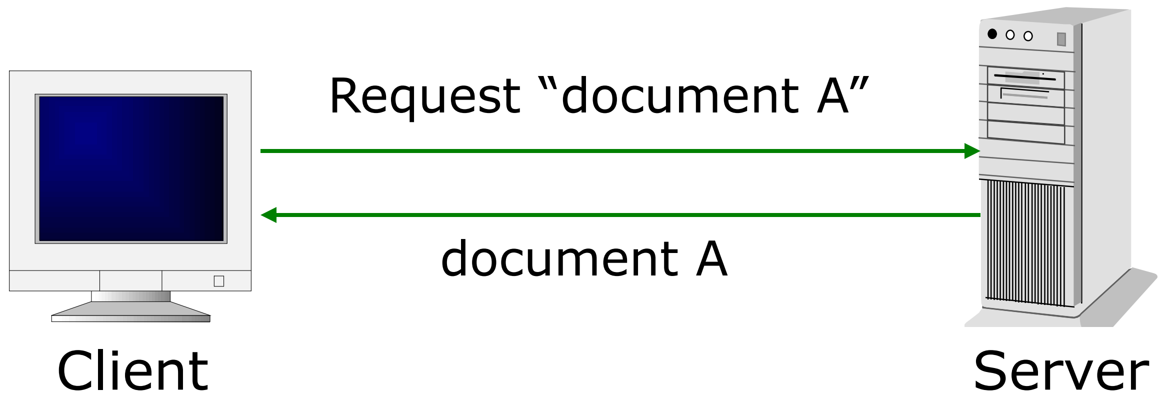
Web Essentials

- **Client**: web browsers, used to surf the Web
- **Server** systems: used to supply information to these browsers
- Computer **networks**: used to support the browser-server communication



Internet v.s. Web

- **The Internet**: a inter-connected computer networks, linked by wires, cables, wireless connections, etc.
- **Web**: a collection of interconnected documents and other resources.
- The world wide web (**WWW**) is accessible via the Internet, as are many other services including email, file sharing, etc.

How does the Internet Work?

- Through communication protocols
- A **communication protocol** is a specification of how communication between two computers will be carried out
  - **IP** (Internet Protocol): defines the packets that carry blocks of data from one node to another
  - **TCP** (Transmission Control Protocol) and **UDP** (User Datagram Protocol): the protocols by which one host sends data to another.
  - Other application protocols: **DNS** (Domain Name Service), **SMTP** (Simple Mail Transmission Protocol), and **FTP** (File Transmission Protocol)

The Internet Protocol (IP)

- A key element of IP is IP address, a 32-bit number
- The Internet authorities assign ranges of numbers to different organizations
- IP is responsible for moving packet of data from node to node
- A packet contains information such as the data to be transferred, the source and destination IP addresses, etc.
- Packets are sent through different local network through gateways
- A checksum is created to ensure the correctness of the data; corrupted packets are discarded
- IP-based communication is unreliable

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

The Transmission Control Protocol (TCP)

- TCP is a higher-level protocol that extends IP to provide additional functionality: reliable communication
- TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received
- Connection
- Acknowledgment

