Lecture Notes 1
Basic Concepts

Anand Tripathi

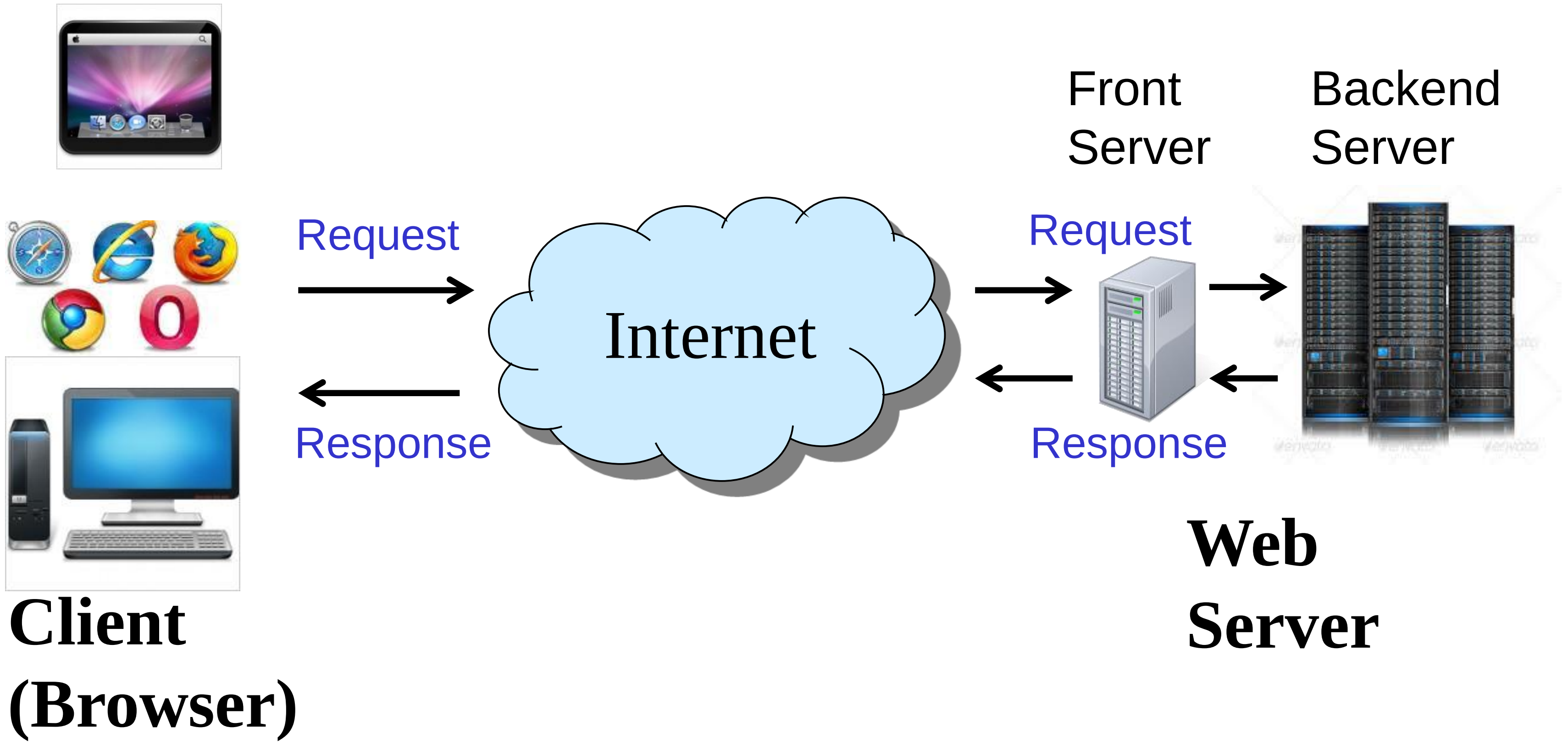
CSci 4131

Internet Programming

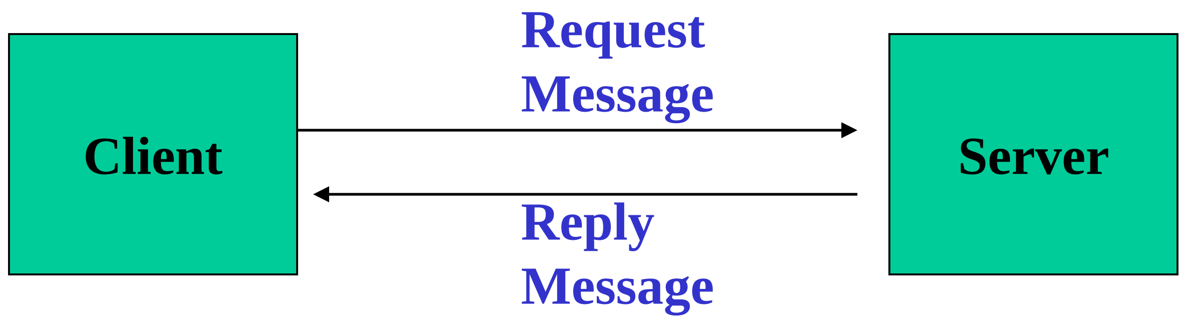
Foundations

- Web Service – Request/Response Paradigm
- TCP/IP (Transmission Control Protocol and Internet Protocol)
- Distributed Application Programming Models:
  - Client-Server Model
