The Web and HTTP

Required reading:
Kurose 2.2

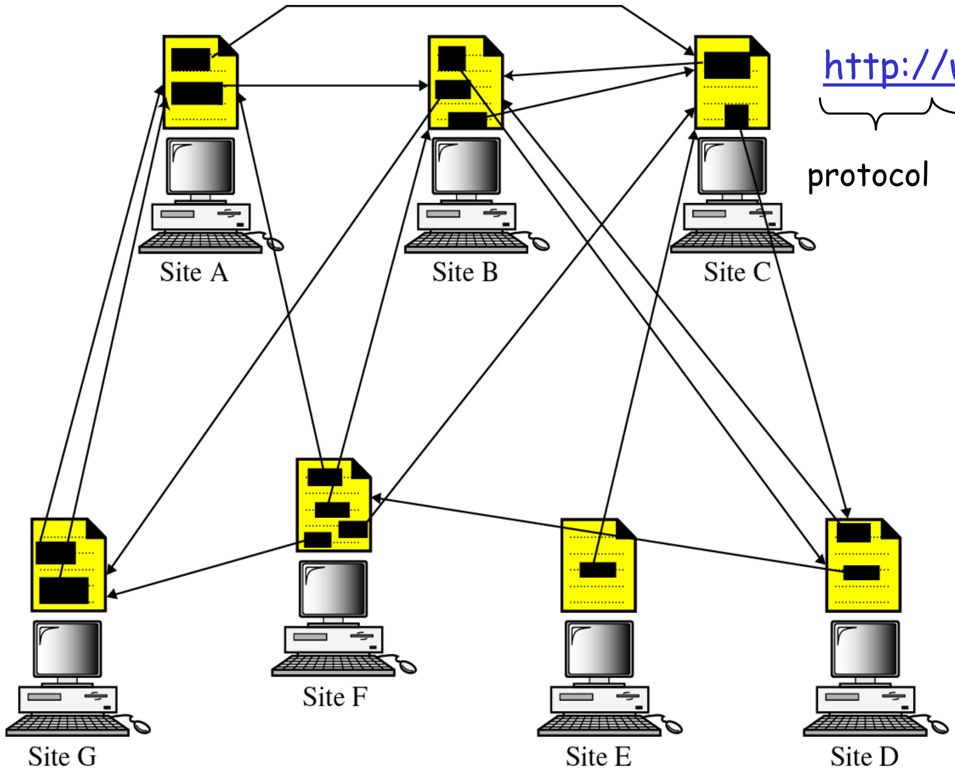
EECS 3214, Winter 2020
Instructor: N. Vlajic

World Wide Web (WWW)

(2) (1) (3)

WWW – **distributed client-server repository of hypertext / hypermedia objects**

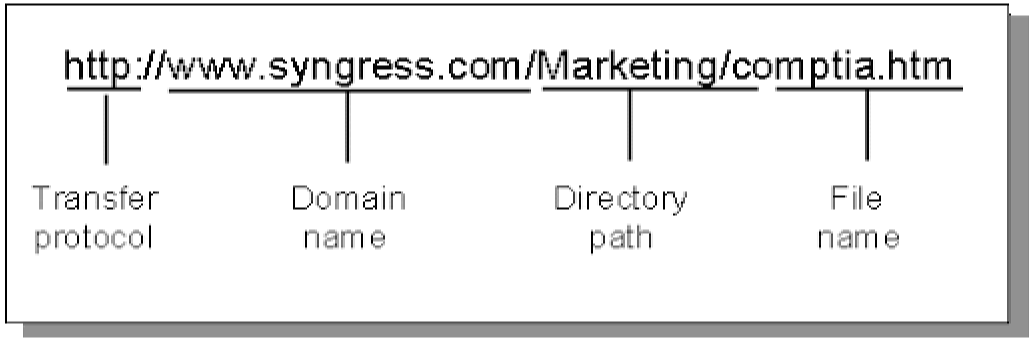
- **hypertext / hypermedia system: information is organized as a set of documents (objects) linked together using the concept of pointers**
  - **each object is addressable by a Uniform Resource Locator (URL)**

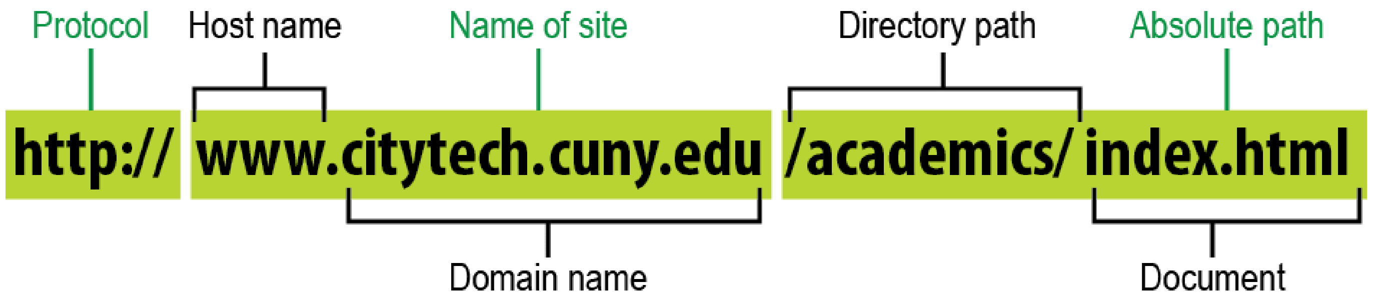


$\underbrace{\text{http://}}_{\text{protocol}}\underbrace{\text{www.someschool.edu}}_{\text{host name}}\underbrace{\text{/someDept/pic.gif}}_{\text{path}}$

World Wide Web (WWW) (cont.)

Example **[three components of URLs …]**

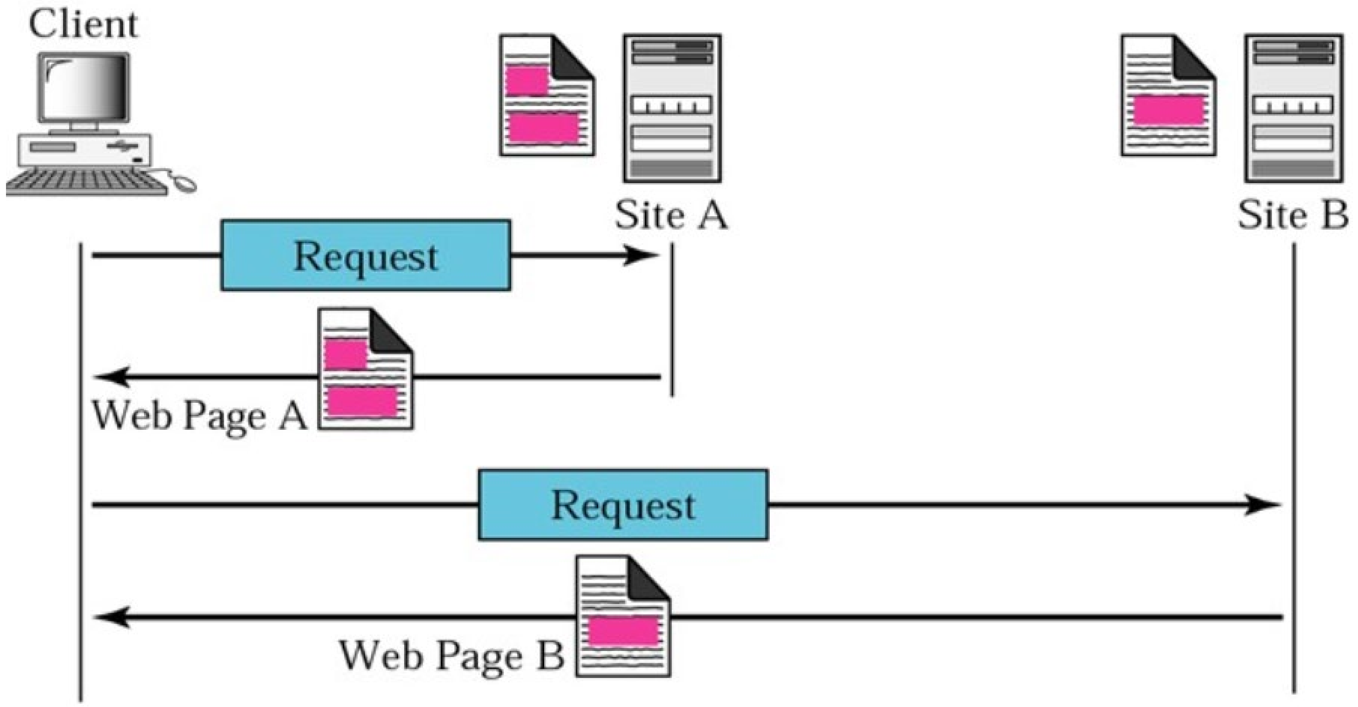




World Wide Web (WWW) (cont.)

