Web Essentials

February 8, 2012

Web Technologies

M.E. (CSE) Semester 2

R S Milton

Department of Computer Science and Engineering

WEB TECHNOLOGIES
A COMPUTER SCIENCE PERSPECTIVE

JEFFREY C. JACKSON

Chapter 1
Web Essentials: Clients, Servers, and Communication

The Internet

- Internet: the network of networks connected via the public backbone and communicating using TCP/IP communication protocol
  - Backbone initially supplied by NSFNET, privately funded (ISP fees) beginning in 1995

Internet Protocols

- Communication protocol: how computers talk
  - Cf. telephone “protocol”: how you answer and end call, what language you speak, etc.
- Internet protocols developed as part of ARPANET research
  - ARPANET began using TCP/IP in 1982
- Designed for use both within local area networks (LAN’s) and between networks

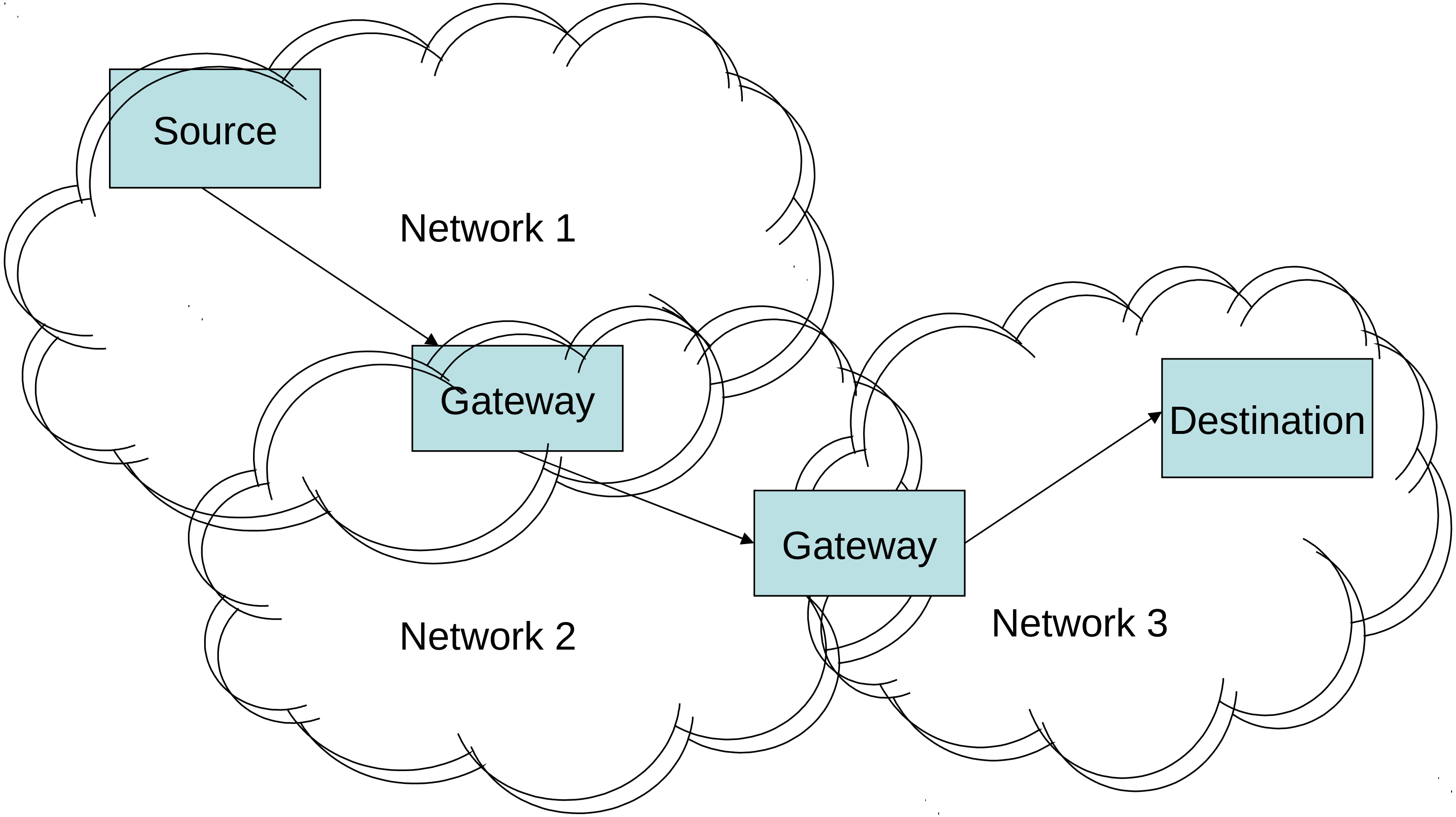
Internet Protocol (IP)

- IP is the fundamental protocol defining the Internet (as the name implies!)
- IP address:
  - 32-bit number (in IPv4)
  - Associated with at most one device at a time (although device may have more than one)
  - Written as four dot-separated bytes, e.g. 192.0.34.166