- Internet Services and Applications

Structure of Web Services



Client-Server Model



1. Client sends a request message to the server process.

   Request contains parameters of the request.

2. Server processes the request.

   Prepares a response message

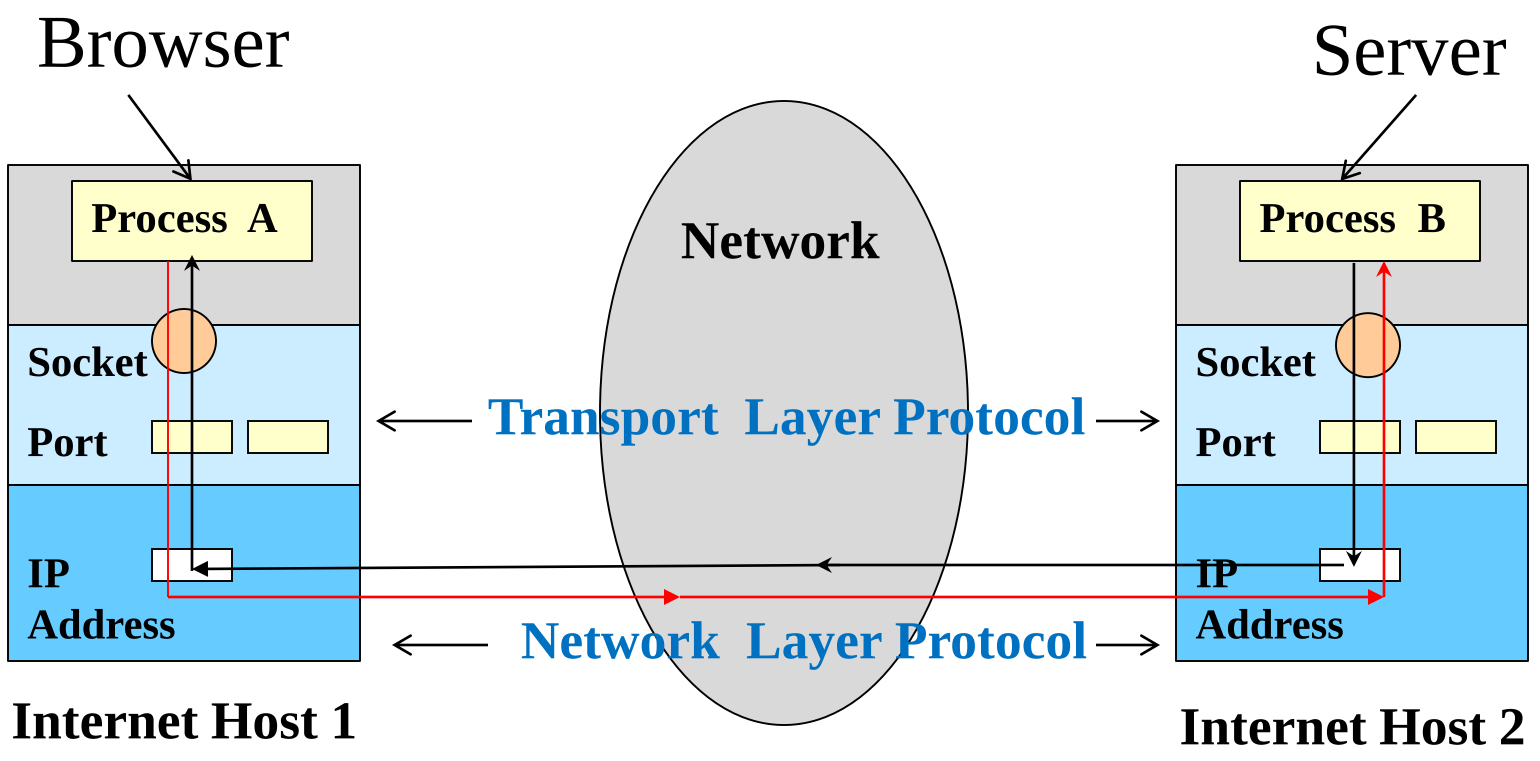
   Sends the response message to the client.