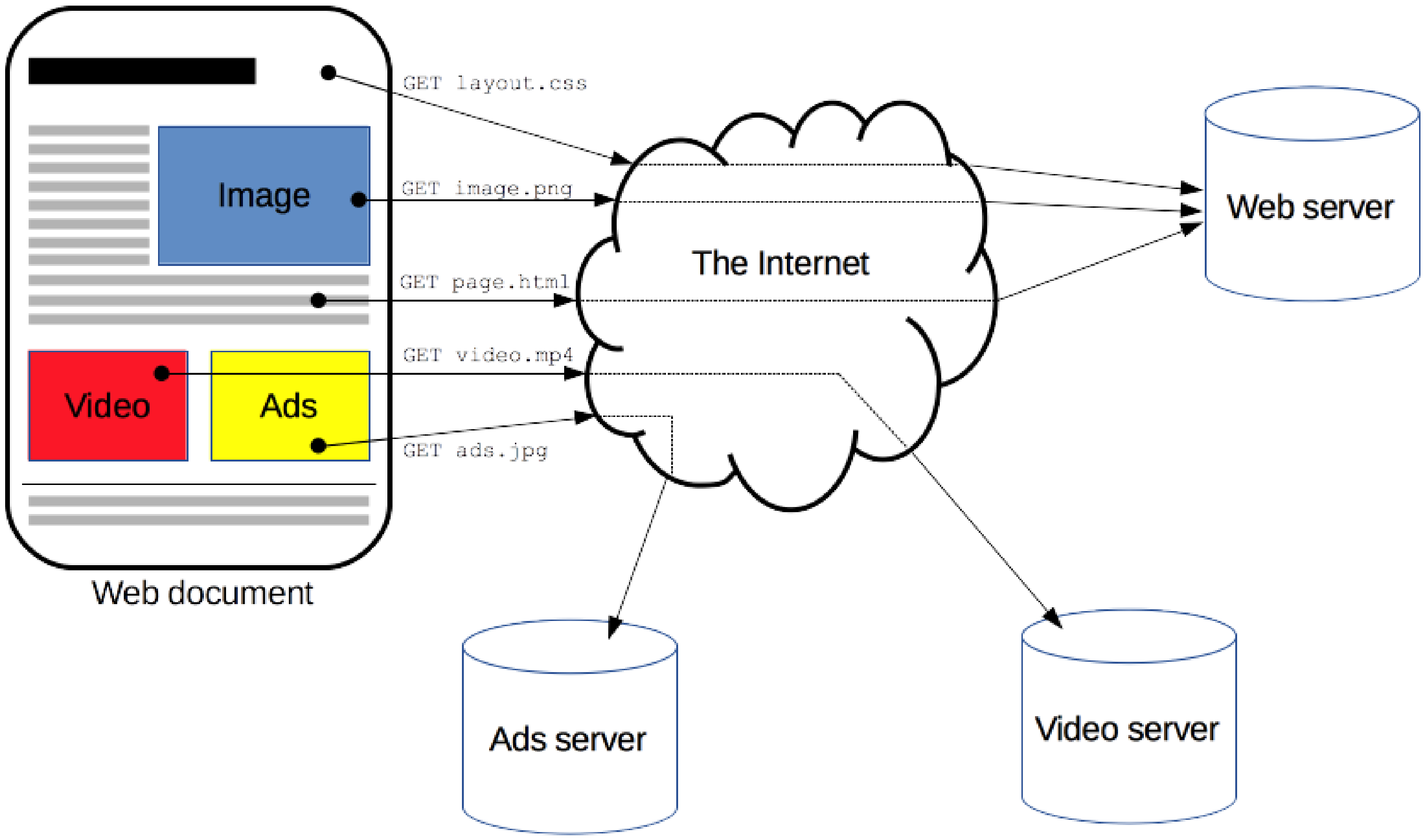
WWW – distributed client-server repository of hypertext / hypermedia objects

- **distributed client-server system: a client deploying a BROWSER can access service (documents / objects) hosted by multiple servers distributed over many locations**



World Wide Web (WWW) (cont.)

Example **[relationship between WWW documents and WWW objects]**



https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

World Wide Web (WWW) (cont.)

Web Browser – client program which enables a user to retrieve, display and interact with WWW objects: HTML files, JPEG images, JavaScript objects, video and audio files, etc.

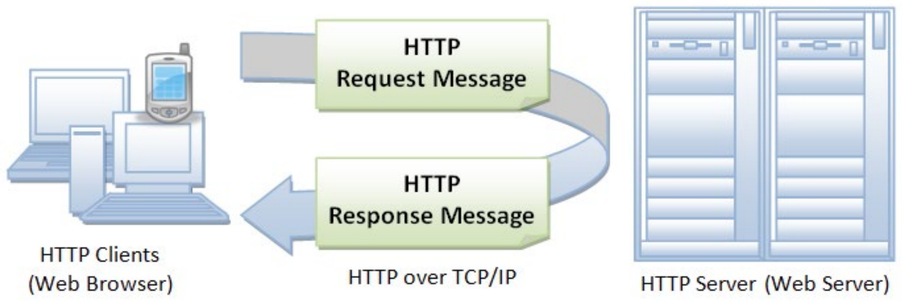
- supports multiple languages / protocols, including:

  (1) **HTML** (HyperText Markup Language) – text language used to construct hypertext documents, i.e. Web pages

  - 'marks' (tags) allow special formatting and linking of objects
  - e.g. <font color=green> this is a <b> great </b> story </font>

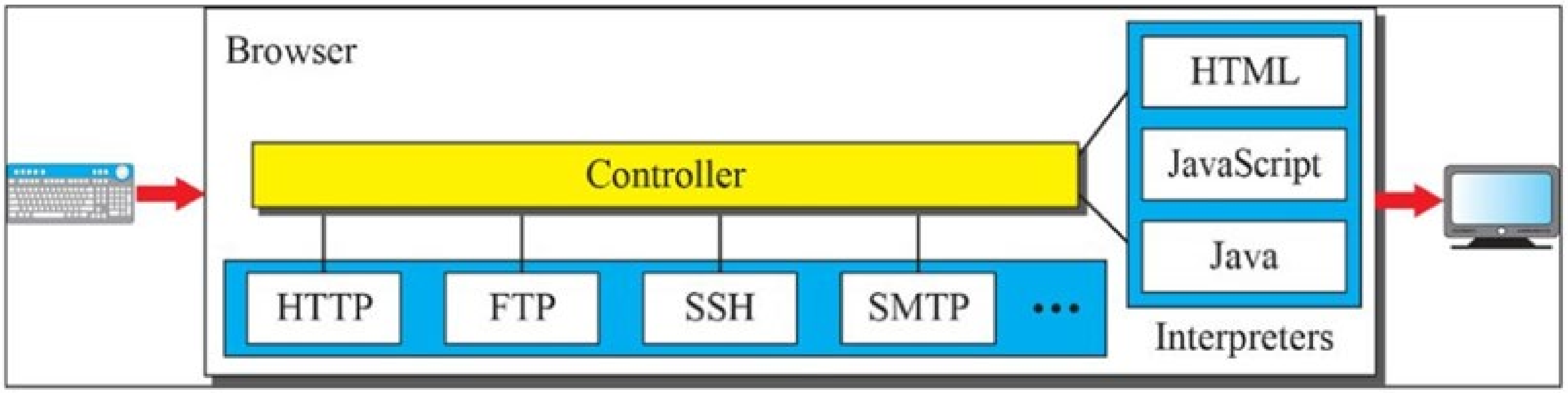
  this is a **great** story

  (2) **HTTP** (Hypertext Transfer Protocol) – used to transfer hypertext objects on WWW



World Wide Web (WWW) (cont.)

Example **[key elements of a browser: controller, client protocols, interpreters]**



- **Controller** receives input from the keyboard or the mouse.
- Controller uses the **client protocols** to access WWW documents / objects.
- Controller uses one of the **interpreters** to display the documents /objects on the screen.

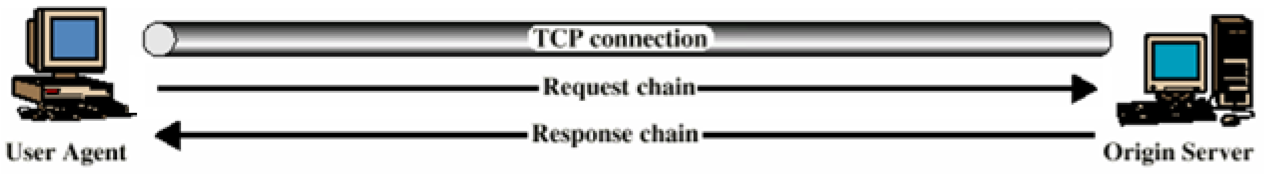
HTTP Overview

HTTP Protocol – client-server application layer protocol – the hart of WWW – defines how Web clients request pages from Web servers and how servers transfer Web pages to clients

- HTTP is stateless protocol – the server listening and handling a request has no memory of any previous request from the same client
- HTTP uses TCP as its underlying transport protocol
  - (1) HTTP client initiates a TCP connection with HTTP server at port 80
  - (2) server accepts TCP connection from client
  - (3) client / server exchange request / response messages through the TCP connection
  - (4) once object(s) are retrieved TCP connection closes
- HTTP-TCP connection can be non-persistent or persistent

Port: ??

Port: 80



HTTP Overview (cont.)

