Introduction

Lecture 1

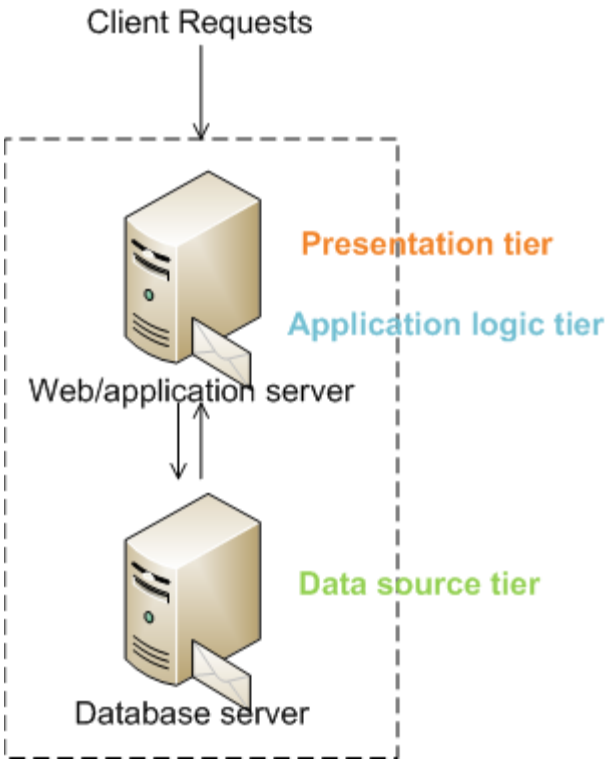
March 7, 2017

Based on Chapter 1 of Fundamentals of Web Development

Outline

- Web Application architecture in general
  - Draft course outline
- Basic concepts behind web browsers
  - Web page
  - HTTP protocol

Web Application Architecture



Simple web app with three principle tier/layers

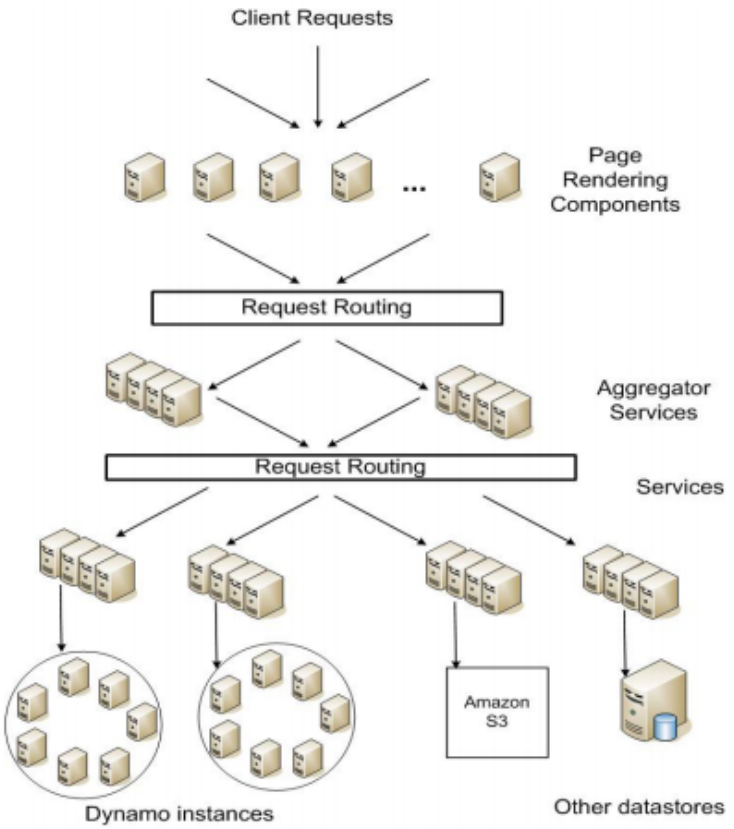
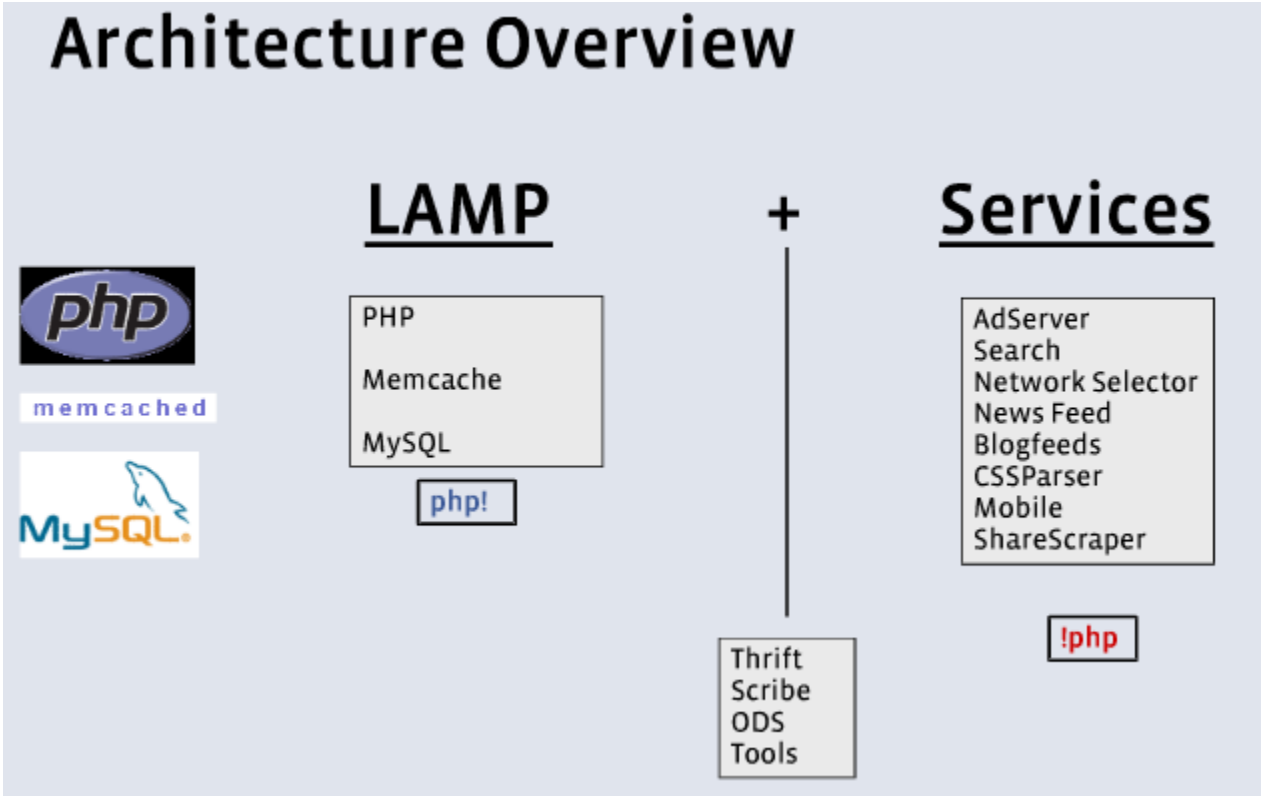


Figure 1: Service-oriented architecture of Amazon's platform [2]

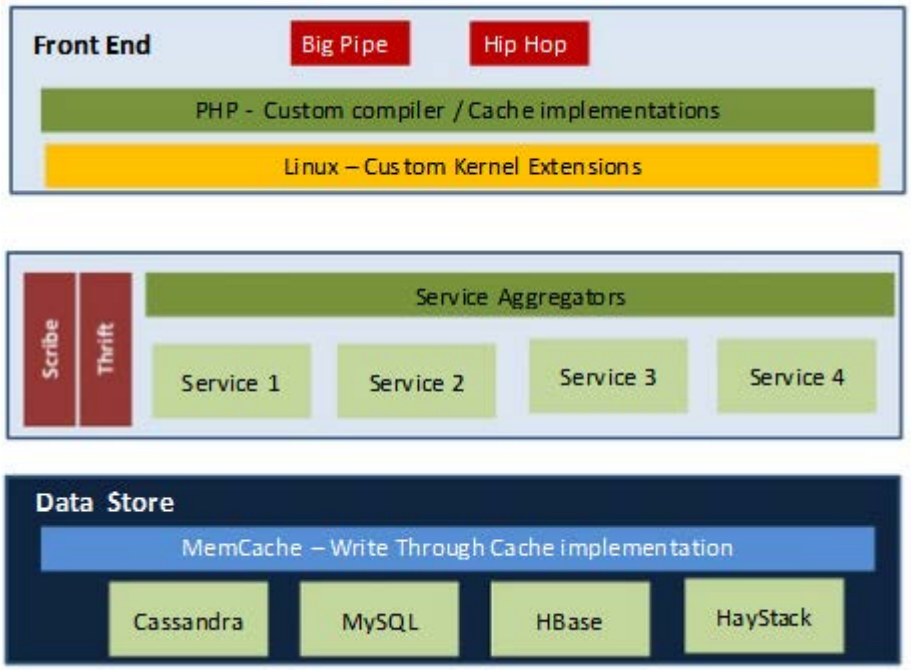
Amazon.com started **10 years** ago as a **monolithic application**, running on a Web server, talking to a database on the back end…. Over time, this grew into **hundreds** of services and a number of application servers that aggregate the information from the services…. If you hit the Amazon.com gateway page, the application calls more than 100 services to collect data and construct the page for you. **A Conversation with Werner Vogels,** ACM Queue, May, **2006** [3]

