CSC 450 – COMPUTER NETWORKS

Lecture 3

Design Philosophy and Applications

HTTP – Hyper Text Transfer Protocol

Recap

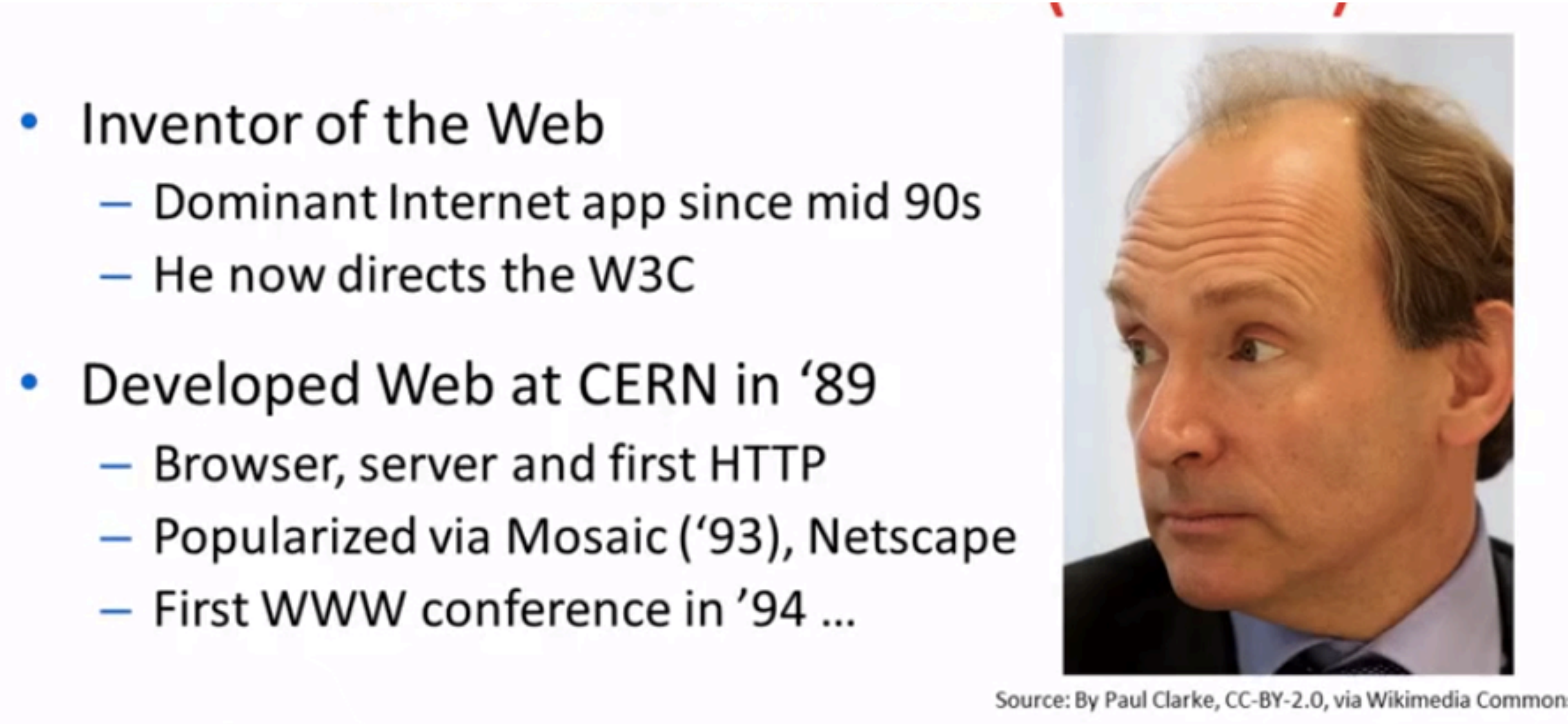
- Layering
- Protocol Stacks
- OSI 7 layer protocol
  - We also know that not all 7 layers are applied
- Important Protocols (overview)
  - IP
  - TCP
  - UDP
- Brief overview of Internet
  - History
  - Services
  - Reliability measures

Todays Class

- A brief overview of Applications
  - Application Programs
  - Application Protocols
- A detailed description of the Application Protocol HTTP

History of the Web as an Application

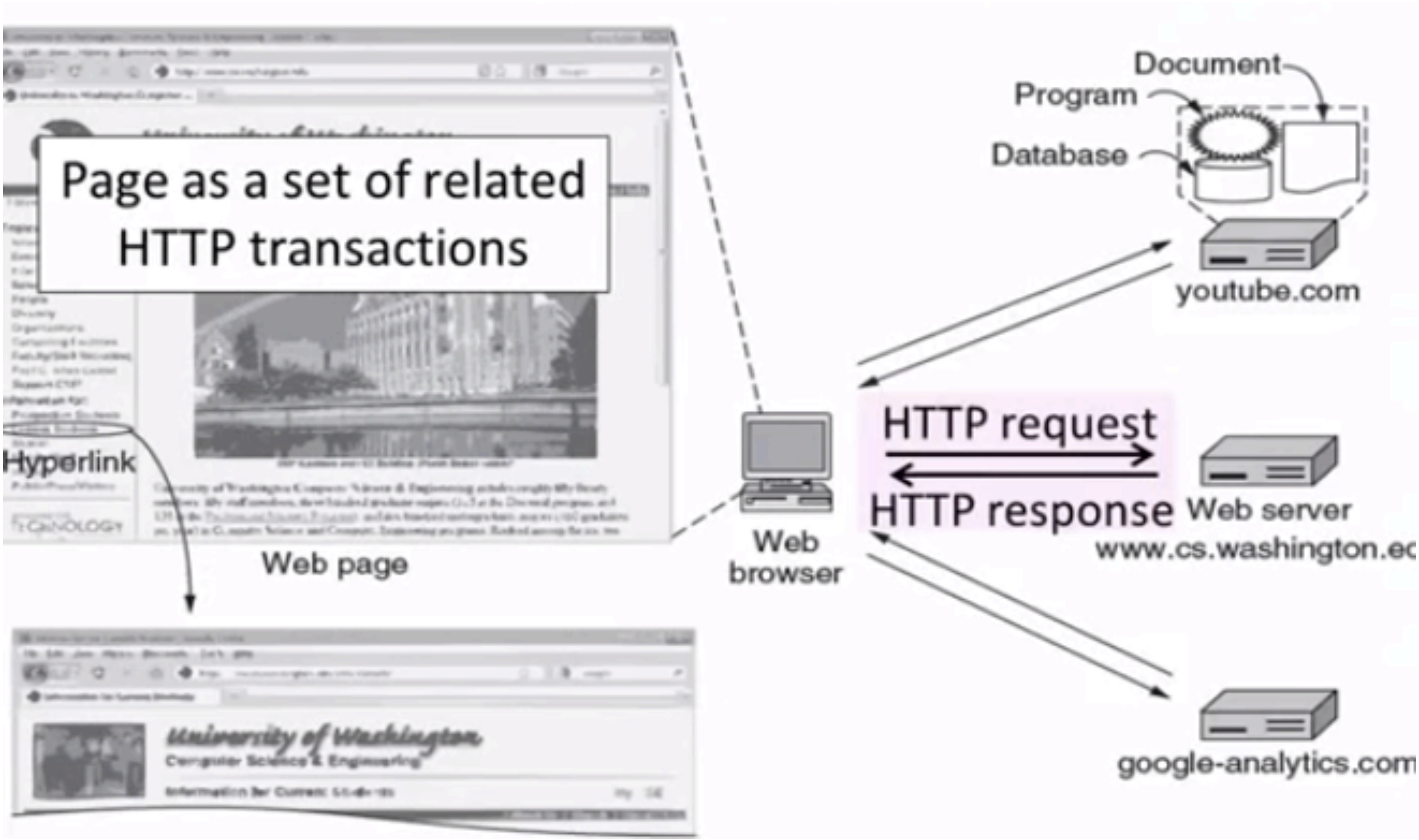
- Sir Tim Berners-Lee (1955-)



- Inventor of the Web
  - Dominant Internet app since mid 90s
  - He now directs the W3C
- Developed Web at CERN in ‘89
  - Browser, server and first HTTP
  - Popularized via Mosaic (‘93), Netscape
  - First WWW conference in ’94 …

