暨南大學
JINAN UNIVERSITY

International School
Jinan University

Computer Networks

L13 – Application Layer II

Lecturer: CUI Lin

Department of Computer Science
Jinan University

Topics

- DHCP: Dynamic Host Configuration Protocol (Chapter 5.6.4)
- DNS: Domain Name System
- The World Wide Web: HTTP
- Electronic Email

World Wide Web

- WWW is invented by Tim Berners-Lee in 1989
- W3C (World Wide Web Consortium), organization oversees the Web's continued development, e.g., standardizing:
  - www.w3.org

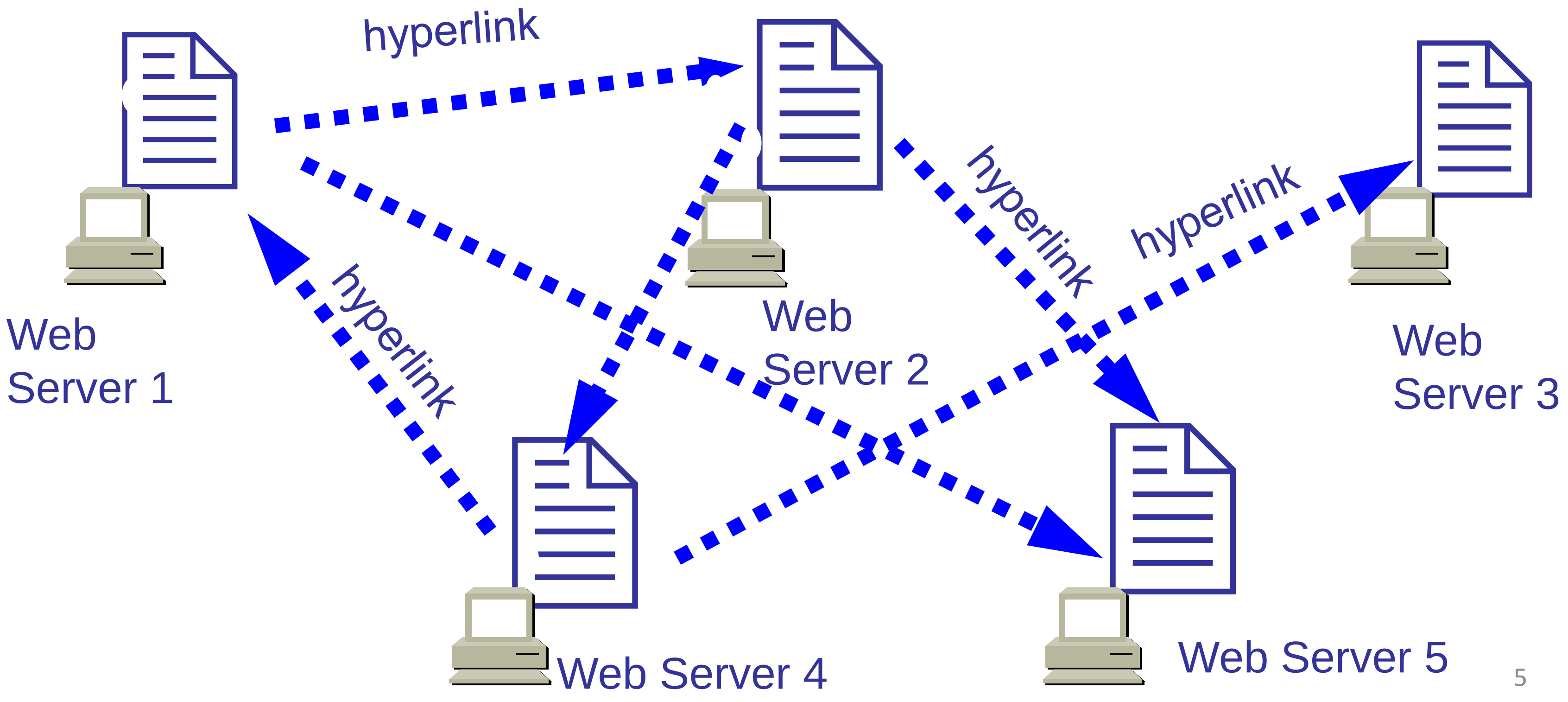
Ref. Wiki

WWW

- WWW: client/server model
  - Client: browser
  - Web server
- From user viewpoint, WWW contain collection of web pages:
  - Web page consists of objects (resources)
  - Object can be HTML file, image, audio file, js/css file,…
  - A web page is introduce by a single HTML file, and contains several objects

WWW

- WWW provides distributed service using hyperlink (超链接):



Three Essential Technologies

- URL (Uniform Resource Locator)
  - A system of globally unique identifiers for resources on the Web and elsewhere
  - Each object is addressable by a unique URL
- HTTP (HyperText Transfer Protocol):
  - Foundation of data communication for WWW
- HTML (HyperText Markup Language):
  - The main markup language for creating web pages and other information that can be displayed in a web browser

URL

- Globally unique identifiers for web objects
- Case insensitive
- URL Format:

```
<protocol>://<host name>:<port>/<path name>
```

- ftp: File Transfer Protocol
- http: hypertext transfer protocol
- https: http with security
- file: local file
- mailto: sending email

URL

- Globally unique identifiers for web objects
- Case insensitive
- URL Format:

$$\text{<protocol>://}\underbrace{\text{<host name>}}_{\text{Domain name of web server}}\text{:<port>/<path name>}$$

URL

- Globally unique identifiers for web objects
- Case insensitive
- URL Format:

```
<protocol>://<host name>:<port>/<path name>
```

Can be ignored,
Default port for HTTP is 80

$$\text{https://}\underbrace{\text{ischool.jnu.edu.cn}}_{\text{Host name}}/\underbrace{\text{a1/23/c20061a434467/page.psp}}_{\text{Path name}}$$

Three Essential Technologies

- URL (Uniform Resource Locator)
  - A system of globally unique identifiers for resources on the Web and elsewhere
  - Each object is addressable by a unique URL
- HTTP (HyperText Transfer Protocol):
  - Foundation of data communication for WWW
- HTML (HyperText Markup Language):
  - The main markup language for creating web pages and other information that can be displayed in a web browser

HTTP

- Based on TCP, default port on server: 80
  - Port other than default should be explicitly indicated
    - http://www.abc.com:8088/
- Stateless (无状态)
  - HTTP server maintains no information about past client requests
- Connectionless
  - No order control at application layer
- Only two basic types of messages, ASCII text
  - Request: sent from client to server
  - Response: replied from server to client

HTTP Architectural Overview

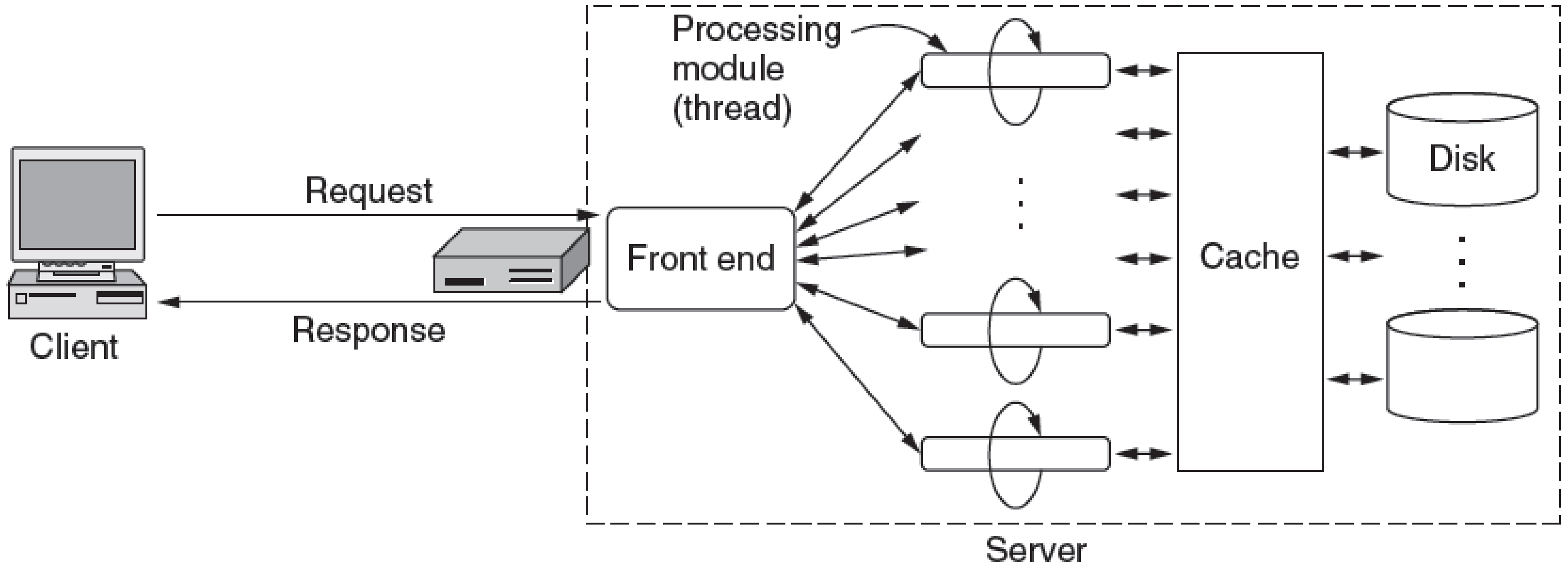
- Steps a client (browser) takes to follow a hyperlink:
  - Determine the protocol (HTTP)
  - Ask DNS for the IP address of server
  - Make a TCP connection to server
  - Send request for the page; server sends it back
  - Fetch other URLs as needed to display the page
  - Close idle TCP connections

HTTP Architectural Overview

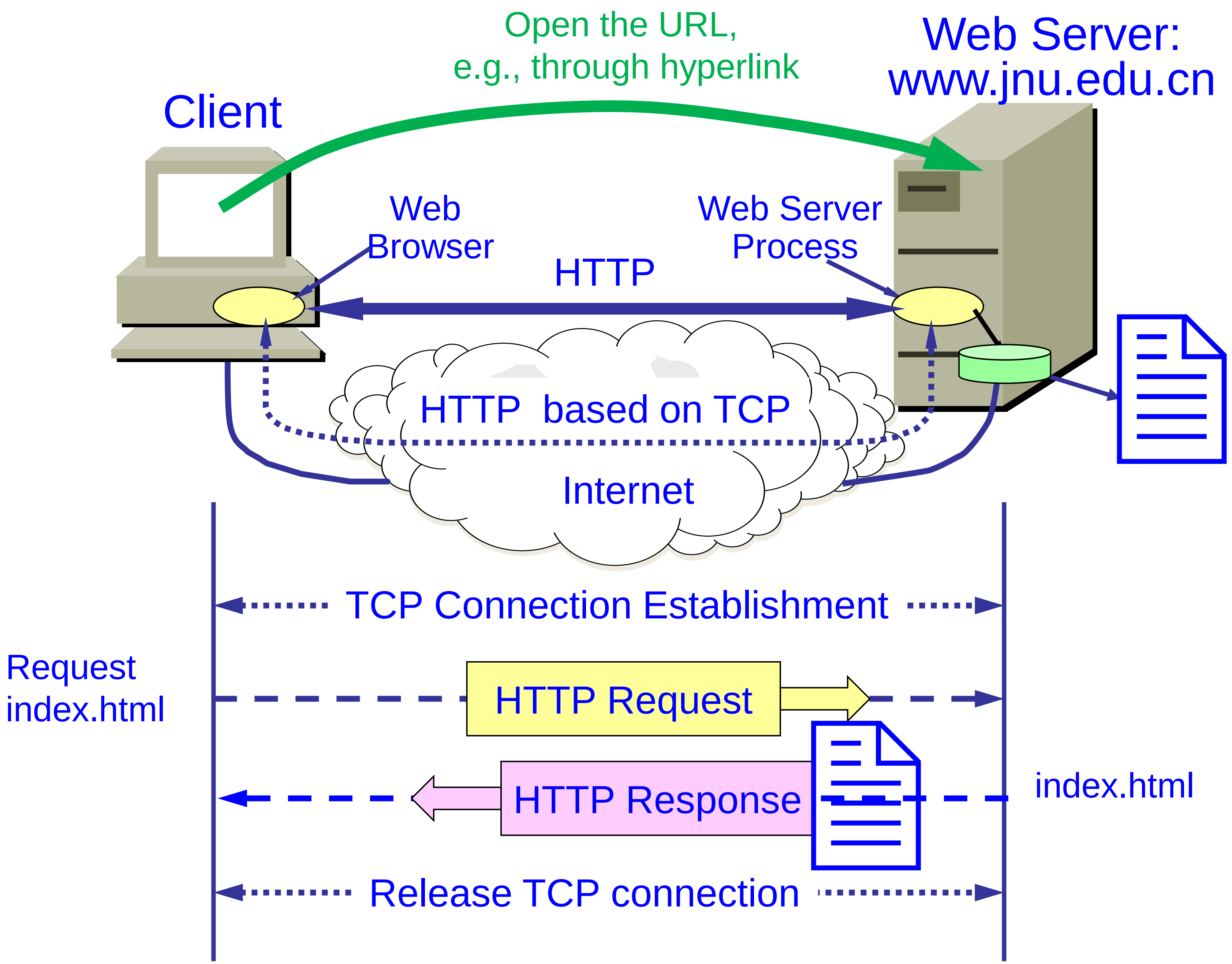
- Steps a server takes to serve pages:
  - Accept a TCP connection from client
  - Get page request and map it to a resource (e.g., file name)
  - Get the resource (e.g., file from disk or database)
  - Send contents of the resource to the client.
  - Release idle TCP connections

A multithreaded Web server

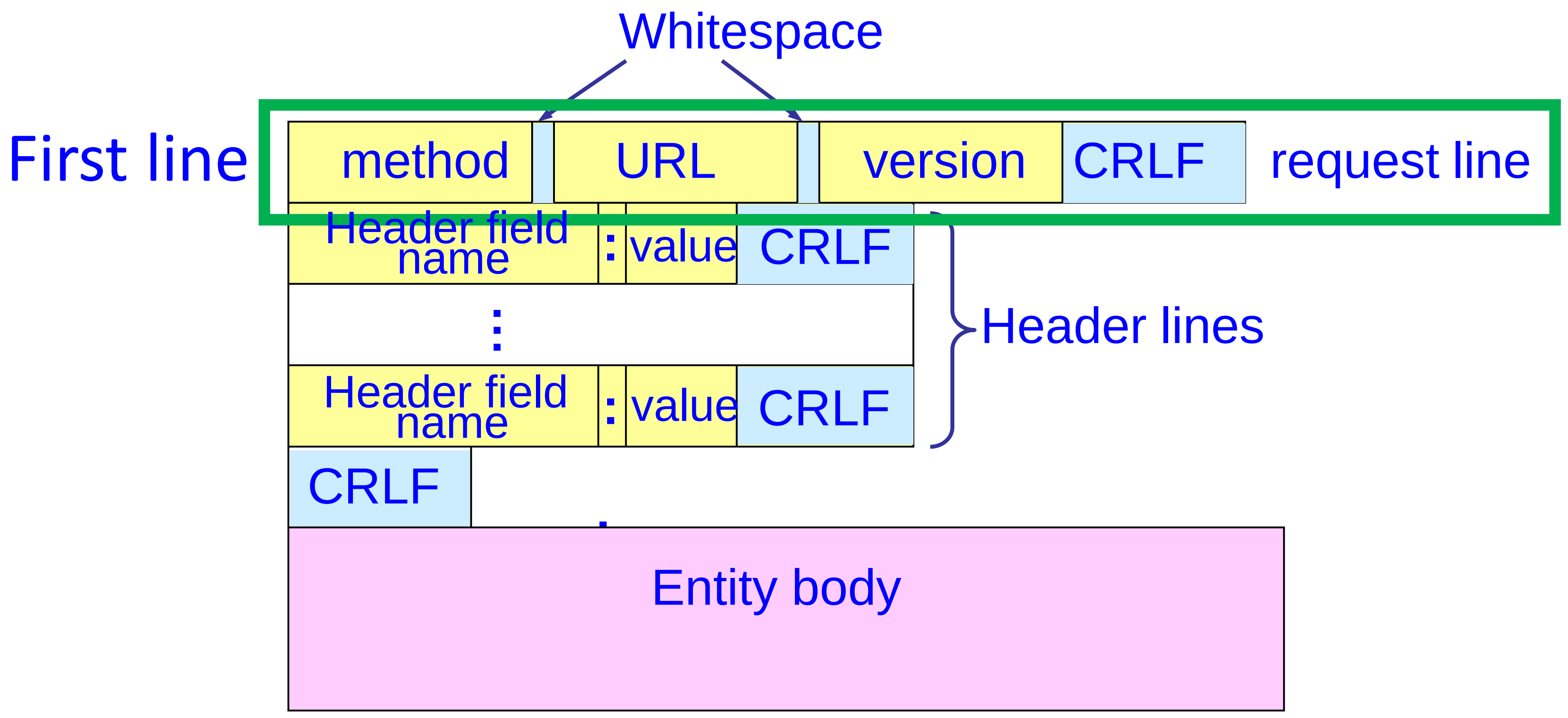
- To scale performance, Web servers can use:
  - Caching, multiple threads, and a front end