Web Application Technology Stack (Facebook)



Aditya Agarwal, Facebook: Science and the Social Graph, Qcon San Francisco 2008 [http://www.infoq.com/presentations/Facebook-Software-Stack**]**

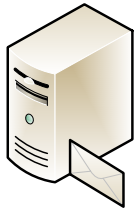
Web Application Technology stack (Facebook)



Facebook Architecture: Breaking it Open, Slide 21 [http://adititechnologiesblog.blogspot.com.au/2012/01/aditi-lead-open-talk-facebook.html]

What are covered in this course

- Simple three tiered web application
  - Client Side Technology/Server Side Technology
  - Communication between Client/Server side
  - Security
- Integrating services from various components
  - web services
- Assumed Knowledge
  - Some programming experience
  - Basic understanding of data model and storage systems



Lecture outline

Week	Topic
1 (07.03)	How the web works
2 (14.03)	Brief Intro to HTML and CSS
3 (21.03)	JavaScript Client side scripting
4 (28.03)	Browser and rendering process
5 (04.04)	Serve side development with servlet and JSP (Assignment 1 due)
6 (11.04)	Server side development with nodejs and expressjs
EASTER BREAK (14/04-23/04)	
7 (25.04)	Anzac Day public holiday
8 (02.05)	Session and Routes
9 (09.05)	Connecting to database
10 (16.05)	Client side framework
11 (23.05)	Security (Quiz)
12 (30.05)	Web Services (Assignment 2 due)
13 (06.06)	Review

Fundamentals of Web Application

- It is of Client/Server architecture
- Client Side Basics
  - Displaying content: **HTML**
  - Displaying content with desirable styles: **CSS**
  - Interacting with content: **JavaScript**
  - A central point of contact: **the browser**
- Server Side Basics
  - A web application written in
    - **Python, PHP, Java, C#, Ruby, JavaScript,…**
  - Some system that stores the content/data: **file system**, **database system, …**
  - Some system that listens/handles common network activities: **web/application server**
- In between
  - A way to locate the server given a name: **DNS services**
  - A way to send Instruction/Content between client and server: **protocols**
  - A way to secure the content: **security mechanism**

Ever changing Fields

- Both HTML and CSS have evolved rapidly
- JavaScript is getting more powerful
  - Lots of libraries
  - It can be used on server side as well
- Browsers become more complicated
- New programming languages/frameworks for writing server applications
- New architecture/programming style to cater for scalability and cloud hosting
  - Eg. Flux vs. MVC as proposed by Facebook
  - When Netflix migrated to AWS in 2010, they have to re-implement many services
- The communication core, e.g. protocols are relatively stable

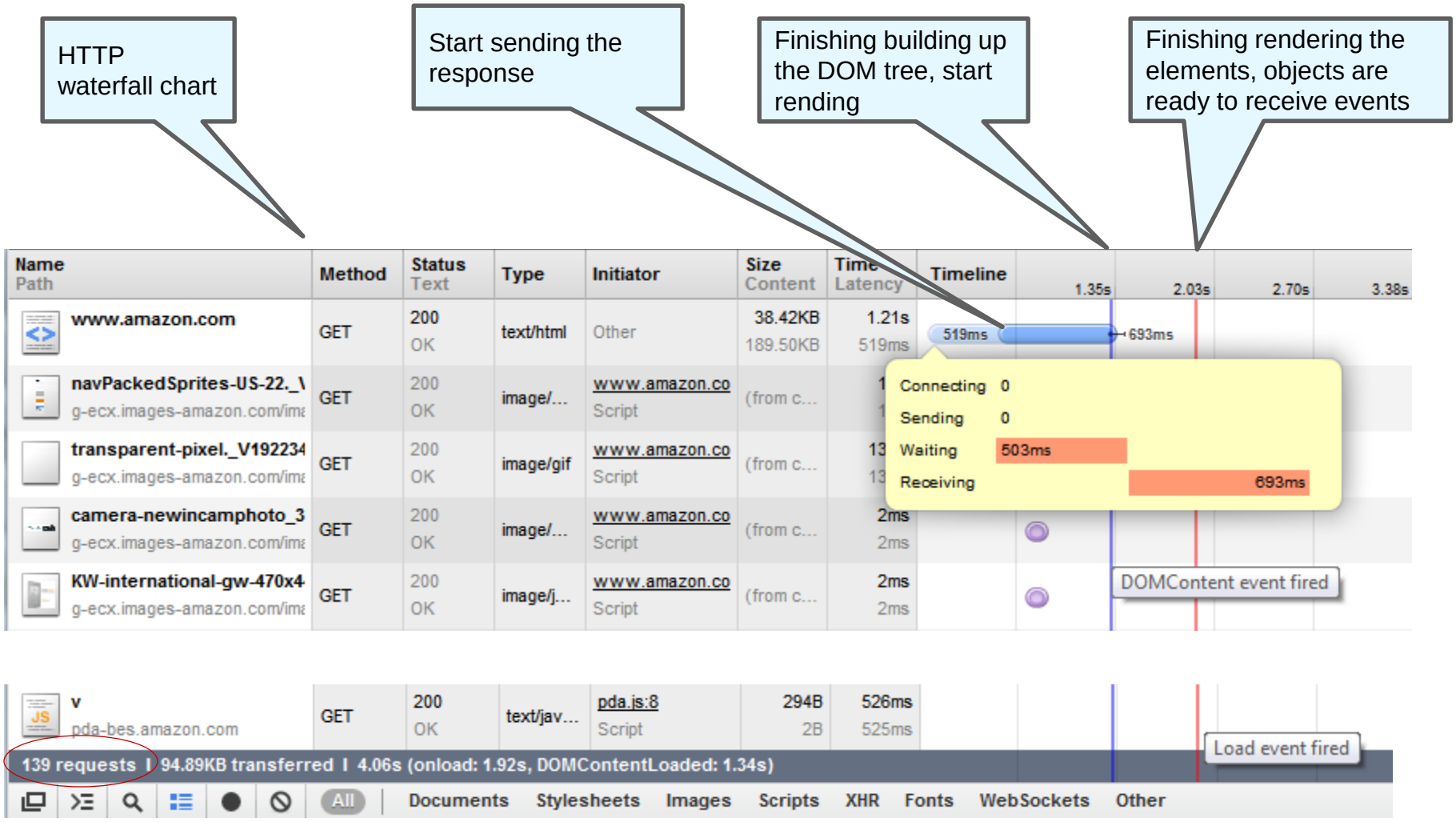
Outline

- Course Information
  - Web application architecture in general
  - What are covered in this course
  - Course resources
  - Assessment package
- Basic concepts of web (from end user perspective)
  - Web page
  - HTTP protocol

Web browser

- Main responsibility of browser
  - Generate and submit requests to web servers
  - **Accept response and render results**
- In detail:
  - **Caching**
  - Authentication and authorization
  - State maintenance
  - **Requesting support data items**
  - **Taking actions in response to other headers and status codes**
  - Rendering complex objects
  - **Dealing with error conditions**