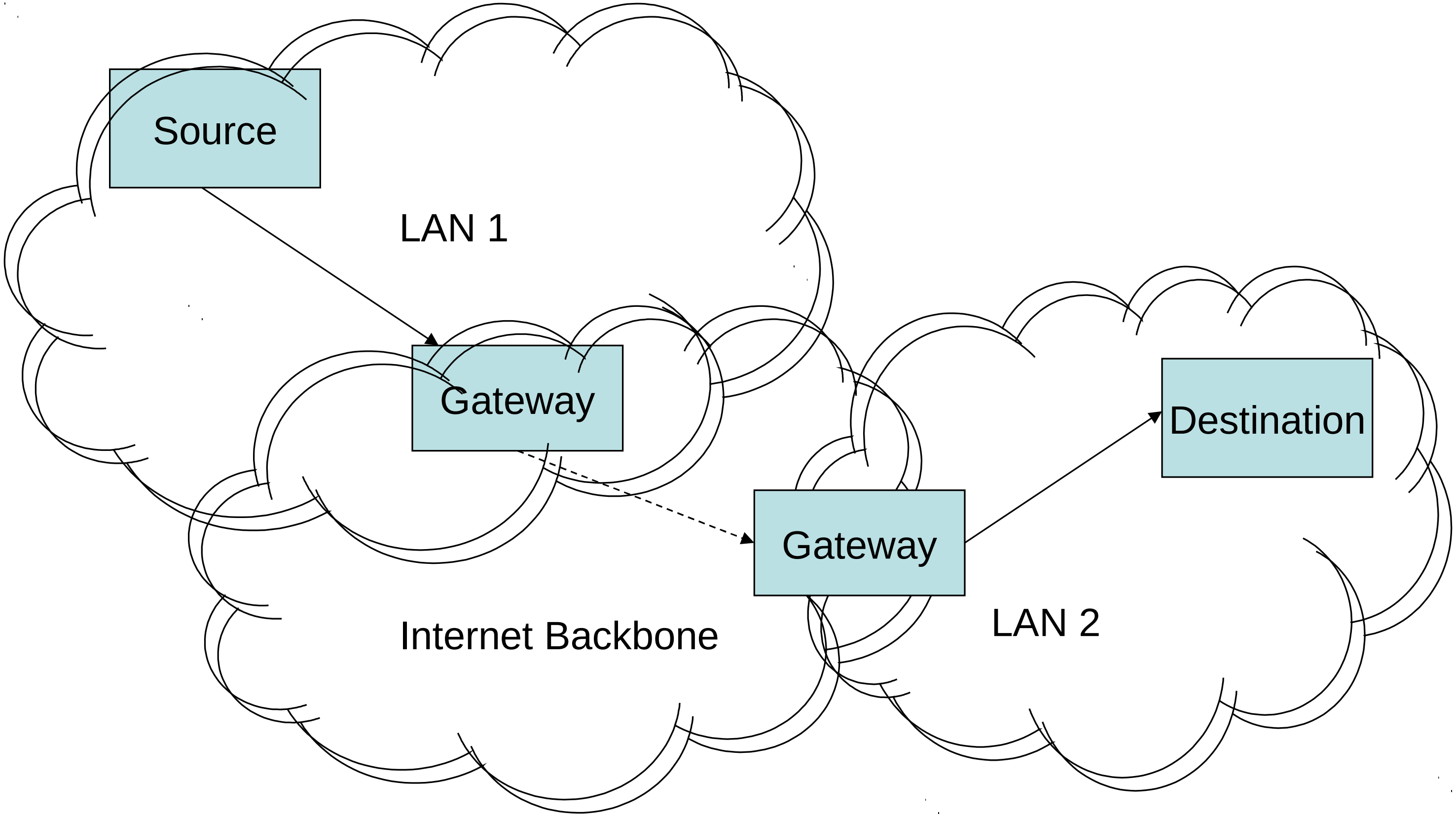
IP

- IP function: transfer data from source device to destination device
- IP source software creates a packet representing the data
  - Header: source and destination IP addresses, length of data, etc.
  - Data itself
- If destination is on another LAN, packet is sent to a gateway that connects to more than one network