Stateful Protocol – **'remembers' information from one transmission to the next and creates a 'context' based on the client's previous requests**

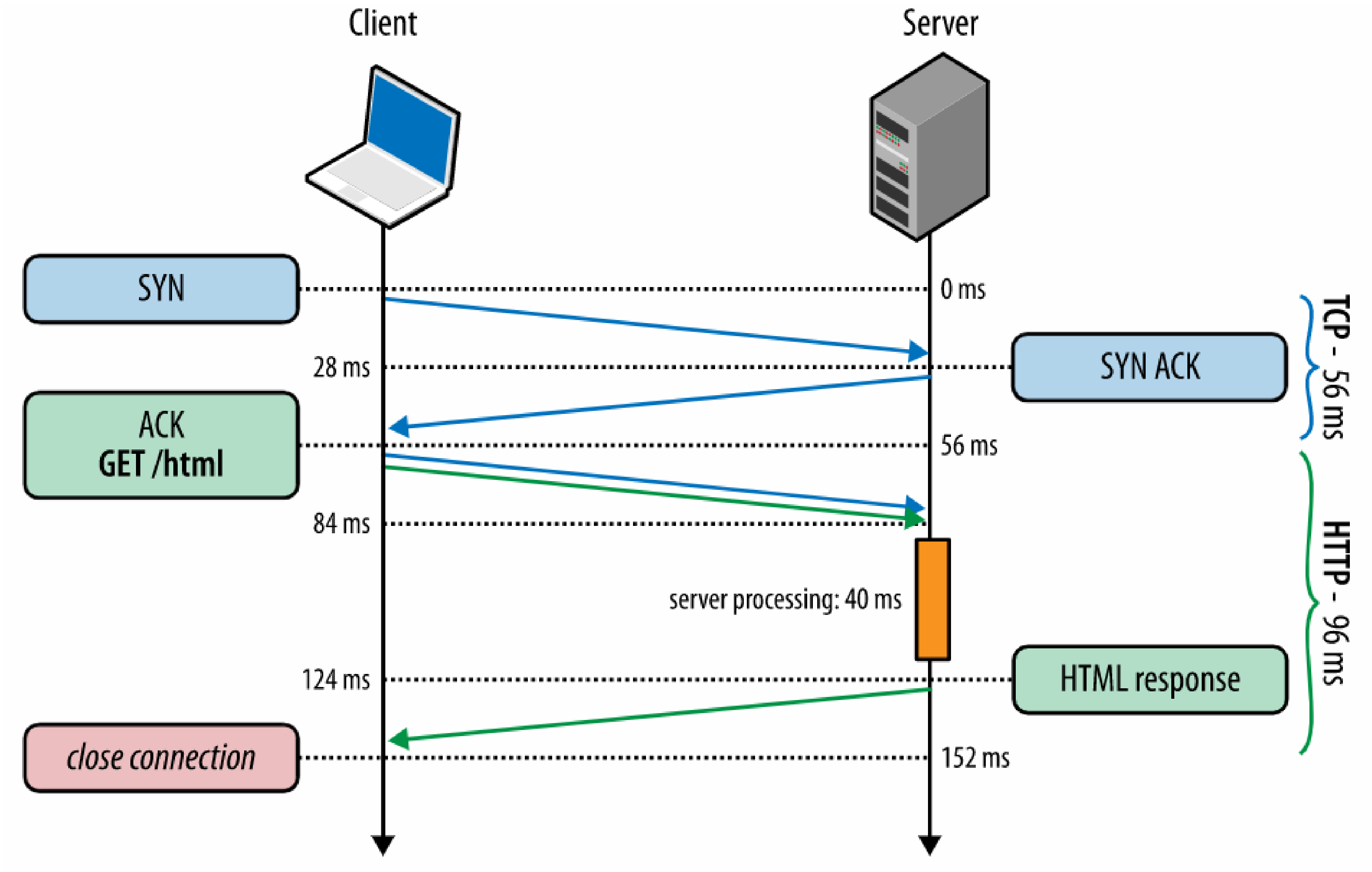
- **e.g. remembers when the last transaction took place, type of last transaction**
- **advantage: less information need to be transported over the network – requests do not have to be self-contained**
- **disadvantage: past history (state) must be maintained; if server/client crashes, their views of "state" may become inconsistent, so they must be reconciled**
- **examples: TCP, FTP, etc.**

Stateless Protocol – **no information about a transaction is maintained (on the server side) after the transaction has been processed**

- **advantage: simplicity + better scalability – no need to keep track of multiple client sessions**
- **disadvantage: processing and bandwidth overhead – each request must be entirely self-contained**
- **examples: UDP, HTTP, etc.**

HTTP Overview (cont.)

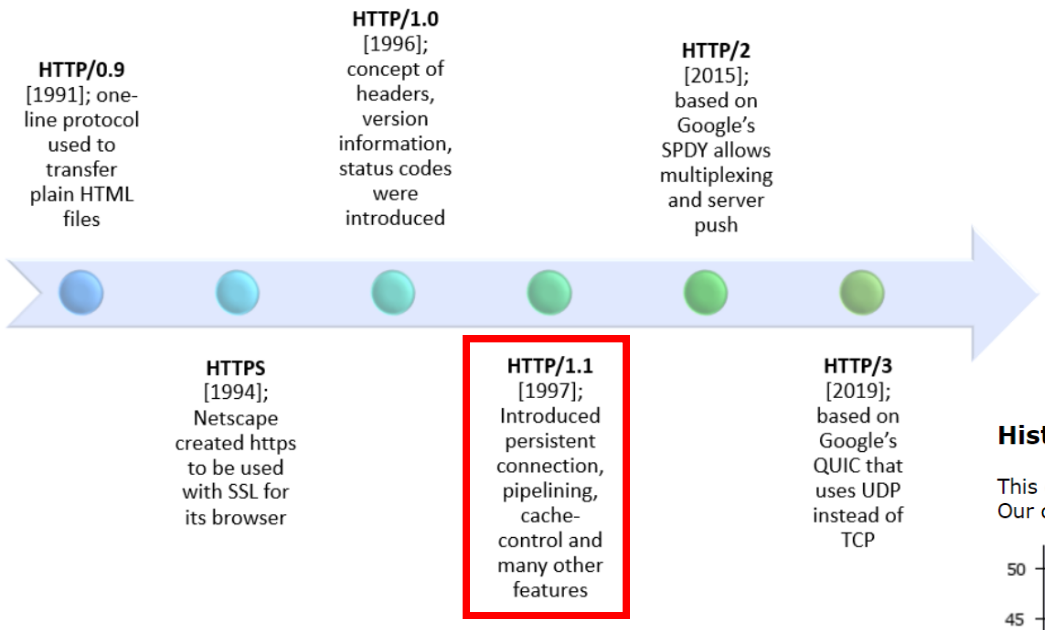
Example **[HTTP request following TCP connection establishment …]**



https://hpbn.co/http1x/

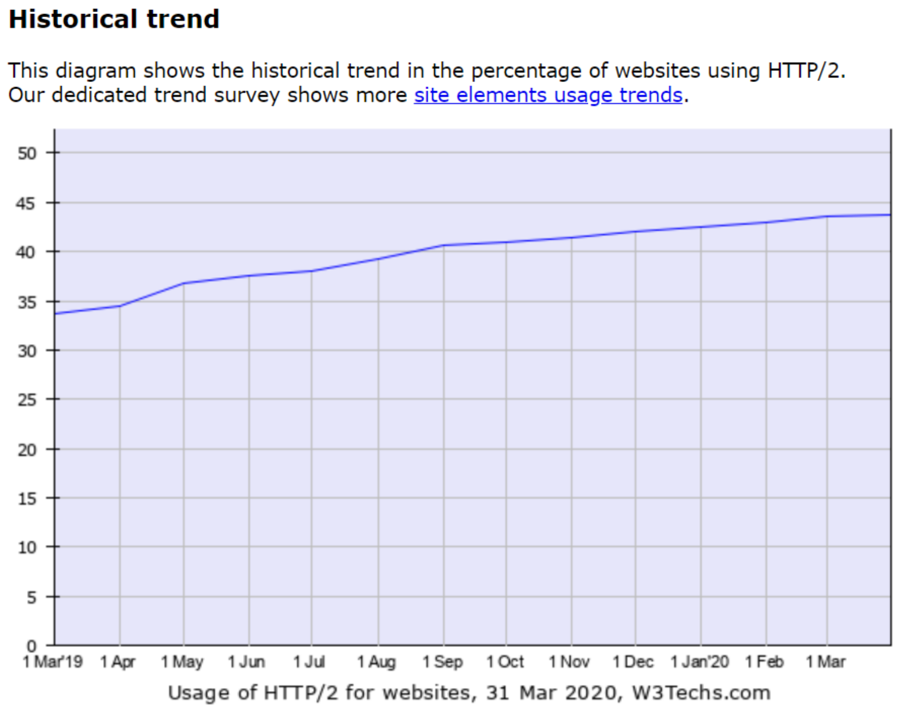
HTTP Overview (cont.)