HTTP Process



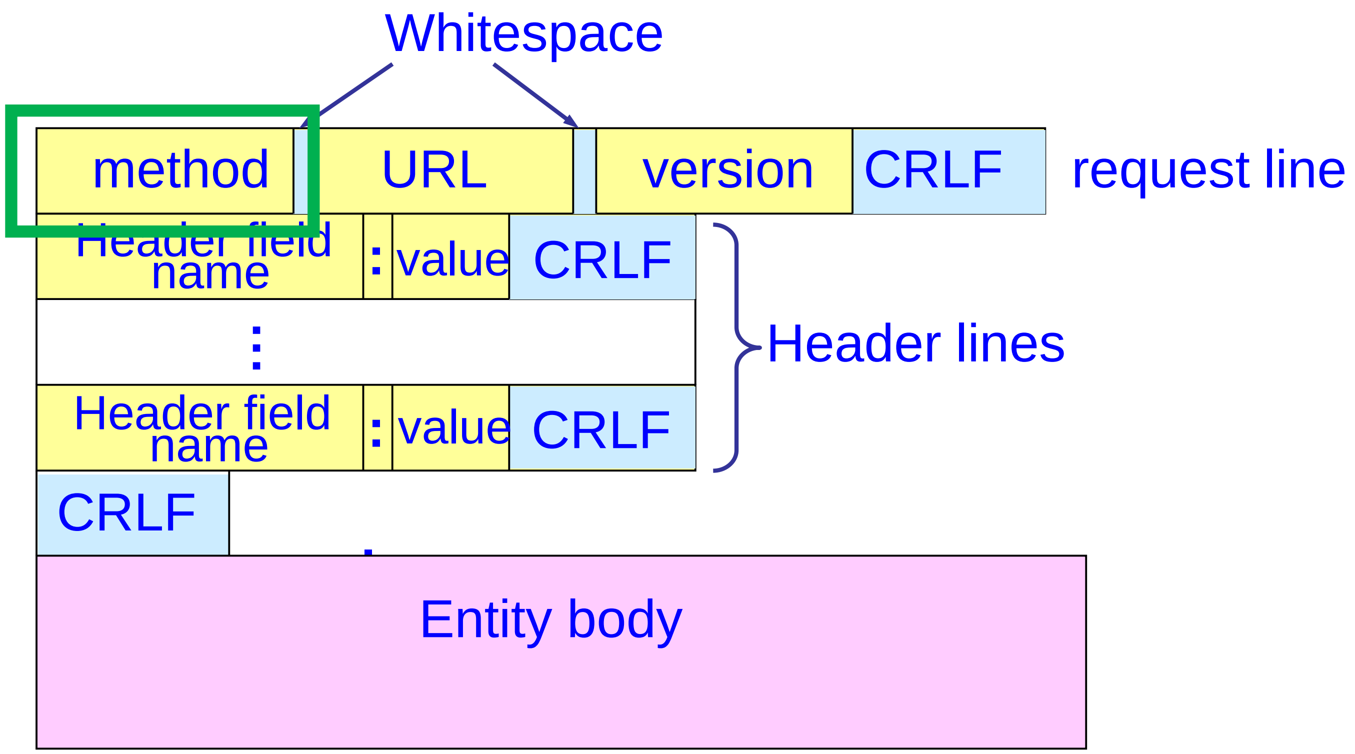
HTTP Request Message



Message include three parts:

- A request line, e.g., GET /images/logo.png HTTP/1.1
- Request Headers, e.g., Accept-Language: en
- An empty line.
- An optional message body.

HTTP Request Message



HTTP defines methods (sometimes referred to as verbs) to indicate the desired action to be performed on the identified resource.

HTTP Request Message: methods

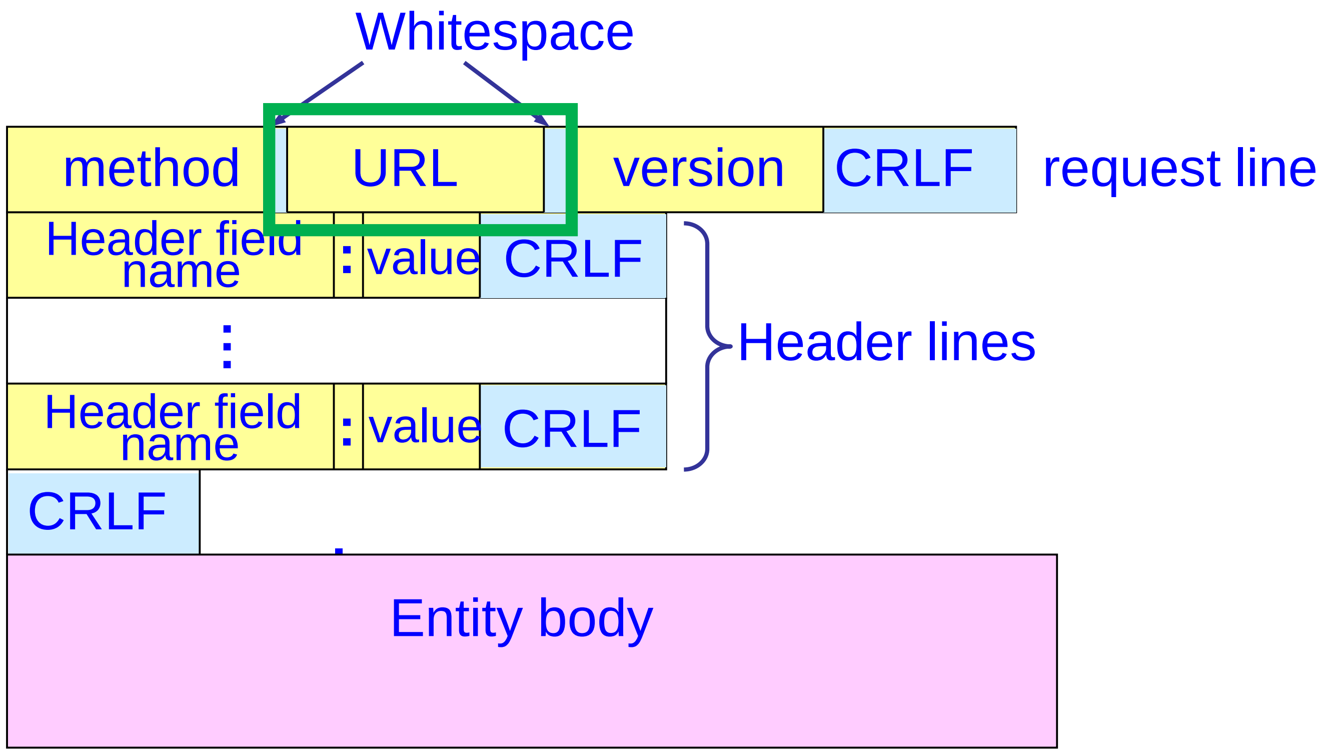
HTTP has several request methods.

Fetch a page → GET

Used to send input data to a server program → POST

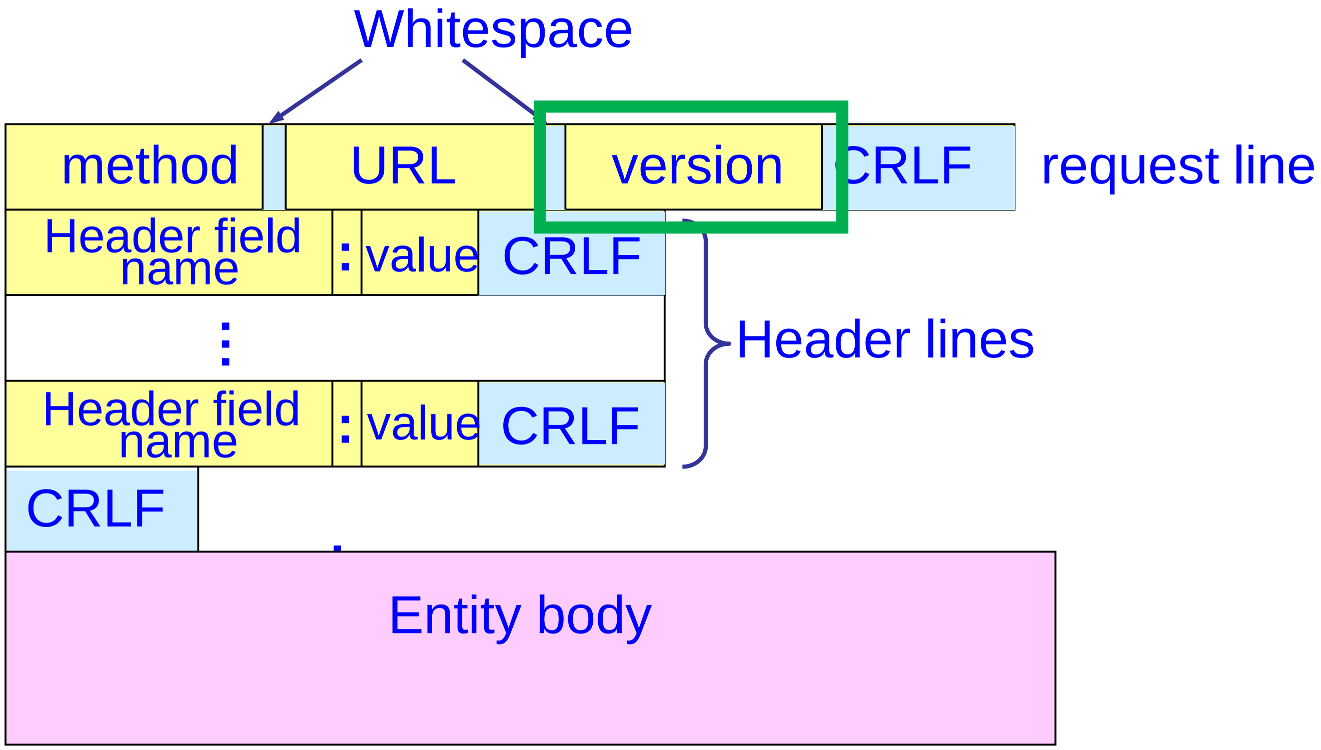
Method	Description
GET	Read a Web page
HEAD	Read a Web page’s header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

HTTP Request Message



URL: the requested resource on web server
e.g., /index.html

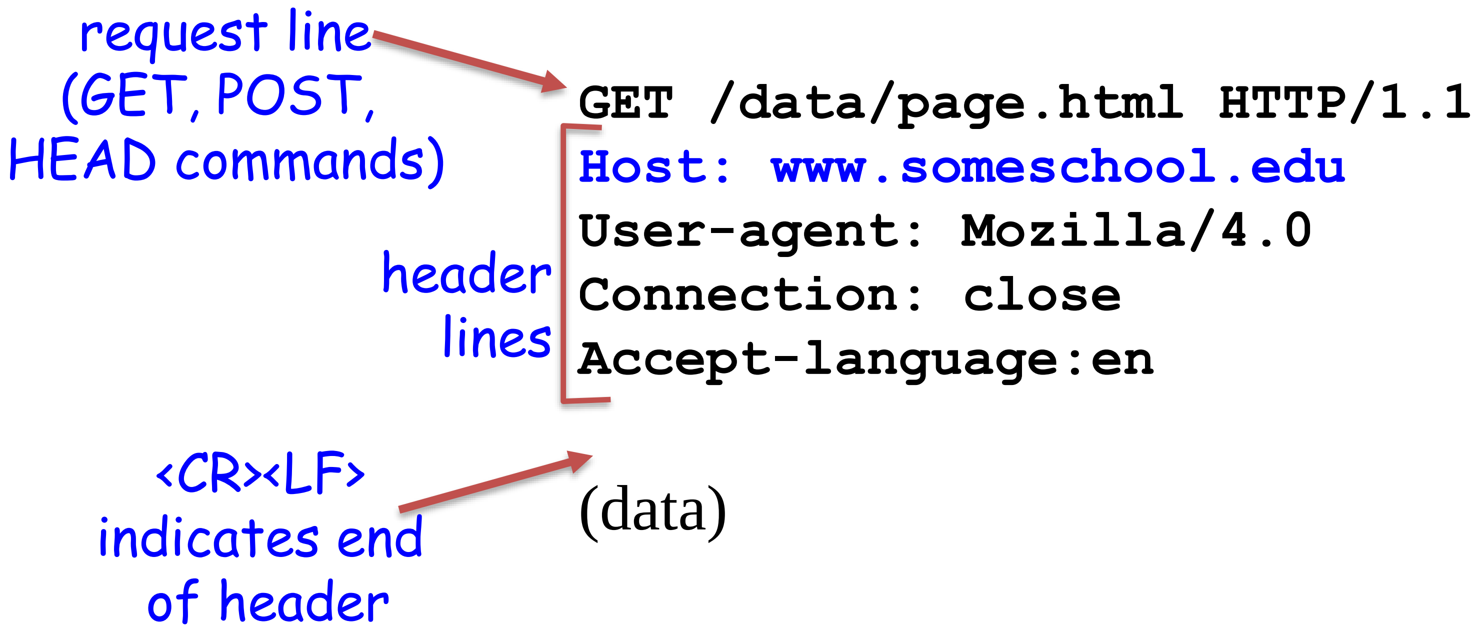
HTTP Request Message



Version: HTTP version, e.g., HTTP/1.1

HTTP Request Message Example

- HTTP request message:
  - ASCII (human-readable format)



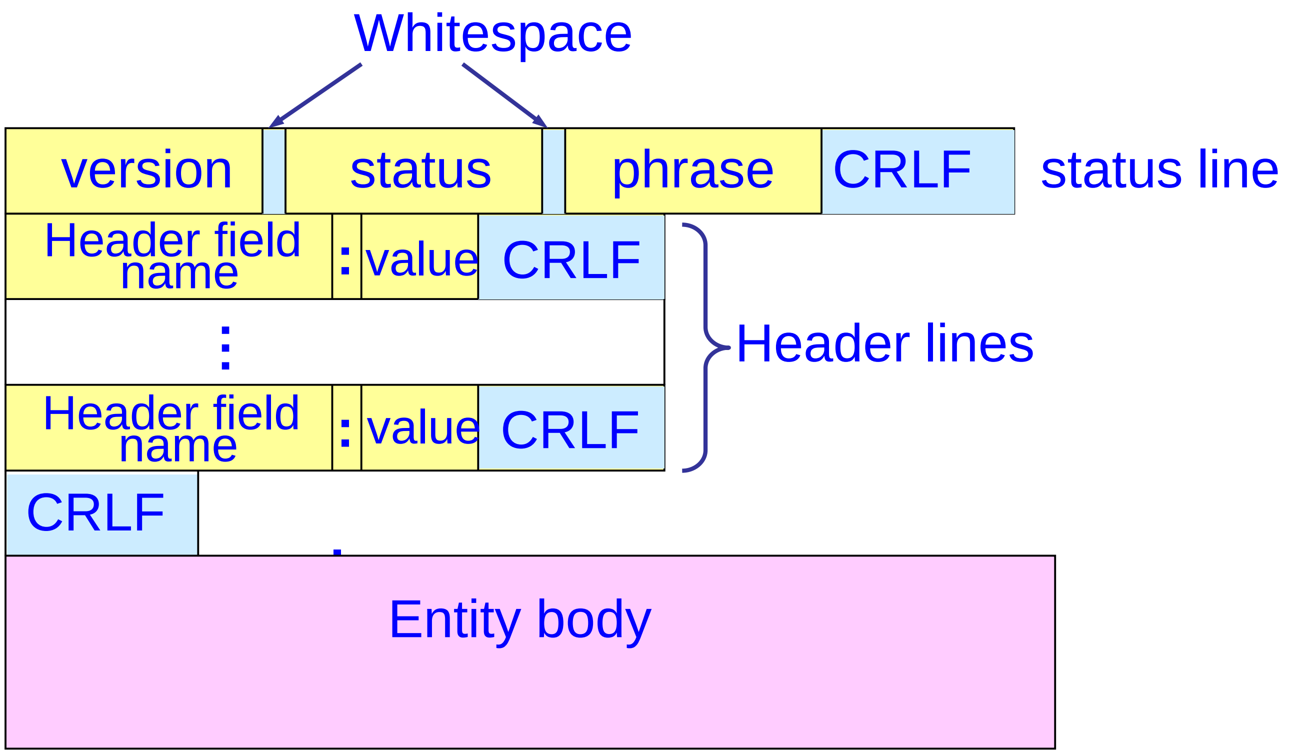
```
GET /data/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:en

(data)
```

request line (GET, POST, HEAD commands)
header lines
<CR><LF> indicates end of header

In the HTTP/1.1, all headers except Host are optional.