IP


Source
Network 1
Gateway
Network 2
Gateway
Destination
Network 3

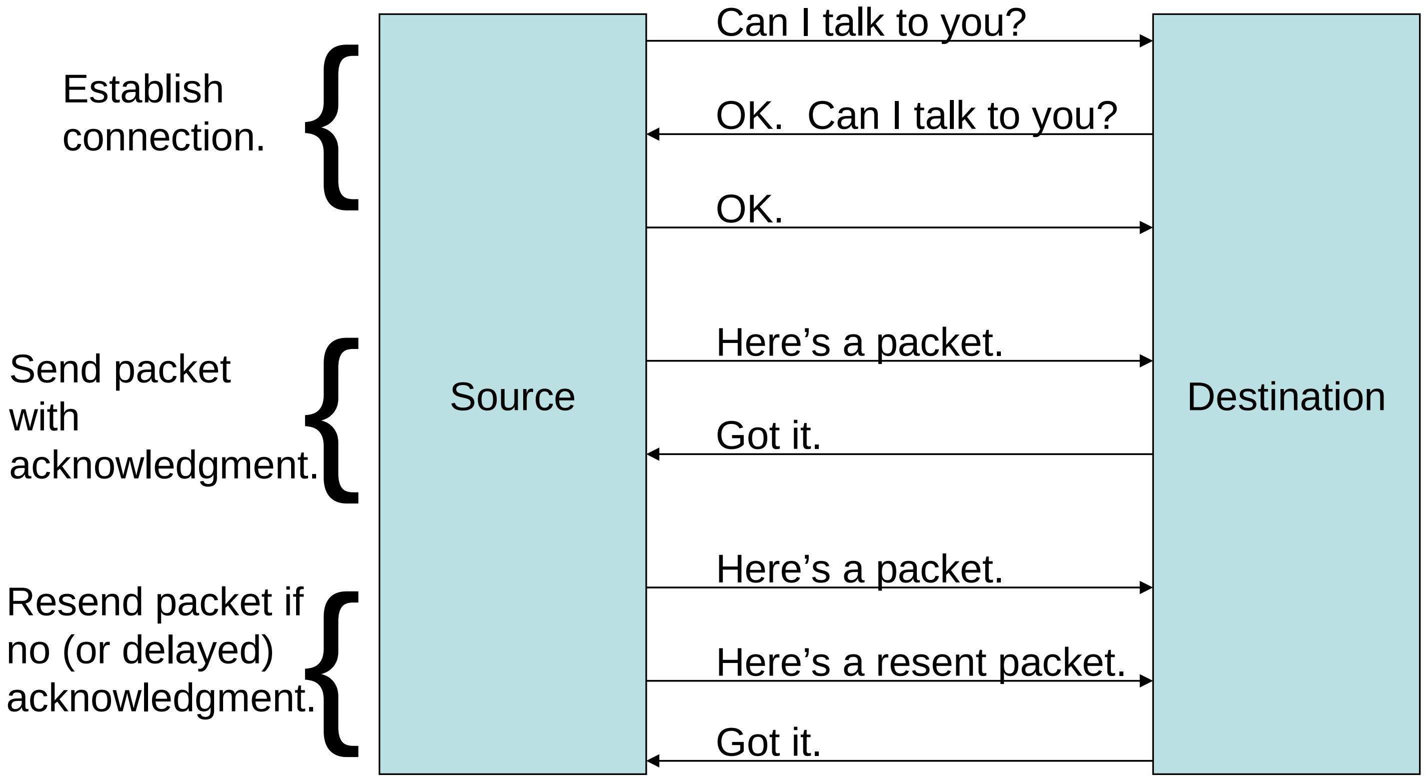
IP


Source
LAN 1
Gateway
Destination
Gateway
Internet Backbone
LAN 2

Transmission Control Protocol (TCP)

- Limitations of IP:
  - No guarantee of packet delivery (packets can be dropped)
  - Communication is one-way (source to destination)
- TCP adds concept of a connection on top of IP
  - Provides guarantee that packets delivered
  - Provide two-way (full duplex) communication