Server Function

The server repeatedly

- waits for a connection on one of its ports;
- when a connection request arrives:
    - Receive the request message
    - Interpret the contents of the request message
    - Process the request
    - Prepares a response message
    - Send the response message to the client
    - Start waiting for the next request.
- A server could be multi-threaded to handle multiple requests concurrently

Network Communication



Message communication takes place using the Internet Protocol (IP)

Internet Protocol

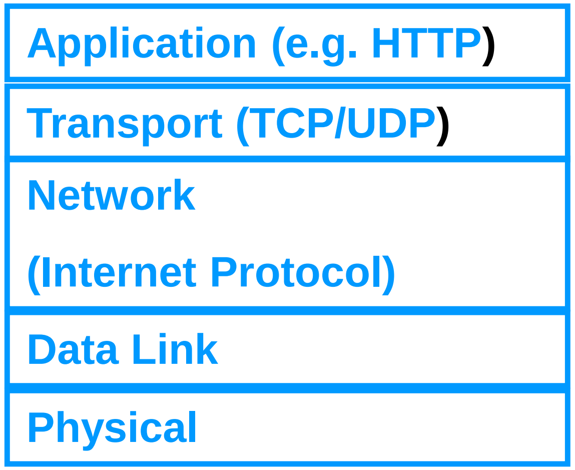
- IP -- Internet Protocol routes IP Packets from one host to another in the internet.
- It is a Network Layer protocol. It is a host-to-host communication protocol.
- Every host in the Internet is assigned a unique IP address, which has 32 bits in IPv4 (IP version 4) and 128 bits in IPv6.

IP Address - Dotted Decimal Format

- IPv4 is commonly used; IPv6 is the new generation protocol.
- IPv4 addresses are specified using the "dotted decimal" notation. Each byte in the 32 bit address is represented by a decimal number. For example: 128.101.228.45
- In order to communicate with another host in the Internet, a host must know the IP address of that host.

Communication Protocol Layers

- The following five-layer model is adopted in the textbook for presentation and discussions.
- It is based on TCP/IP and the OSI model.



Transport Layer Protocols
Transmission Control Protocol (TCP)

- TCP (Transmission Control Protocol) supports:
  - Connection-oriented protocol
  - Reliable bidirectional communication channel between two processes at any hosts in the Internet.
    - Pipes in two directions
    - No loss
    - In-order delivery of the bytes in the message stream
  - A TCP connection supports communication of any amount of data in either direction.
  - It's a connection-oriented protocol: this involves three IP packets to be exchanged to establish a connection

Transport Layer Protocols
User Datagram Protocol (UDP)

Many applications do not require retransmissions of packets for reliability, as in TCP.

- Example: voice-over-IP

- UDP (User Datagram Protocol) support communication with fixed maximum size packets.
  - Packet delivery is on "best effort" basis
    - No guarantee of delivery
  - It's a connection-less protocol (so, no overhead of establishing a connection).