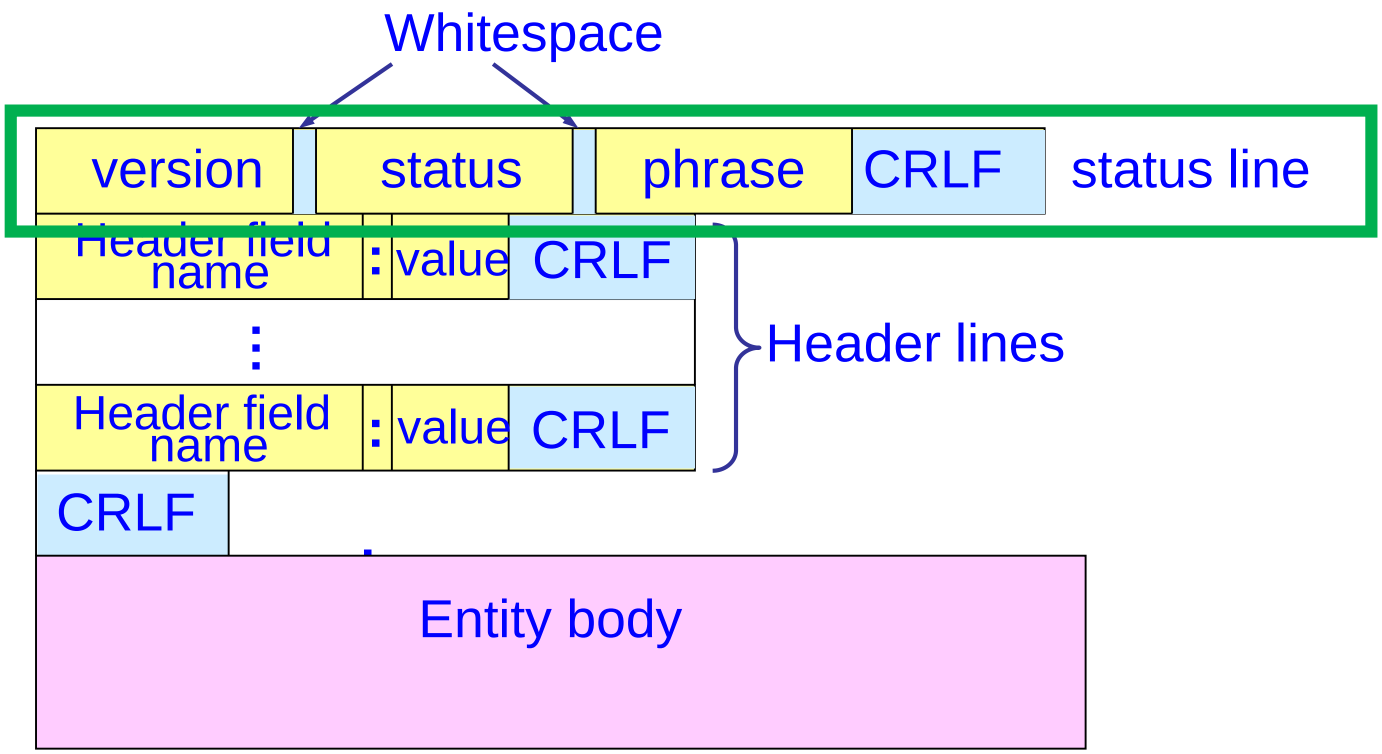
HTTP Response Message



The response message consists of the following:

- A Status-Line: e.g., HTTP/1.1 200 OK
- Response Headers, such as Content-Type: text/html
- An empty line
- An optional message body

HTTP Response Message

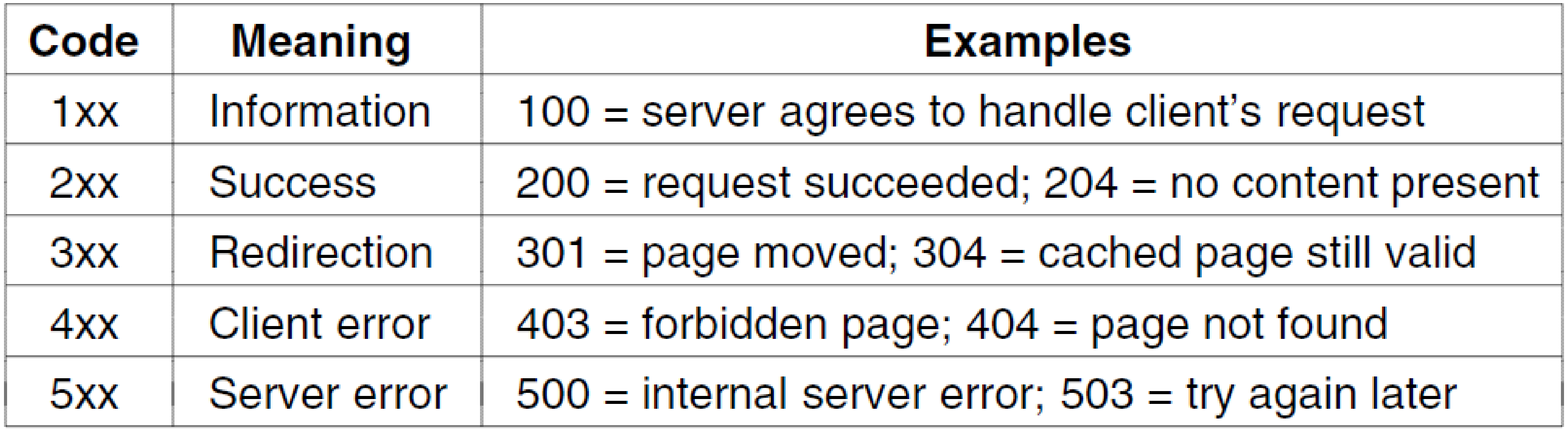


First line of response is the Status-Line*: HTTP version, status code, status phrase*

Example: HTTP/1.1 200 OK

HTTP Response: Status Code

- Response codes tell the client the results of the request:



Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Refer: HTTP response status codes and reason phrases

HTTP Response Message Example

status line
(protocol
status code
status phrase)

header
lines

data, e.g.,
requested
HTML file

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 2013 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 2013 …...
Content-Length: 6821
Content-Type: text/html