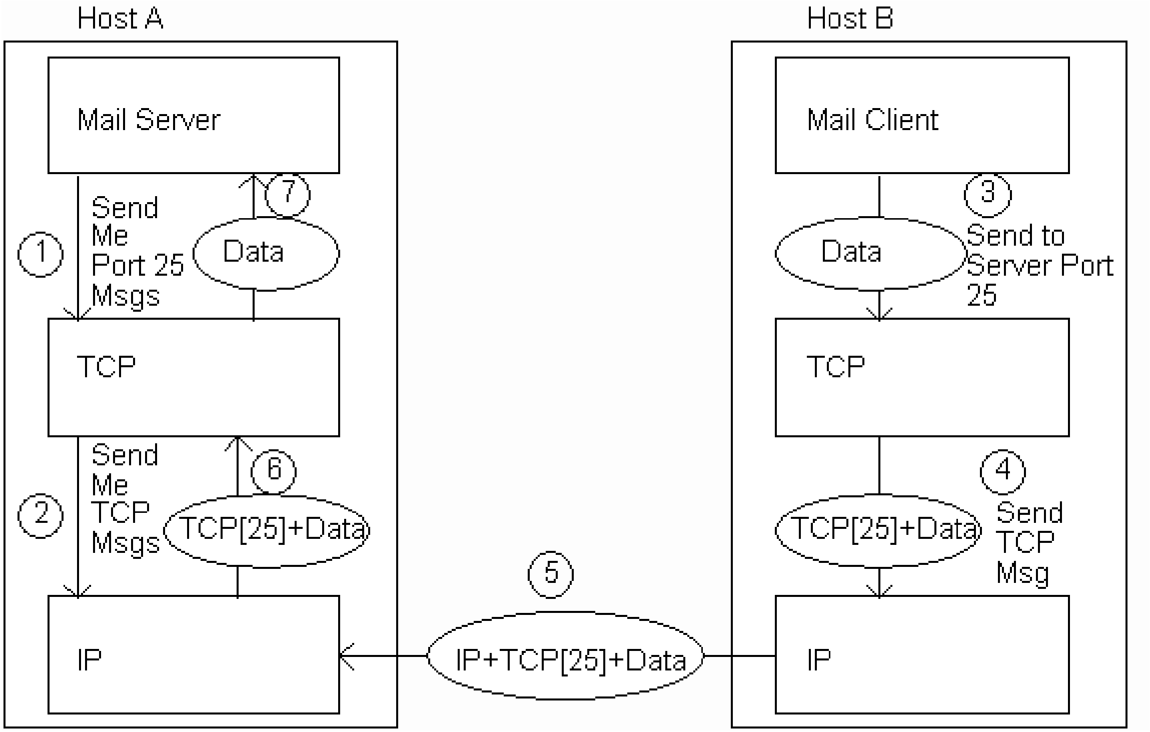
TCP


Establish connection.
Send packet with acknowledgment.
Resend packet if no (or delayed) acknowledgment.
Source
Destination
Can I talk to you?
OK. Can I talk to you?
OK.
Here’s a packet.
Got it.
Here’s a packet.
Here’s a resent packet.
Got it.

TCP

- TCP also adds concept of a port
  - TCP header contains port number representing an application program on the destination computer
  - Some port numbers have standard meanings
    - Example: port 25 is normally used for email transmitted using the Simple Mail Transfer Protocol (SMTP)
  - Other port numbers are available first-come-first served to any application

TCP


Host A
Host B
Mail Server
Mail Client
1
Send Me Port 25 Msgs
7
Data
3
Data
Send to Server Port 25
TCP
TCP
2
Send Me TCP Msgs
6
TCP[25]+Data
4
TCP[25]+Data
Send TCP Msg
5
IP
IP+TCP[25]+Data
IP

User Datagram Protocol (UDP)

- Like TCP in that:
  - Builds on IP
  - Provides port concept
- Unlike TCP in that:
  - No connection concept
  - No transmission guarantee
- Advantage of UDP vs. TCP:
  - Lightweight, so faster for one-time messages

Domain Name Service (DNS)

- DNS is the “phone book” for the Internet
  - Map between host names and IP addresses
  - DNS often uses UDP for communication
- Host names
  - Labels separated by dots, e.g., `www.example.org`
  - Final label is *top-level domain*
    - Generic: .com, .org, etc.
    - Country-code: .us, .il, etc.

DNS

- Domains are divided into second-level domains, which can be further divided into subdomains, etc.
  - E.g., in `www.example.com`, `example` is a second-level domain
- A host name plus domain name information is called the fully qualified domain name of the computer
  - Above, `www` is the host name, `www.example.com` is the FQDN

1. The Internet

- ▷ Communication infrastructure
- ▷ Layered architecture
- ▷ Network layer — IP protocol
- ▷ Transport layer — TCP protocol
- ▷ Application layer — protocol defined by the application

2. Communication protocol

▷ Message format

▷ Sequence of messages exchanged between the sender and the receiver

Higher-level Protocols

- Many protocols build on TCP
  - Telephone analogy: TCP specifies how we initiate and terminate the phone call, but some other protocol specifies how we carry on the actual conversation
- Some examples:
  - SMTP (email)
  - FTP (file transfer)
  - HTTP (transfer of Web documents)

3. Internet and World Wide Web

- ▷ Internet is the communication infrastructure
- ▷ WWW is an information management technology
- ▷ Several other technologies

World Wide Web

- Originally, one of several systems for organizing Internet-based information
  - Competitors: WAIS, Gopher, ARCHIE
- Distinctive feature of Web: support for hypertext (text containing links)
  - Communication via Hypertext Transport Protocol (HTTP)
  - Document representation using Hypertext Markup Language (HTML)

World Wide Web

- The Web is the collection of machines (Web servers) on the Internet that provide information, particularly HTML documents, via HTTP.
- Machines that access information on the Web are known as Web clients. A Web browser is software used by an end user to access the Web.

Hypertext Transport Protocol (HTTP)

- HTTP is based on the request-response communication model:
  - Client sends a request
  - Server sends a response
- HTTP is a stateless protocol:
  - The protocol does not require the server to remember anything about the client between requests.

HTTP

- Normally implemented over a TCP connection (80 is standard port number for HTTP)
- Typical browser-server interaction:
  - User enters Web address in browser
  - Browser uses DNS to locate IP address
  - Browser opens TCP connection to server
  - Browser sends HTTP request over connection
  - Server sends HTTP response to browser over connection
  - Browser displays body of response in the client area of the browser window

HTTP

- The information transmitted using HTTP is often entirely text
- Can use the Internet's Telnet protocol to simulate browser request and view server response

HTTP

Connect

```
$ telnet www.example.org 80
Trying 192.0.34.166...
Connected to www.example.com
(192.0.34.166).
Escape character is ’^]’.
```

Send Request

```
GET / HTTP/1.1
Host: www.example.org

```

Receive Response

```
HTTP/1.1 200 OK
Date: Thu, 09 Oct 2003 20:30:49 GMT
…
```

HTTP Request

- Structure of the request:
  - start line
  - header field(s)
  - blank line
  - optional body

HTTP Request

- Structure of the request:
  - **start line**
  - header field(s)
  - blank line
  - optional body

HTTP Request

- Start line
  - Example: `GET / HTTP/1.1`
- Three space-separated parts:
  - HTTP request method
  - Request-URI
  - HTTP version