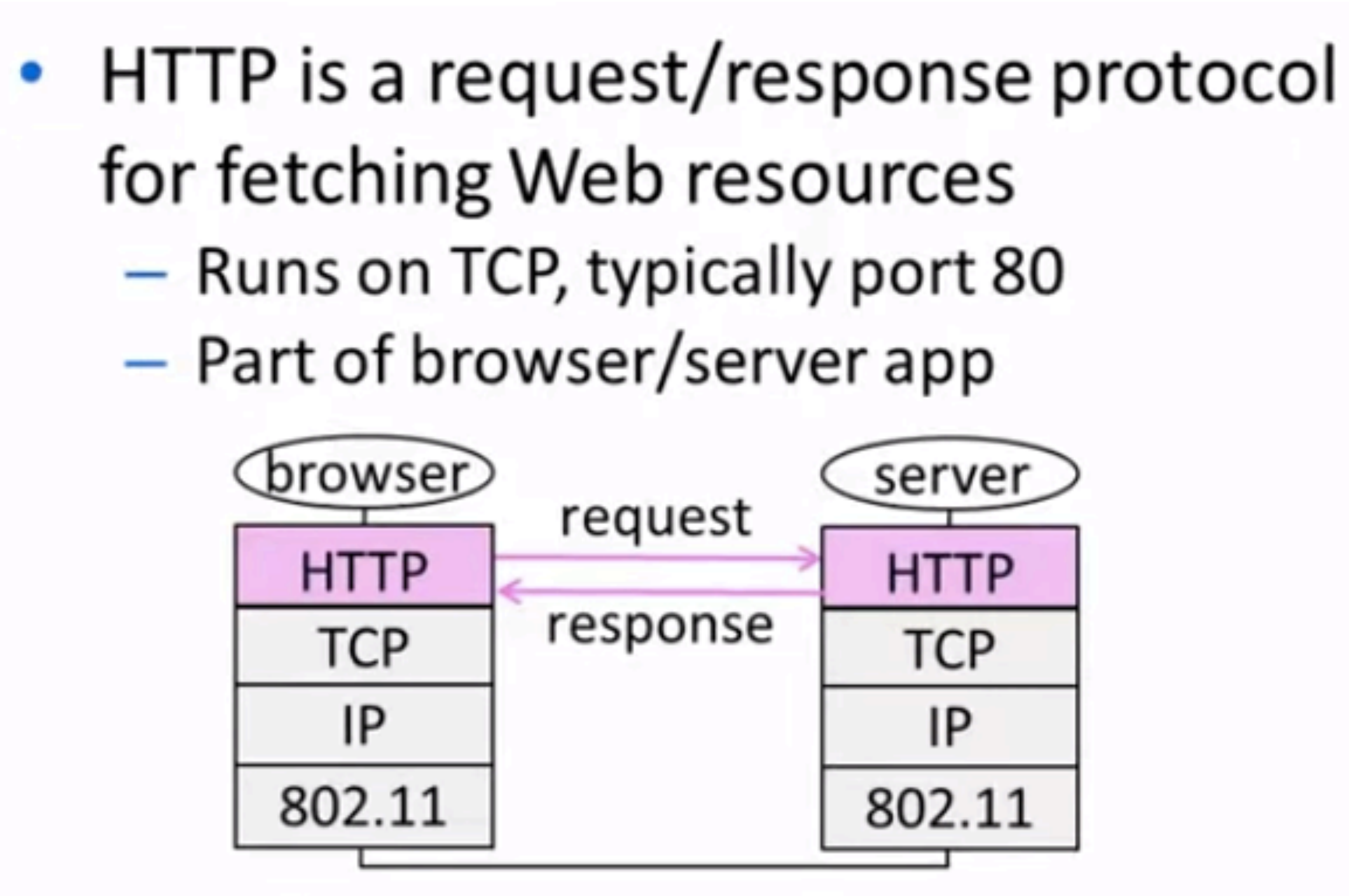
Source: By Paul Clarke, CC-BY-2.0, via Wikimedia Common

Web Context



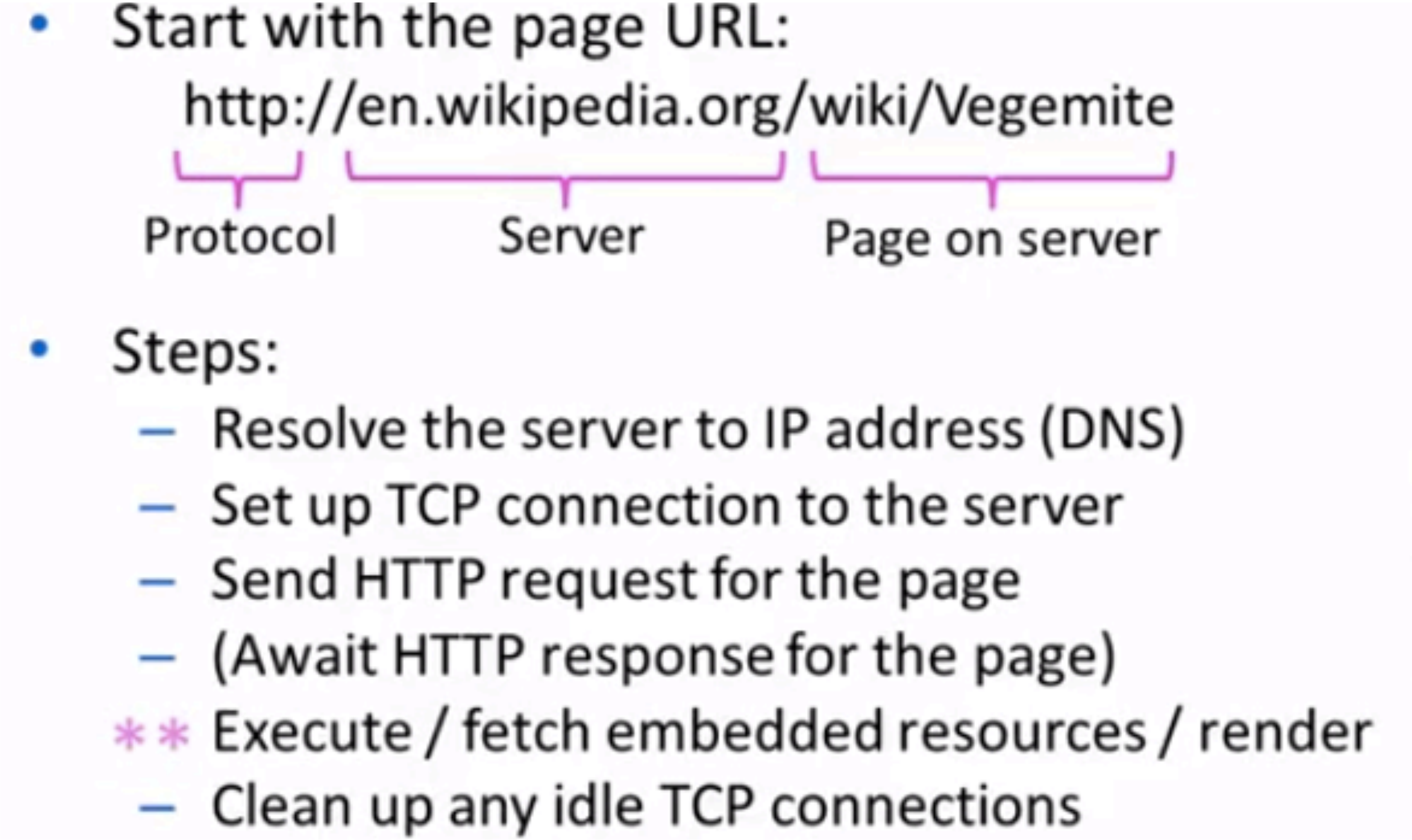
Page as a set of related HTTP transactions

HTTP Context



- HTTP is a request/response protocol for fetching Web resources
  - Runs on TCP, typically port 80
  - Part of browser/server app

Fetching a Web page with HTTP



- Start with the page URL:

  http://en.wikipedia.org/wiki/Vegemite

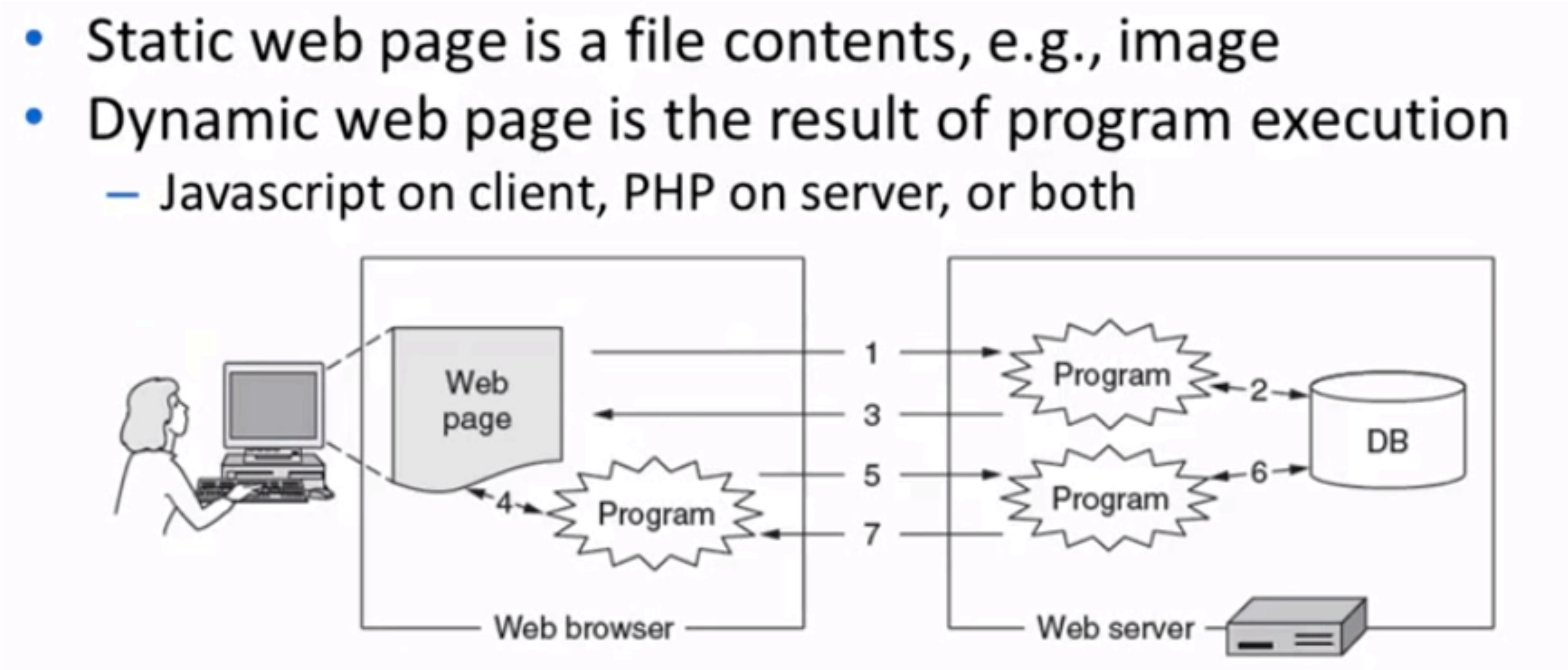
  Protocol Server Page on server

- Steps:
  - Resolve the server to IP address (DNS)
  - Set up TCP connection to the server
  - Send HTTP request for the page
  - (Await HTTP response for the page)

  ** Execute / fetch embedded resources / render

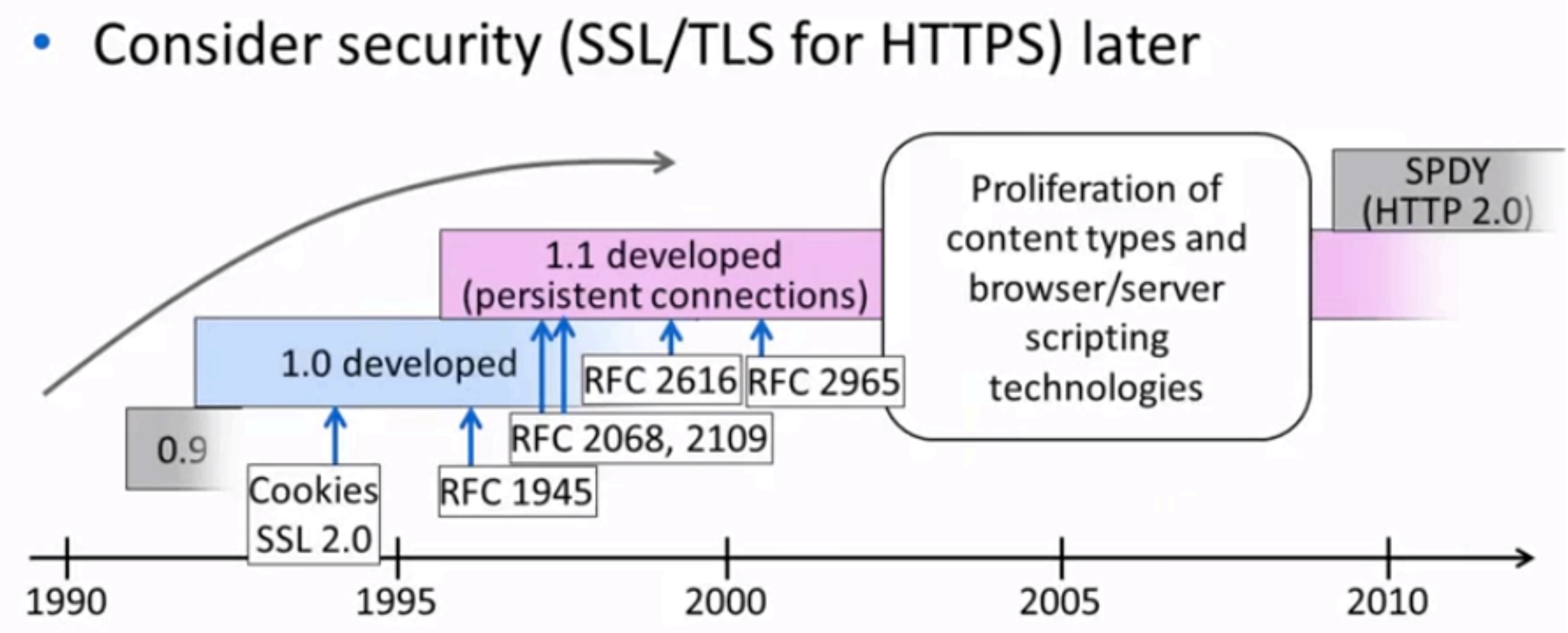
  - Clean up any idle TCP connections

Static vs Dynamic Web pages



- Static web page is a file contents, e.g., image
- Dynamic web page is the result of program execution
  - Javascript on client, PHP on server, or both

Evolution of HTTP



- Consider security (SSL/TLS for HTTPS) later

0.9

1.0 developed

1.1 developed (persistent connections)

Cookies SSL 2.0

RFC 1945

RFC 2068, 2109

RFC 2616

RFC 2965

Proliferation of content types and browser/server scripting technologies

SPDY (HTTP 2.0)