Evolution of HTTP Protocol


HTTP/0.9 [1991]; one-line protocol used to transfer plain HTML files
HTTP/1.0 [1996]; concept of headers, version information, status codes were introduced
HTTP/2 [2015]; based on Google's SPDY allows multiplexing and server push
HTTPS [1994]; Netscape created https to be used with SSL for its browser
HTTP/1.1 [1997]; Introduced persistent connection, pipelining, cache-control and many other features
HTTP/3 [2019]; based on Google's QUIC that uses UDP instead of TCP

https://cheapsslsecurity.com/p/http2-vs-http1/

Prevalence of HTTP/2



Historical trend

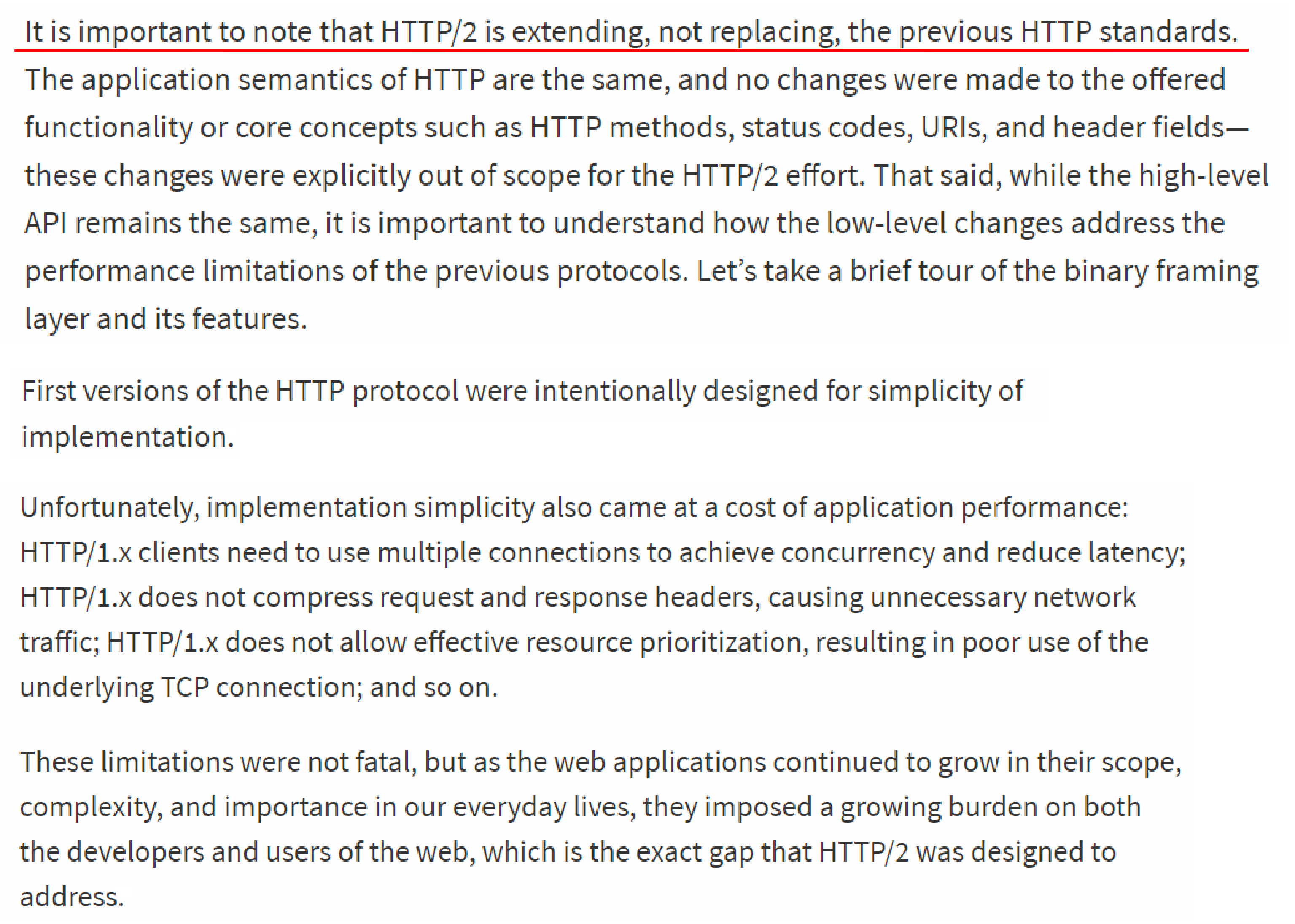
This diagram shows the historical trend in the percentage of websites using HTTP/2.
Our dedicated trend survey shows more site elements usage trends.

Usage of HTTP/2 for websites, 31 Mar 2020, W3Techs.com

https://w3techs.com/technologies/details/ce-http2

HTTP Overview (cont.)

Example **[further details about HTTP/2]**



It is important to note that HTTP/2 is extending, not replacing, the previous HTTP standards. The application semantics of HTTP are the same, and no changes were made to the offered functionality or core concepts such as HTTP methods, status codes, URIs, and header fields—these changes were explicitly out of scope for the HTTP/2 effort. That said, while the high-level API remains the same, it is important to understand how the low-level changes address the performance limitations of the previous protocols. Let's take a brief tour of the binary framing layer and its features.

First versions of the HTTP protocol were intentionally designed for simplicity of implementation.

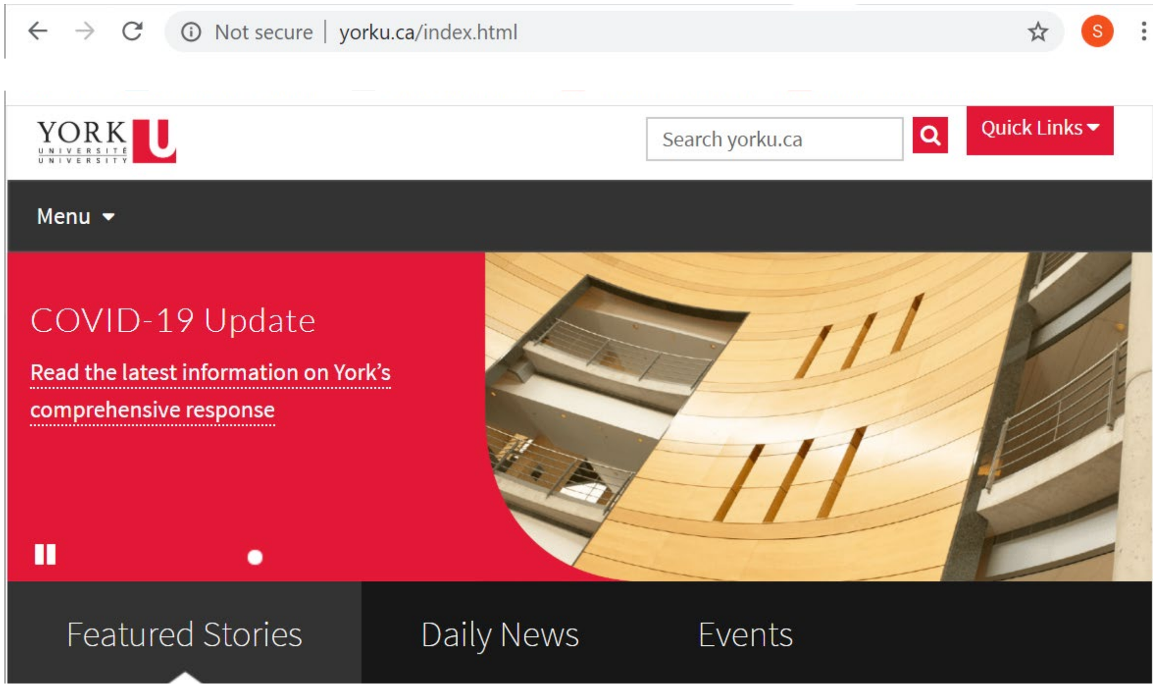
Unfortunately, implementation simplicity also came at a cost of application performance: HTTP/1.x clients need to use multiple connections to achieve concurrency and reduce latency; HTTP/1.x does not compress request and response headers, causing unnecessary network traffic; HTTP/1.x does not allow effective resource prioritization, resulting in poor use of the underlying TCP connection; and so on.

These limitations were not fatal, but as the web applications continued to grow in their scope, complexity, and importance in our everyday lives, they imposed a growing burden on both the developers and users of the web, which is the exact gap that HTTP/2 was designed to address.

https://hpbn.co/http2/#brief-history-of-spdy-and-http2

HTTP Overview (cont.)

Example **[Is my browser using HTTP/1.1 or HTTP/2 ??]**


Not secure | yorku.ca/index.html
YORK UNIVERSITÉ UNIVERSITY
Search yorku.ca
Quick Links
Menu
COVID-19 Update
Read the latest information on York's comprehensive response
Featured Stories
Daily News
Events

How could you learn whether this Web page, and its embedded objects, have been retrieved (by your browser) using HTTP/1.1 or HTTP/2 ???