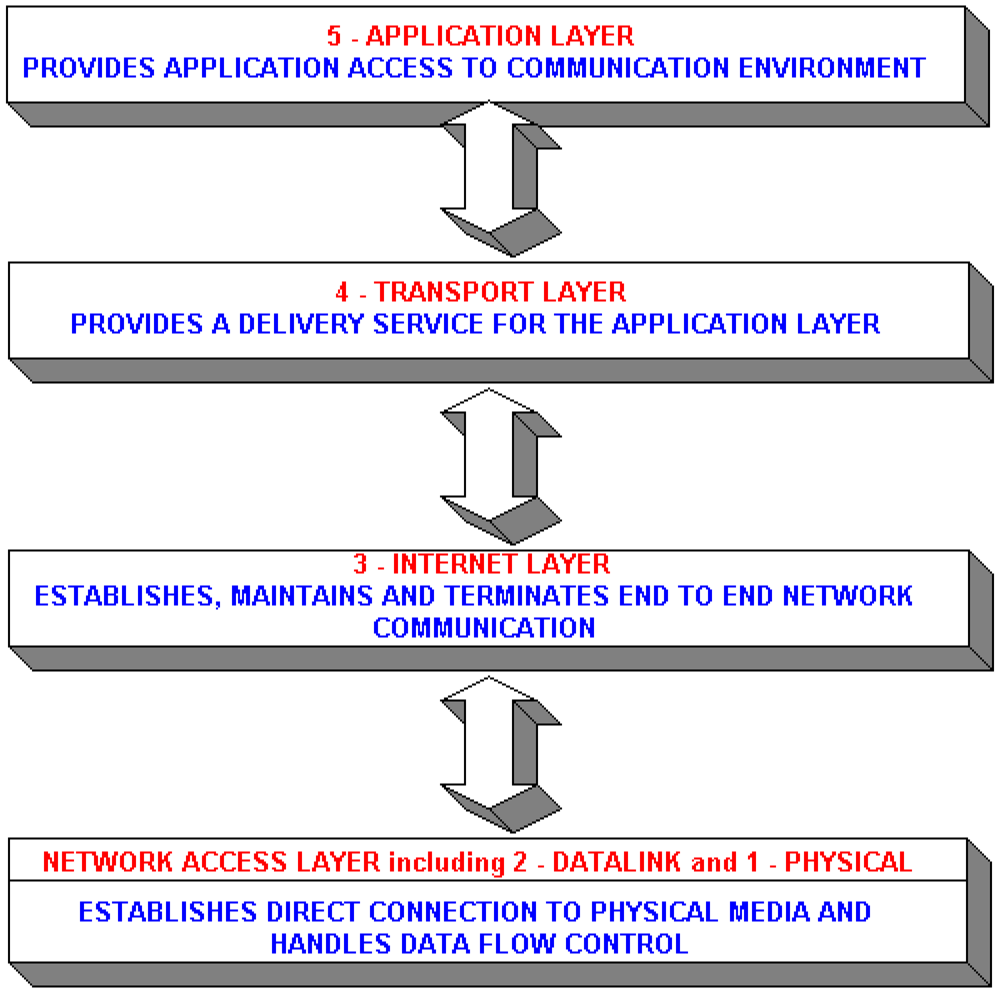
User Datagram Protocol (UDP)

- 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.

TCP/IP Protocol Suites


5 - APPLICATION LAYER
PROVIDES APPLICATION ACCESS TO COMMUNICATION ENVIRONMENT
4 - TRANSPORT LAYER
PROVIDES A DELIVERY SERVICE FOR THE APPLICATION LAYER
3 - INTERNET LAYER
ESTABLISHES, MAINTAINS AND TERMINATES END TO END NETWORK COMMUNICATION
NETWORK ACCESS LAYER including 2 - DATALINK and 1 - PHYSICAL
ESTABLISHES DIRECT CONNECTION TO PHYSICAL MEDIA AND HANDLES DATA FLOW CONTROL

HTTP, FTP, Telnet, DNS, SMTP, etc.

TCP, UDP

IP (IPv4, IPv6)

The World Wide Web (WWW)

- **WWW** is a system of interlinked, hypertext documents that runs over the Internet
- Two types of software:
  - **Client**: a system that wishes to access the information provided by servers must run client software (e.g., web browser)
  - **Server**: an internet-connected computer that wishes to provide information to others must run server software
  - Client and server applications communicate over the Internet by following a protocol built on top of TCP/IP – HyperText Transport Protocol (**HTTP**)

Basics of the WWW

- **Hypertext**: a format of information which allows one to move from one part of a document to another or from one document to another through **hyperlinks**
- Uniform Resource Locator (**URL**): unique identifiers used to locate a particular resource on the network
- **Markup language**: defines the structure and content of hypertext documents

Web Client: Browser

- Makes HTTP requests on behalf of the user
  - Reformat the URL entered as a valid HTTP request
  - Use DNS to convert server’s host name to appropriate IP address
  - Establish a TCP connection using the IP address
  - Send HTTP request over the connection and wait for server’s response
  - Display the document contained in the response
    - If the document is not a plain-text document but instead is written in HTML, this involves rendering the document (positioning text, graphics, creating table borders, using appropriate fonts, etc.)