When you send a request from a browser

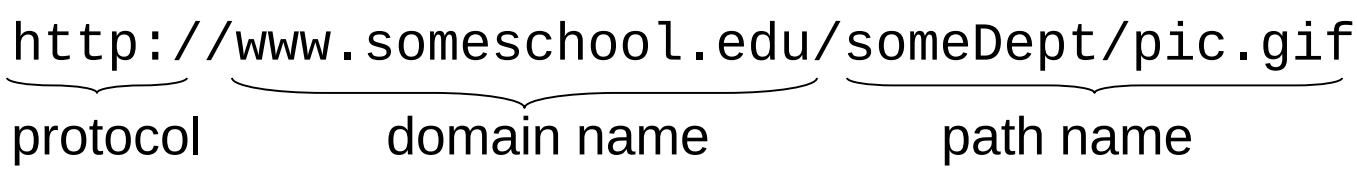


World Wide Web

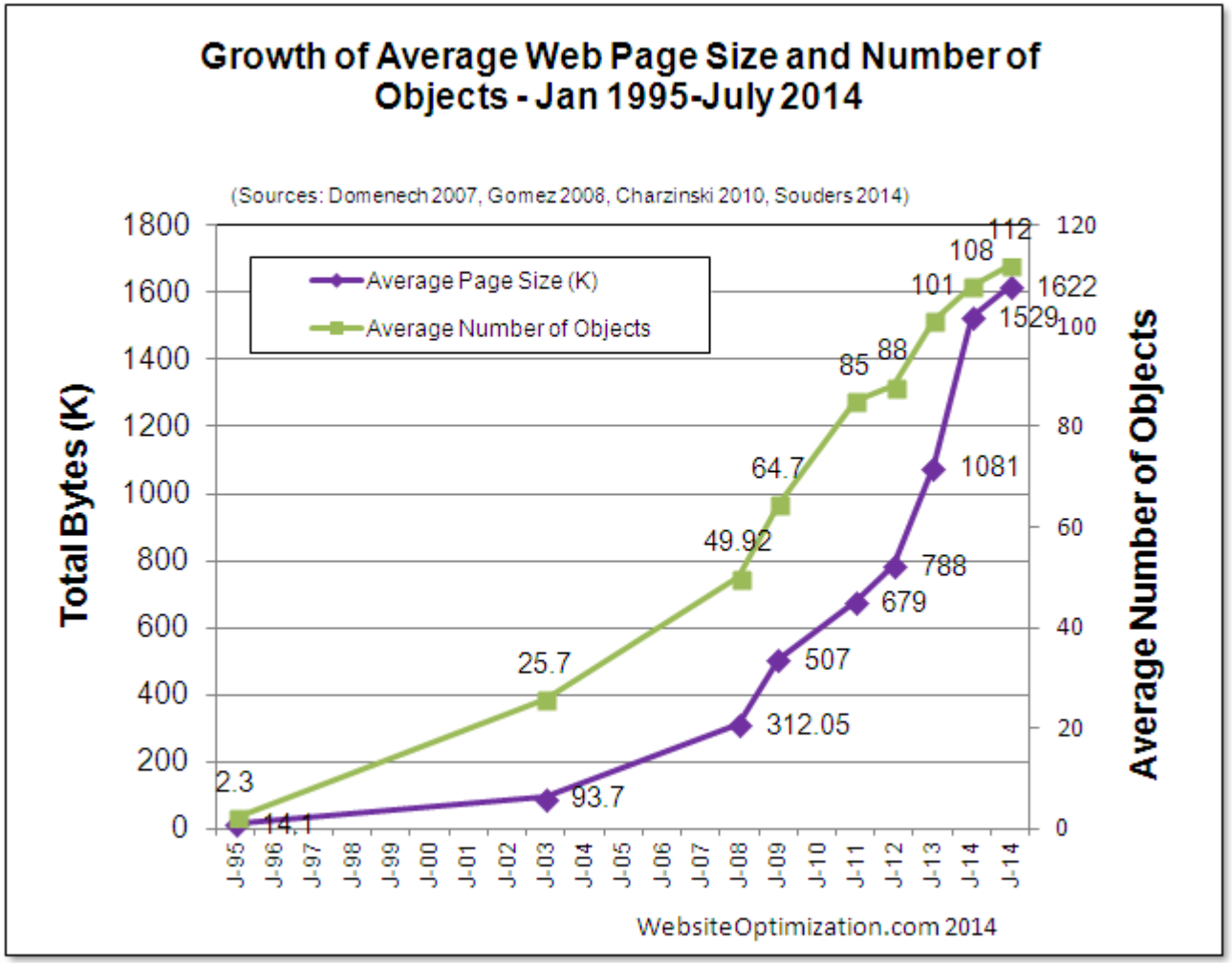
- The invention of the WWW is usually attributed to the British Sir Tim Berners-Lee, who, along with the Belgian Robert Cailliau, published a proposal in 1990 for a hypertext system while both were working at CERN in Switzerland.
- Core Features of the Web
  - A URL to uniquely identify a resource on the WWW.
  - The HTTP protocol to describe how requests and responses operate.
  - A software program (later called web server software) that can respond to HTTP requests.
  - HTML to publish documents.
  - A program (later called a browser) to make HTTP requests from URLs and that can display the HTML it receives.

A web page

- Jargons everyone is familiar with
  - Web page consists of objects
  - Object can be HTML file, scripts, JPEG image, video/audio file,…
  - Web page consists of a base HTML-file which includes several referenced objects
  - Each object is addressable by a URL (Uniform Resource Locator)
  - Example URL:



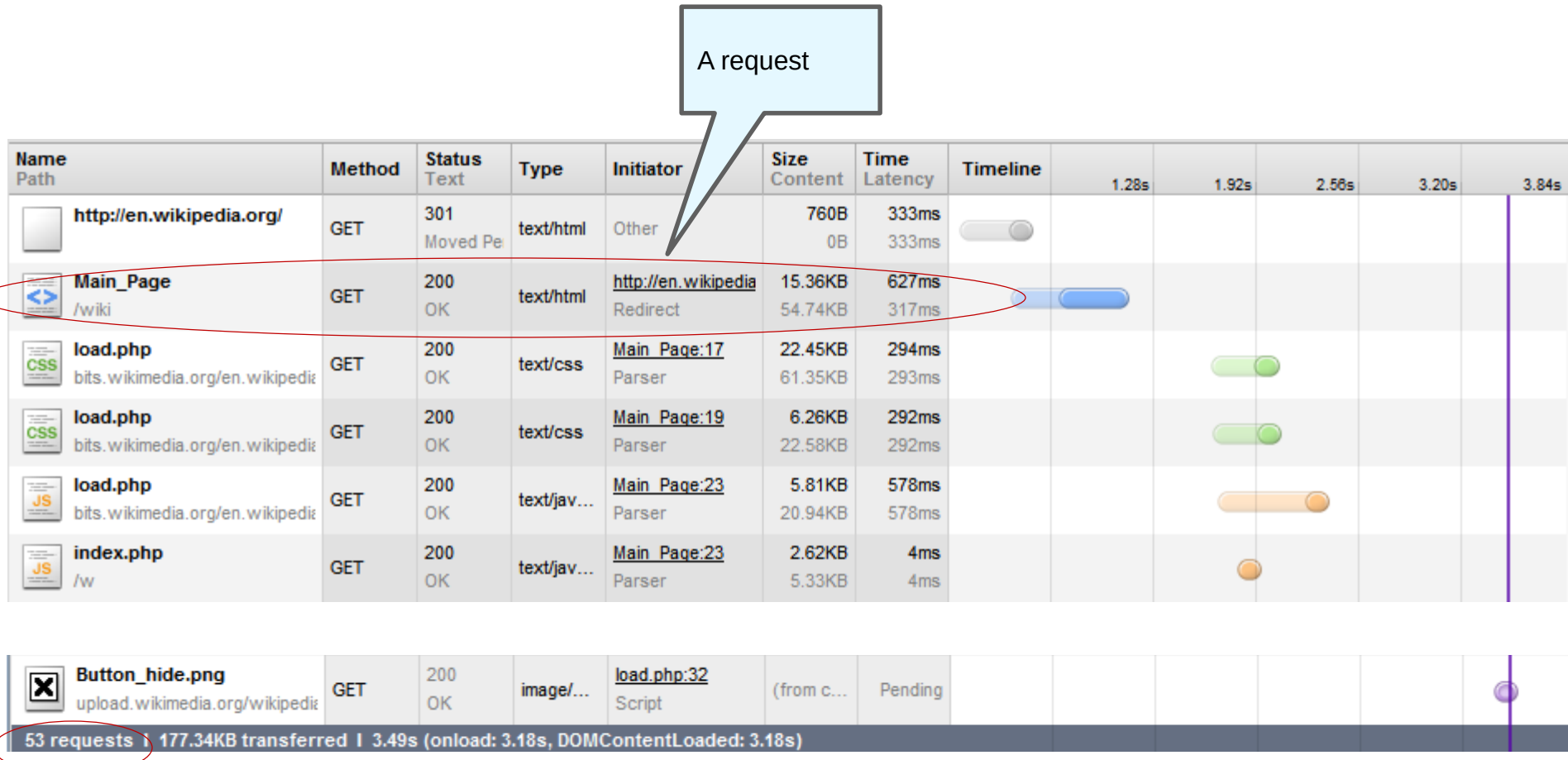
A web page is not just some text



Average Web Page Breaks 1600K
http://www.websiteoptimization.com/speed/tweak/average-web-page/