1990

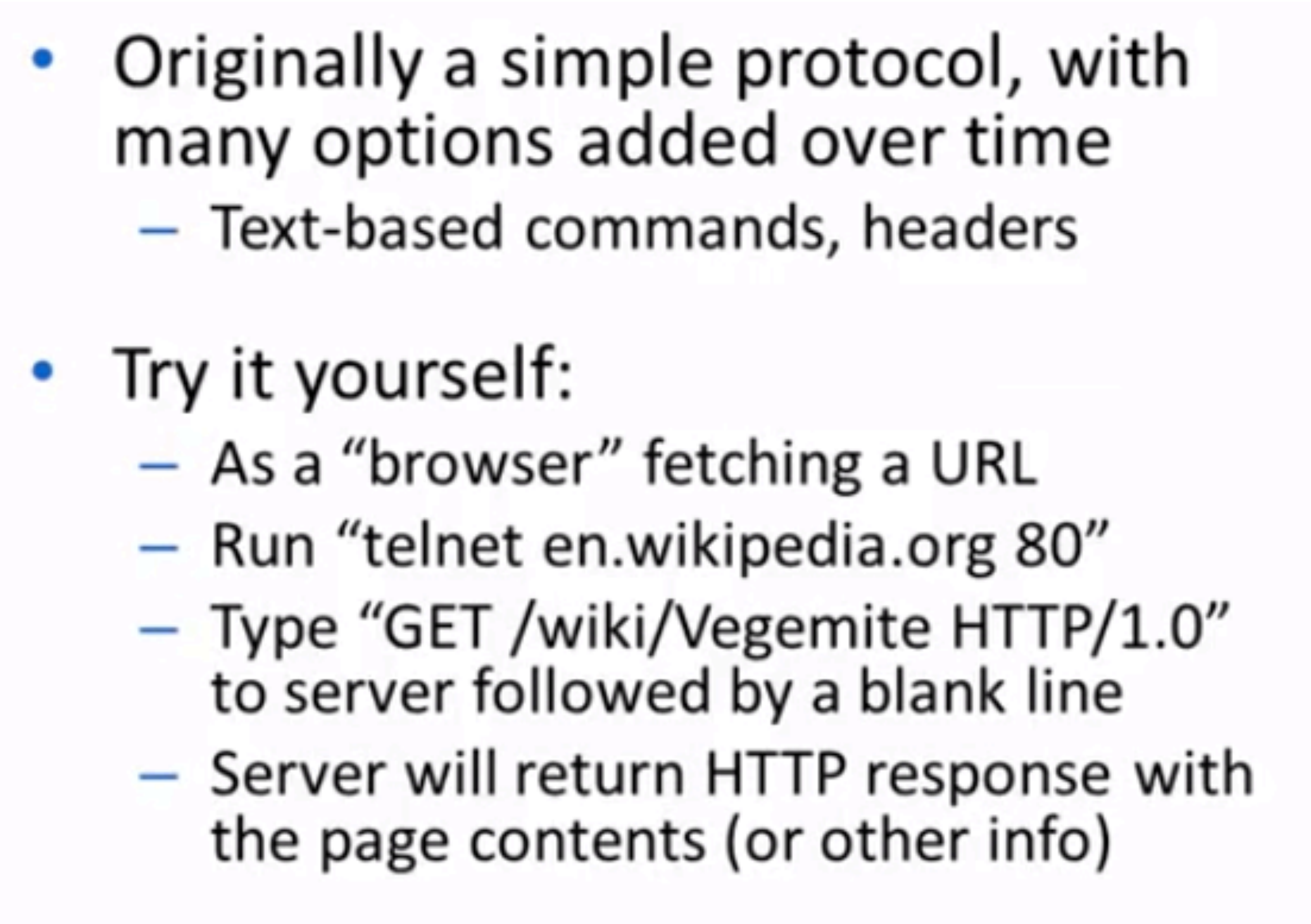
1995

2000

2005

2010

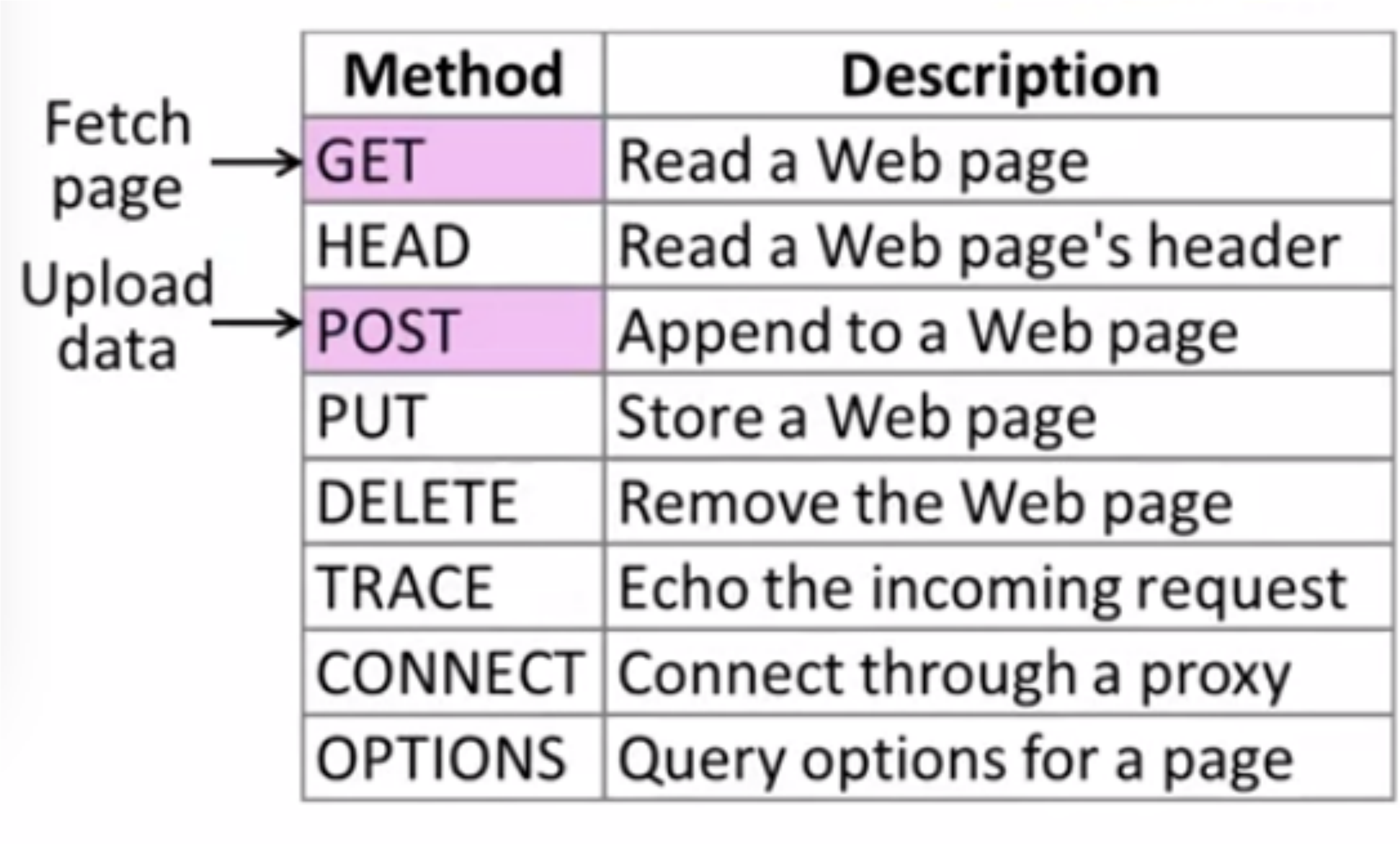
HTTP Protocol



- Originally a simple protocol, with many options added over time
  - Text-based commands, headers
- Try it yourself:
  - As a “browser” fetching a URL
  - Run “telnet en.wikipedia.org 80”
  - Type “GET /wiki/Vegemite HTTP/1.0” to server followed by a blank line
  - Server will return HTTP response with the page contents (or other info)