HTTP Request

- Start line
  - Example: `GET / HTTP/1.1`
- Three space-separated parts:
  - HTTP request method
  - Request-URI
  - **HTTP version**
    - We will cover 1.1, in which version part of start line must be exactly as shown

HTTP Request

- Start line
  - Example: `GET / HTTP/1.1`
- Three space-separated parts:
  - HTTP request method
  - **Request-URI**
  - HTTP version

HTTP Request

- Uniform Resource Identifier (URI)
  - Syntax: `*scheme : scheme-depend-part*`
    - Ex: In `http://www.example.com/` the scheme is `http`
  - Request-URI is the portion of the requested URI that follows the host name (which is supplied by the required Host header field)
    - Ex: `/` is Request-URI portion of `http://www.example.com/`

URI

- URI’s are of two types:
  - – Uniform Resource Name (URN)
    - Can be used to identify resources with unique names, such as books (which have unique ISBN’s)
    - Scheme is `urn`
  - – Uniform Resource Locator (URL)
    - Specifies location at which a resource can be found
    - In addition to `http`, some other URL schemes are `https`, `ftp`, `mailto`, and `file`

HTTP Request

- Start line
  - Example: `GET / HTTP/1.1`
- Three space-separated parts:
  - **HTTP request method**
  - Request-URI
  - HTTP version

HTTP Request

- Common request methods:
  - GET
    - Used if link is clicked or address typed in browser
    - No body in request with GET method
  - POST
    - Used when submit button is clicked on a form
    - Form information contained in body of request
  - HEAD
    - Requests that only header fields (no body) be returned in the response

HTTP Request

- Structure of the request:
  - start line
  - **header field(s)**
  - blank line
  - optional body

HTTP Request

- Header field structure:
  - *`field name : field value`*
- Syntax
  - Field name is not case sensitive
  - Field value may continue on multiple lines by starting continuation lines with white space
  - Field values may contain MIME types, quality values, and wildcard characters (*'s)

Multipurpose Internet Mail Extensions (MIME)

- Convention for specifying content type of a message
  - In HTTP, typically used to specify content type of the body of the response
- MIME content type syntax:
  - *`top-level type / subtype`*
- Examples: text/html, image/jpeg

HTTP Quality Values and Wildcards

- Example header field with quality values:

```
accept:
  text/xml,text/html;q=0.9,
  text/plain;q=0.8, image/jpeg,
  image/gif;q=0.2,*/*;q=0.1
```

- Quality value applies to all preceding items
- Higher the value, higher the preference
- Note use of wildcards to specify quality 0.1 for any MIME type not specified earlier

HTTP Request

- Common header fields:
  - Host: host name from URL (required)
  - User-Agent: type of browser sending request
  - Accept: MIME types of acceptable documents
  - Connection: value `close` tells server to close connection after single request/response
  - Content-Type: MIME type of (POST) body, normally application/x-www-form-urlencoded
  - Content-Length: bytes in body
  - Referer: URL of document containing link that supplied URI for this HTTP request

HTTP Response

- Structure of the response:
  - status line
  - header field(s)
  - blank line
  - optional body

HTTP Response

- Structure of the response:
  - **status line**
  - header field(s)
  - blank line
  - optional body

HTTP Response

- Status line
  - – Example: HTTP/1.1 200 OK
- Three space-separated parts:
  - – HTTP version
  - – status code
  - – reason phrase (intended for human use)

HTTP Response

- Status code
  - Three-digit number
  - First digit is class of the status code:
    - 1=Informational
    - 2=Success
    - 3=Redirection (alternate URL is supplied)
    - 4=Client Error
    - 5=Server Error
  - Other two digits provide additional information

HTTP Response

- Structure of the response:
  - status line
  - **header field(s)**
  - blank line
  - optional body