Web Servers

- Main functionalities:
  - Server waits for connect requests
  - When a connection request is received, the server creates a new process to handle this connection
  - The new process establishes the TCP connection and waits for HTTP requests
  - The new process invokes software that maps the requested URL to a resource on the server
  - If the resource is a file, creates an HTTP response that contains the file in the body of the response message
  - If the resource is a program, runs the program, and returns the output

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       { GET / HTTP/1.1
Request      Host: www.example.org

Receive    { HTTP/1.1 200 OK
Response     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

- 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/`

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

- 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

- 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

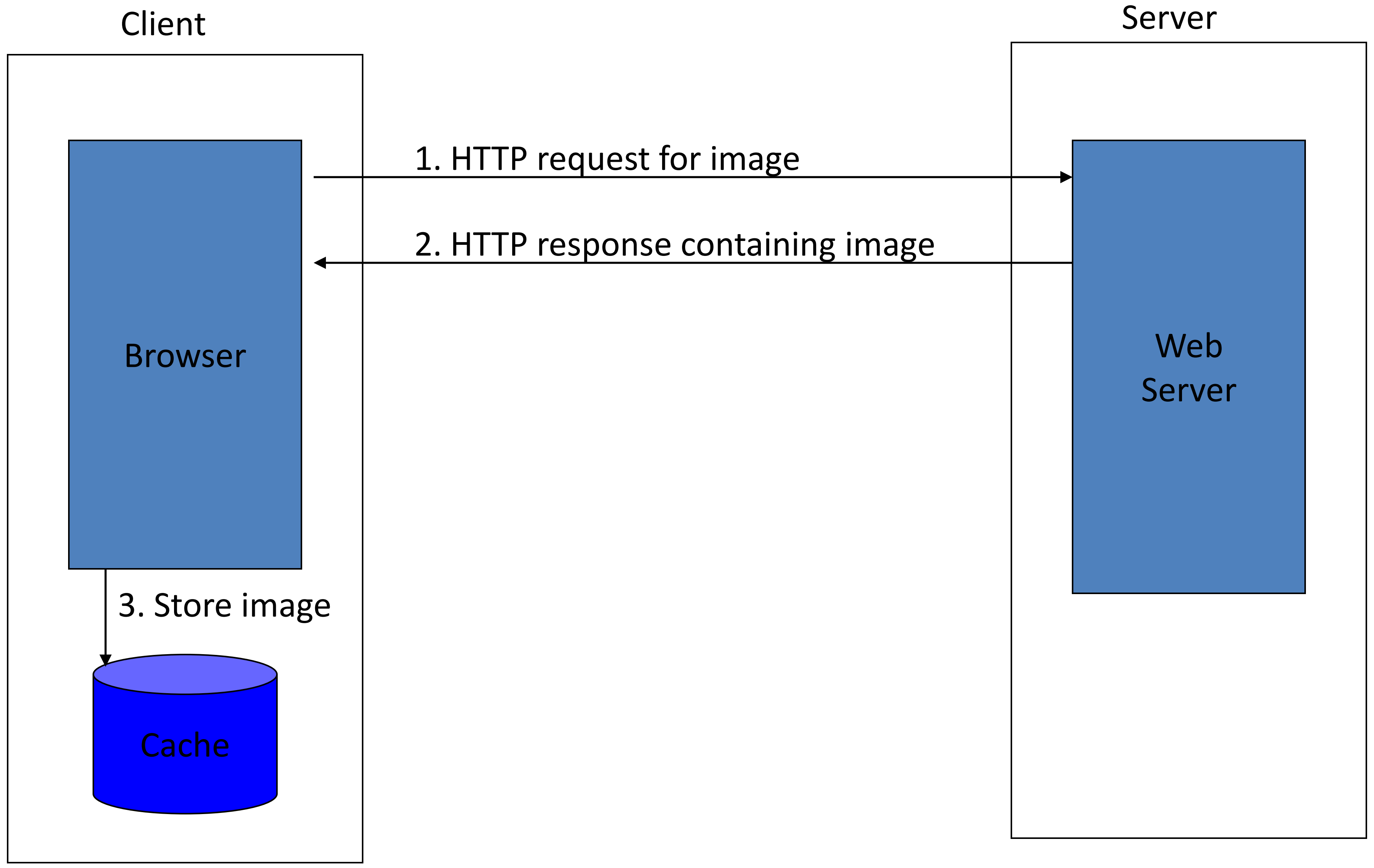
HTTP Response

- 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)