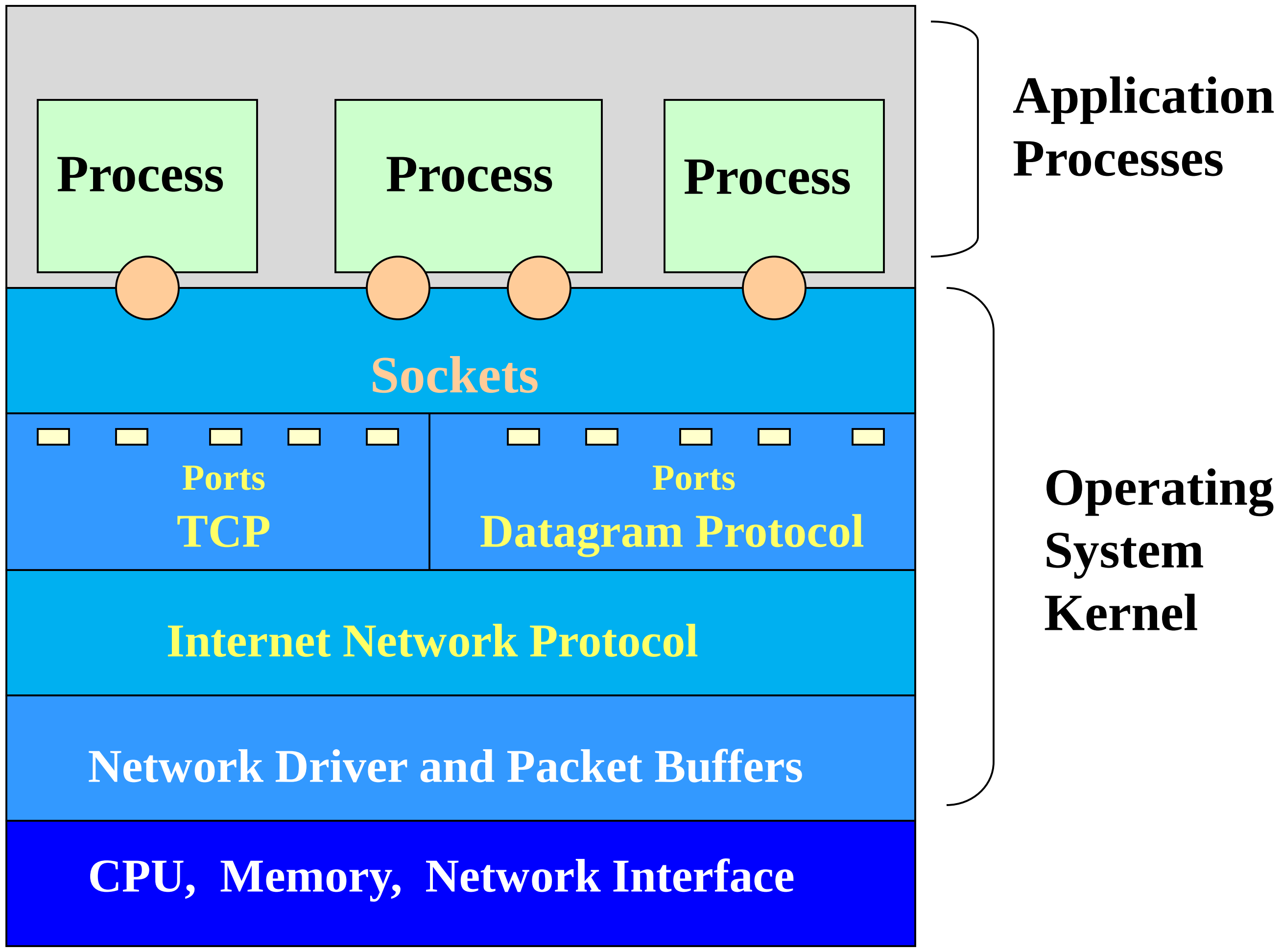
Sockets and Ports

- A socket represents kernel-supported entity using which a process communicates with another process in the network.
- A socket is similar to a file descriptor in UNIX. In fact, it is referenced through an an entry in the UNIX file descriptor table of the process.
- A port is a protocol specific entity which acts as a communication end-point for messages routed to a host.
- Ports are maintained by the kernel. It is identified by a 16 bit number.