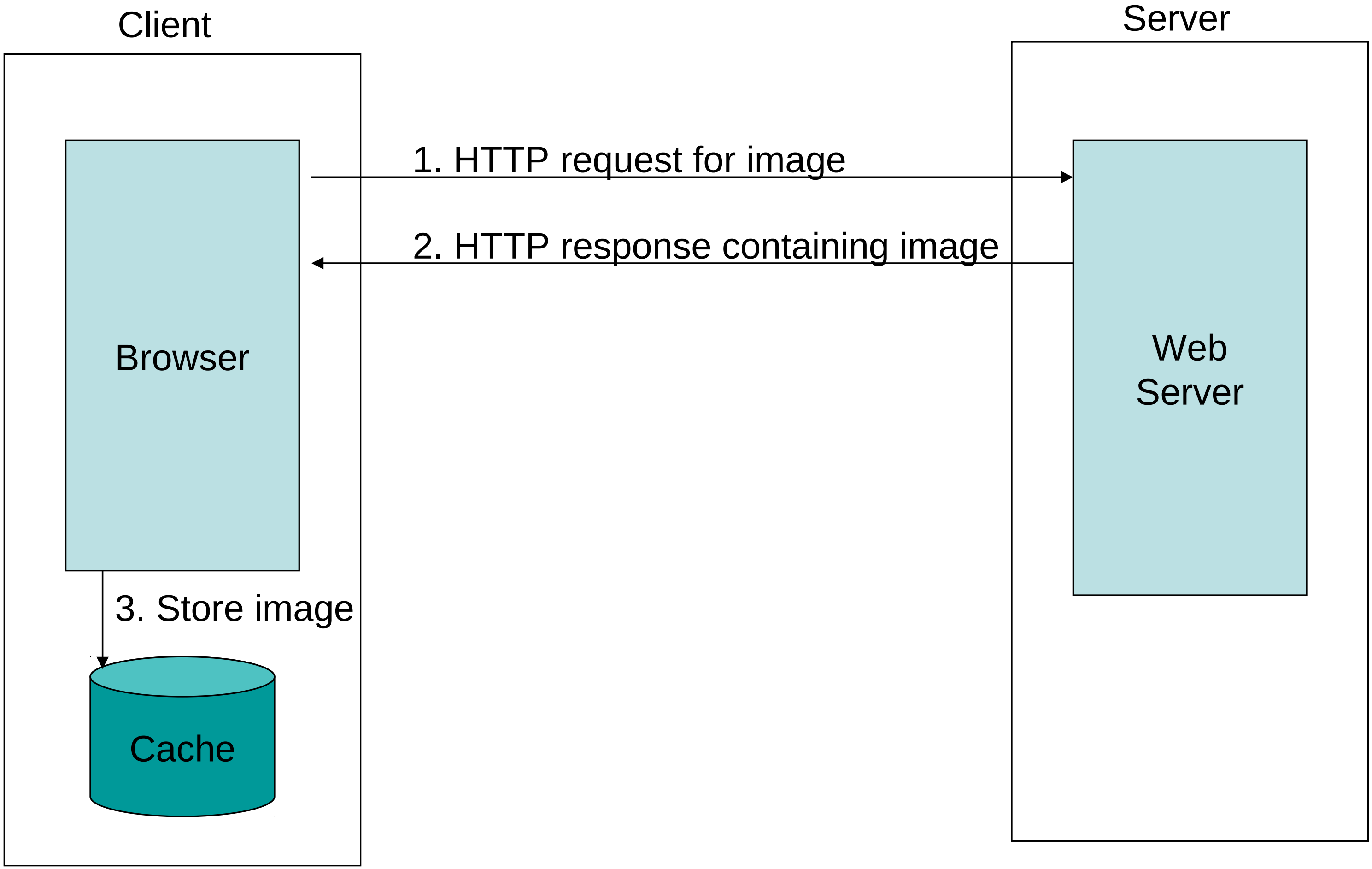
HTTP Response

- Common header fields:
  - Connection, Content-Type, Content-Length
  - Date: date and time at which response was generated (required)
  - Location: alternate URI if status is redirection
  - Last-Modified: date and time the requested resource was last modified on the server
  - Expires: date and time after which the client’s copy of the resource will be out-of-date
  - ETag: a unique identifier for this version of the requested resource (changes if resource changes)