<html>
<head>
      <title> Computer Networks</title>
</head>
<body text=“#00000”>
      <img src=“images/new.gif/>
      <h1>Hi There!</h1>
      <a href=“more.html”>Click here</a>
</body>
</html>
```

HTTP Headers

Many headers carry key informations:

Function	**Example Headers**
Browser capabilities (client → server)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Caching related (mixed directions)	If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag
Browser context (client → server)	Cookie, Referer, Authorization, Host
Content delivery (server → client)	Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie

very imp

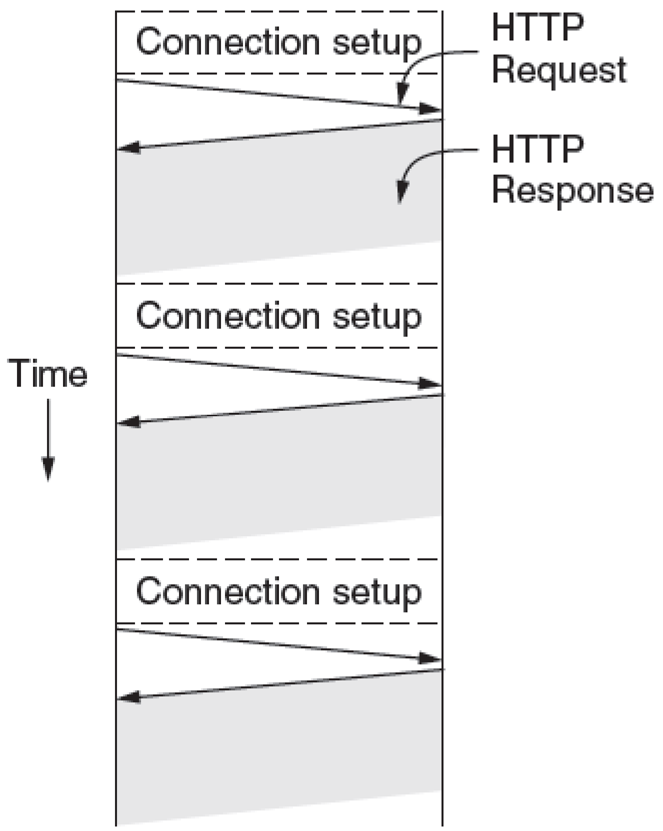
HTTP Connections

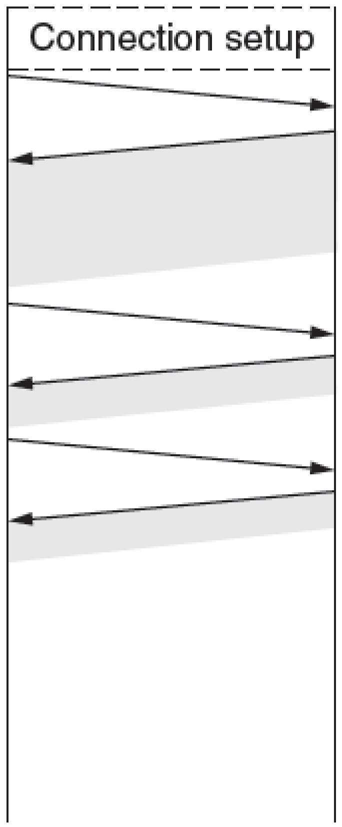
- Nonpersistent HTTP (非持续HTTP)
  - At most one object is sent over a TCP connection
  - Example: for a web page with multiple images, a TCP connection will be setup for each image
- Persistent HTTP (since HTTP/1.1) (持续HTTP)
  - Multiple objects can be sent over single TCP connection between client and server.
  - Further improvement using Pipeline: send another request before the previous response has arrived

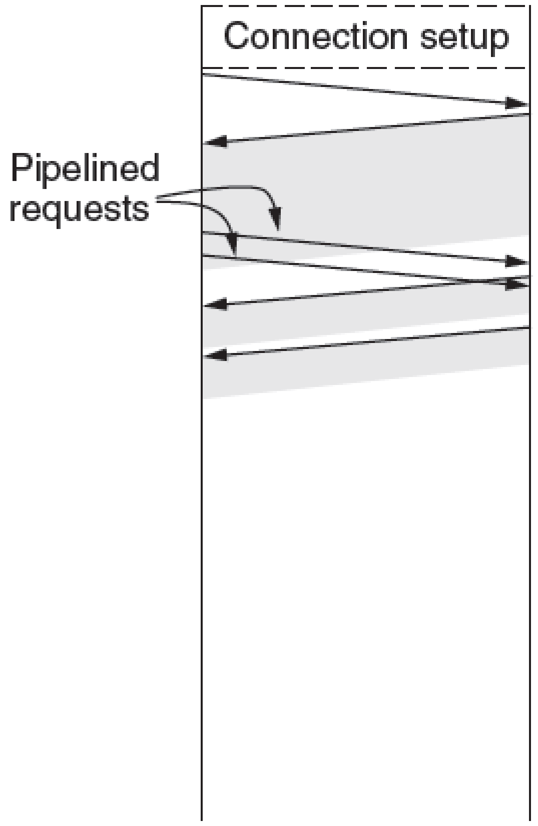
HTTP/1.1 was finalized in RFC 2616 in 1999

HTTP Connections

- HTTP uses persistent connections to improve performance







Nonpersistent HTTP:
One connection for each request

Persistent HTTP:
A persistent connection and sequential requests

Persistent HTTP (Pipeline):
A persistent connection and pipelined requests

HTTP Connections

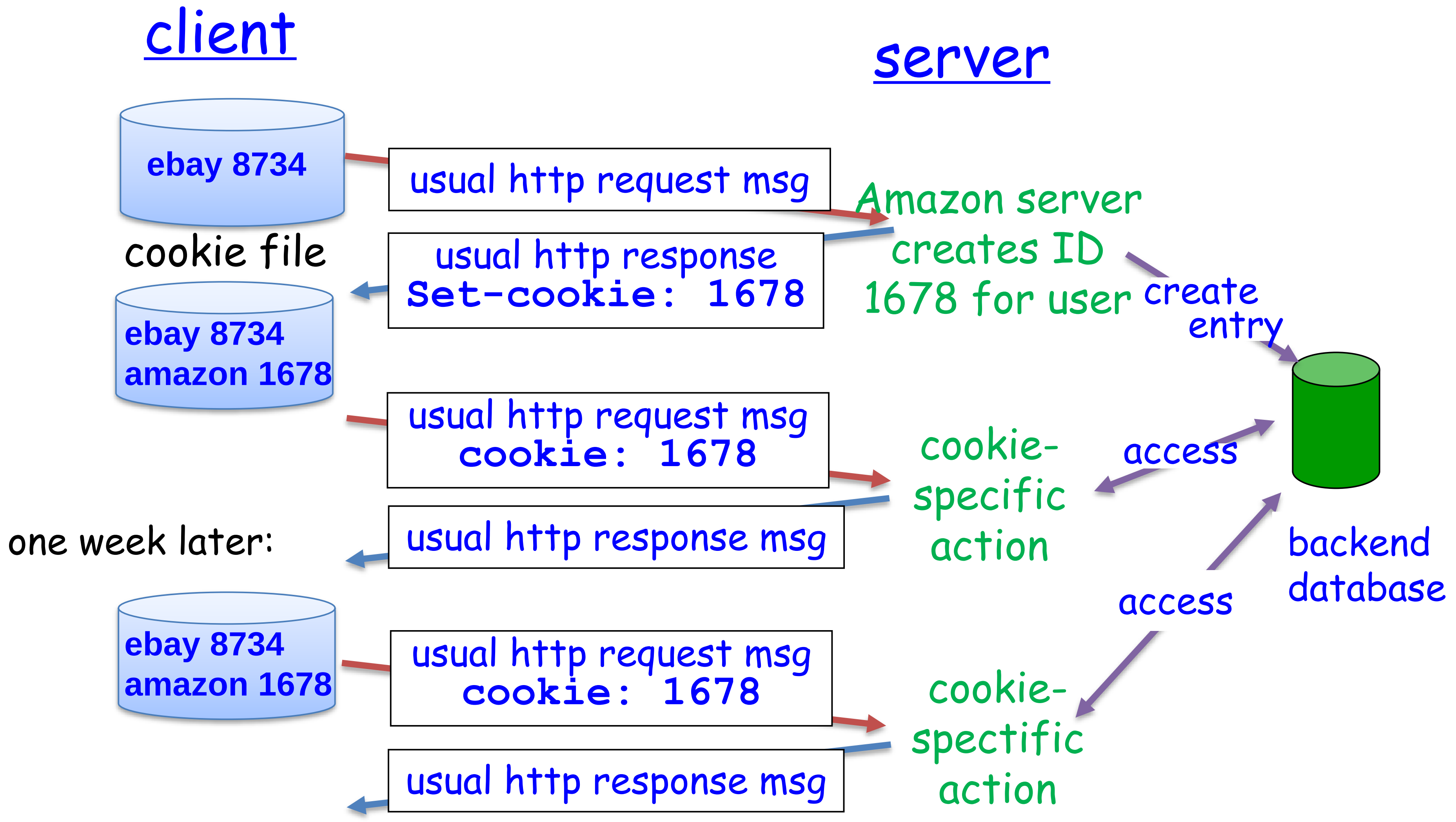
- Persistent connection is efficient because:
  - Time is not wasted for setting up additional connections
  - No *slow start* (TCP congestion) during following objects transmission, so connection is faster

Example: Chrome Developer Tools

HTTP Cookie

- HTTP is a stateless protocol.
- However, some web applications implement states or server side sessions, e.g., cookies
- Cookies support stateful client/server interactions
  - Server creates unique ID for each user
  - Cookies kept by both user’s host and backend database of web server
  - HTTP messages carry cookies (state)

HTTP Cookie


client
server
ebay 8734
cookie file
usual http request msg
Amazon server creates ID 1678 for user
usual http response
Set-cookie: 1678
ebay 8734
amazon 1678
create entry
usual http request msg
cookie: 1678
cookie-specific action
access
usual http response msg
one week later:
backend database
access
ebay 8734
amazon 1678
usual http request msg
cookie: 1678
cookie-specific action
usual http response msg

HTTP Cookie

- What cookies can bring:
  - authorization
  - shopping carts
  - recommendations
  - user session state

Google Cookie Example

- Google.com -> Search Settings
  - Search Language preference : English
  - SafeSearch Filtering: Strict Filtering
  - Number of Results: 50

Name	PREF
Value	ID=8d21b0e1ab97f420:U=b921ae77674ff57c:FF=1:LD=en:NR=50:CR=2:TM=1259142035:LM=1292848080:GM=1:SG=1:S=gT76_VBX8TqvHEcU
Domain	.google.com
Path	/
Secure	No
Expires	Thu, 19 Dec 2013 12:28:00 GMT

HTTP Caching

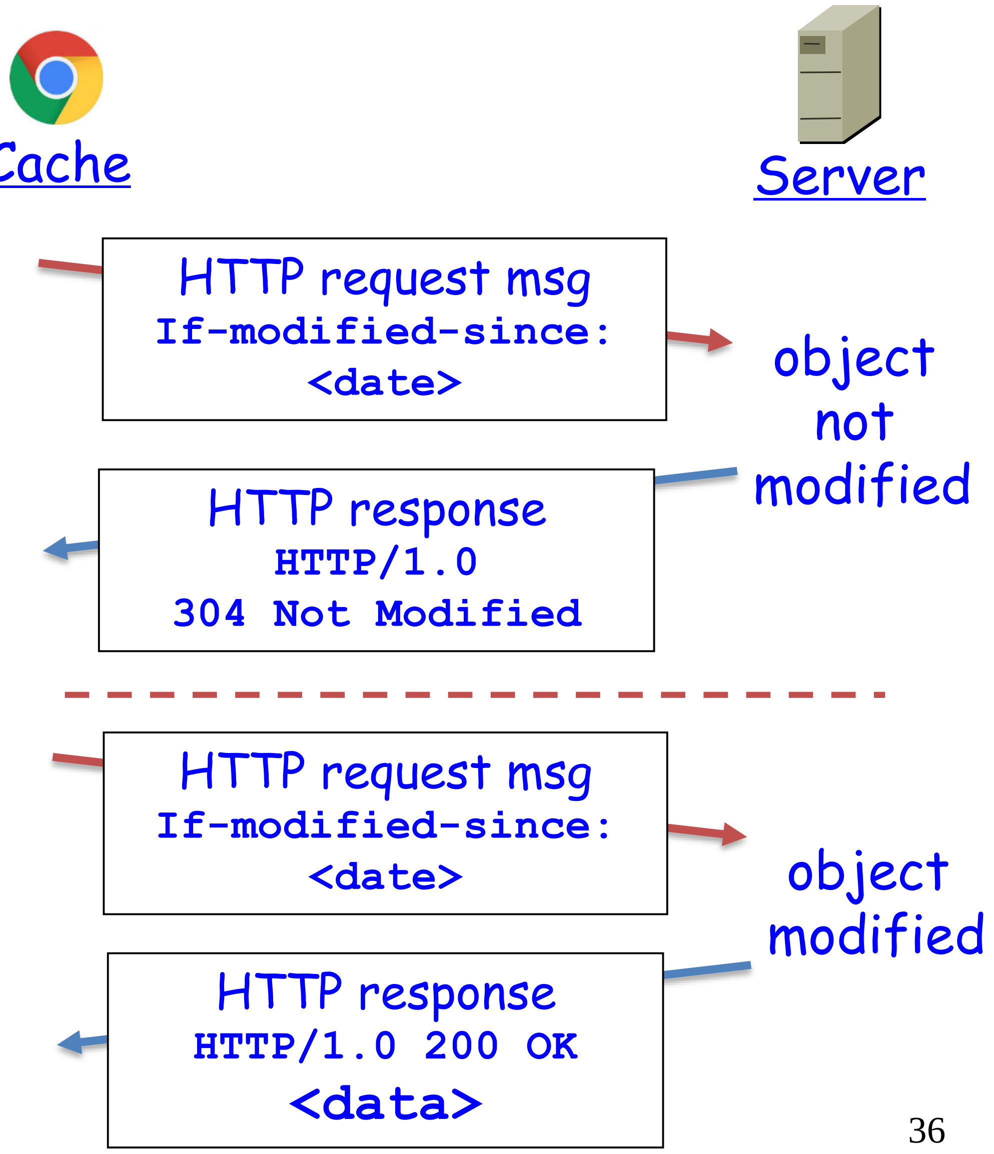
- People often revisit webpages
- Caching: Browser can cache the fetched webpage for subsequent use.
- Client don’t need to request the page if the browser has a known fresh copy
  - Reduce response time and traffic
  - Browsers need storage spaces for cached copies
- **Difficulty**: Pages may be changed. How to determine the cached copy is fresh enough and don’t need to fetch again?

HTTP Caching

- Page validation:
  - *Expires* header in HTTP response, indicating when the page must be fetched again
  - Problem: not all responses have *Expires* header
- Conditional GET:
  - Cached copy has the time of *Last-Modified*
  - Client sends this time using the *If-Modified-Since* header in HTTP request

Conditional GET

- **Goal**: don't send object if cache has up-to-date cached version
- **Cache**: specify date of cached copy in HTTP request
- **Server**: response contains no object if cached copy is up-to-date. Otherwise, send back a normal response.

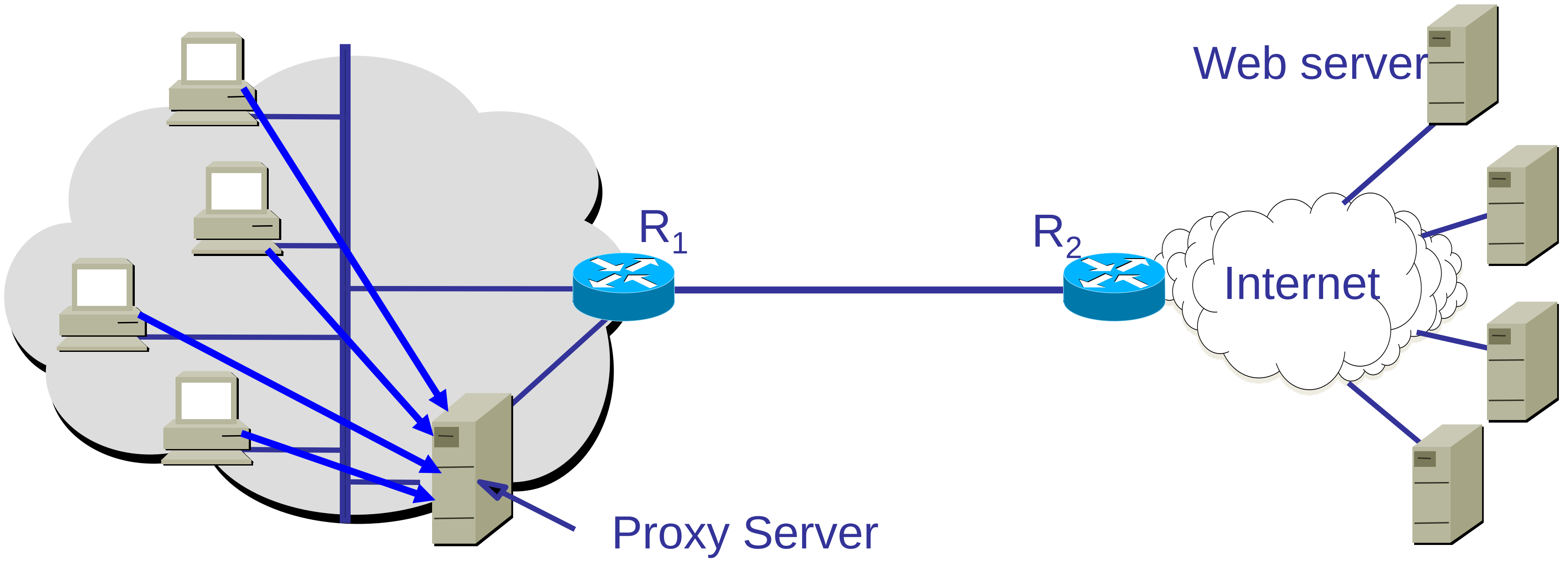


Caching: Proxy Server

- Caching can be performed at a proxy server besides browser
- Proxy Server acts as both client and server
- Proxy Server is typically installed by ISP (university, company, residential ISP)

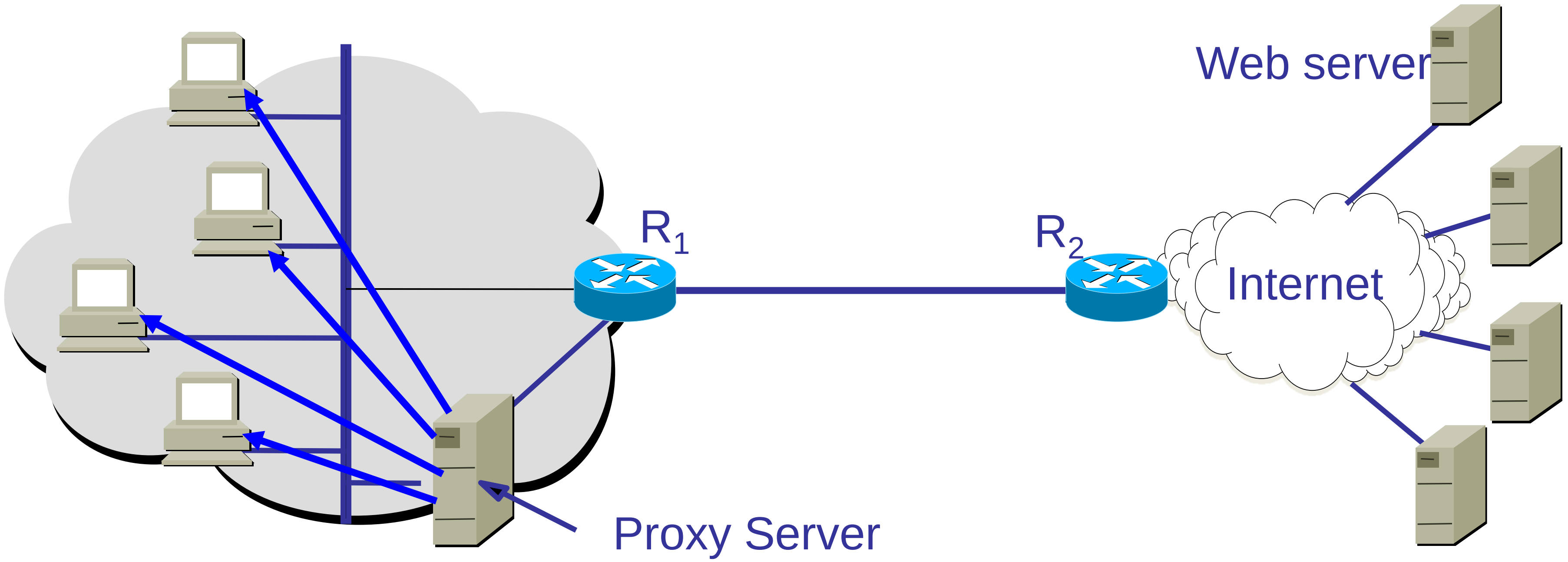
Caching: Proxy Server

Browsers send HTTP request to a proxy server



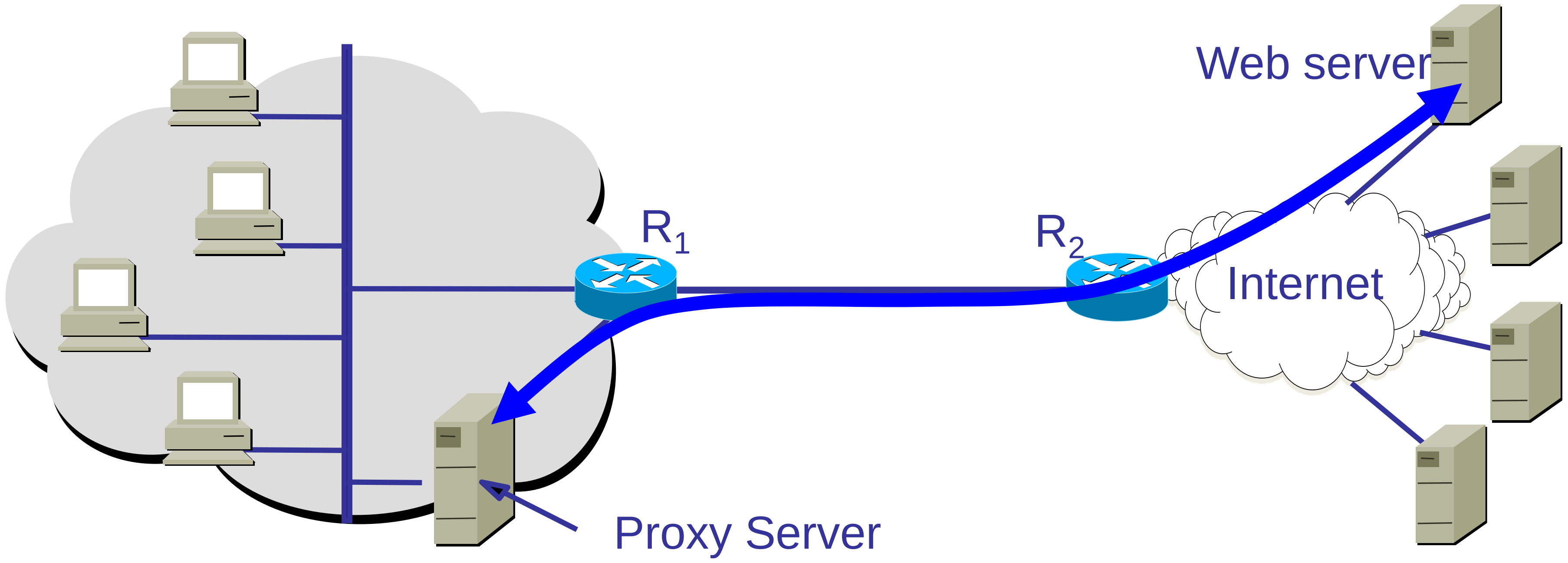
Caching: Proxy Server

Proxy server sends back HTTP response if it has an up-to-date cached copy



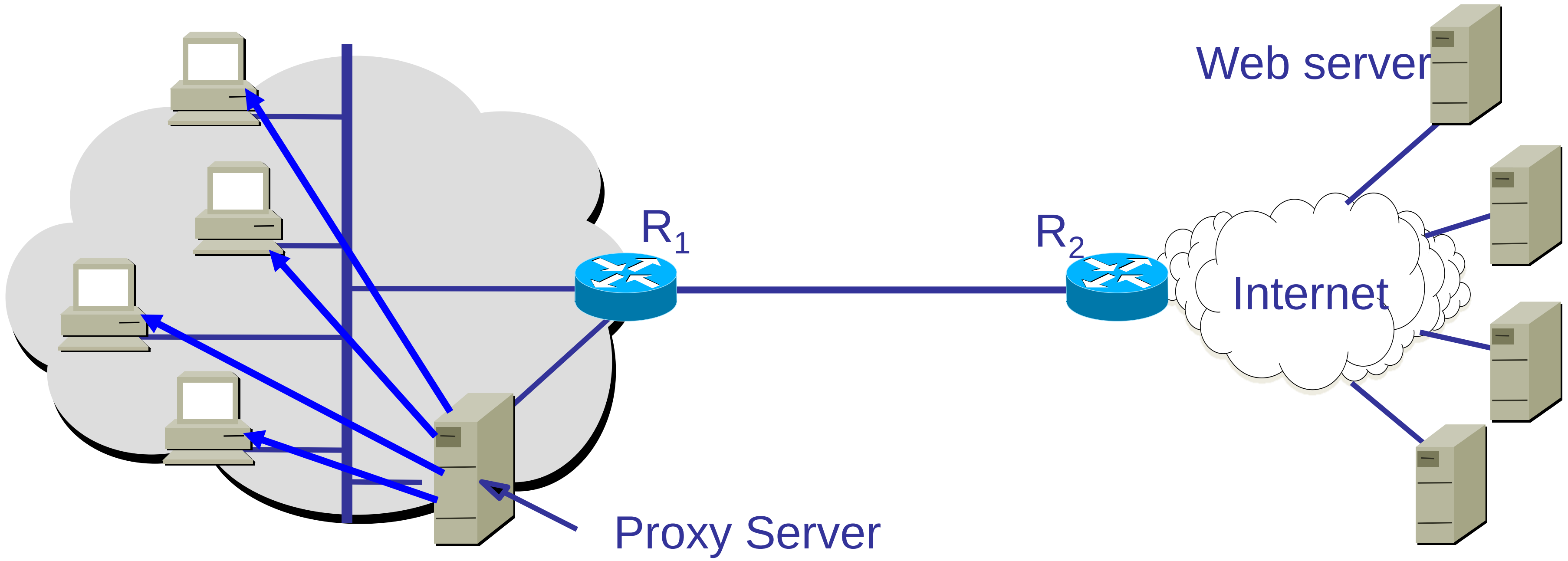
Caching: Proxy Server

Otherwise, proxy server sends HTTP request to web server to fetch a fresh copy



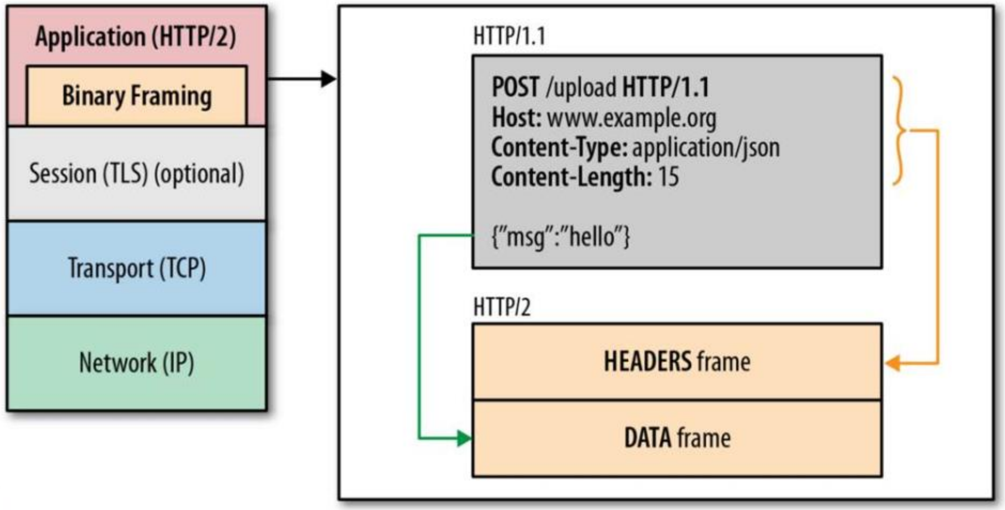
Caching: Proxy Server

Cache the web page and send HTTP response to users.



HTTP/2

- HTTP/2, published in RFC 7540 in May 2015
  - The first new version of HTTP since HTTP 1.1
  - As of Oct. 2021, ~ 50% of the top 10 million websites supported HTTP/2
- Main features:
  - One TCP connection
  - Request → Stream
    - Streams are multiplexed
    - Streams are prioritized
  - Binary framing layer
    - Prioritization
    - Flow control
    - Server push
  - Header compression (HPACK)