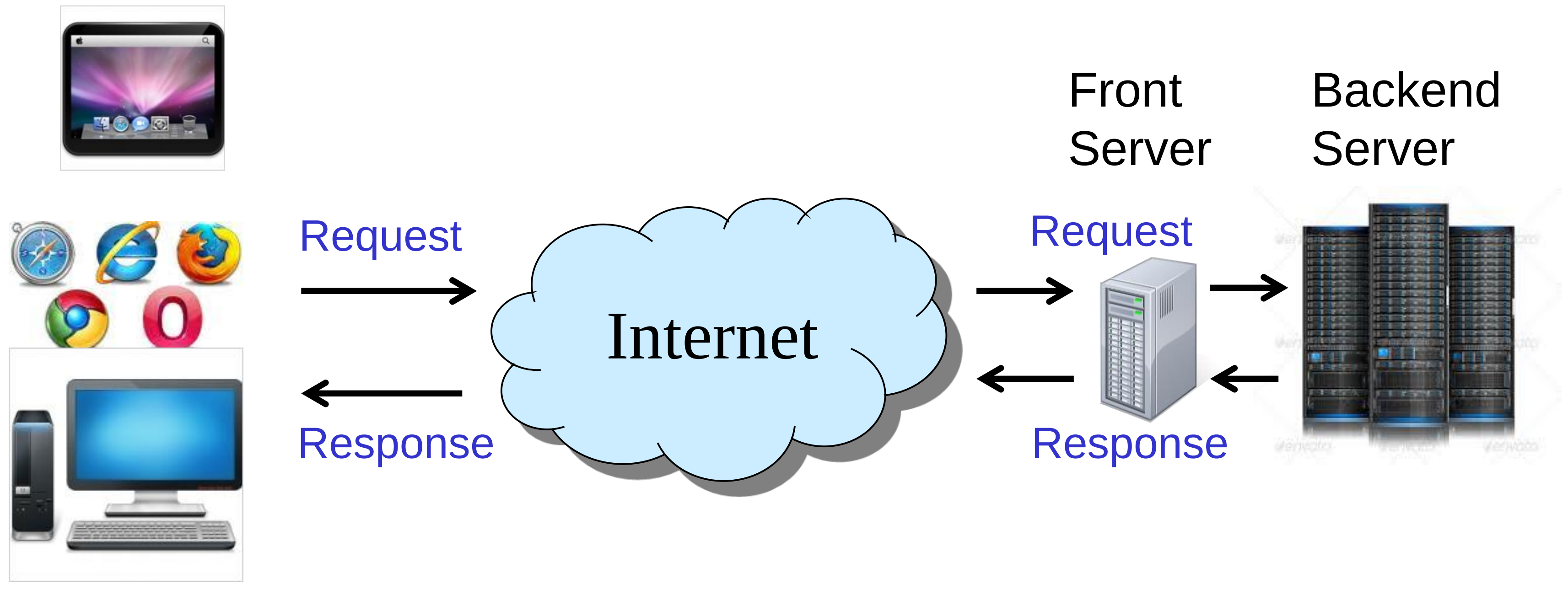
Sockets and Ports

- On a host, different protocols (such as TCP and UDP) can have ports with the same number, but such ports represent different entities.
- A process needs to "bind" one of its sockets to a port in order to communicate through that socket.
- An <IP address, Protocol, Port number> tuple uniquely identifies a communication point in the network.

Sockets and Ports



Structure of Web Services



$$\begin{tikzcd}
\text{Client (Browser)} \arrow[r, shift left, "\text{HTTP Request Message}"] & \text{Web Server} \arrow[l, shift left, "\text{HTTP Response Message}"]
\end{tikzcd}$$

Web Services and Mashups

- A web resource can be accessed by application programs or other web servers using the HTTP protocol.
  - We are not limited to using human-centered browser based access.
- Using mashups a web application or a web service can aggregate contents from many other services to provide new functionalities and services.
  - Web services can communicate with each other using XML based messages, in addition to HTML.