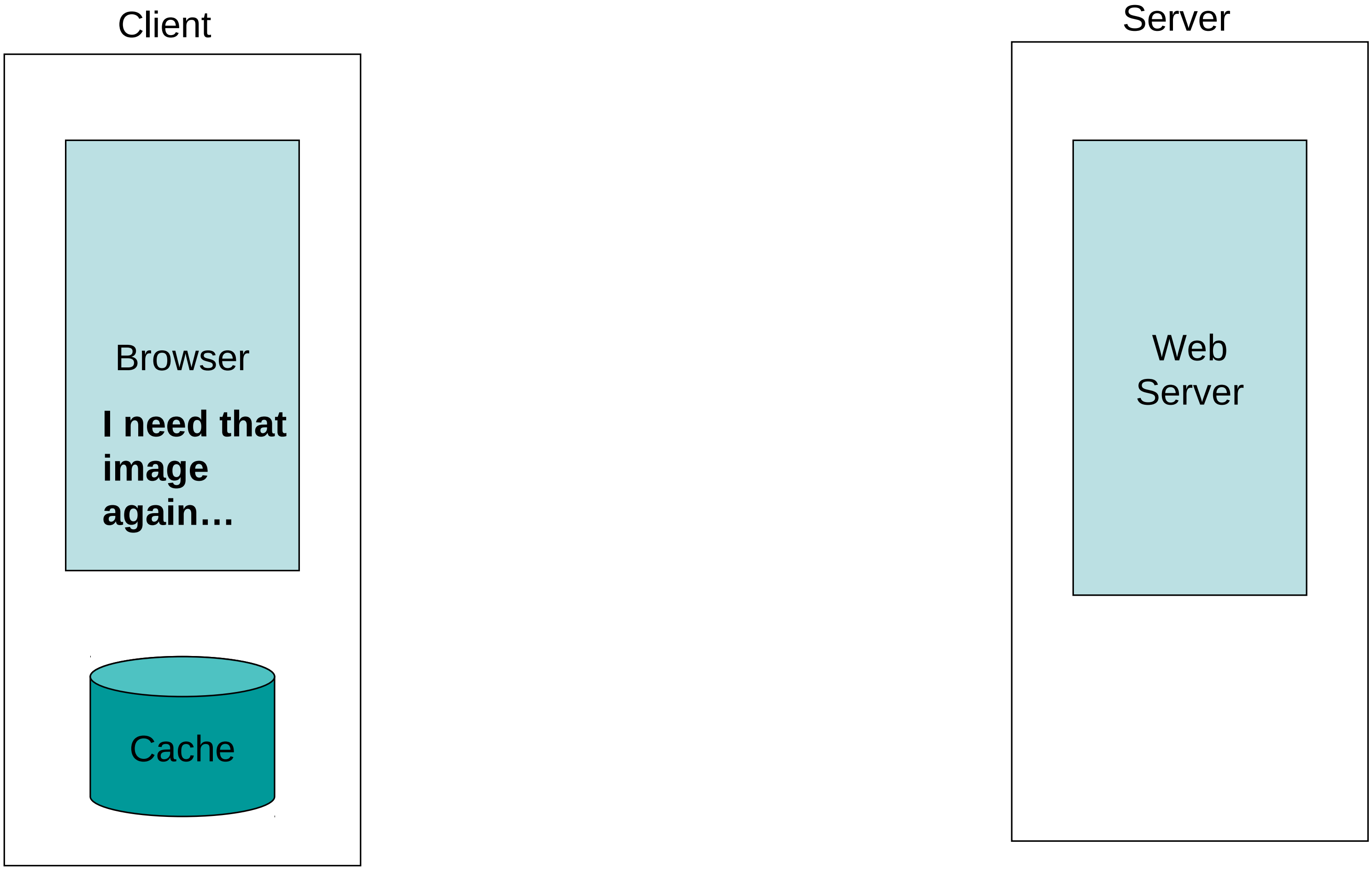
Client Caching

- A cache is a local copy of information obtained from some other source
- Most web browsers use cache to store requested resources so that subsequent requests to the same resource will not necessarily require an HTTP request/response
  - Ex: icon appearing multiple times in a Web page