A web page consists of many objects

53 requests to show the English wikipedia page on your browser



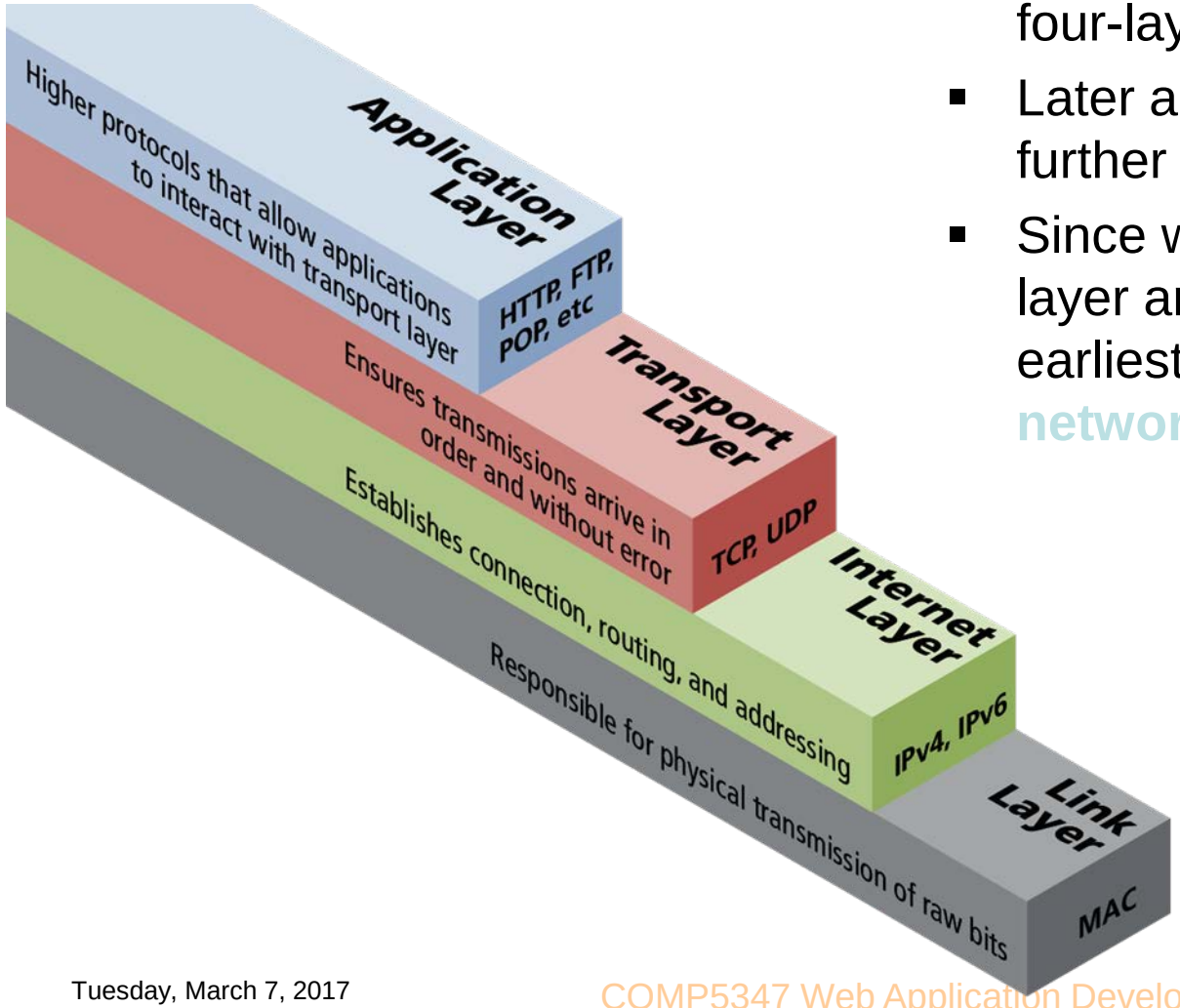
Support for content types

- To view content on the web, your browser might do
  - Render the content as a text page or an HTML page
  - Present content inline
  - Run scripts
  - Launch a helper application capable of presenting non-HTML content
  - Get confused into showing the content of an HTML file (or a server-side script) as plain text without attempting to render it or execute it
- Browser determines the content type and performs actions appropriate for that type
  - HTTP borrows its content typing system from Multipurpose Internet Mail Extension(MIME)
    - Content-encoding
    - Content-type

What is a Protocol?

- The internet exists today because of a suite of interrelated communications protocols.
  - The research network ARPANET was created in the 1960s. It was funded and controlled by the United States government, and was used exclusively for academic and scientific purposes.
  - The early network started small with just a handful of connected campuses in 1969 and grew to a few hundred by the early 1980s.
  - To promote the growth and unification of the disparate networks a suite of protocols was invented to unify the networks together.
  - By 1981, new networks built in the US began to adopt the TCP/IP communication model, while older networks were transitioned over to it.
- A **protocol** is a set of rules that partners in communication use when they communicate.

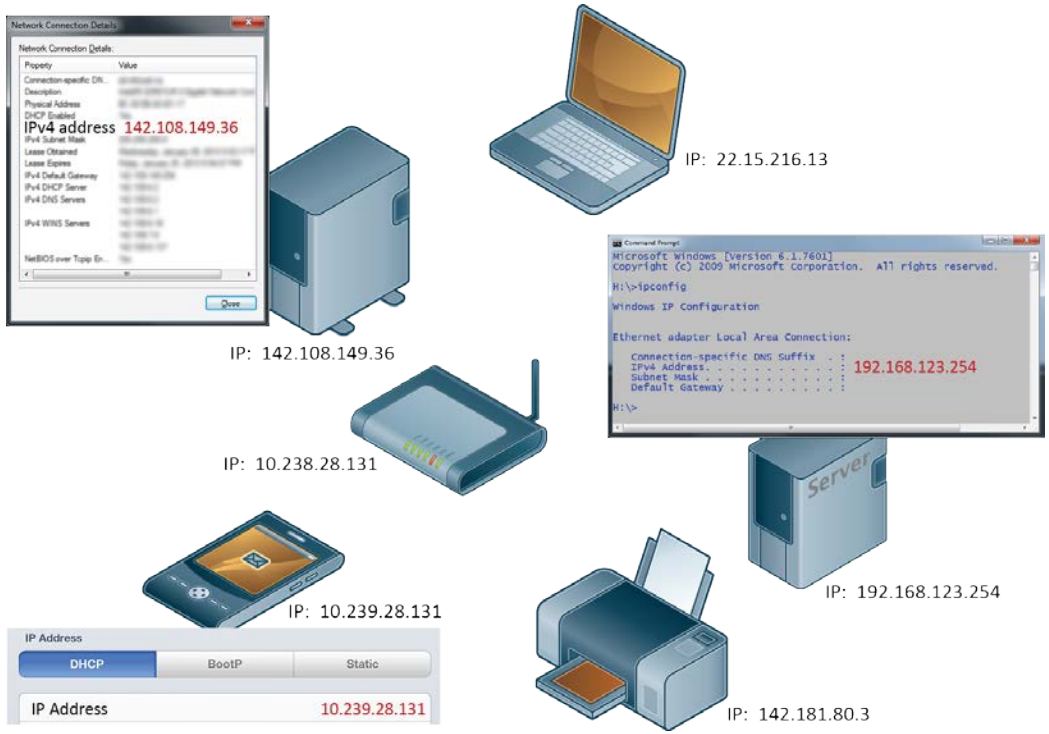
A Layered Architecture



- The TCP/IP Internet protocols were originally abstracted as a four-layer stack.
- Later abstractions subdivide it further into five or seven layers.
- Since we are focused on the top layer anyhow, we will use the earliest and simplest four-layer network model.

Internet Protocol

- The Internet uses the **Internet Protocol** (**IP**) addresses to identify destinations on the Internet.
- Every device connected to the Internet has an **IP address**, which is a numeric code that is meant to uniquely identify it.