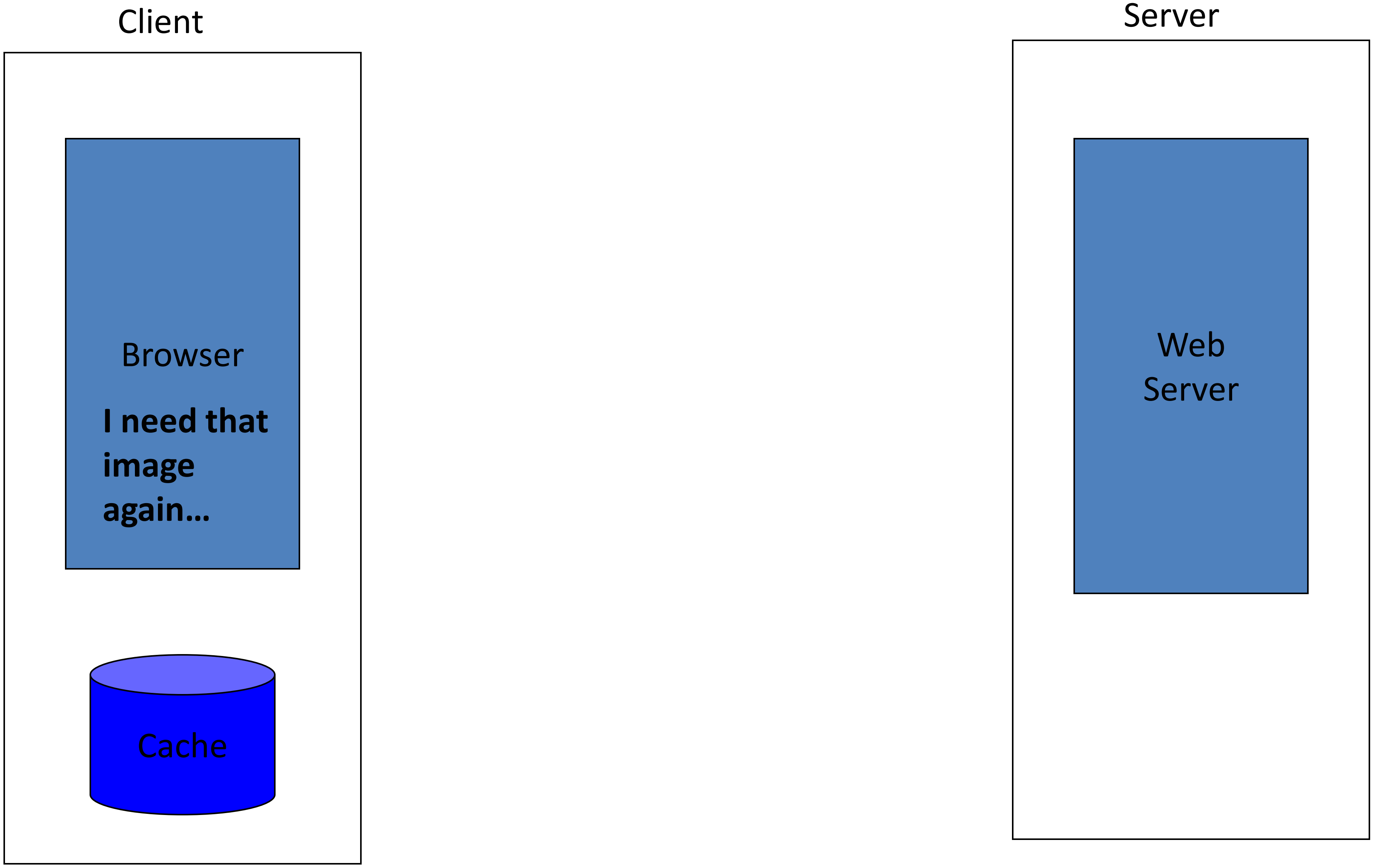
Client Caching

- 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

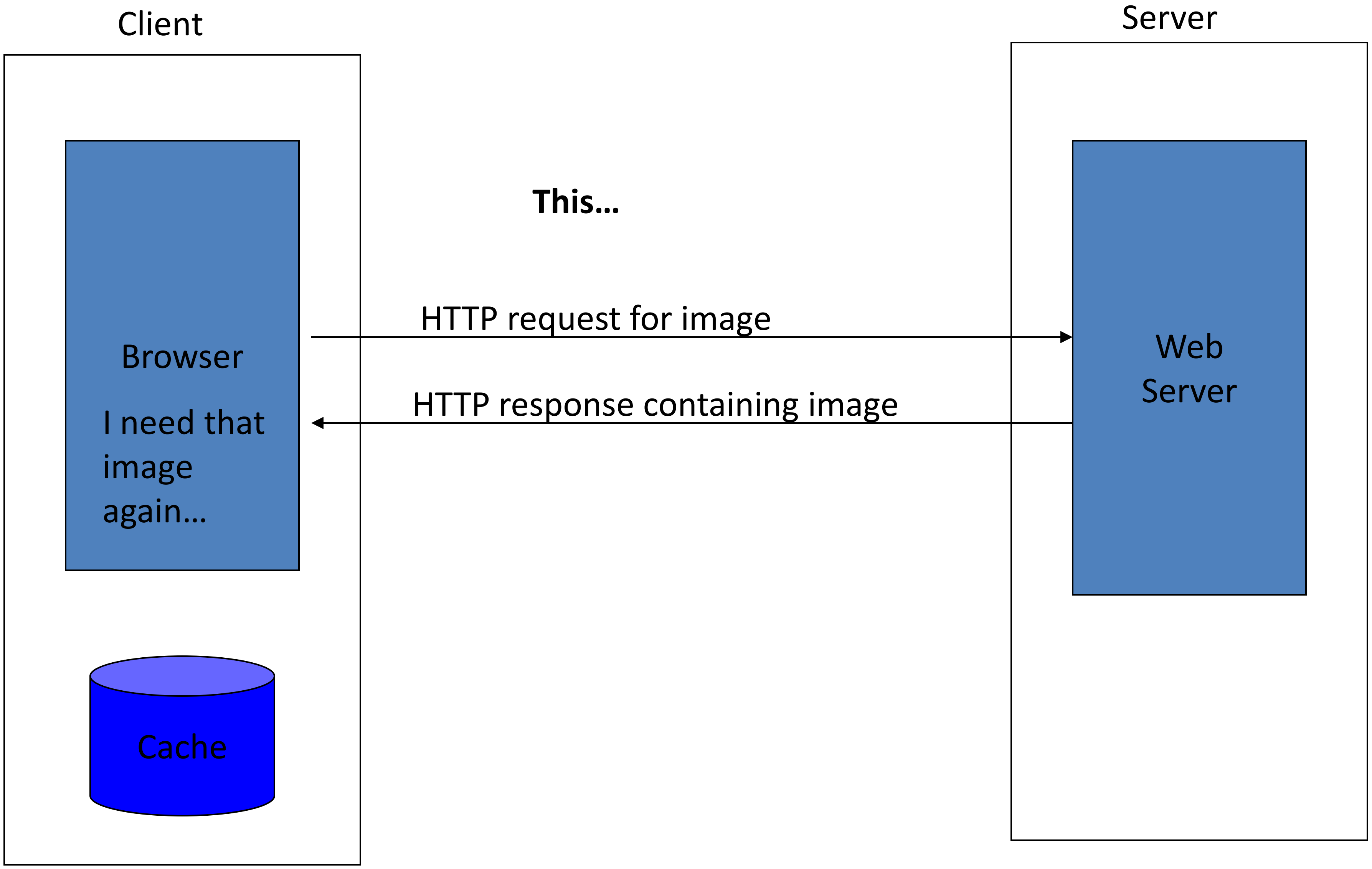
Client Caching



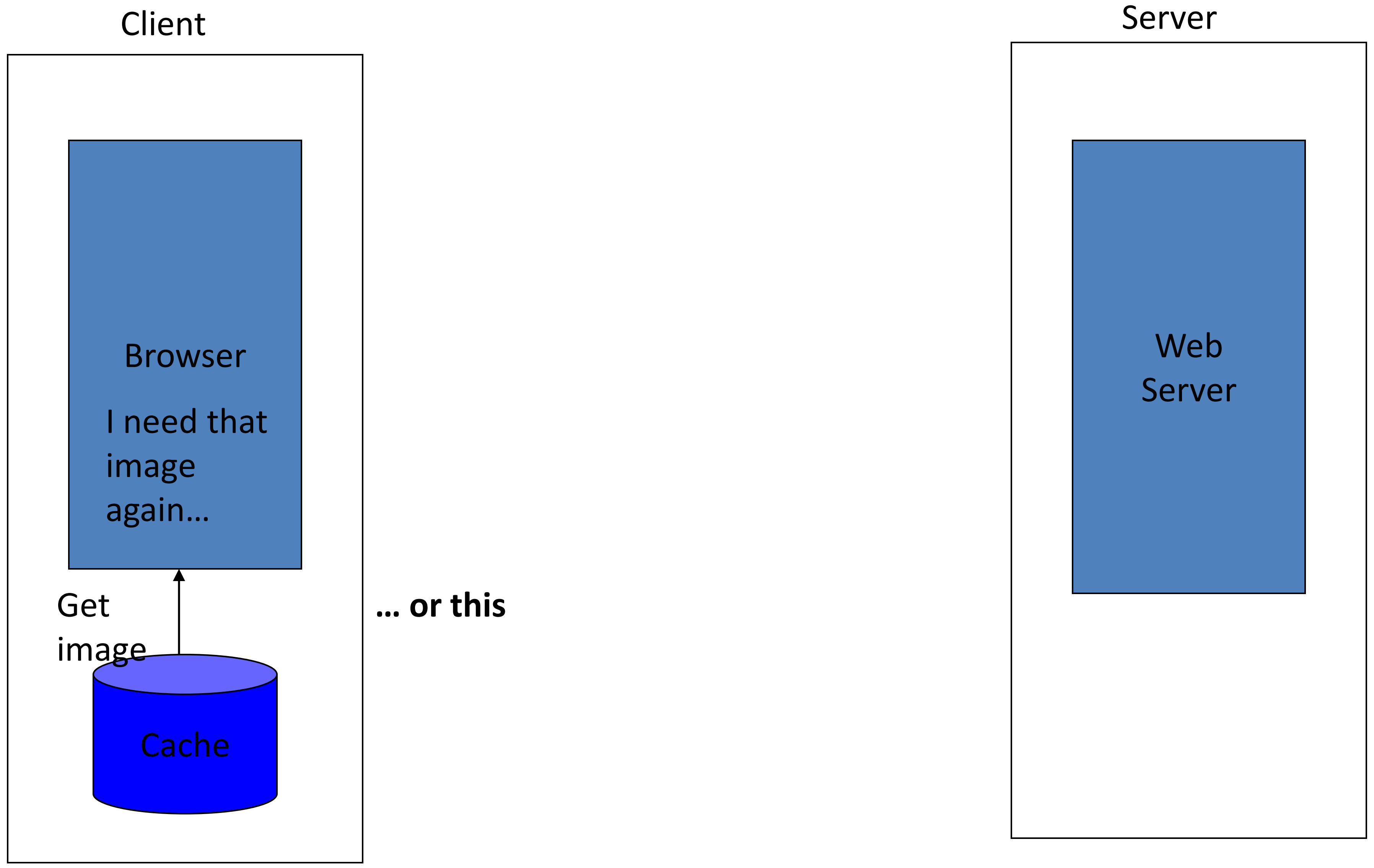
Client Caching



Client 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)

Static Web: HTML/XHTML, CSS

- **HTML** stands for **H**yper**T**ext **M**arkup **L**anguage
  - It is a text file containing small markup tags that tell the Web browser how to display the page
- **XHTML** stands for e**X**tensible **H**yper**T**ext **M**arkup **L**anguage
  - It is identical to HTML 4.01
  - It is a stricter and cleaner version of HTML
- **CSS** stands for **C**ascading **S**tyle **S**heets
  - It defines how to display HTML elements

Why Programmability?

- What’s the drawback to simple document model?
  - Static
  - Assume that documents are created before they are requested