Client Caching



Client Caching



Client

Browser

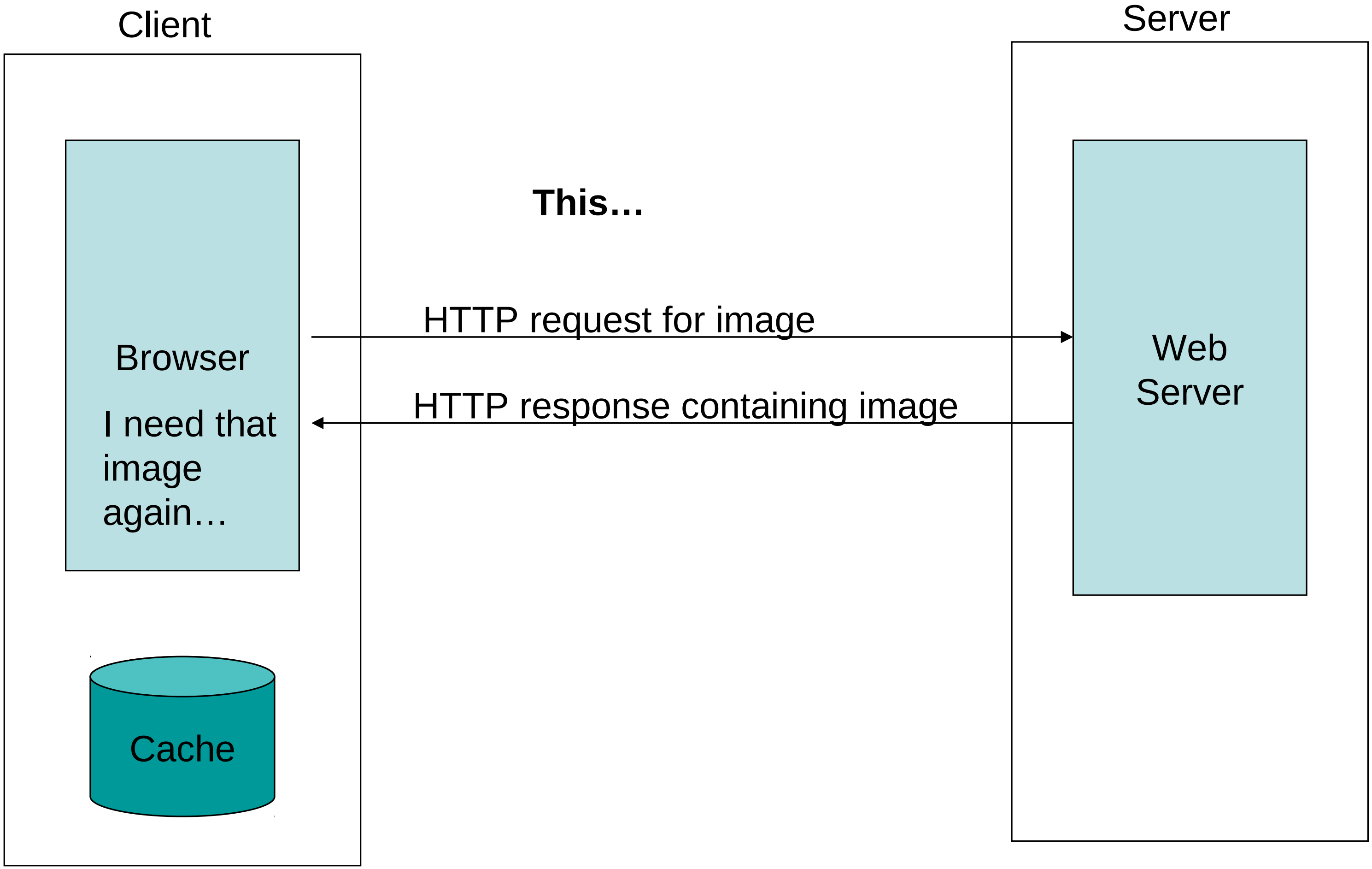
I need that image again…

Cache

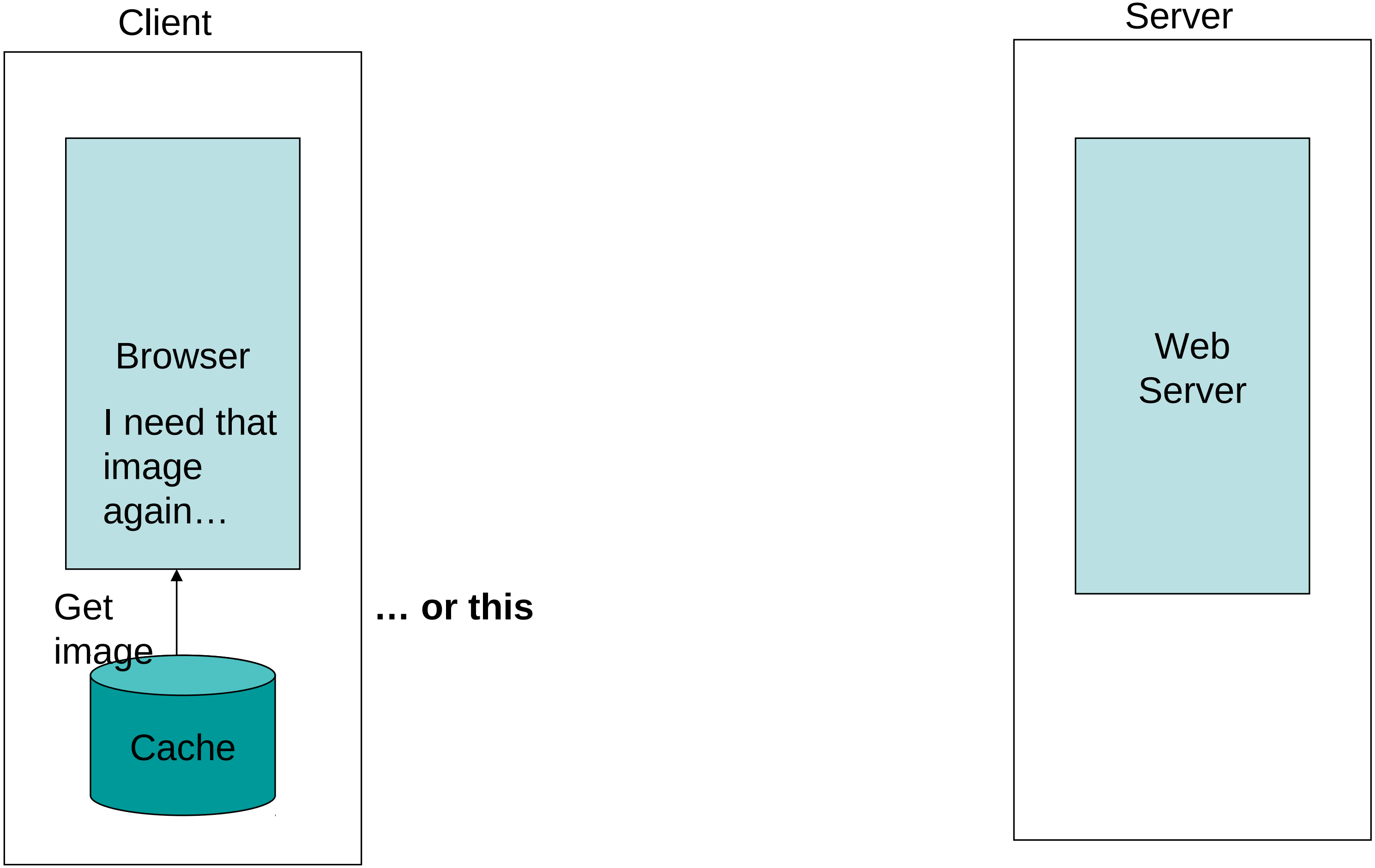
Server

Web Server

Client Caching



Client Caching



Client Caching

- Cache advantages
  - (Much) faster than HTTP request/response
  - Less network traffic
  - Less load on server
- Cache disadvantage
  - Cached copy of resource may be invalid (inconsistent with remote version)

Client Caching

- Validating cached resource:
  - Send HTTP HEAD request and check Last-Modified or ETag header in response
  - Compare current date/time with Expires header sent in response containing resource
  - If no Expires header was sent, use heuristic algorithm to estimate value for Expires
    - Ex: Expires = 0.01 * (Date – Last-Modified) + Date