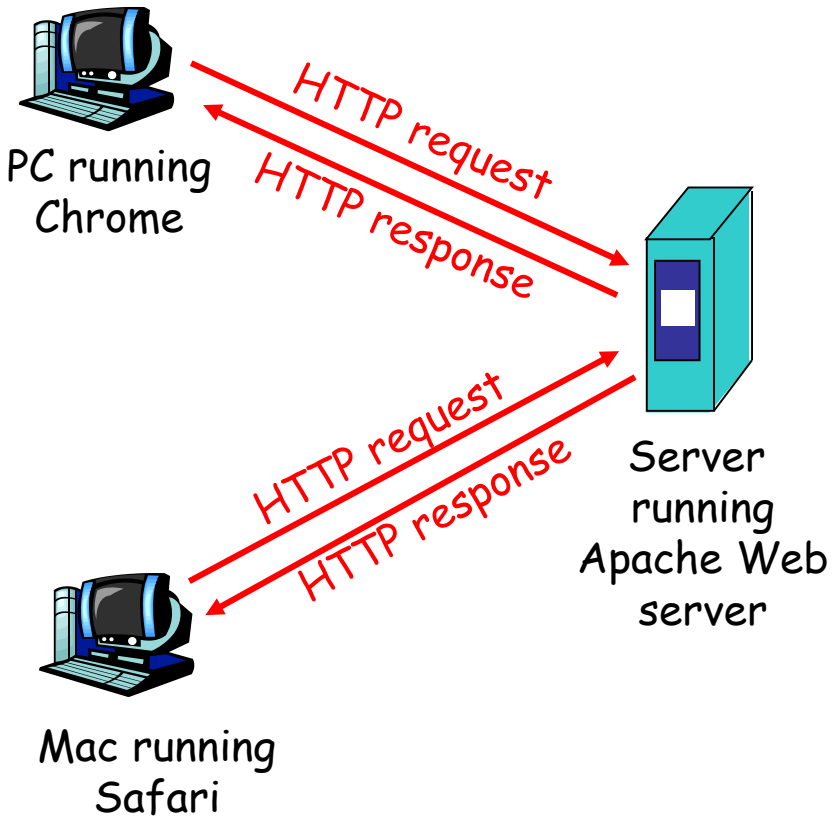


Transport Layer and Application Layer

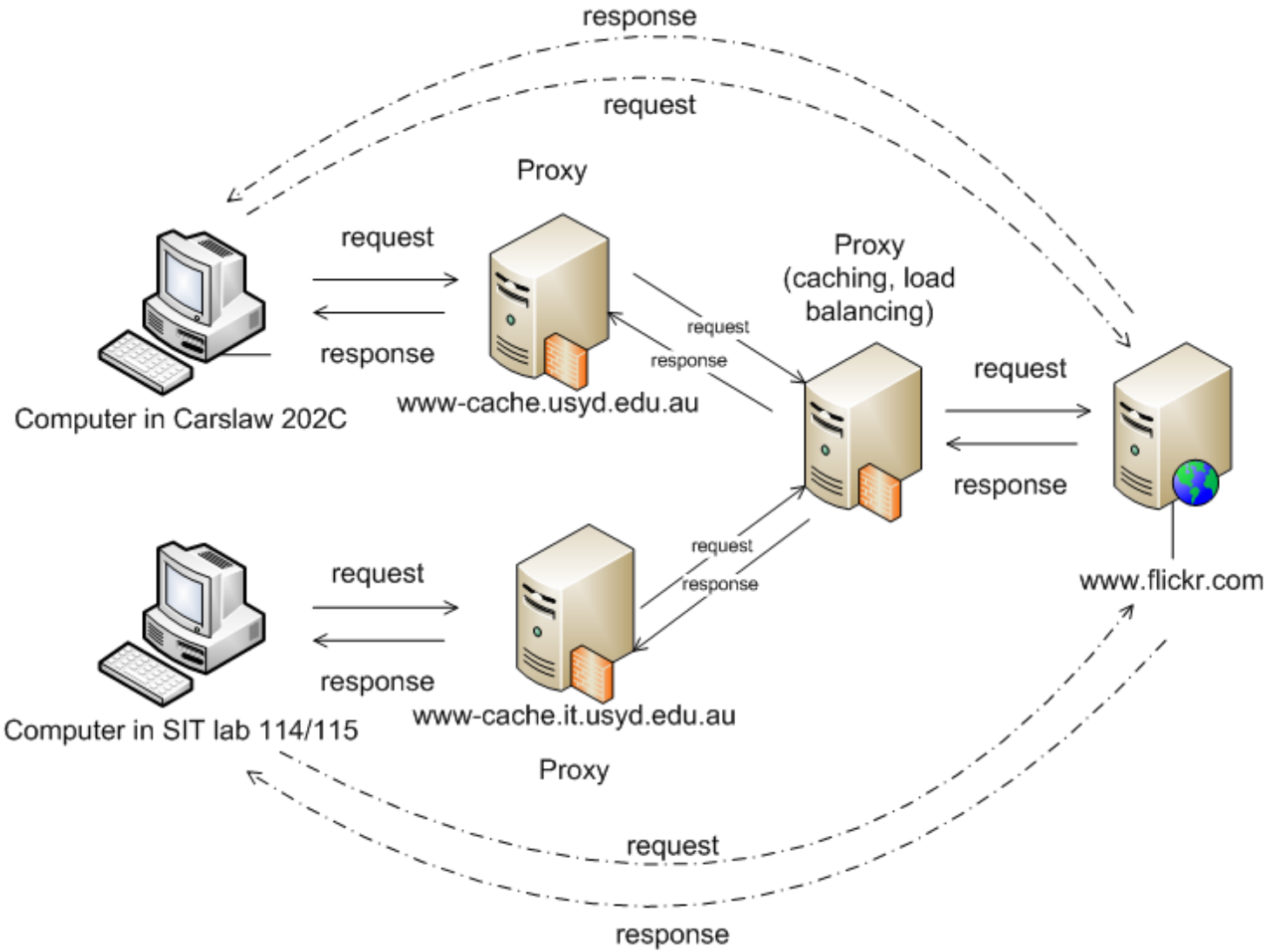
- Internet layer protocol (e.g. IP) provides logical communication between hosts identified by **`IP`** address
- Transport layer protocol (e.g. TCP) provides logical communication between processes running on the hosts identified by **`ip:port`**
  - In particular, TCP ensures that transmissions arrive, in order, and without error
- Application layer protocol (e.g. HTTP) defines the syntax and semantics of the message send between processes

HTTP: hypertext transfer protocol

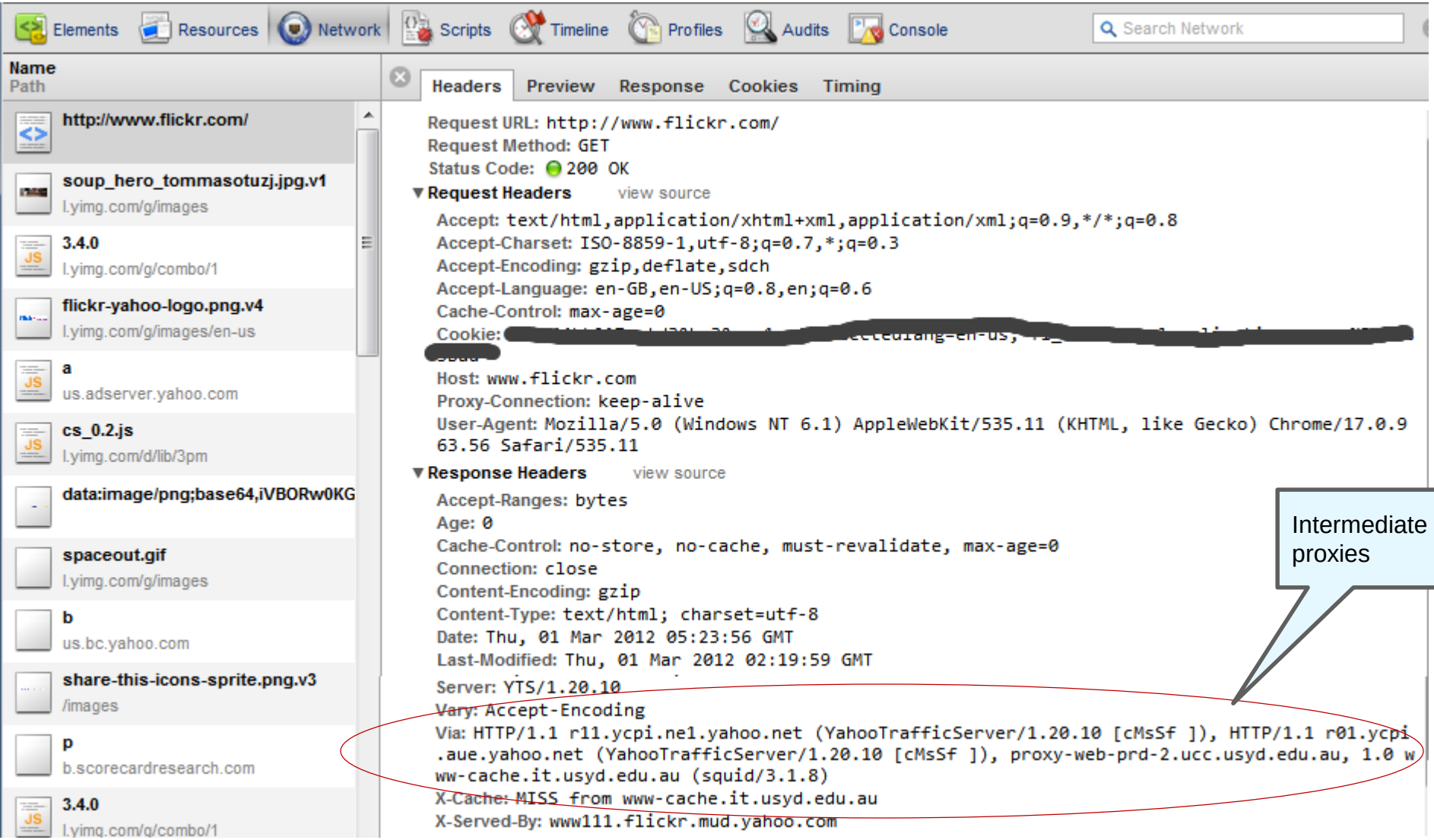
- Web's application layer protocol
  - RFC 2616 (http://www.w3.org/protocols/rfc2616/rfc2616.html)
- Request-Response paradigm
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* Web server sends objects in response to requests