- What are examples of information that might be part of web documents that may not be known before they are requested?

Client-Side Programmability

- Scripting language: a lightweight programming language
- Browser scripting: **JavaScript**
  - Designed to add interactivity to HTML pages
  - Usually embedded into HTML pages
  - What can a JavaScript Do?
    - Put dynamic text into an HTML page
    - React to events
    - Read and write HTML elements
    - Validate data before it is submitted to a server
    - Create cookies
    - …

Server-Side Programmability

- The requests cause the response to be generated
- Server scripting:
  - **CGI/Perl**: Common Gate Way Interface (*.pl, *.cgi)
  - PHP: Open source, strong database support (*.php)
  - ASP: Microsoft product, uses .Net framework (*.asp)
  - Java via JavaServer Pages (*.jsp)
  - …

CGI

- Common Gateway Interface:
  - CGI provides a way by which a web server can obtain data from (or send data to) database, and other programs, and present that data to viewers via the web.
  - A CGI program can be written in any programming language, but Perl is one of the most popular

Web languages/technologies

- Hypertext Markup Language (HTML): used for writing web pages
- Cascading Style Sheets (CSS): stylistic info for web pages
- PHP Hypertext Processor (PHP): dynamically create pages on a web server
- JavaScript: interactive and programmable web pages (with libraries like jQuery)
- Asynchronous JavaScript and XML (Ajax): retrieving data for web applications from pre-loaded web pages
- eXtensible Markup Language (XML) and JavaScript Obejct Notation (JSON): meta-languages for organizing data
- Ruby on Rails: web application development framework
- C#: a Microsoft .net language used in web development as well as in other contexts