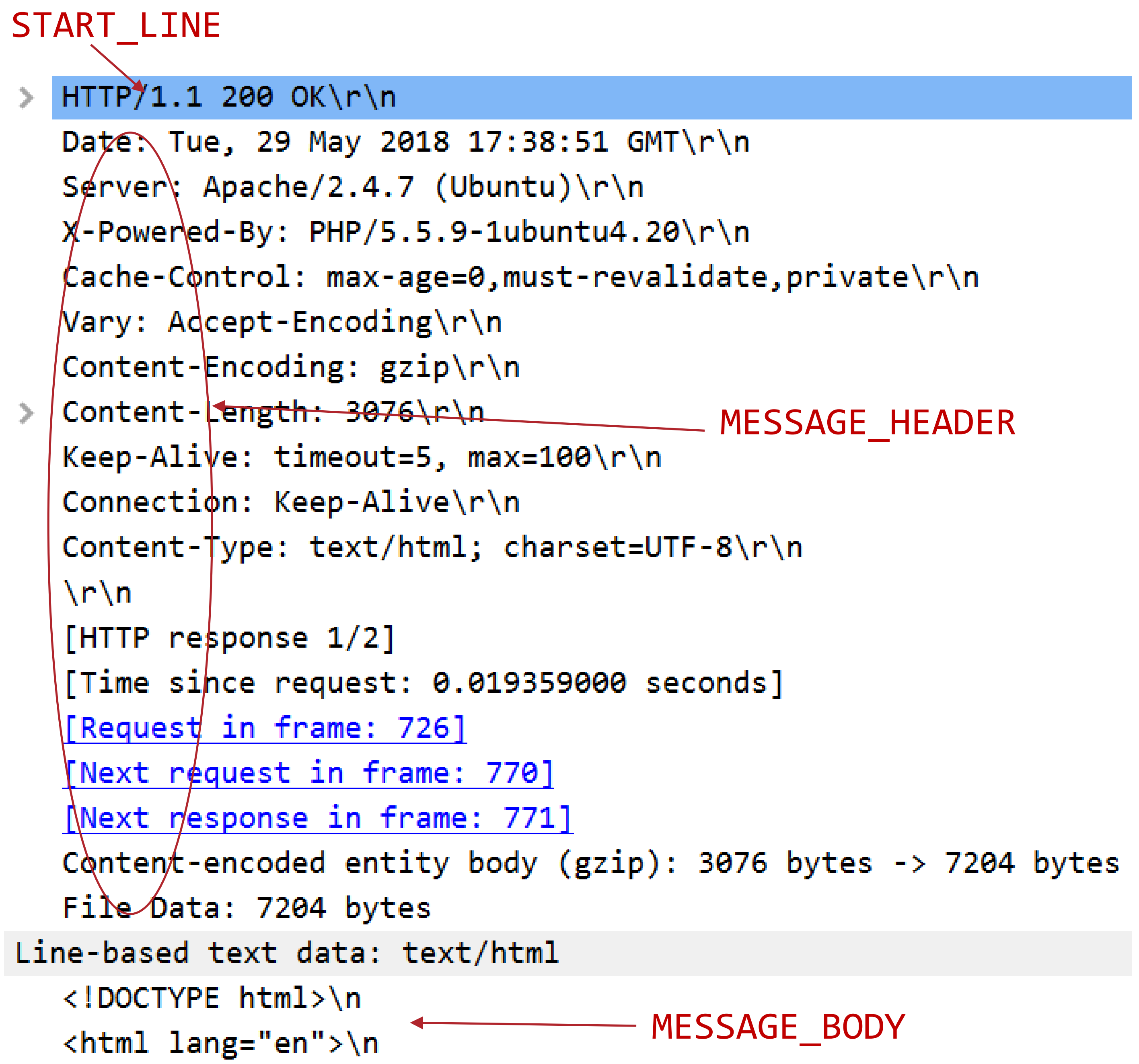


Table 9.2 Five Types of HTTP Result Codes

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



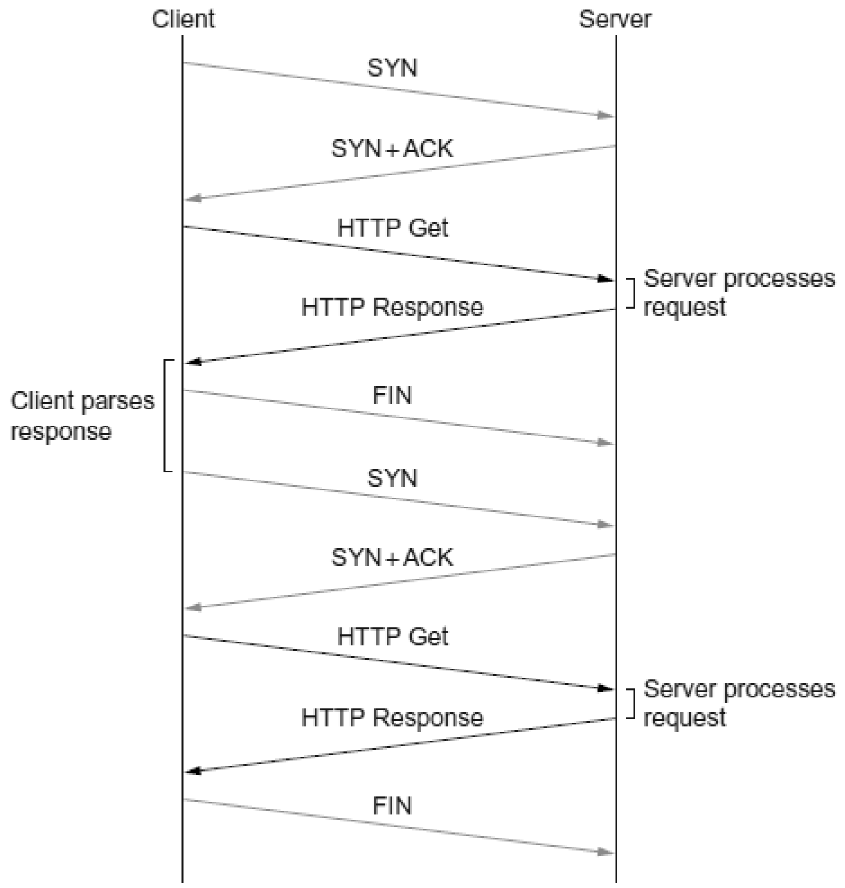
```
HTTP/1.1 200 OK\r\n
Date: Tue, 29 May 2018 17:38:51 GMT\r\n
Server: Apache/2.4.7 (Ubuntu)\r\n
X-Powered-By: PHP/5.5.9-1ubuntu4.20\r\n
Cache-Control: max-age=0,must-revalidate,private\r\n
Vary: Accept-Encoding\r\n
Content-Encoding: gzip\r\n
Content-Length: 3076\r\n
Keep-Alive: timeout=5, max=100\r\n
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
```