```
POST /upload HTTP/1.1
Host: www.example.org
Content-Type: application/json
Content-Length: 15

{"msg":"hello"}
```

HTTP/1.x vs HTTP/2

HTTP/3

- HTTP/3 is the proposed successor (Internet Draft) to HTTP/2
  - based on “Hypertext Transfer Protocol (HTTP) over QUIC” (renamed to HTTP/3 in November 2018)
  - Builds upon HTTP/2
  - QUIC: a transport protocol based on UDP
- Support of HTTP/3
  - Chrome, Cloudflare, Firefox (2019.11),…
  - As of Dec. 2021, supported by 73% of running web browsers, and 24% of the top 10 million website

HTTP-over-QUIC to be renamed HTTP/3

Why QUIC? Why UDP?

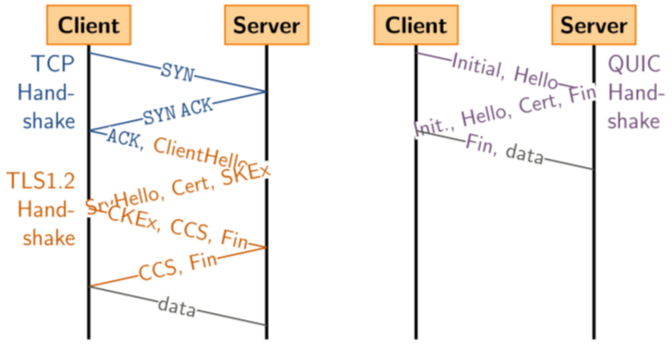
- TCP is now so ossified that introducing any changes to its basic operation must be done very carefully
  - This harms the ability of the protocol to meet the needs of its users.
  - TCP may not receive major revision in foreseeable future.
- Thus, we restart all over again on UDP
  - Implementing and improving all the components of TCP upon UDP and in userland.

Major changes of QUIC

- Low connection establish time with built-in encryption
- Better congestion control mechanism
  - Monotonically increased packet number and stream offset
  - No Reneging
  - More ACK block
  - ACK delay time
- Multiplexing without head-of-line blocking
- Connection migration

Example: Handshake of QUIC compared to TCP with TLS1.2

- QUIC aims to be nearly equivalent to a TCP connection but with much-reduced latency through two major changes:
  - Greatly reduce overhead during connection setup.
  - Use UDP rather than TCP as its basis, which does not include loss recovery.



Example: Handshake of QUIC compared to TCP with TLS1.2

Three Essential Technologies

- URL (Uniform Resource Locator)
  - A system of globally unique identifiers for resources on the Web and elsewhere
  - Each object is addressable by a unique URL
- HTTP (HyperText Transfer Protocol):
  - Foundation of data communication for WWW
- HTML (HyperText Markup Language):
  - The main markup language for creating web pages and other information that can be displayed in a web browser

HTML

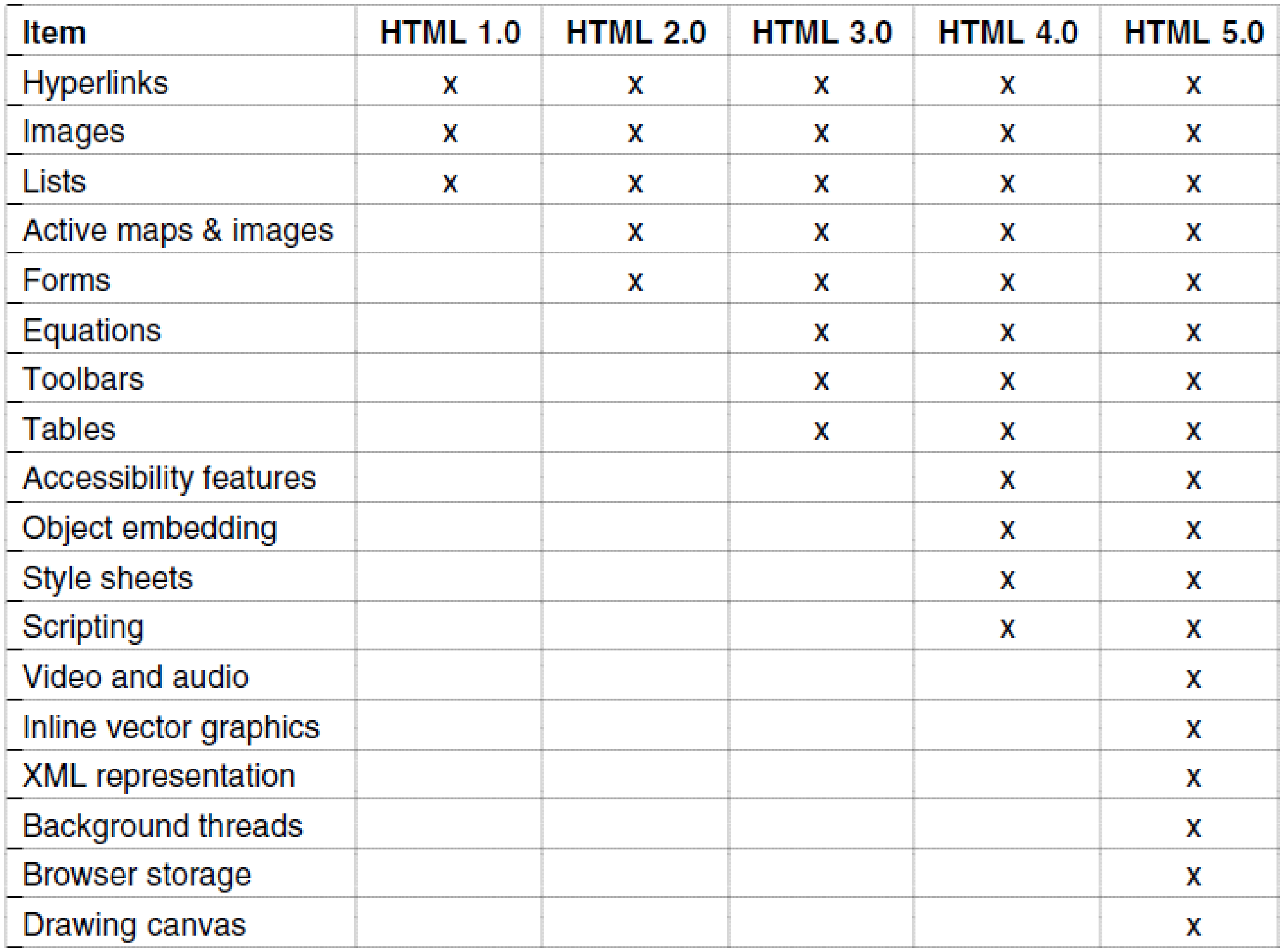
- HTML (HyperText Markup Language):
  - A standardized language to produce and display web pages that include text, graphics, video, hyperlinks, etc.
- HTML are printable ASCII text
- HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets, e.g., <img>
- HTML common filename extension: .html, .htm

HTML Example

```
<html>
<head>
      <meta name=“Author” content=“Anonymous”>
      <title> Linux Web Server Performance</title>
</head>
<body text=“#00000”>
      <img width=31 height=11 src=“ibmlogo.gif”/>
      <img src=“images/new.gif/>
      <h1>Hi There!</h1>
      Here’s lots of cool linux stuff!
      <a href=“more.html”>Click here</a> for more!
</body>
</html>
```

HTML

Progression of features through HTML 5.0



Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	x	x	x	x	x
Images	x	x	x	x	x
Lists	x	x	x	x	x
Active maps & images		x	x	x	x
Forms		x	x	x	x
Equations			x	x	x
Toolbars			x	x	x
Tables			x	x	x
Accessibility features				x	x
Object embedding				x	x
Style sheets				x	x
Scripting				x	x
Video and audio					x
Inline vector graphics					x
XML representation					x
Background threads					x
Browser storage					x
Drawing canvas					x

HTML 5 is finalized in Oct. 2014

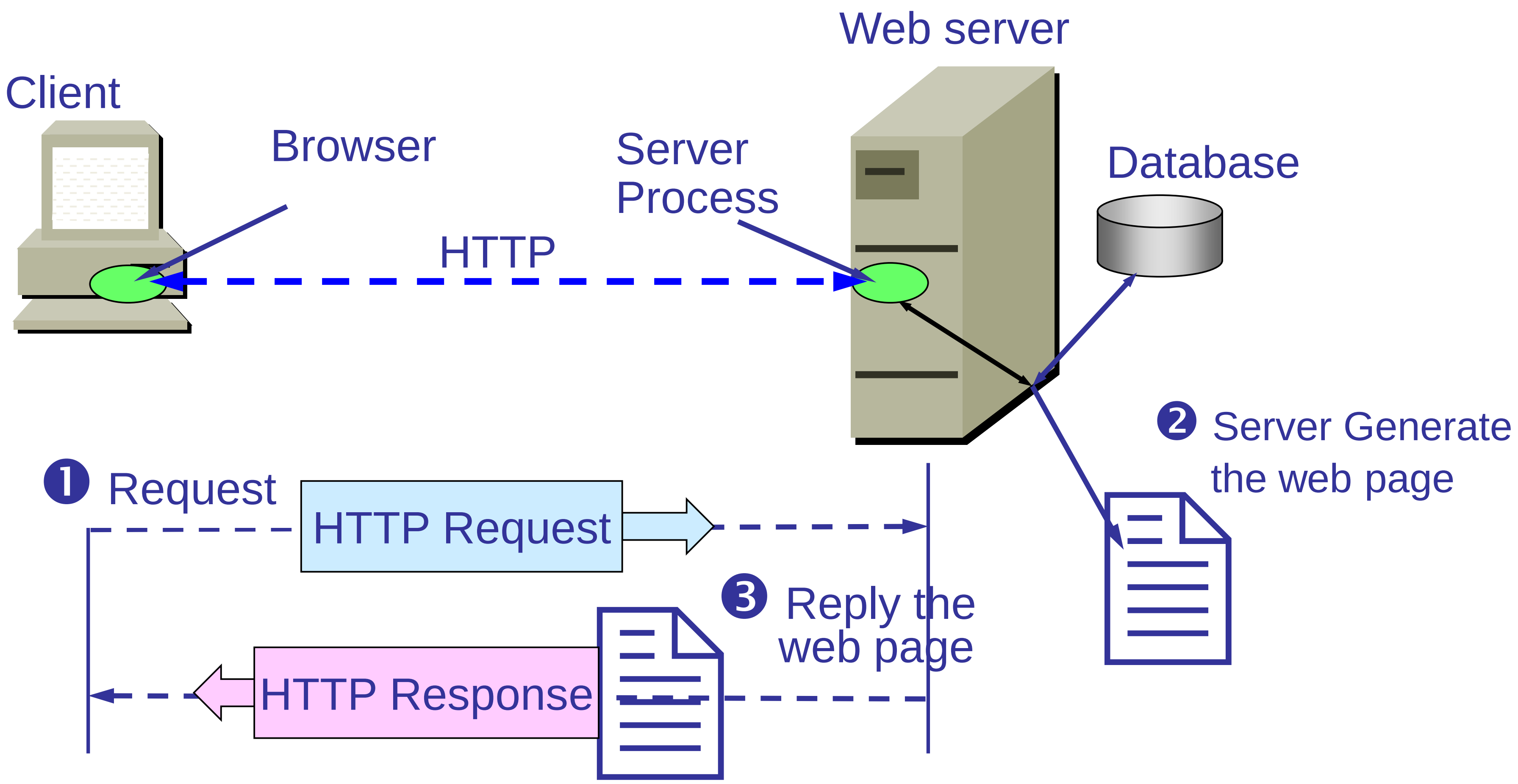
Static Web Pages

- Static Web pages are simply files
  - Have the same contents for each viewing
- Can be visually rich but no interaction with user:
  - HTML that mixes text and images, etc.
  - Forms that gather user input
  - Style sheets that tailor presentation
  - Vector graphics, videos, and more (over) . . .

Dynamic Pages

- Dynamic pages are generated by programs running at the server (with a database) and the client
  - Pages vary each time like using an application
- Accepting and processing client's input data
- Can be implemented on both
  - Server-side
  - Client-side

Server-side Dynamic Web Page



Server-side Dynamic Web Page

- Two example ways: CGI and embedded scripts
- CGI (Common Gateway Interface), RFC 3875
  - Call back-end programs, accepting client's input and generating HTML pages in response
  - Language: Python, Ruby, Perl, etc.

CGI Example