HTTP Protocol

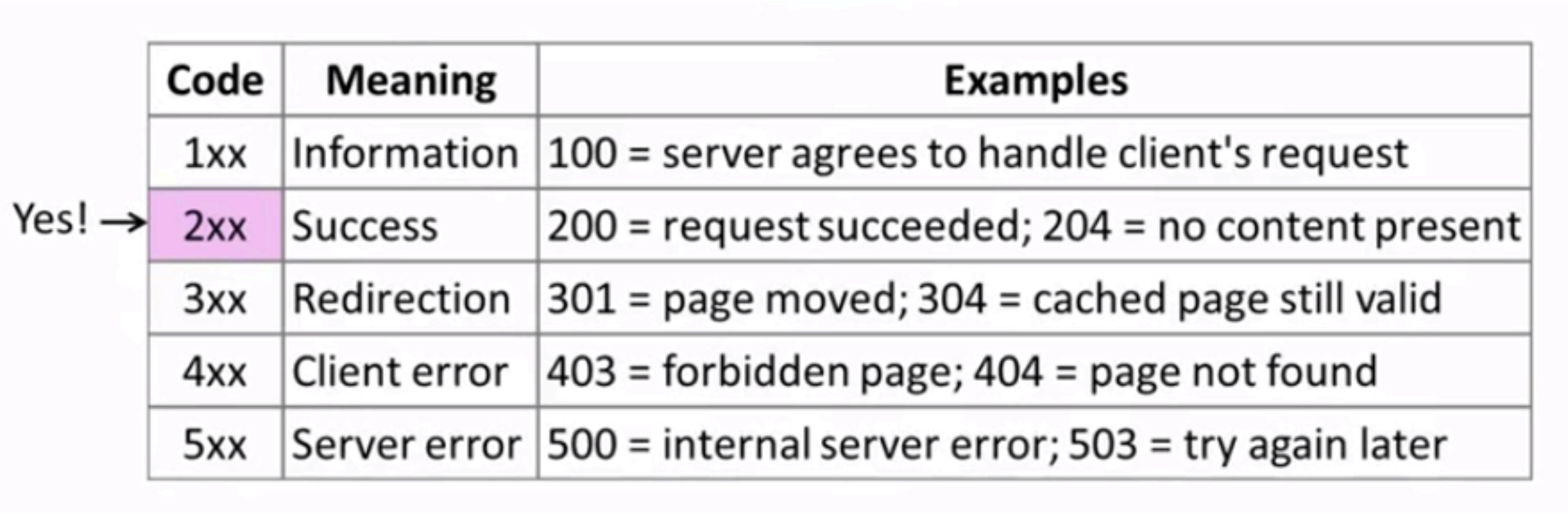
- Commands used in the request


Fetch page
Upload data

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 Protocol

- Codes returned with the response

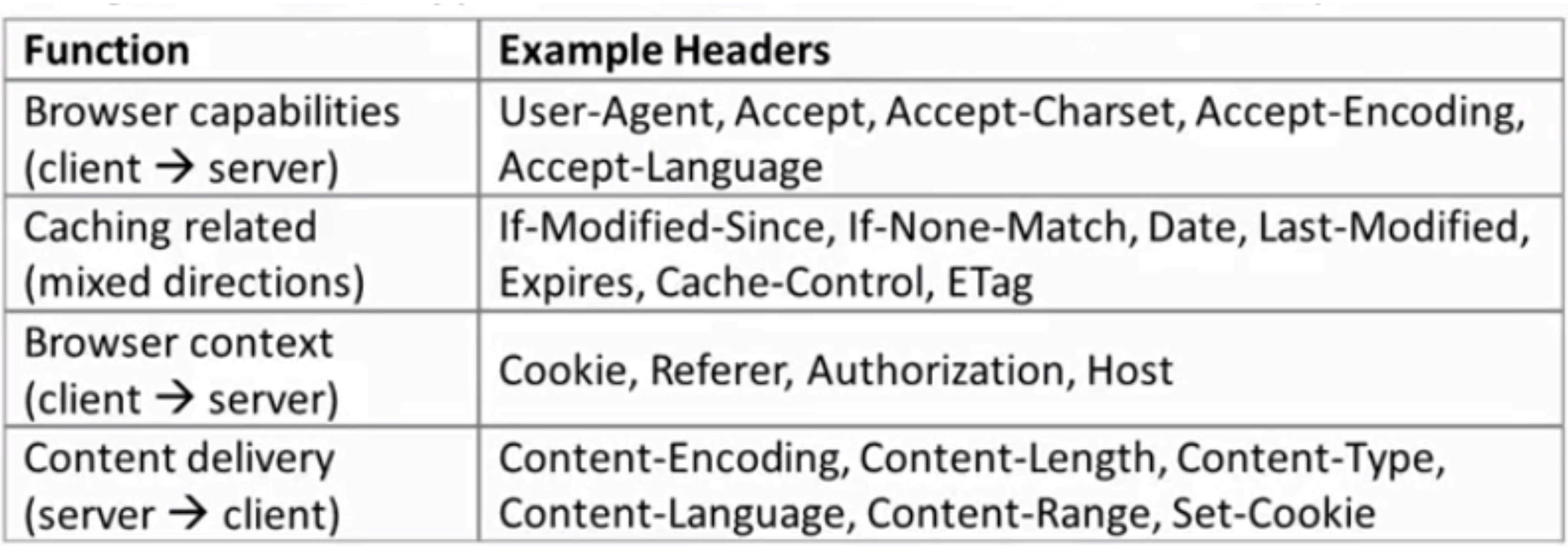


Yes! →

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

HTTP Protocol

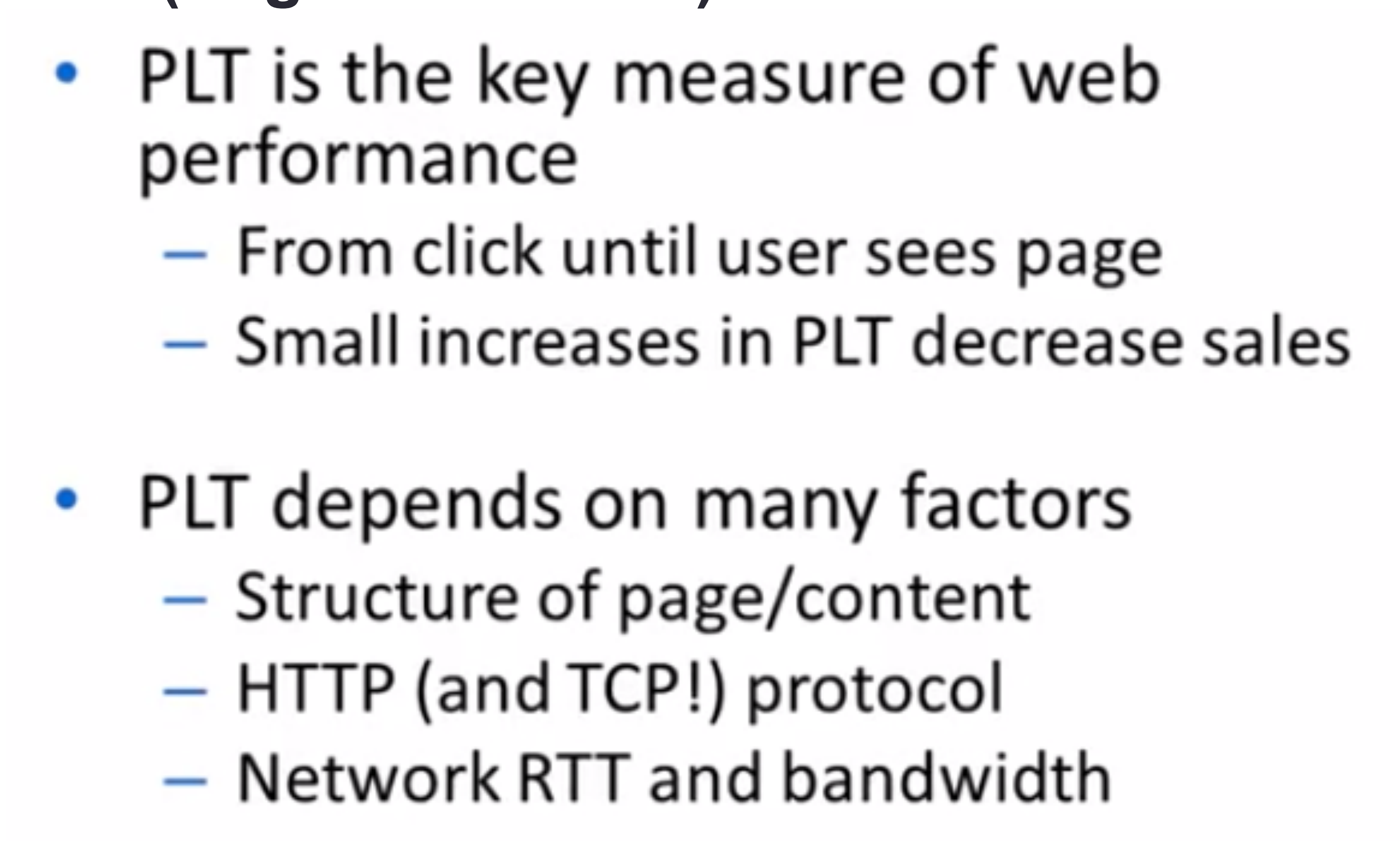
- Many header fields specify capabilities and content
  - Eg., Content –Type: text/html, Cookie:lect=9-1-http