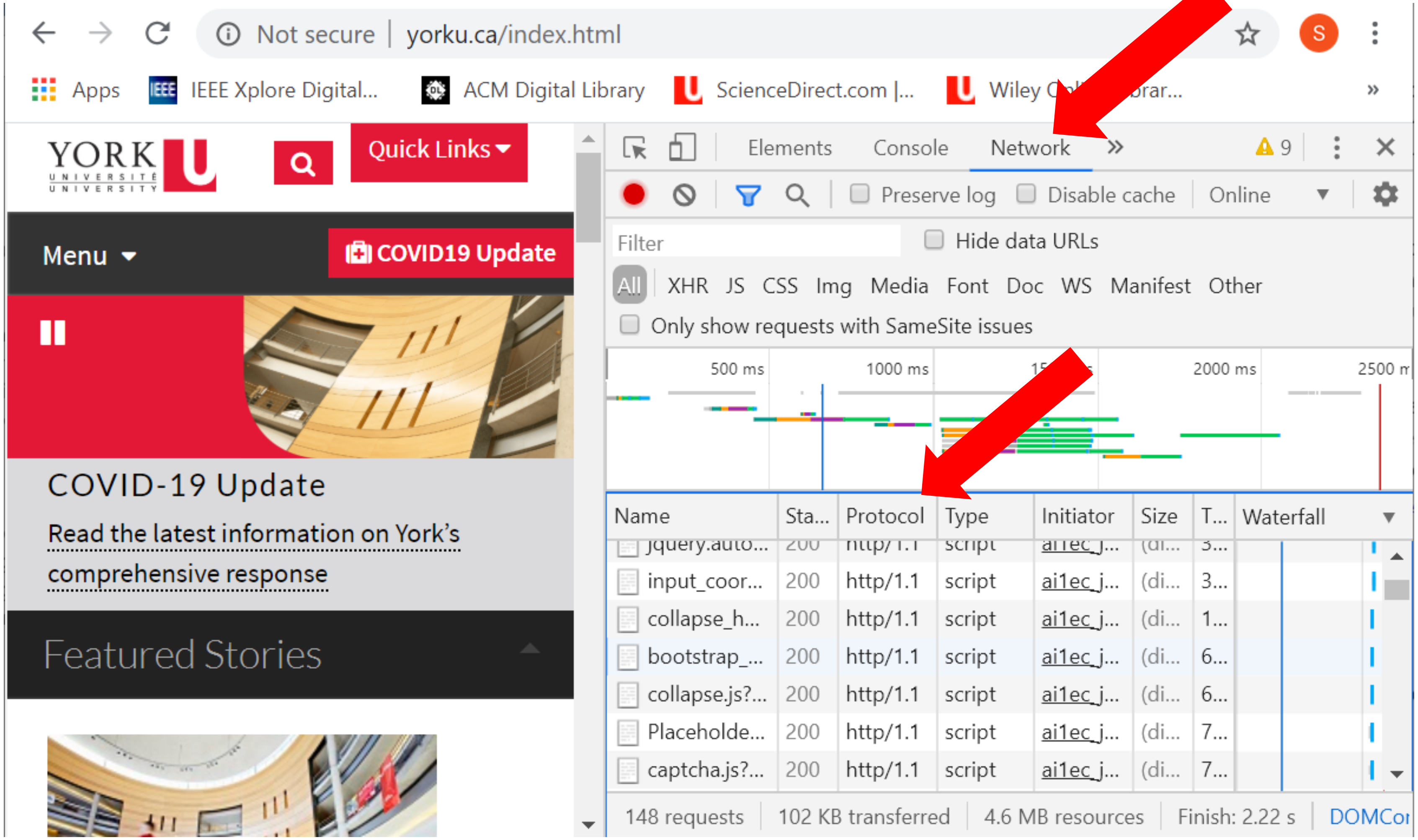
Use Wireshark ?!

Only if the site is not encrypted (not using HTTPS) !!!

HTTP Overview (cont.)

Example **[Is my browser using HTTP/1.1 or HTTP/2 ??]**

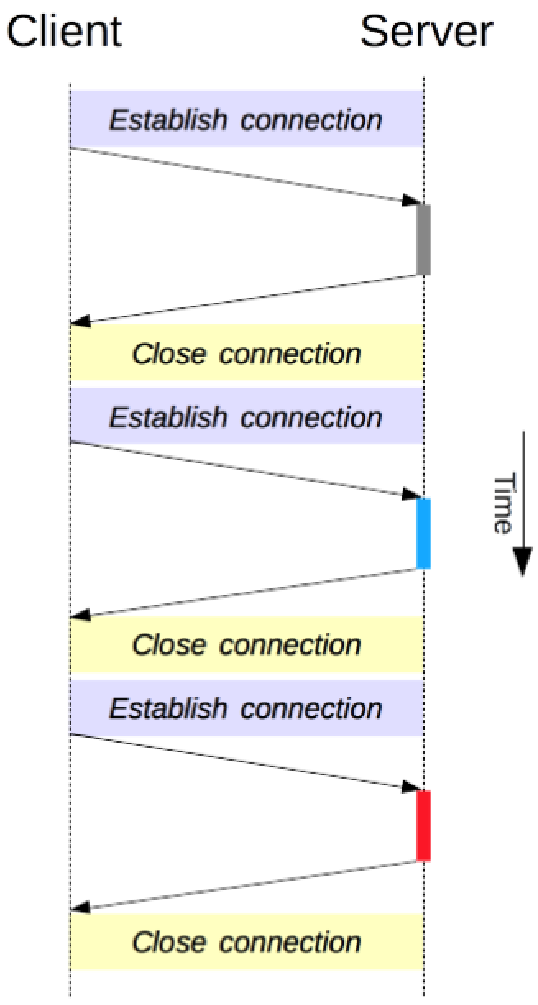
Use: Chrome -> Settings -> More Tools -> Developer Tools -> Network



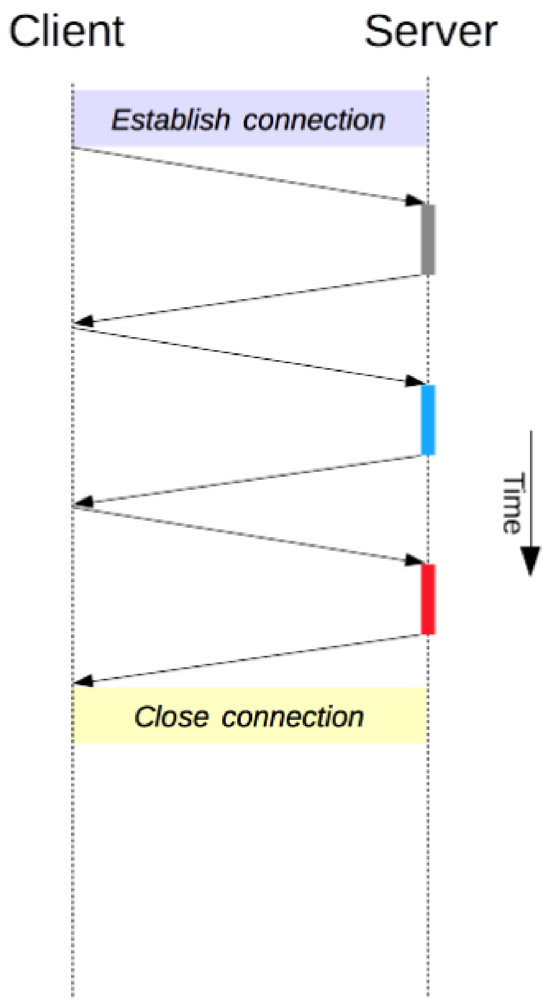
Name	Sta...	Protocol	Type	Initiator	Size	T...	Waterfall
jquery.auto...	200	http/1.1	script	ai1ec_j...	(di...	3...	
input_coor...	200	http/1.1	script	ai1ec_j...	(di...	3...	
collapse_h...	200	http/1.1	script	ai1ec_j...	(di...	1...	
bootstrap_...	200	http/1.1	script	ai1ec_j...	(di...	6...	
collapse.js?...	200	http/1.1	script	ai1ec_j...	(di...	6...	
Placeholde...	200	http/1.1	script	ai1ec_j...	(di...	7...	
captcha.js?...	200	http/1.1	script	ai1ec_j...	(di...	7...	