Demo

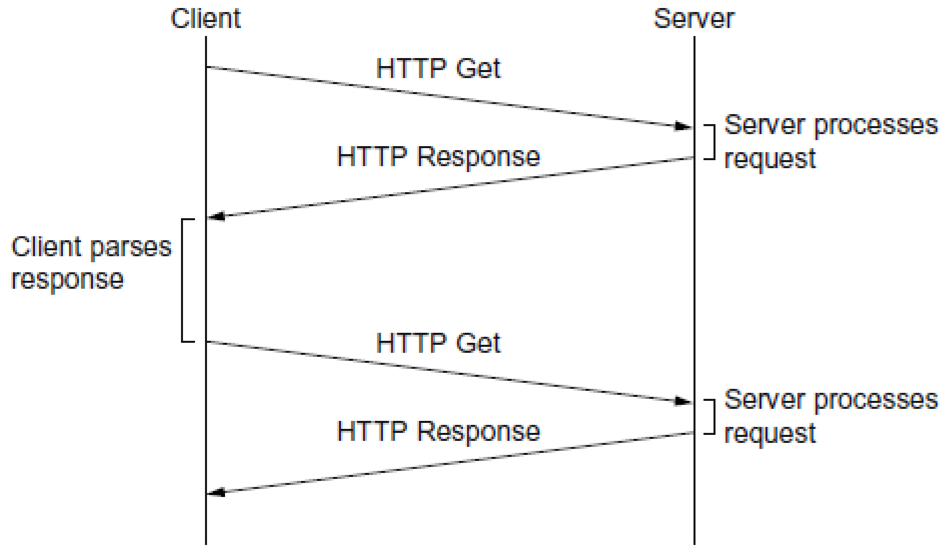
- HTTP Protocol
  - http://sist-admission.shanghaitech.edu.cn/
  - Wireshark
  - Telnet
    - GET / HTTP/1.1
    - host: sist-admission.shanghaitech.edu.cn

HTTP Connections

- Non-persistent HTTP

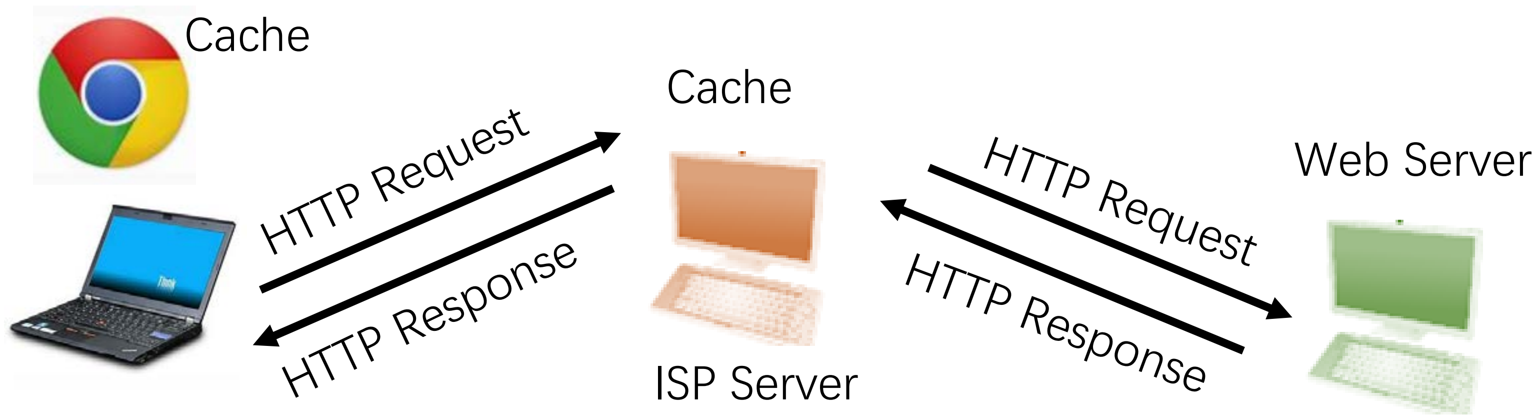


- Persistent HTTP



Web Caching

- 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
- Why
  - Reduce response time for client request
  - Reduce traffic



Demo

- Web caching for shanghaitech.edu.cn

Reference

- Textbook 9.1