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

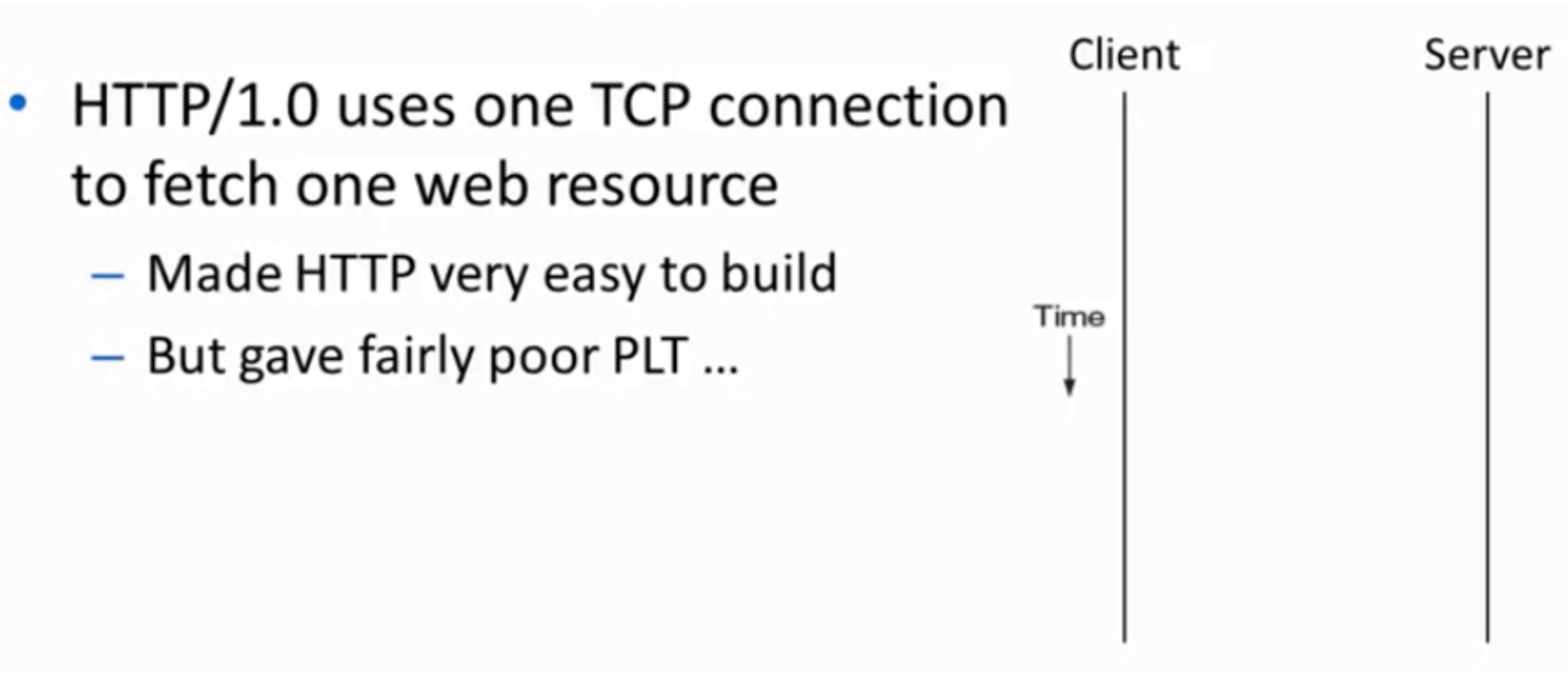
HTTP Performance

- **PLT (Page Load Time)**



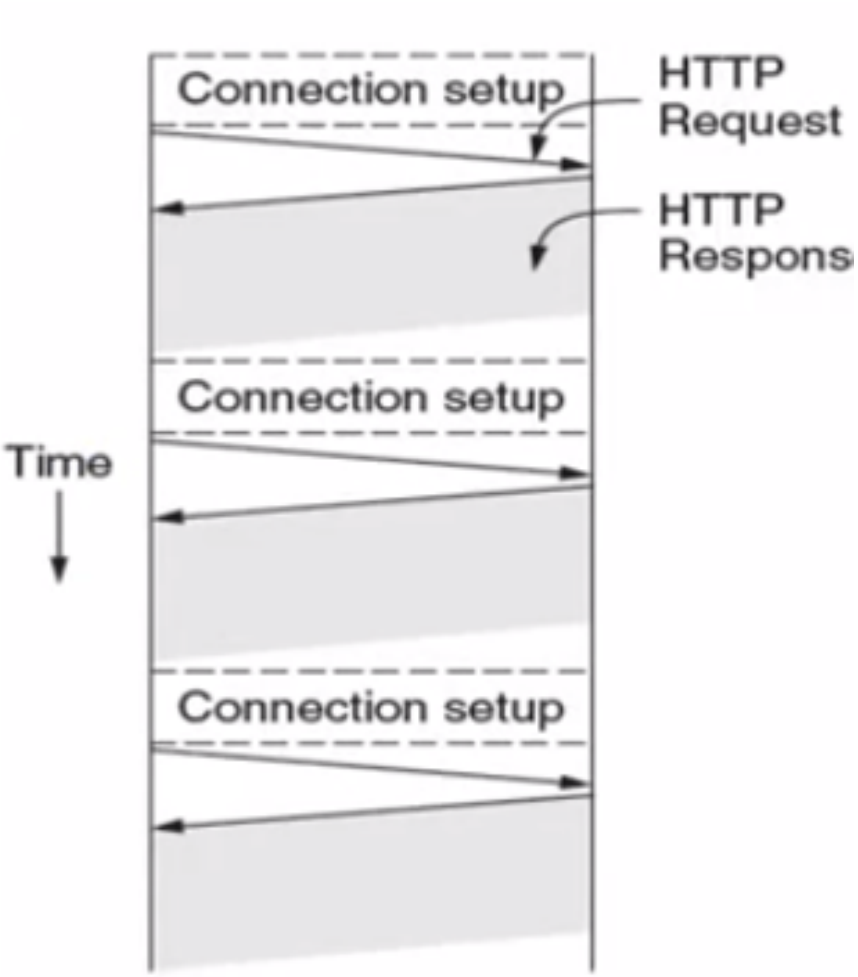
- PLT is the key measure of web performance
  - From click until user sees page
  - Small increases in PLT decrease sales
- PLT depends on many factors
  - Structure of page/content
  - HTTP (and TCP!) protocol
  - Network RTT and bandwidth

Early Performance

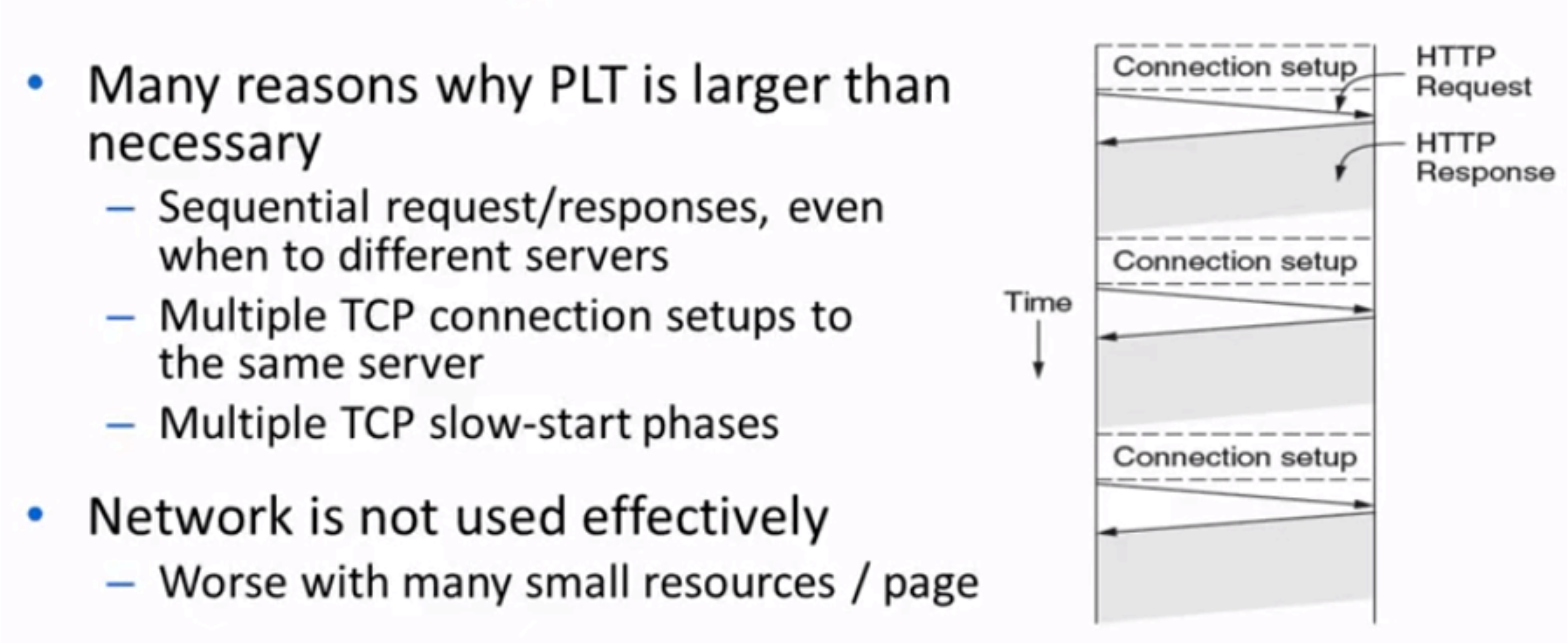


- HTTP/1.0 uses one TCP connection to fetch one web resource
  - Made HTTP very easy to build
  - But gave fairly poor PLT …

Early Performance

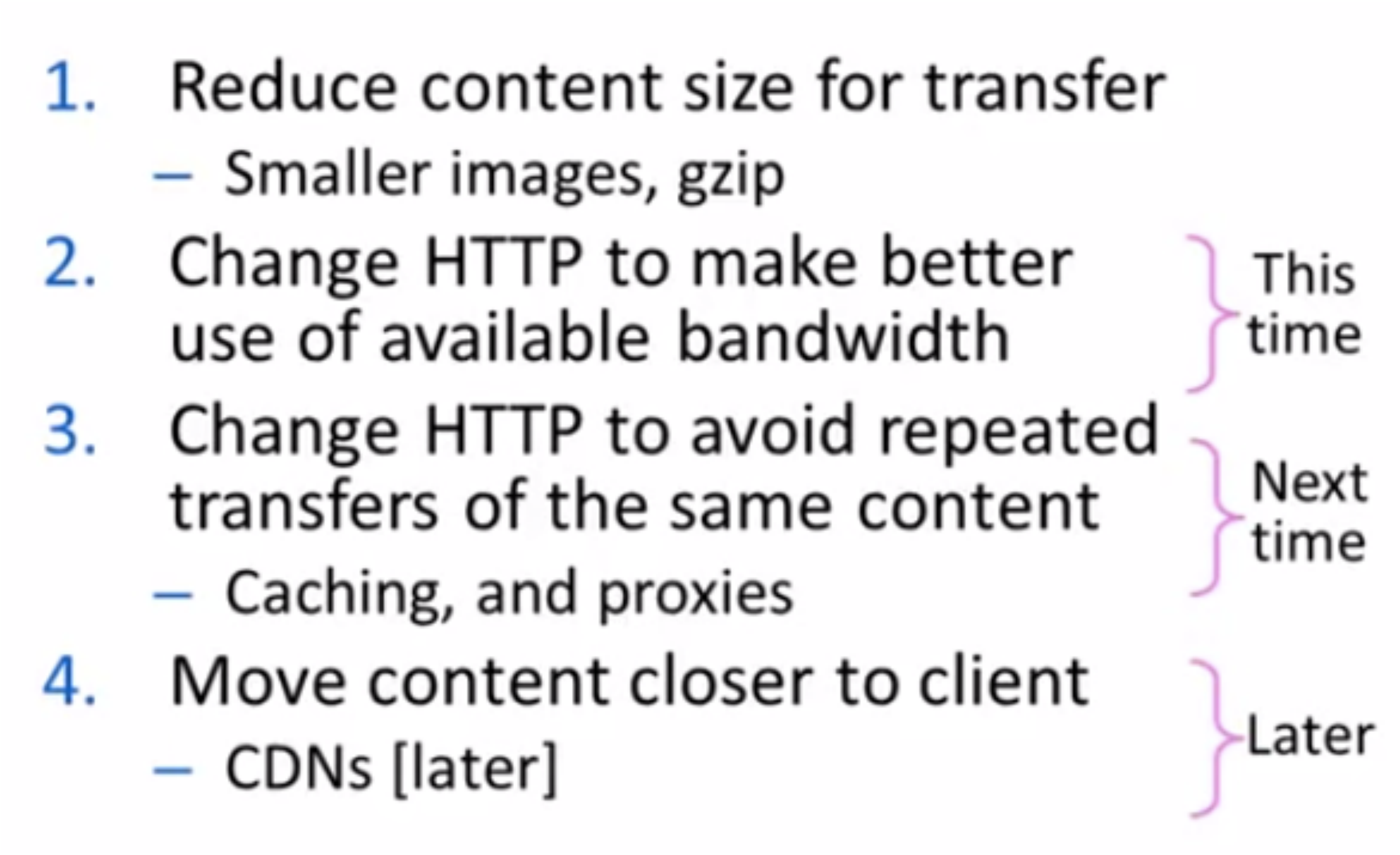


Early Performance



- Many reasons why PLT is larger than necessary
  - Sequential request/responses, even when to different servers
  - Multiple TCP connection setups to the same server
  - Multiple TCP slow-start phases
- Network is not used effectively
  - Worse with many small resources / page