148 requests | 102 KB transferred | 4.6 MB resources | Finish: 2.22 s | DOMCor

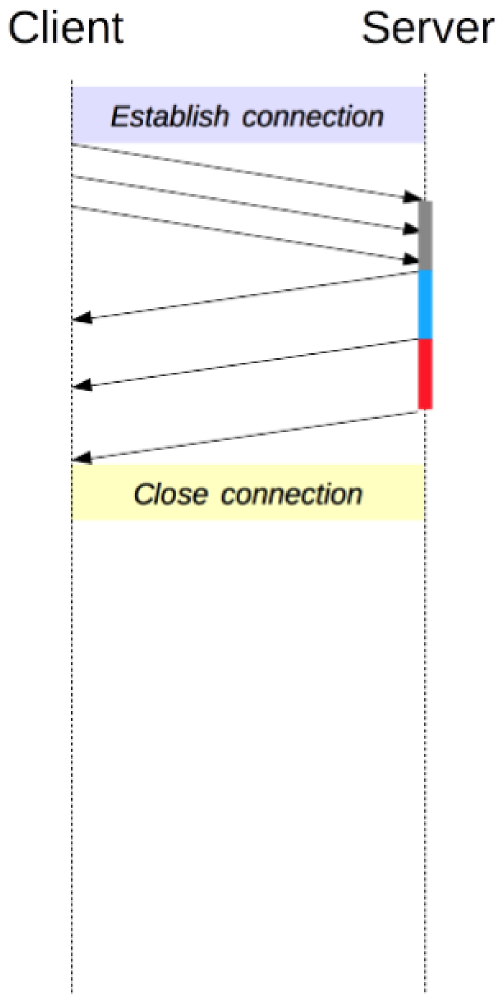
HTTP: Non-persistent vs. Persistent



non-persistent



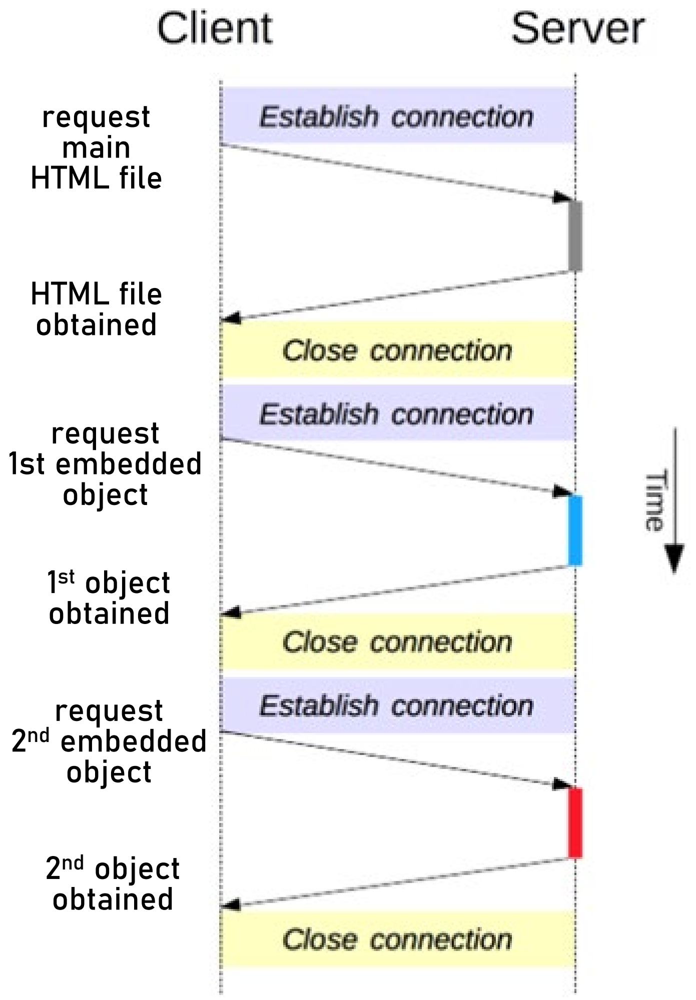
persistent



persistent
with pipelining

HTTP: Non-persistent vs. Persistent (cont.)

Non-persistent HTTP Connection – **each TCP connection transports only one HTTP request message and one HTTP response message**



- **retrieval strategy**:

  (1) client opens a TCP connection and sends request for an object

  (2) server sends response and closes connection

  (3) client examines retrieved object; if it contains references to other objects, steps (1) to (3) are repeated for each of these objects

- **to retrieve a Web-page with N embedded pictures, (N+1) TCP connections must be opened and closed**

- **disadvantage**:
  - **high overhead is imposed on server – multiple TCP buffers need to be used**
  - **slow-start procedure at the beginning of each TCP connection $\Rightarrow$ longer overall retrieval time**

retrieval time per object = 2*RTT + transmission time

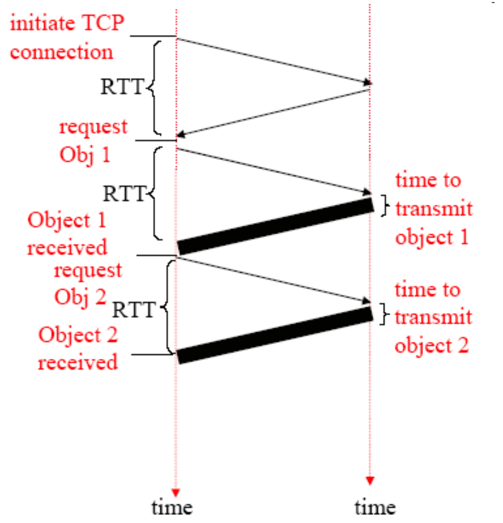
RTT = propagation + queueing + processing delay !!!

HTTP: Non-persistent vs. Persistent (cont.)

Persistent HTTP Connection – **server leaves the TCP connection open after sending a response – subsequent requests and responses can be sent over the same connection (default in HTTP 1.1)**

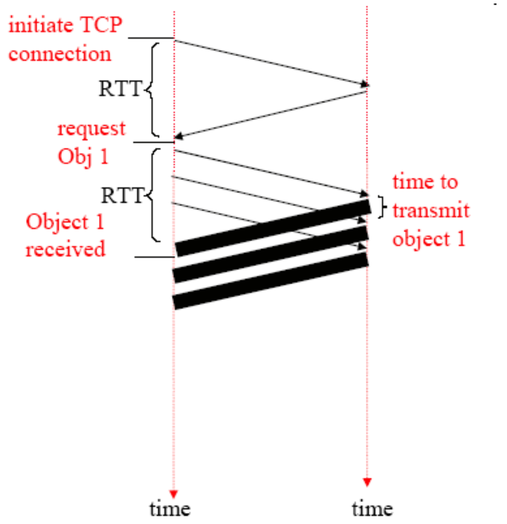
- **2 versions of persistent HTTP:**

(a) without pipelining – HTTP client issues new request only when the previous response / object has been received



retrieval time per object = RTT + transmission time

(b) with pipelining – HTTP client issues a request as soon as it encounters a reference



one RTT for all objects

HTTP: Non-persistent vs. Persistent (cont.)

Example **[non-persistent vs. persistent HTTP]**

Assume a Web page consists of 1 base HTML page and M images (each of size O bits). Data-rate on the link is R[bps]. What is the overall retrieval time in case of:

(a) non-persistent HTTP,

(b) persistent HTTP with pipelining.

(a) non-persistent HTTP:

establish (M+1) TCP connections in series	= (M+1) RTT
request/receive (M+1) objects of size O over a link of datarate R	= (M+1) [RTT + O/R]

$$\boxed{\text{overall retrieval time} = 2 \cdot (M+1) \cdot RTT + (M+1) \cdot \frac{O}{R}}$$

(b) persistent HTTP with pipelining:

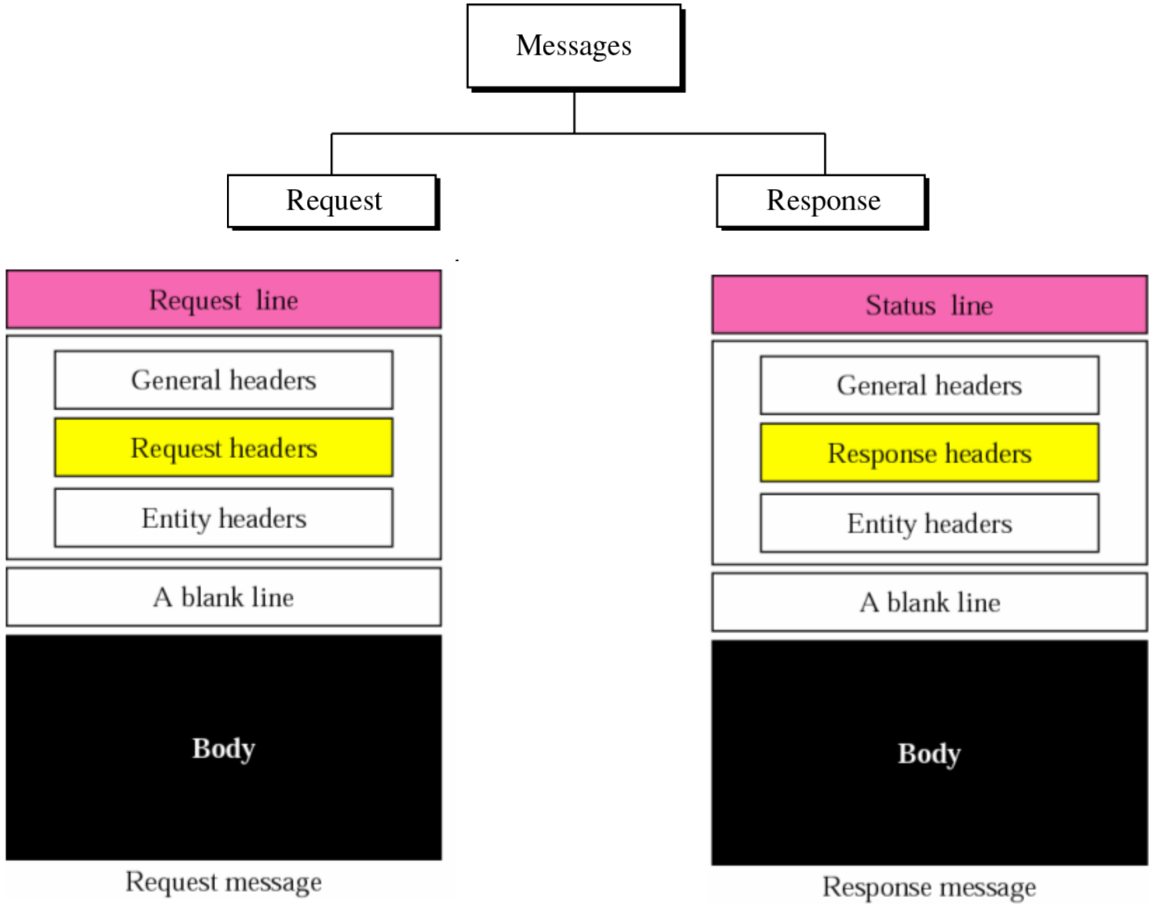
establish TCP connection	= 1 RTT
request/receive base HTML file	= RTT + O/R
request/receive all images	= RTT + M*O/R

$$\boxed{\text{overall retrieval time} = 3 \cdot RTT + (M+1) \cdot \frac{O}{R}}$$

HTTP Messages

HTTP Messages – HTTP consists of two types of messages: requests from clients to servers, and responses from servers to clients

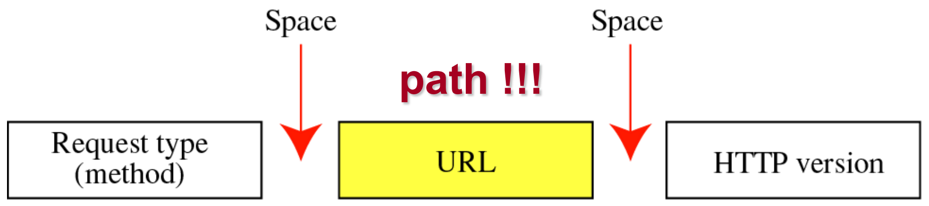
- both message types are written in ordinary ASCII (human readable) format



HTTP Messages (cont.)