```
#!/usr/bin/env python3

print("Content-Type: text/html\n\n")  # html markup follows

print("""
<html>
  <Title>Hello in HTML</Title>
<body>
  <p>Hello There!</p>
  <p><b>Hi There!</b></p>
</body>
</html> """)
```

Server-side Dynamic Web Page

- Embedded Scripts:
  - Embedded scripts inside HTML pages and execute them by server itself to generate web pages
  - Example: PHP (PHP: Hypertext Preprocessor), JSP (JavaServer Pages), ASP.NET (Active Server Pages .NET)

PHP Example

Web page that gets form input and calls a server program

```
<html>
<body>
<form action="action.php" method="post">
<p> Please enter your name: <input type="text" name="name"> </p>
<p> Please enter your age: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

PHP server program that creates a custom Web page

PHP scripts

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```

Resulting Web page (for inputs “Barbara” and “32”)

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```

Client-side Dynamic Web Page

- Dynamic HTML: Response to mouse movements or directly interact with users
- JavaScript:
  - a most popular scripting language
- VBScript:
  - For windows platforms
- Java Applets:
  - Java programs compiled by JVM (Java Virtual Machine)

JavaScript Example

JavaScript program produces result page in the browser

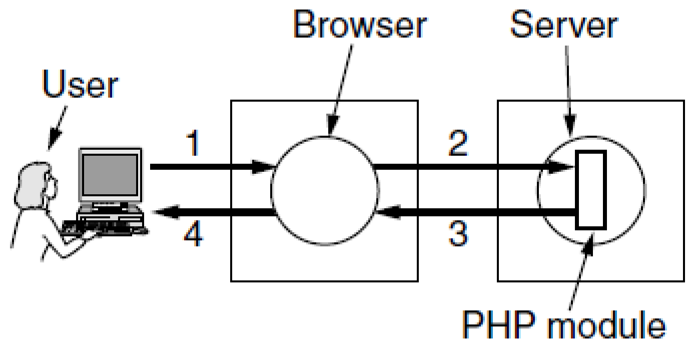
```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
   var person = test_form.name.value;
   var years = eval(test_form.age.value) + 1;
   document.open();
   document.writeln("<html> <body>");
   document.writeln("Hello " + person + ".<br>");
   document.writeln("Prediction: next year you will be " + years + ".");
   document.writeln("</body> </html>");
   document.close();
}
</script>
</head>
```