Request-response paradigm

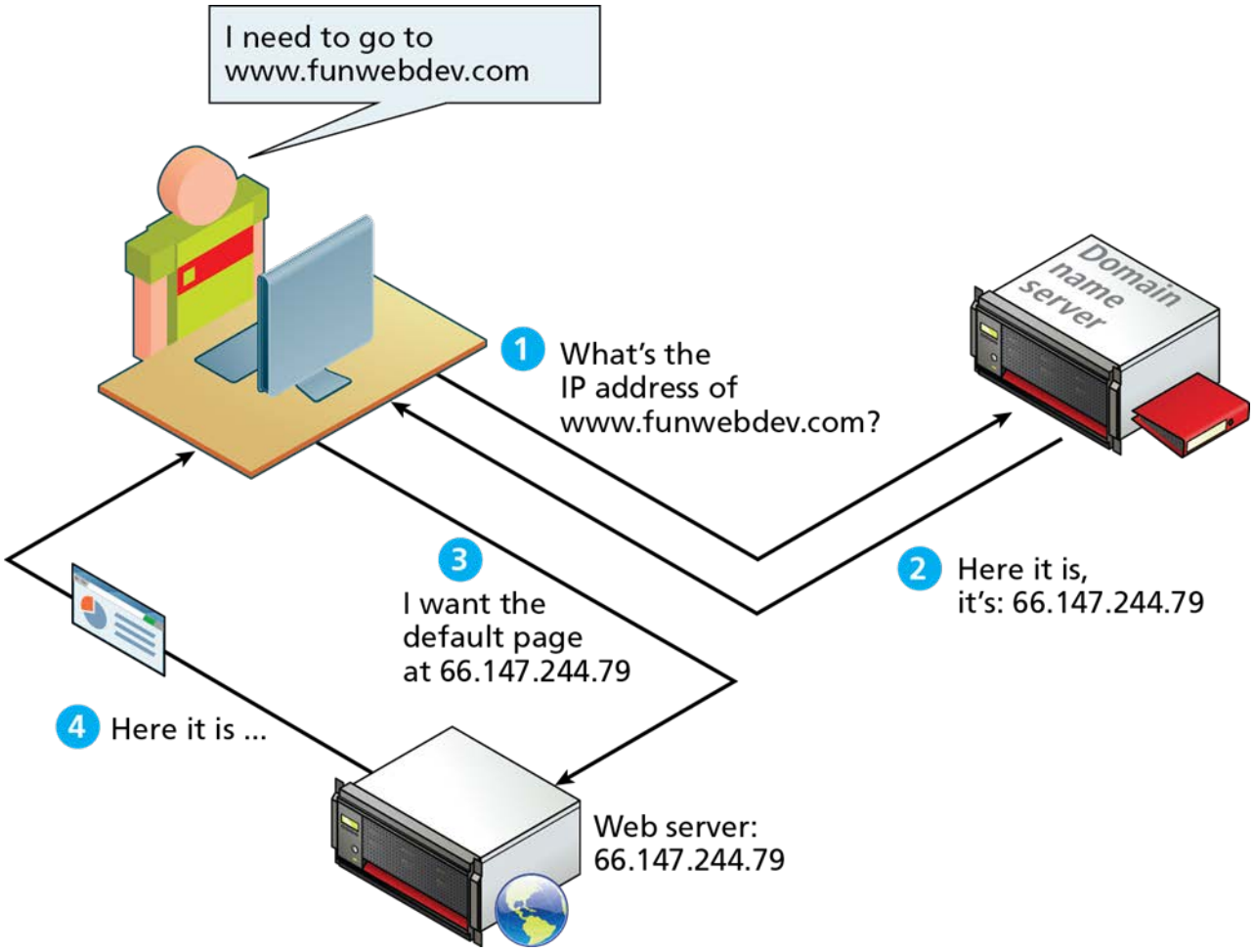


Intermediate proxies

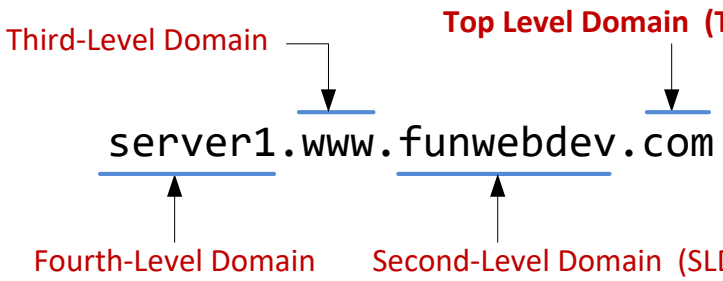


Domain Name System

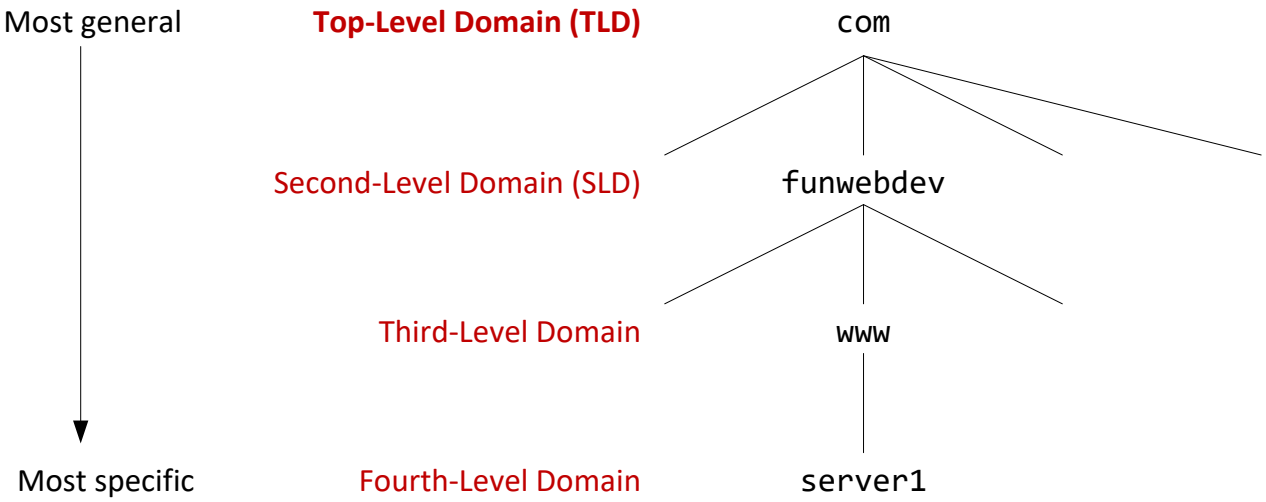
- Instead of IP addresses, we use the **Domain Name System** (**DNS**) to identify hosts on Internet



Domain Levels



Generic top-level domains (gTLD)
Country code top-level domain (ccTLD)



HTTP (cont'd)

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

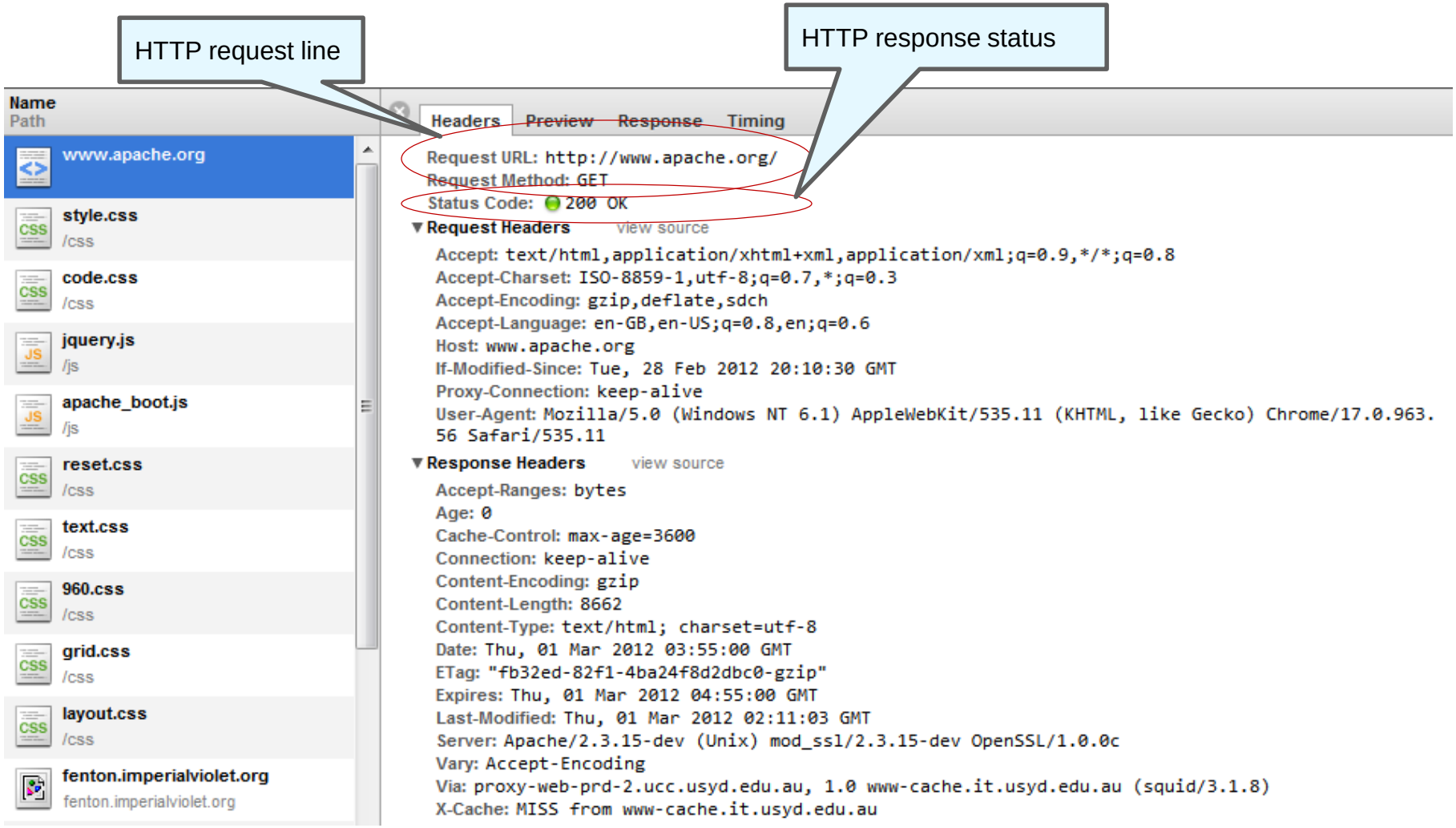
- server maintains no information about past client requests
- In contrast to protocols like FTP, SMTP