Ways to Decrease PLT



1. Reduce content size for transfer
   - Smaller images, gzip
2. Change HTTP to make better use of available bandwidth

This time

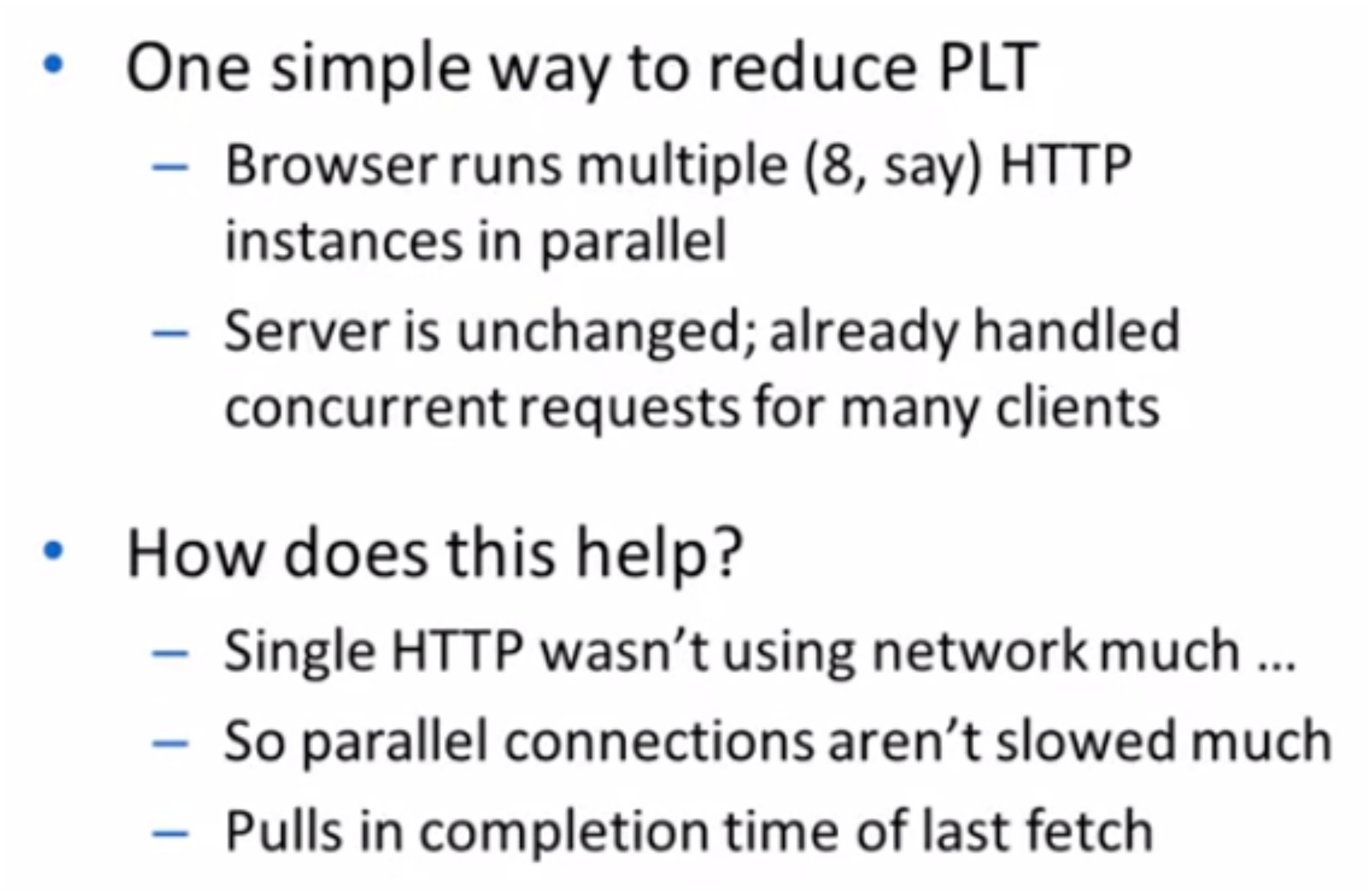
3. Change HTTP to avoid repeated transfers of the same content
   - Caching, and proxies

Next time

4. Move content closer to client
   - CDNs [later]

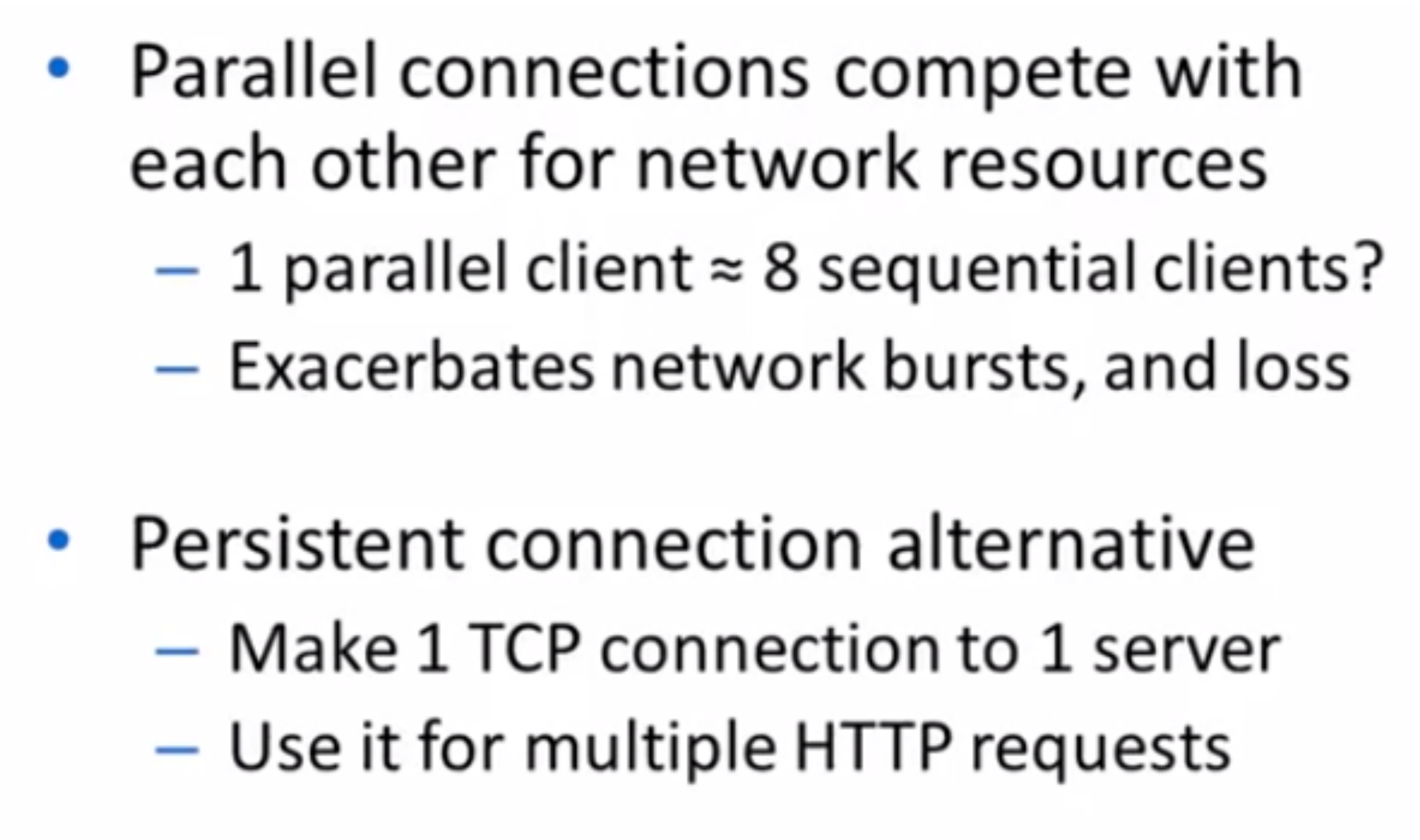
Later

Parallel Connections



- One simple way to reduce PLT
  - Browser runs multiple (8, say) HTTP instances in parallel
  - Server is unchanged; already handled concurrent requests for many clients
- How does this help?
  - Single HTTP wasn't using network much …
  - So parallel connections aren't slowed much
  - Pulls in completion time of last fetch