Request Line in Request Message

– has three fields: method, URL, and HTTP version field

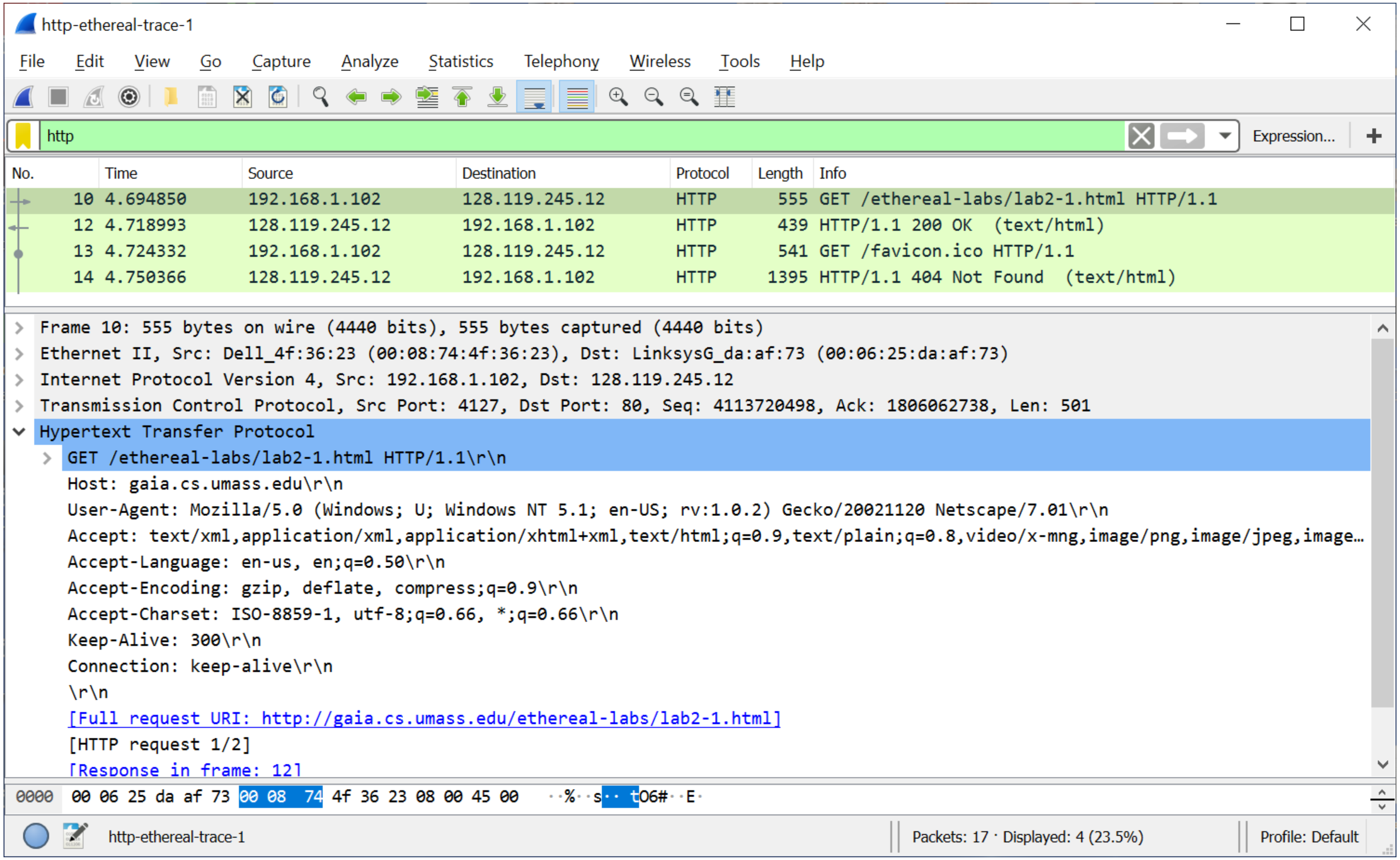


- **method** field can take on several values, including: GET, HEAD, and POST – majority of HTTP requests use the GET method
  - **GET** is used when client wants to retrieve a document from server
  - **HEAD** is used when client wants some information about document but not document itself
  - **POST** is when client provides some information for server – e.g. input to server

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

HTTP Messages (cont.)

Example [HTTP request message in Wireshark]


http-ethereal-trace-1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...

```
No.  Time      Source          Destination     Protocol  Length  Info
10   4.694850  192.168.1.102   128.119.245.12  HTTP      555     GET /ethereal-labs/lab2-1.html HTTP/1.1
12   4.718993  128.119.245.12  192.168.1.102   HTTP      439     HTTP/1.1 200 OK  (text/html)
13   4.724332  192.168.1.102   128.119.245.12  HTTP      541     GET /favicon.ico HTTP/1.1
14   4.750366  128.119.245.12  192.168.1.102   HTTP      1395    HTTP/1.1 404 Not Found  (text/html)
```

```
> Frame 10: 555 bytes on wire (4440 bits), 555 bytes captured (4440 bits)
> Ethernet II, Src: Dell_4f:36:23 (00:08:74:4f:36:23), Dst: LinksysG_da:af:73 (00:06:25:da:af:73)
> Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 4127, Dst Port: 80, Seq: 4113720498, Ack: 1806062738, Len: 501
v Hypertext Transfer Protocol
  > GET /ethereal-labs/lab2-1.html HTTP/1.1\r\n
    Host: gaia.cs.umass.edu\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.0.2) Gecko/20021120 Netscape/7.01\r\n
    Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image…
    Accept-Language: en-us, en;q=0.50\r\n
    Accept-Encoding: gzip, deflate, compress;q=0.9\r\n
    Accept-Charset: ISO-8859-1, utf-8;q=0.66, *;q=0.66\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/ethereal-labs/lab2-1.html]
    [HTTP request 1/2]
    [Response in frame: 12]
```

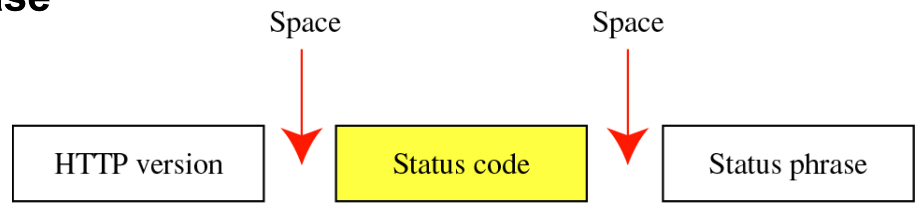
```
0000  00 06 25 da af 73 00 08  74 4f 36 23 08 00 45 00   ··%··s·· tO6#··E·
```

http-ethereal-trace-1 | Packets: 17 · Displayed: 4 (23.5%) | Profile: Default

HTTP Messages (cont.)

Status Line in Response Message

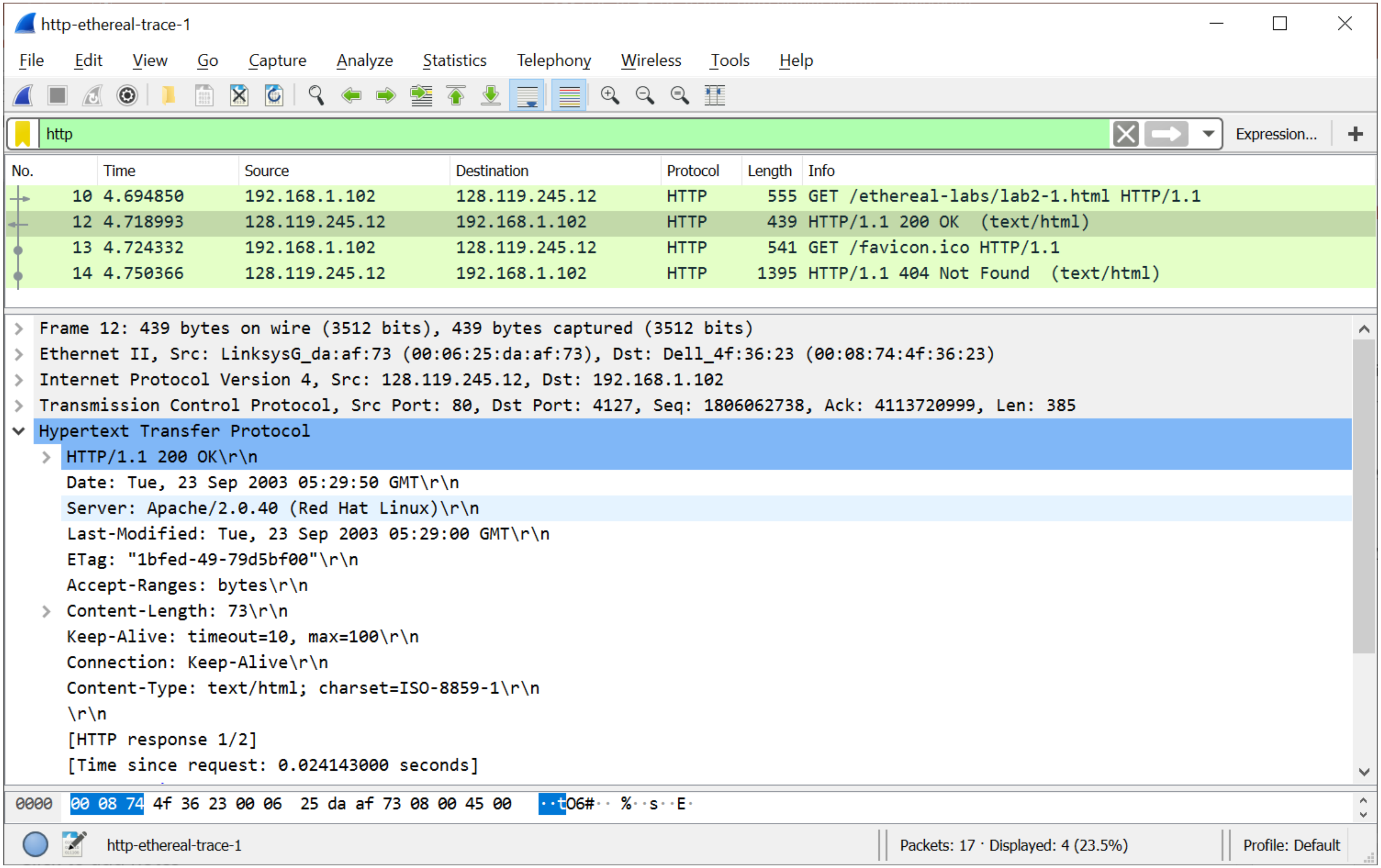
– has three fields: HTTP version, status code, status phrase