aside

Protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

Request/Response message



Request Methods

HTTP/1.0

- GET
- POST
- HEAD
  - asks server to leave requested object out of response

HTTP/1.1

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field
- …

HTTP Response Status Code

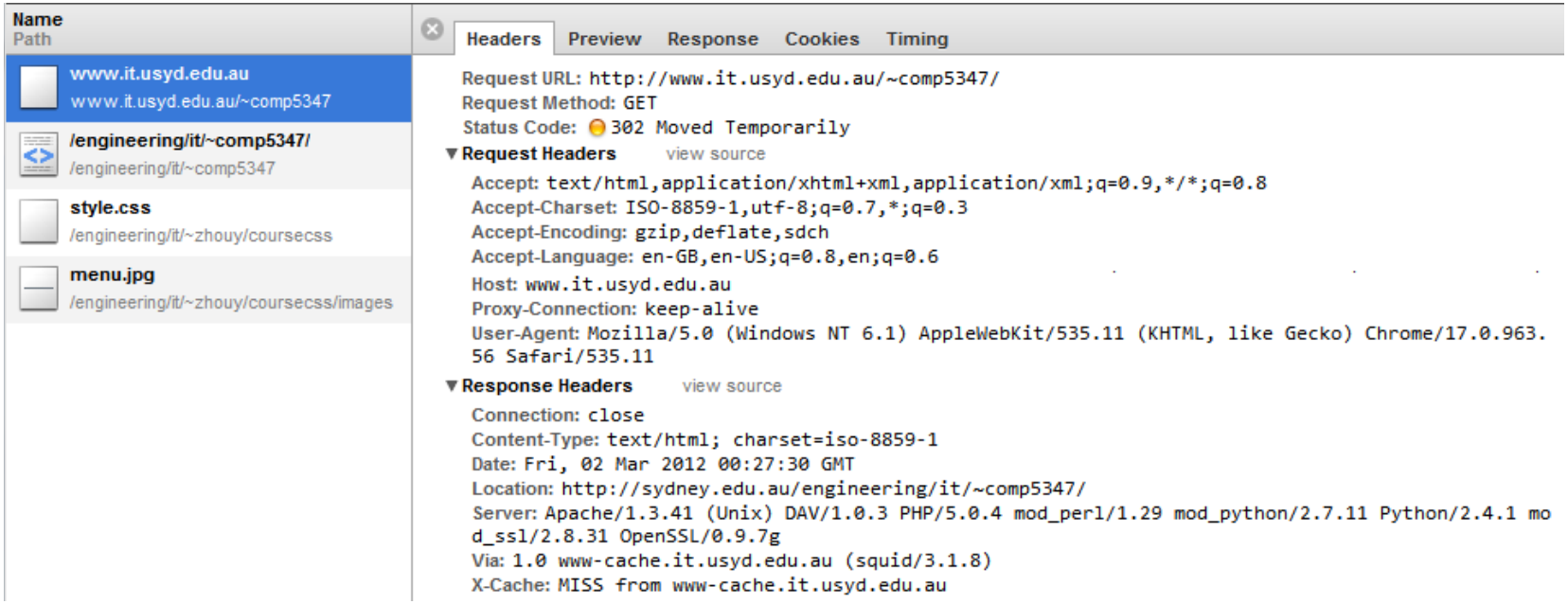
- 3 digit response code
  - 1XX – informational
  - 2XX – success
    - 200 OK
  - 3XX – redirection
    - 301 Moved Permanently
    - 302 Found/Moved temporary
    - 303 Moved Temporarily
    - 304 Not Modified
  - 4XX – client error
    - 404 Not Found
  - 5XX – server error
    - 505 HTTP Version Not Supported

Redirection status code

- 301 Moved Permanently
  - You can easily observe a 301 code by not including the trailing "/". E.g. go to rp-www.it.usyd.edu.au/~comp5347

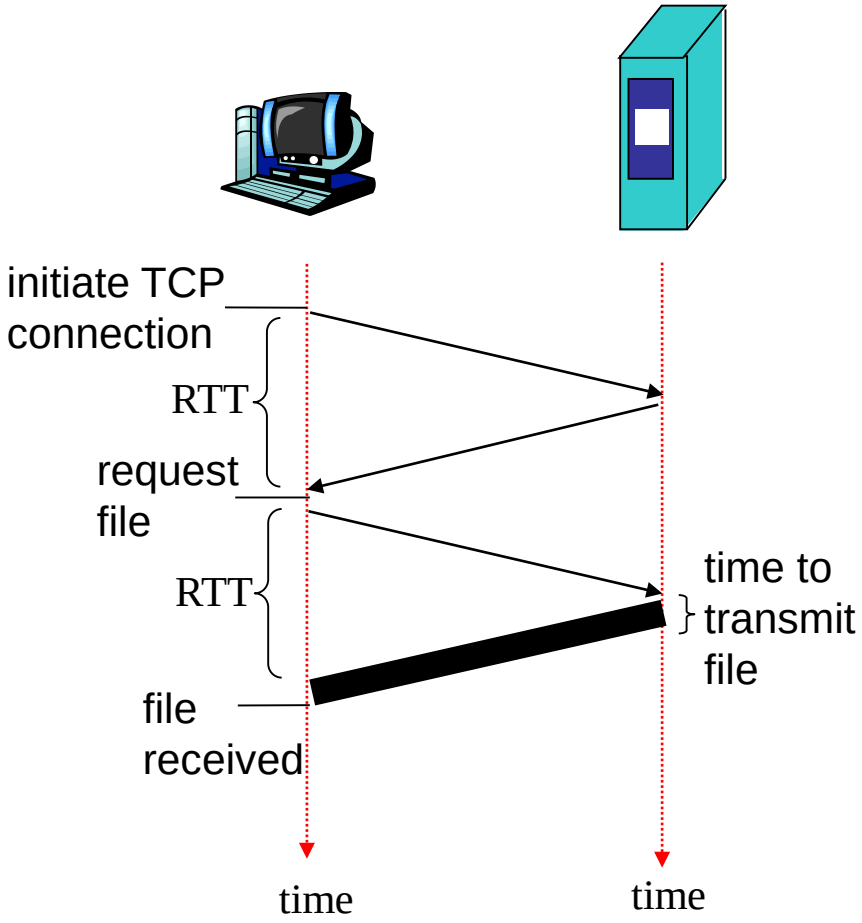
~comp5347 rp-www.it.usyd.edu.au	GET	301 Moved Permanentl	text/html	Other	405B 0B	12ms 11ms
/~comp5347/ /~comp5347	GET	200 OK	text/html	http://rp-www.it.u Redirect	4.47KB 4.04KB	50ms 49ms

- 302 code can be observed as well

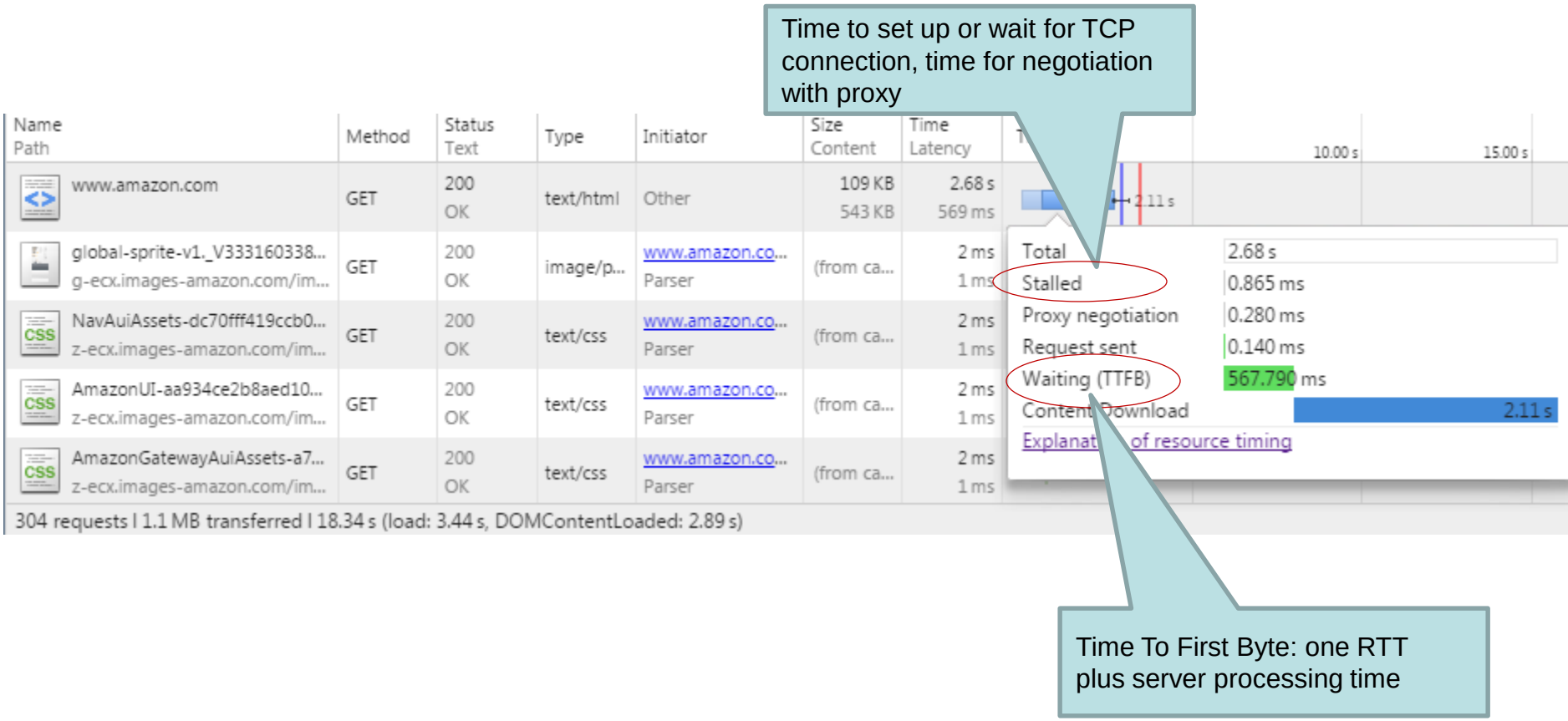


HTTP Performance

- Response Time Modeling
  - Definition of RTT: time to send a small packet to travel from client to server and back.
  - Response time:
  - one RTT to initiate TCP connection
  - one RTT for HTTP request and first few bytes of HTTP response to return
  - file transmission time
  - total = 2RTT+transmit time