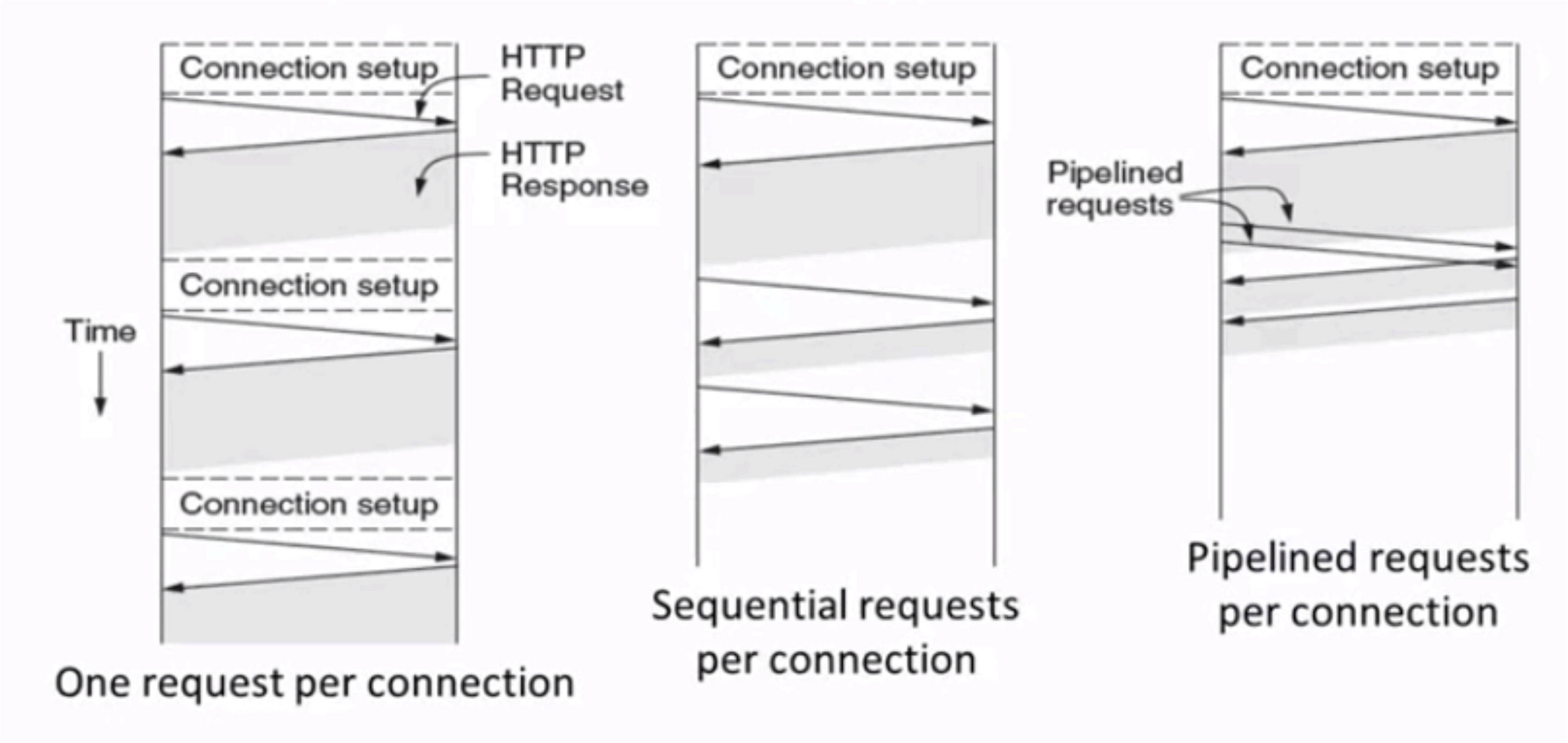
Persistent Connections



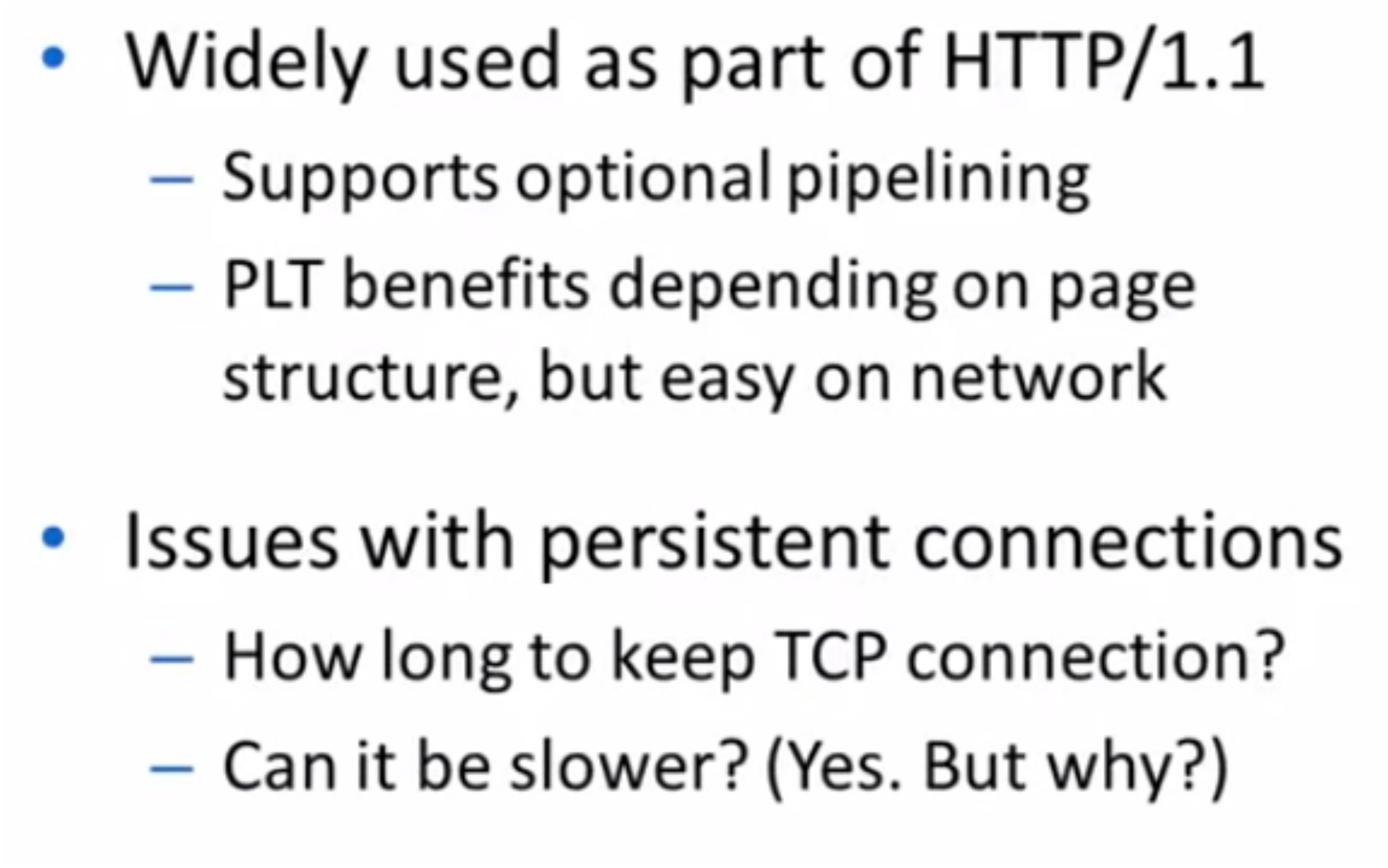
- Parallel connections compete with each other for network resources
  - 1 parallel client ≈ 8 sequential clients?
  - Exacerbates network bursts, and loss

- Persistent connection alternative
  - Make 1 TCP connection to 1 server
  - Use it for multiple HTTP requests

Comparison


Connection setup
HTTP Request
HTTP Response
Connection setup
Time
Connection setup
One request per connection
Connection setup
Sequential requests per connection
Connection setup
Pipelined requests
Pipelined requests per connection

Overview of Persistent Connections



- Widely used as part of HTTP/1.1
  - Supports optional pipelining
  - PLT benefits depending on page structure, but easy on network
- Issues with persistent connections
  - How long to keep TCP connection?
  - Can it be slower? (Yes. But why?)

Next Class

- An overview of the HTTP formats
- An overview of Catching and Proxies