- *status code* is 3-digit integer that indicates the response to a received request; *status phrase* gives short textual explanation of the status code
  - **200 OK** – request succeeded, information returned
  - **301 Moved Permanently** – requested object has been permanently moved
  - **400 Bad Request** – syntax error in request
  - **404 Not Found** – requested document does not exist on this server
  - **503 Service Unavailable** – service temporarily unavailable

```
HTTP/1.1  200  OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998
...
```

HTTP Messages (cont.)



http-ethereal-trace-1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...

```
No.   Time       Source          Destination     Protocol  Length  Info
10    4.694850   192.168.1.102   128.119.245.12  HTTP      555     GET /ethereal-labs/lab2-1.html HTTP/1.1
12    4.718993   128.119.245.12  192.168.1.102   HTTP      439     HTTP/1.1 200 OK  (text/html)
13    4.724332   192.168.1.102   128.119.245.12  HTTP      541     GET /favicon.ico HTTP/1.1
14    4.750366   128.119.245.12  192.168.1.102   HTTP      1395    HTTP/1.1 404 Not Found  (text/html)
```

```
> Frame 12: 439 bytes on wire (3512 bits), 439 bytes captured (3512 bits)
> Ethernet II, Src: LinksysG_da:af:73 (00:06:25:da:af:73), Dst: Dell_4f:36:23 (00:08:74:4f:36:23)
> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.1.102
> Transmission Control Protocol, Src Port: 80, Dst Port: 4127, Seq: 1806062738, Ack: 4113720999, Len: 385
v Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Date: Tue, 23 Sep 2003 05:29:50 GMT\r\n
    Server: Apache/2.0.40 (Red Hat Linux)\r\n
    Last-Modified: Tue, 23 Sep 2003 05:29:00 GMT\r\n
    ETag: "1bfed-49-79d5bf00"\r\n
    Accept-Ranges: bytes\r\n
  > Content-Length: 73\r\n
    Keep-Alive: timeout=10, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=ISO-8859-1\r\n
    \r\n
    [HTTP response 1/2]
    [Time since request: 0.024143000 seconds]
```

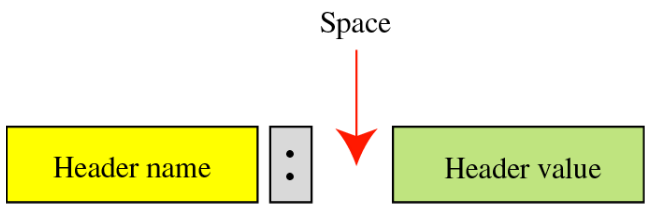
```
0000  00 08 74 4f 36 23 00 06  25 da af 73 08 00 45 00   ··tO6#·· %··s··E·
```

http-ethereal-trace-1   Packets: 17 · Displayed: 4 (23.5%)   Profile: Default

HTTP Messages (cont.)

Headers – exchange additional information between the client and the server

- header line is made of: header name + colon + space + header value



(1) **GENERAL HEADER** – gives general information about the message and can be present in both a request and response

- it provides general info on how the message itself should be processed & handled, but are not related to the content of the message

Header	Description
cache-control	specifies info about caching (e.g. no-cache, max-age)
connection	specifies whether connection should be closed or not
date	shows the date and time at which the message originated
MIME-version	shows the MIME version used
…	

HTTP Messages (cont.)

(2) **REQUEST HEADER** – can be present only in a request message – it specifies the client's configuration and the nature of client's request

Header	Description
accept	shows the media format the client can accept
accept-language	shows the language the client can accept
host	specifies the Internet host of the requested resource
if-modified-since	send the document if newer than specified date
user-agent	identifies the client program
…	

(3) **RESPONSE HEADER** – can be present only in a response message – it specifies the server's configuration and special information about the request

Header	Description
public	shows the list of HTTP methods supported by this server
retry-after	shows how long the service is expected to be unavailable
server	shows the server name and version number
set-cookie	defines a name=value pair associated with this URL
…	

HTTP Messages (cont.)

(4) **ENTITY HEADER** – gives information about the body of the document / message – mostly present in <u>response</u> message

- it conveys information the client needs to properly process and display the resource/entity carried in this message, such as its type and encoding method

Header	Description
content-encoding	specifies the encoding scheme
content-language	specifies the language
content-length	shows the length of the document
content-type	specifies the media type
expires	gives the date and time when contents may change
last-modified	gives the date and time of the last change
location	specifies the location of the created or moved document
…	

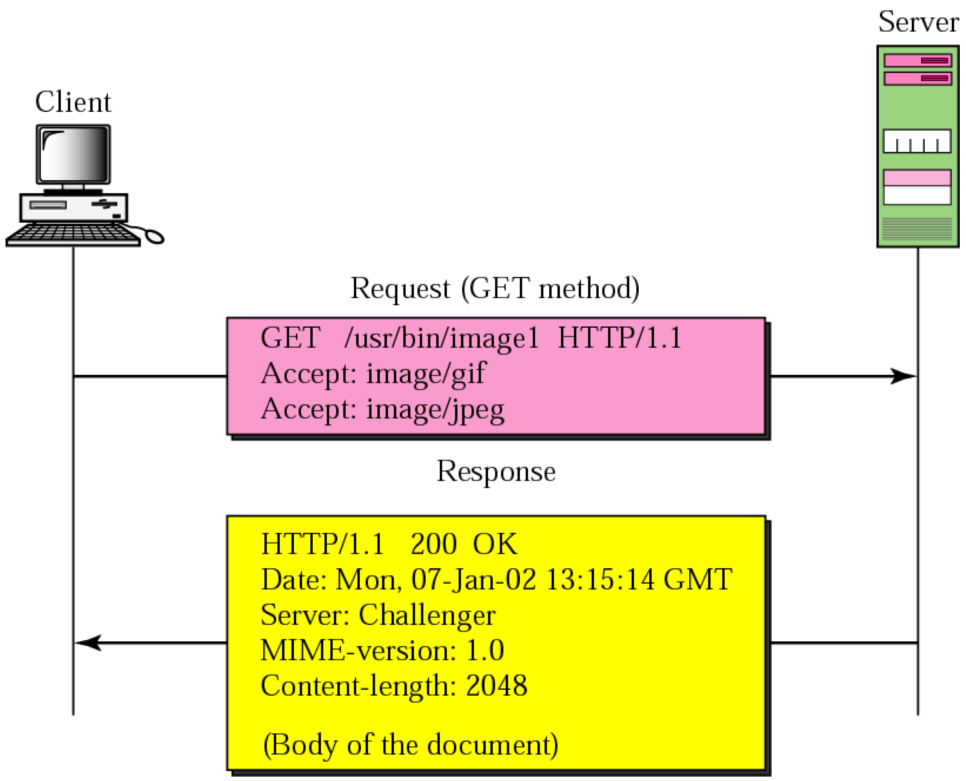
HTTP Messages (cont.)

Example [HTTP retrieval using GET]

In this example the GET method is used to retrieve an image with the path /usr/bin/image1.

The request line shows the method (GET), the URL, and the HTTP version (1.1). The header has two lines that show the client can accept images in the GIF and JPEG format. The request does not have a body.

The response message contains the status line and four lines of header. The header line define the date, server, MIME version, and length of the document. The body of the document follows the header.



```
GET   /usr/bin/image1  HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

```
HTTP/1.1   200  OK
Date: Mon, 07-Jan-02 13:15:14 GMT
Server: Challenger
MIME-version: 1.0
Content-length: 2048

(Body of the document)
```