HTTP performance



Nonpersistent HTTP v. Persistent HTTP

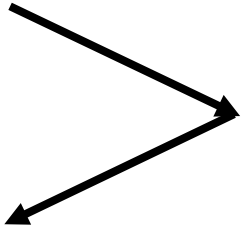
- Nonpersistent HTTP
  - At most one object is sent over a TCP connection.
  - HTTP/1.0 uses nonpersistent HTTP
- Persistent HTTP
  - Multiple objects can be sent over single TCP connection between client and server.
  - HTTP/1.1 uses persistent connections in default mode

Nonpersistent HTTP

- Suppose user enters URL `www.someSchool.edu/someDepartment/home.index`

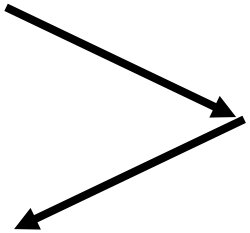
- 1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

(contains text, references to 10 jpeg images)



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

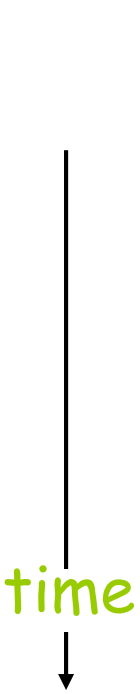
- 2. HTTP client sends HTTP *request message* into TCP connection socket. Message indicates that client wants object someDepartment/home.index



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

time

Nonpersistent HTTP (cont)



- 5. HTTP client receives response message containing html file, displays html. Parsing html file, finds **10** referenced jpeg objects
- 4. HTTP server closes TCP connection.
- 6. Steps 1-5 repeated for each of 10 jpeg objects

Ignoring server processing time, the approximate response time for HTTP/1.0 is 22 RTT + 11 transmit time

The approximate response time for HTTP/1.1 is 12 RTT + 11 transmit time

Caching in HTTP

- Goal of caching in HTTP
  - Eliminate the need to send requests in many cases
    - reduce the number of network round-trips required for many operations
    - an "expiration" mechanism
  - Eliminate the need to send full responses in many other cases.
    - reduce network bandwidth requirements
    - a "validation" mechanism
- Level of caches: server side, client side (proxy and **browser**)
- Cache Correctness
  - It has been checked for equivalence with what the origin server would have returned by revalidating the response with the origin server
  - It is "fresh enough“. In the default case, this means it meets the least restrictive freshness requirement of the client, origin server, and cache
  - It is an appropriate 304 (Not Modified), 305 (Proxy Redirect), or error (4xx or 5xx) response message.

Caching in HTTP (cont)

- Expiration Model
  - Server-Specified Expiration
    - e.g., Cache-Control: no-cache
    - Cache-Control: max-age=60
  - Heuristic Expiration
    - The HTTP/1.1 specification does not provide specific algorithms
- Validation Model
  - When a cache has a stale entry that it would like to use as a response to a client's request, it first has to check with the origin server (or possibly an intermediate cache with a fresh response) to see if its cached entry is still usable.
  - No overhead of re-transmitting the whole response when the entry is valid, but incurs overhead in RTT.

Caching in HTTP (cont)

- Validation Model
  - Last-Modified Dates
  - Entity Tag Cache Validators
    - Entity tags are used for comparing two or more entities from the same requested resource.
  - Requestor side:
    - If-Match
      - If-Match: "xyzzy"
    - If-None-Match
    - If-Modified-Since
      - If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT

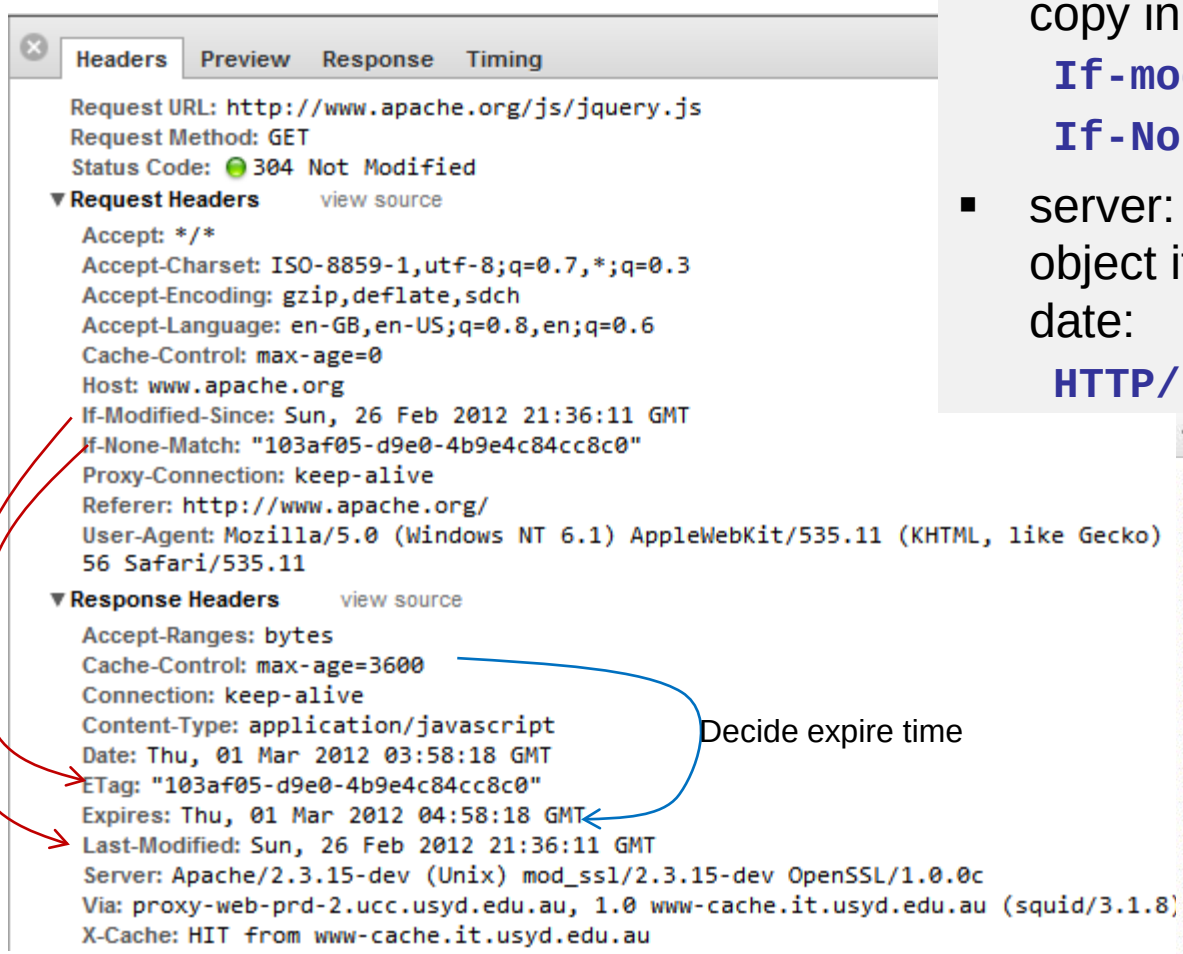
Conditional GET: Browser caching

- **Goal:** don't send object if client has up-to-date cached version
- client: specify date of cached copy in HTTP request

  `If-modified-since: <date>`

  `If-None-Match: <Etag>`
- server: response contains no object if cached copy is up-to-date:

  `HTTP/1.1 304 Not Modified`



References

1. Leon Shklar and Rich Rosen, Web application architecture. 2nd edition, Wiley, Chapter 3&7
2. Randy Connolly and Ricardo Hoar, Fundamentals of Web Development, chapter 1
3. Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati,Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels, ***Dynamo: Amazon's Highly Available Key-Value Store***, In Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP'07), October 2007, Stevenson, Washington, USA.
4. **A Conversation with Werner Vogels,** ACM Queue, May, 2006 [http://queue.acm.org/detail.cfm?id=1142065]
5. **Facebook Architecture: Breaking it Open**, Slide 21 [http://adititechnologiesblog.blogspot.com.au/2012/01/aditi-lead-open-talk-facebook.html]
6. **Aditya Agarwal, Facebook: Science and the Social Graph, Qcon San Francisco 2008 [**http://www.infoq.com/presentations/Facebook-Software-Stack**]**
7. **Abel Avram,** Facebook: MVC Does Not Scale, Use Flux Instead [Updated], May 15, 2014, InfoQ [http://www.infoq.com/news/2014/05/facebook-mvc-flux]