First page with form, gets input and calls program above

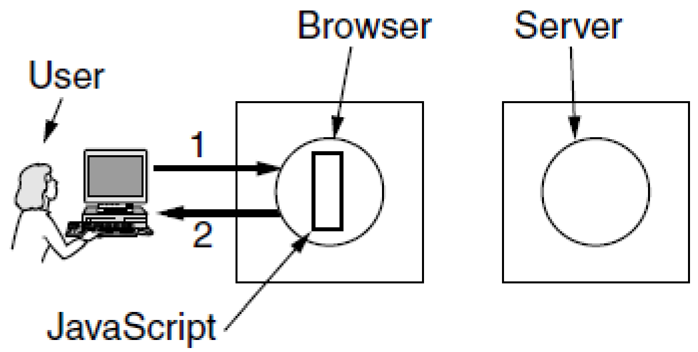
```
<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

Dynamic Web Pages

- The difference between server and client programs



Server-side scripting with PHP



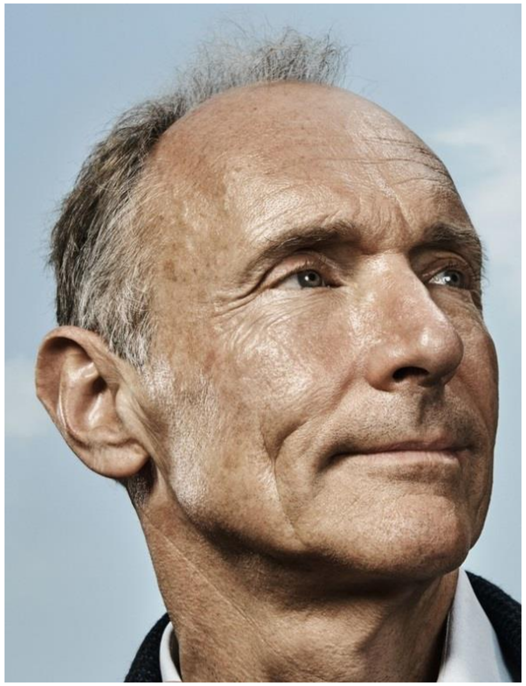
Client-side scripting with JavaScript

Dynamic Pages & Web Applications

- Web applications use a set of technologies that work together, e.g.
  - HTML: present information as pages.
  - DOM: Document Object Model, change parts of pages while they are viewed.
  - XML: let programs exchange data with the server.
  - AJAX: Asynchronous way to send and retrieve XML data.
  - JavaScript as a language to bind all this together.

Tim Berners-Lee: I WAS DEVASTATED

- “I WAS DEVASTATED”: TIM BERNERS-LEE, THE MAN WHO CREATED THE WORLD WIDE WEB, HAS SOME REGRETS
- Solid (Social Linked Data), 2018
  - a web decentralization project led by Tim Berners-Lee
  - Run from MIT
  - Wiki



Topics

- DHCP: Dynamic Host Configuration Protocol (Chapter 5.6.4)
- DNS: Domain Name System
- The World Wide Web: HTTP
- Electronic Email

China’s First Email

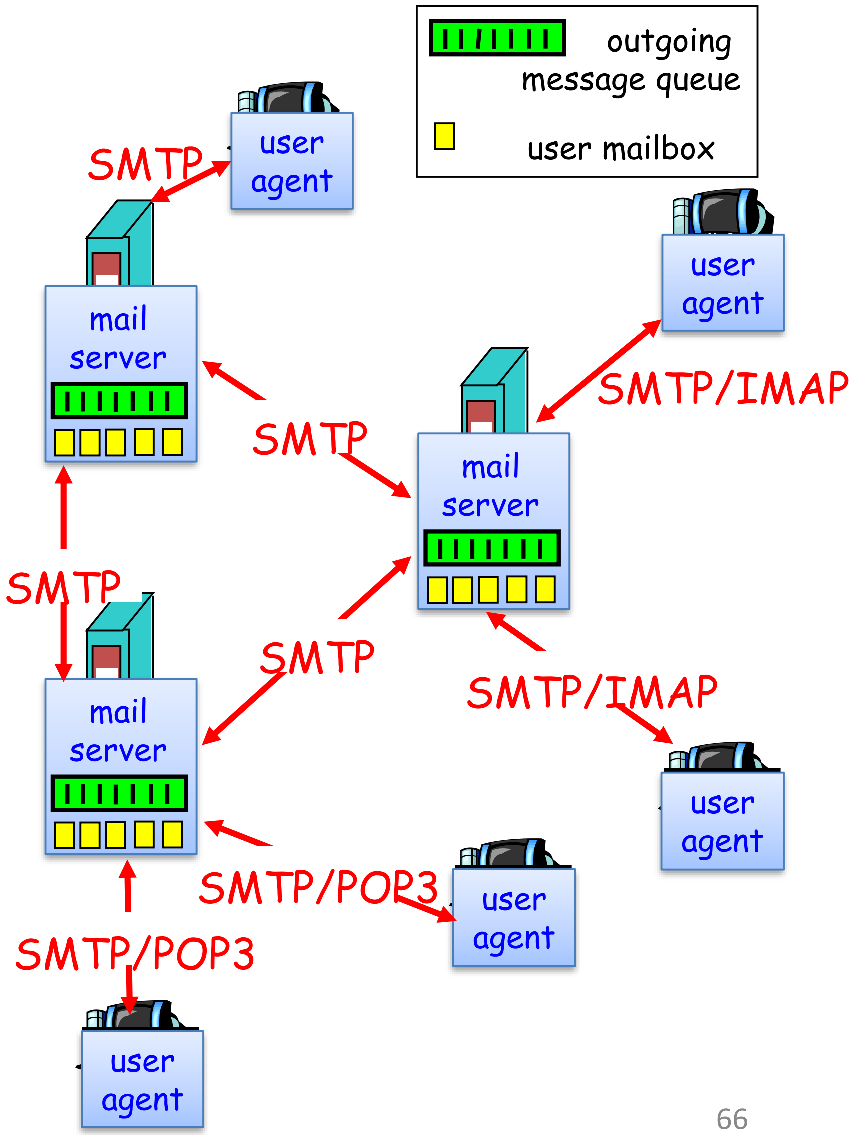
- Sep. 14th, 1987 21:07, sent from Beijing to Germany

"Across the Great Wall we can reach every corner in the world"

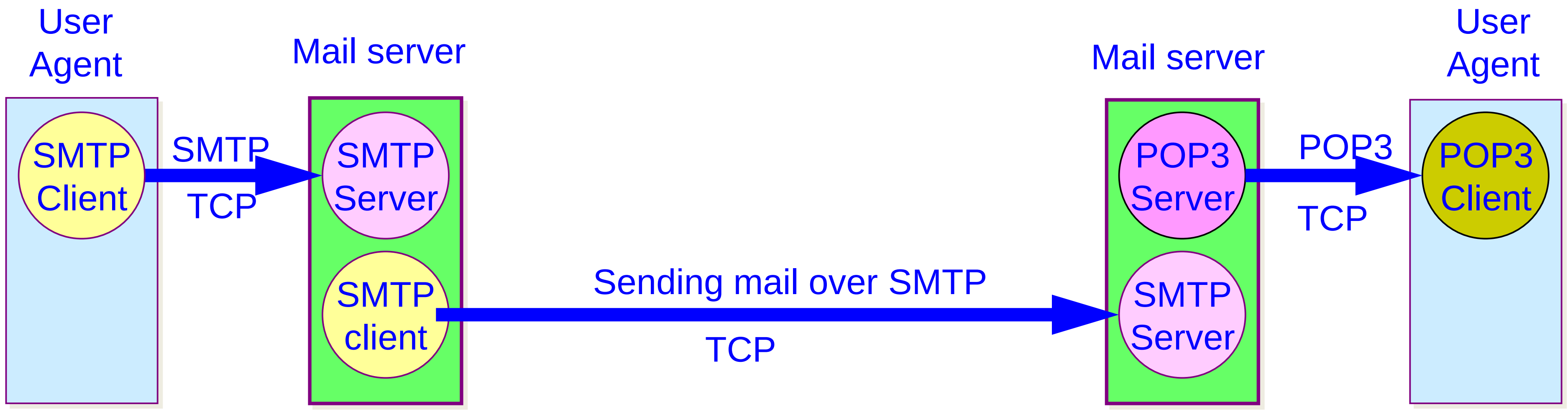
- Read more [here]

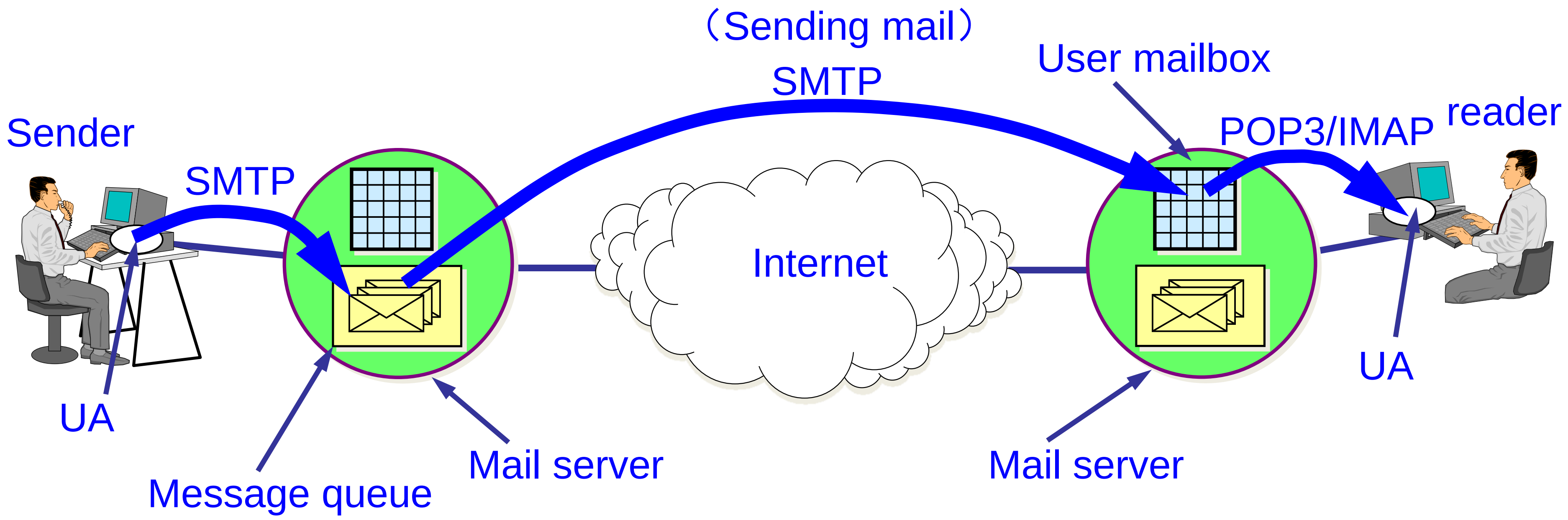
Electronic Mail

- Four components
  - User agents
    - mail reader/composer
    - Outlook, Apple Mail, Thunderbird, Foxmail
  - Mail servers (Message transfer agents)
    - mail.google.com
    - mail.jnu.edu.cn
  - Mail transfer protocol
    - SMTP
  - Mail access protocol
    - POP3/IMAP



Mail Components





Mail Server

- Mailbox
  - Contains incoming messages for user
- Message queue
  - Outgoing (to be sent) mail messages

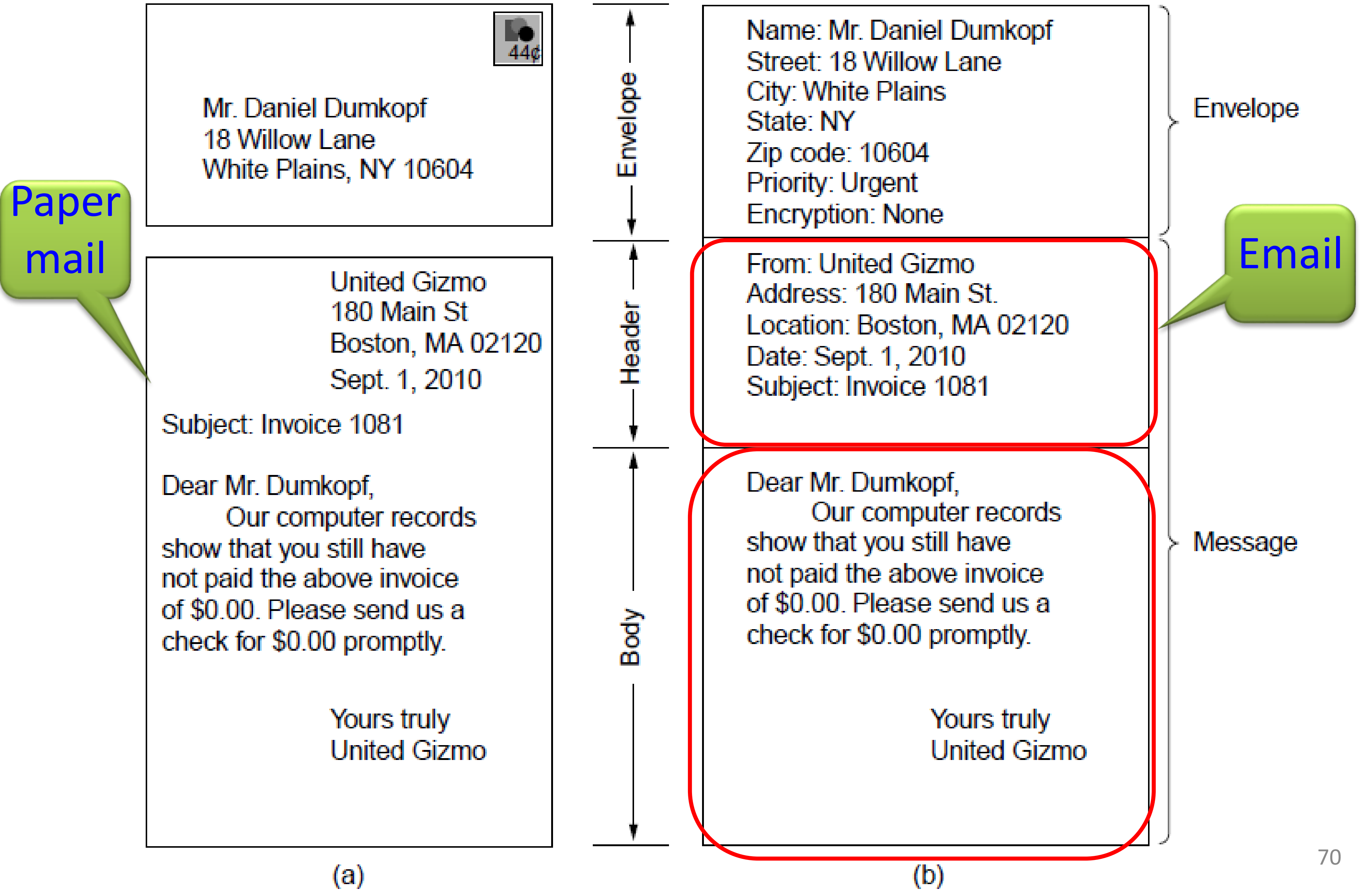
Email Message Format

- Email address:

[username] @ [domain name of mail server]

- Email message is encapsulated in the *envelop*
  - Contains info. needed for transporting the message, e.g., destination, priority, security level
- Message content includes two parts:
  - Header: control info. for User Agent (UA)
  - Body: real message that user to send

Envelopes and messages



44¢

Mr. Daniel Dumkopf
18 Willow Lane
White Plains, NY 10604

United Gizmo
180 Main St
Boston, MA 02120
Sept. 1, 2010

Subject: Invoice 1081

Dear Mr. Dumkopf,
Our computer records show that you still have not paid the above invoice of $0.00. Please send us a check for $0.00 promptly.

Yours truly
United Gizmo

(a)

Name: Mr. Daniel Dumkopf
Street: 18 Willow Lane
City: White Plains
State: NY
Zip code: 10604
Priority: Urgent
Encryption: None

From: United Gizmo
Address: 180 Main St.
Location: Boston, MA 02120
Date: Sept. 1, 2010
Subject: Invoice 1081

Dear Mr. Dumkopf,
Our computer records show that you still have not paid the above invoice of $0.00. Please send us a check for $0.00 promptly.

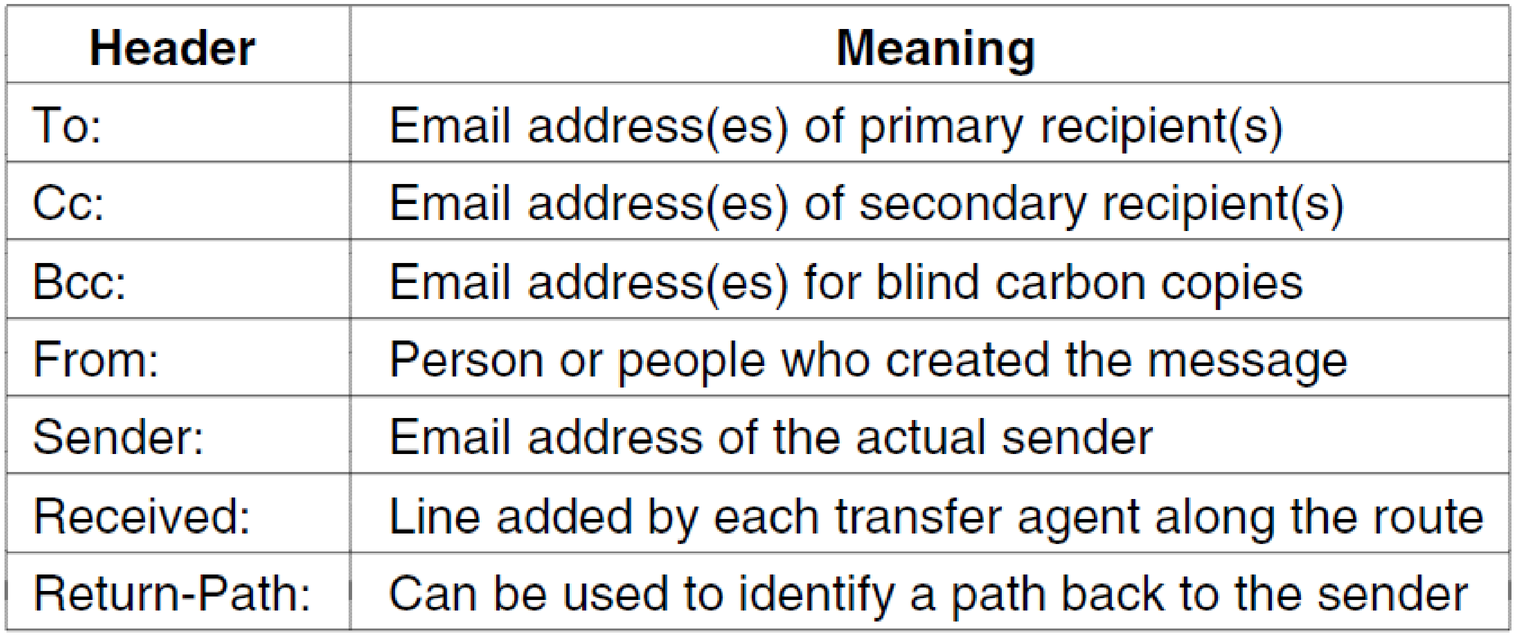
Yours truly
United Gizmo

(b)

Message Formats (1)

1

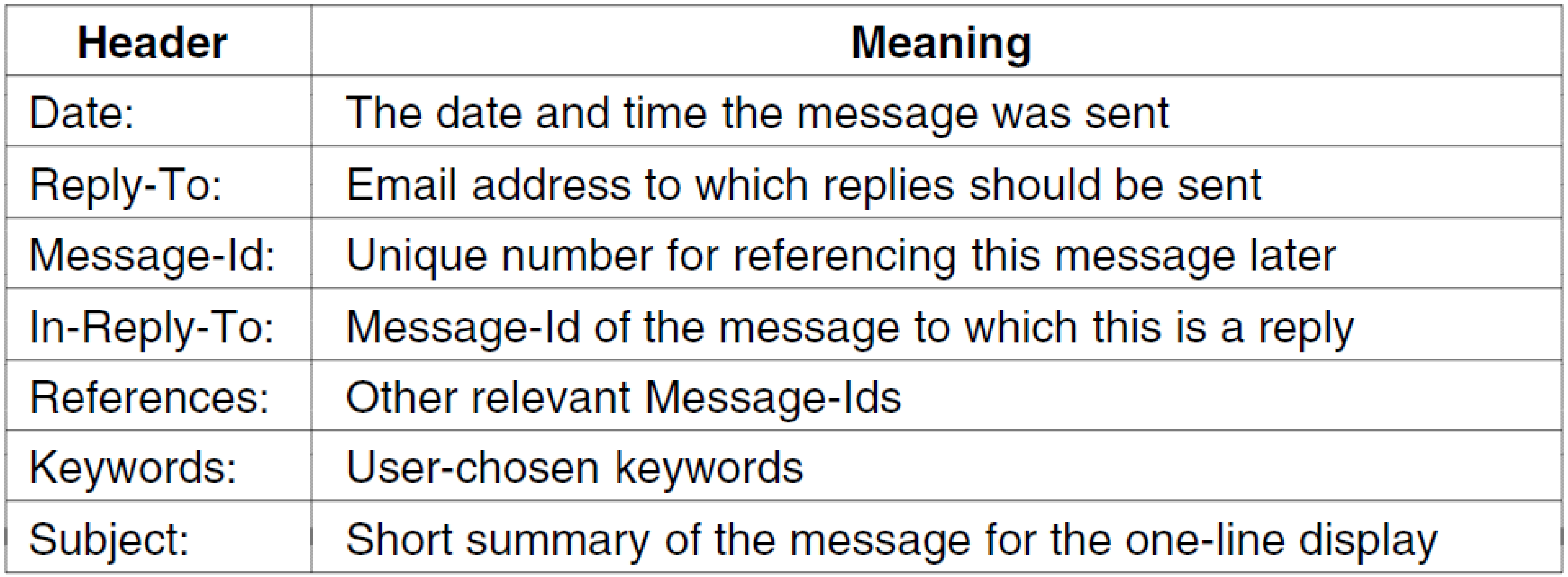
- The Internet Message Format, RFC822, 5322
- Header fields related to message transport;
  - Each header field is readable ASCII text for a single line
  - Header and message body are separated by a blank line



Header	Meaning
To:	Email address(es) of primary recipient(s)
Cc:	Email address(es) of secondary recipient(s)
Bcc:	Email address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	Email address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

Message Formats (2)

- Other header fields useful for user agents



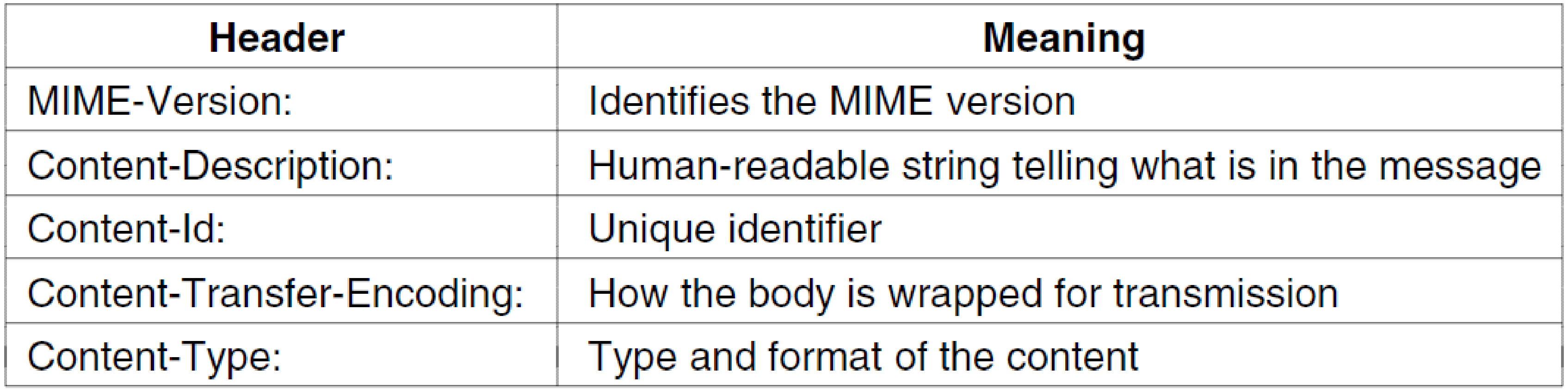
Header	Meaning
Date:	The date and time the message was sent
Reply-To:	Email address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User-chosen keywords
Subject:	Short summary of the message for the one-line display

MIME

- In early Internet (ARPANET), emails are written in English and expressed in ASCII
  - ASCII char has 7 bits, no more than 1000 chars each line
- MIME (Multipurpose Internet Mail Extensions), RFCs 2045~2047, etc.
  - Use the basic email format, but add new structure and rules for non-ASCII messages
  - Base64 encoding:
    - Groups of 24bits are broken up into four 6-bits units
    - 64 units coded by: A~Z, a~z, 0~9, +, /
  - Quoted-printable encoding, ref. wiki for more:
    - For message with only few non-ASCII chars
    - Chars above 127: “=” and two hexadecimal digits

MIME Header Fields

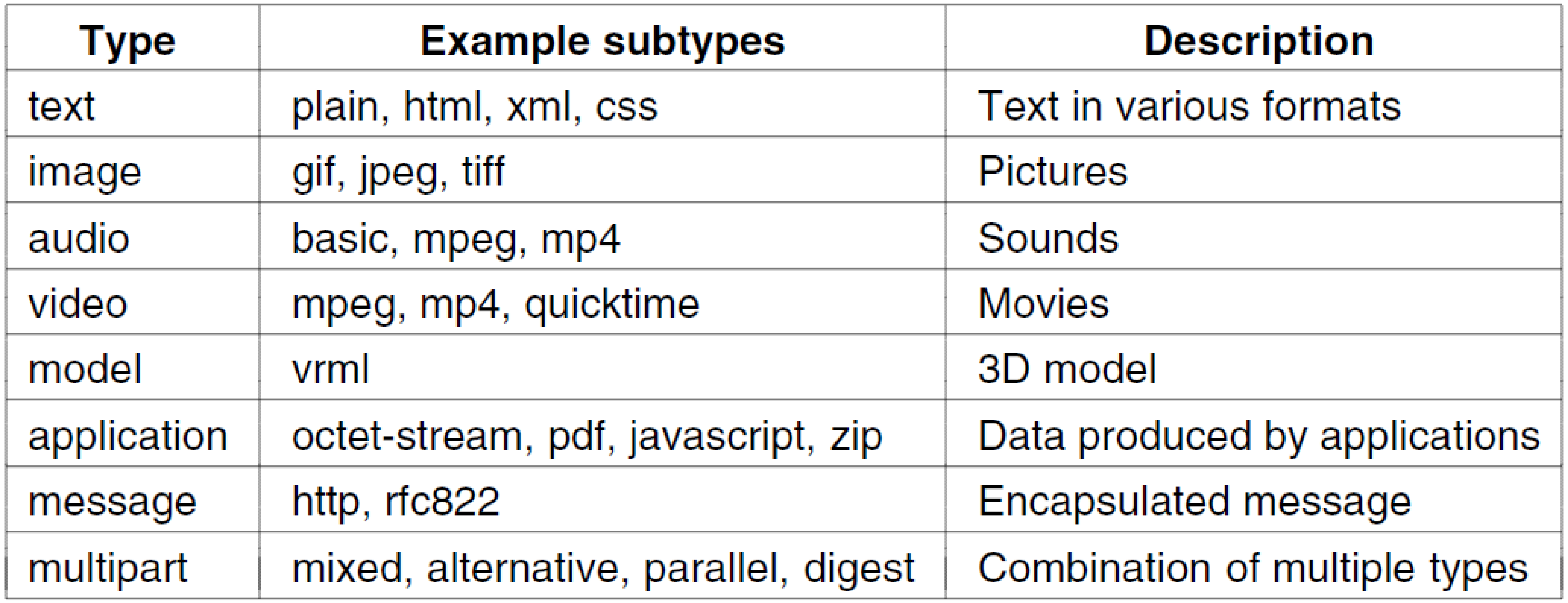
- MIME header fields used to describe what content is in the body of the message



Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

MIME Types Example

- Common MIME content types and subtypes



Type	Example subtypes	Description
text	plain, html, xml, css	Text in various formats
image	gif, jpeg, tiff	Pictures
audio	basic, mpeg, mp4	Sounds
video	mpeg, mp4, quicktime	Movies
model	vrml	3D model
application	octet-stream, pdf, javascript, zip	Data produced by applications
message	http, rfc822	Encapsulated message
multipart	mixed, alternative, parallel, digest	Combination of multiple types

An Email Message Example

A multipart message containing HTML and audio alternatives:

```
From: alice@cs.washington.edu
To: bob@ee.uwa.edu.au
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@cs.washington.edu>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/html

<p>Happy birthday to you<br>
Happy birthday to you<br>
Happy birthday dear <b> Bob </b><br>
Happy birthday to you</p>
```

One part (HTML)

```
--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
        access-type="anon-ftp";
        site="bicycle.cs.washington.edu";
        directory="pub";
        name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Another (audio)

See a Gmail Example

Simple Mail Transfer Protocol

- SMTP: A simple ASCII protocol, RFC2821
- SMTP Works on port 25, TCP
- Direct delivery: from sender's server to receiver's server
- Message:
  - Must be 7-bit ASCII encoded
- Extended SMTP (ESMTP), RFC 5321
  - with authentication and encryption

Three Phases Submission of SMTP

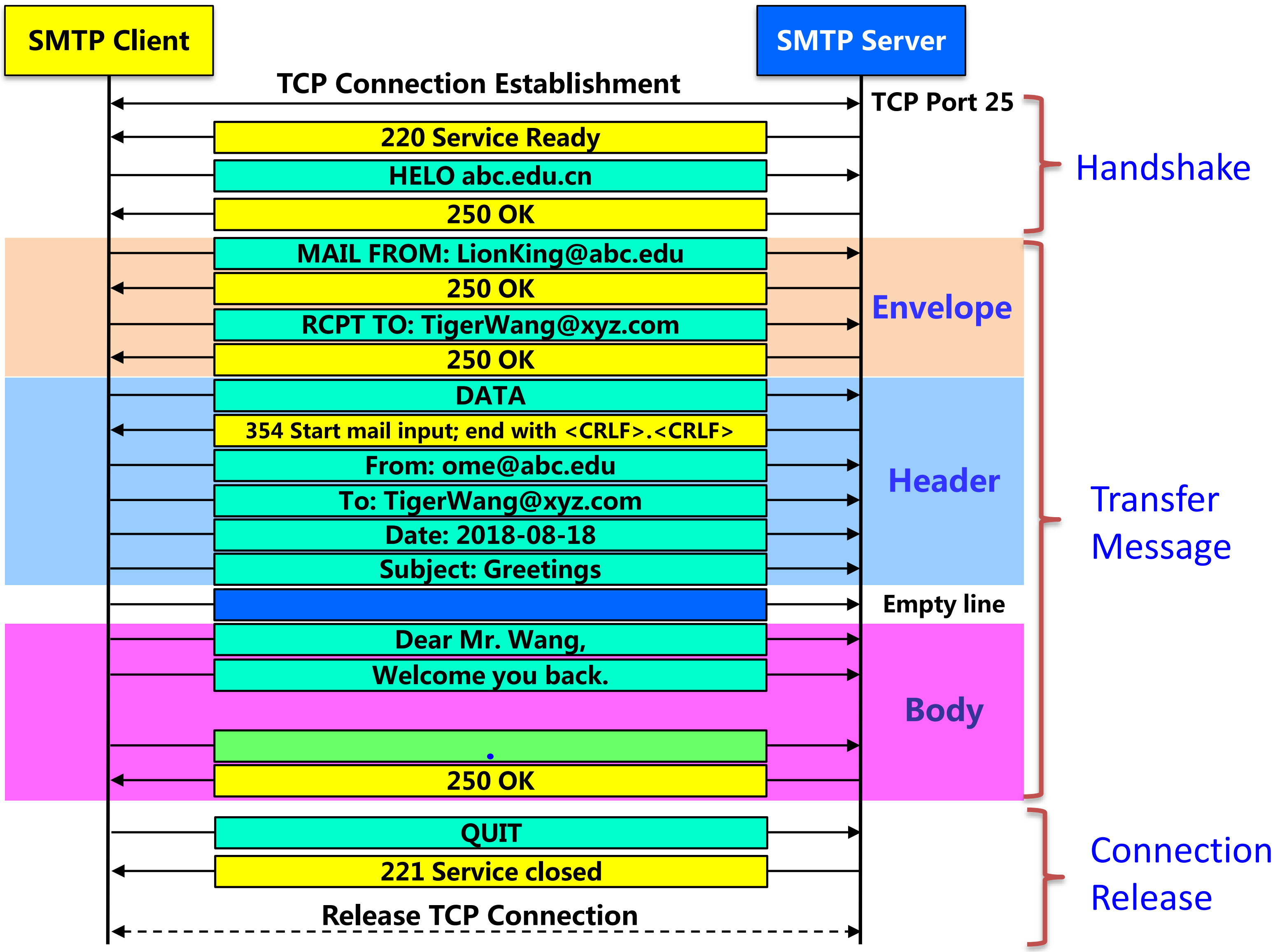
- Handshake
  - TCP connection setting up between client and server
- Transfer Message
- Close
  - Release TCP connection

SMTP

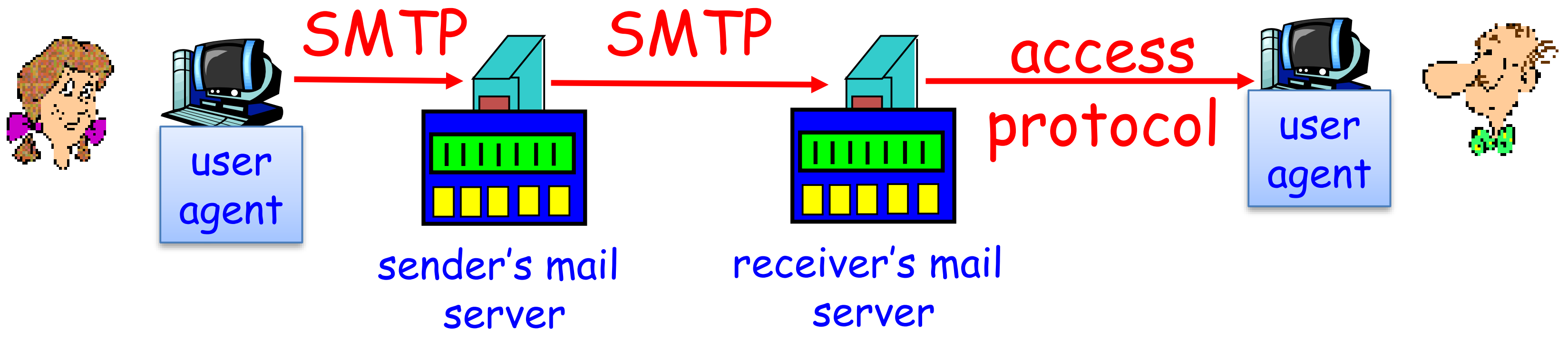
- Client/server model
  - “client”: sending mail server
  - “server”: receiving mail server

- Commands and responses
  - Commands: ASCII text
  - Response: status code and phrase

Sample SMTP Interaction

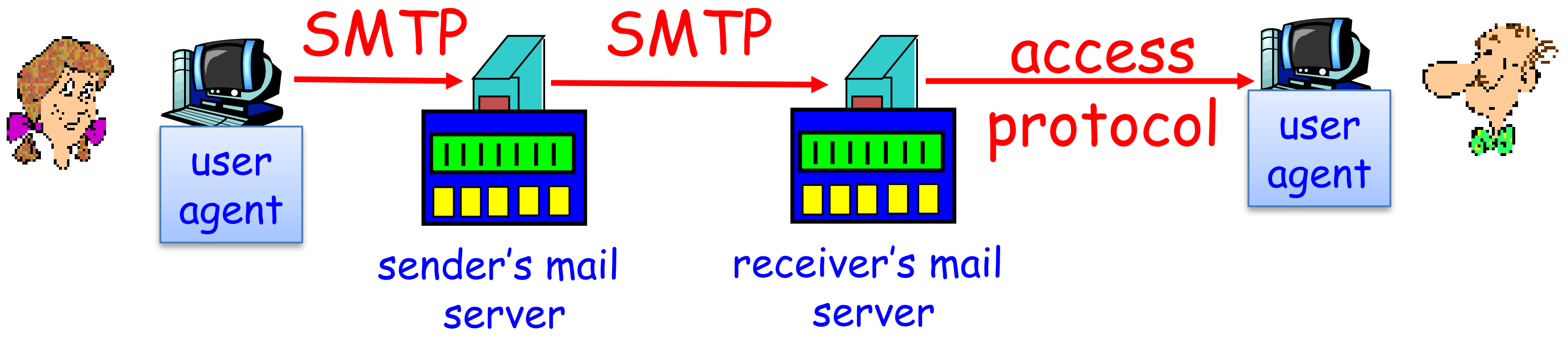


Message Access Protocol - POP3



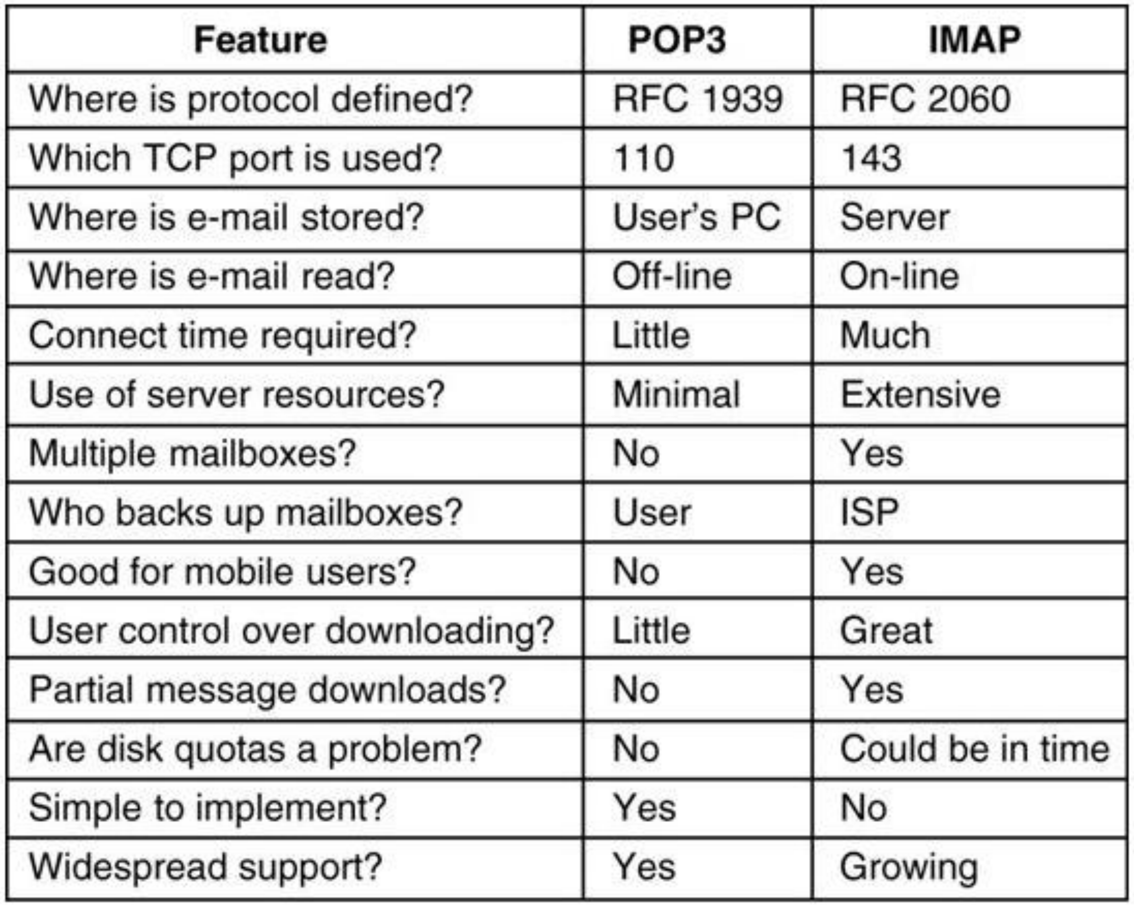
- POP3 (Post Office Protocol, version 3), RFC1939
  - Simple protocol supports few features and less secure
  - Authorization (agent <-->server) and download emails to UA computer
  - Client (UA) / Server (mail server)

Message Access Protocol - IMAP



- IMAP (Internet Message Access Protocol), RFC3501
  - More features (more complex)
  - Manipulation of stored messages on server
  - Client (UA) / Server (mail server)

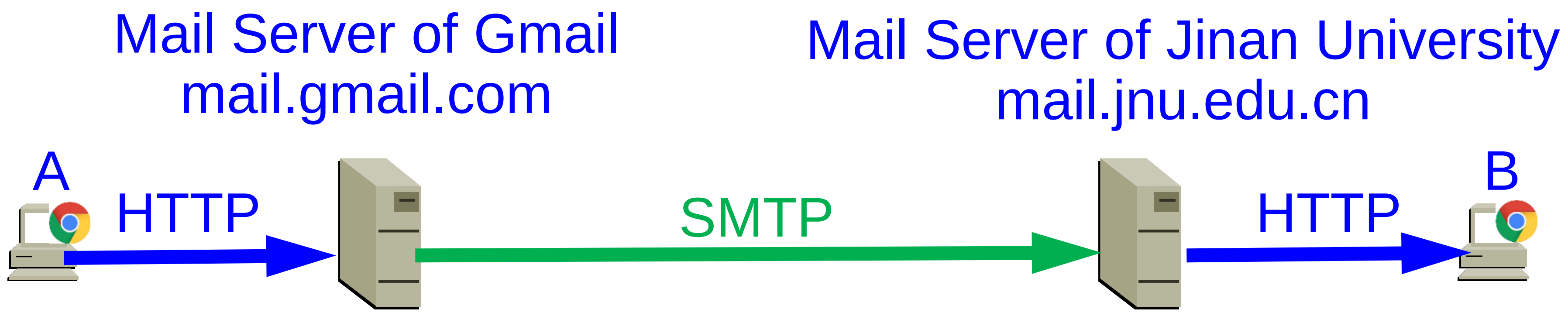
POP3 vs. IMAP



Feature	POP3	IMAP
Where is protocol defined?	RFC 1939	RFC 2060
Which TCP port is used?	110	143
Where is e-mail stored?	User's PC	Server
Where is e-mail read?	Off-line	On-line
Connect time required?	Little	Much
Use of server resources?	Minimal	Extensive
Multiple mailboxes?	No	Yes
Who backs up mailboxes?	User	ISP
Good for mobile users?	No	Yes
User control over downloading?	Little	Great
Partial message downloads?	No	Yes
Are disk quotas a problem?	No	Could be in time
Simple to implement?	Yes	No
Widespread support?	Yes	Growing

Web Email

- HTTP between user computer can mail server
- SMTP between mail servers



Review

- URL Uniform Resource Locator)
  - Globally unique identifier
- HTTP (Hypertext Transfer Protocol)
  - Communication between server and client
- HTML (HyperText Markup Language)
  - Produce and display webpage
- Electronic Email

Thank you!

Q & A