Client –Server Model on the Internet

- A server provides certain services to clients on the Internet.
- For example: File Server, Web Server, FTP Server, DNS Server etc
- To communicate with the server, a client must know:
  - IP address of the host where the server is running
  - Protocol (TCP/UDP) to be used.
  - Port number

Standard Services on Internet

- An Internet service is implemented at the application layer, using TCP or UDP for message communication.
- There are many standard, well-known services over the Internet:
  - File Transfer Protocol (FTP)
  - Remote login using Telnet
  - WWW using the HTTP protocol
  - Domain Name System (DNS) for name service
  - Simple Mail Transfer Protocol (SMTP) for email service
  - Simple Network Management (SNMP) Protocol for managing network components

RESERVED PORTS

- Many commonly used services (such as web service, ftp service etc) on a host in the Internet are accessed through well-known ports TCP and UDP ports. These are called *well-known ports* .

  Clients are generally assigned short-lived *ephemeral ports* .
  Internet Assigned Numbers Authority (IANA) divides ports into three groups:

1. Well-known ports - these are in the range 0 - 1023. The same port number is assigned to a given service for use with TCP and UDP.
2. Registered ports: 1024 - 49151. Not controlled by IANA. IANA registers and lists these ports as a service to the community.
3. Dynamic, private, or ephemeral ports: 49152 - 65535.

TRANSPORT LEVEL PROTOCOLS USED BY VARIOUS INTERNET APPLICATIONS

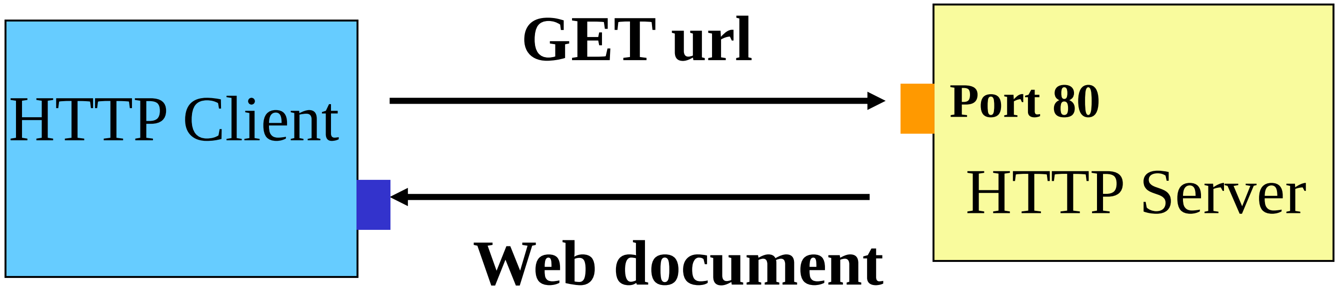
Application	Transport Protocol	Port
FTP (file transfer)	TCP	21
SSH	TCP	22
Telnet (remote login)	TCP	23
SMTP (mail transfer)	TCP	25
Gopher	TCP	70
Finger	TCP	79
HTTP (hypertext transfer)	TCP	80
SHTTP (Secure hypertext transfer)	TCP	443
POP2	TCP	109
POP3	TCP	110
Sun RPC	TCP, UDP	111
NNTP (network news)	TCP	119
NTP (Time protocol)	UDP, TCP	123

Stateful vs. Stateless Service

- *Stateless Protocol*
  - In a *stateless protocol* , the server does not maintain any information about a client's previous requests.
  - Each request from a client must contain its full context and information needed by the server to process the request.
    Examples: HTTP, Sun's Network File System.
  - Recovery of a server from a crash is easier.
- *Stateful Protocol*
  - The server maintains some information about a client's previous requests.
  - A client's request can be processed based on the context of its previous requests.
  - Crash recovery of a server becomes complex.

Web Servers and Clients

- A web client (e.g. browser) and a web service communicate using the HTTP protocol.
  - It is in the form of request and reply.
- HTTP Request commands: GET, or POST



Uniform Resource Locator (URL)

- A user or application specifies a URL to the client agent (browser).
- A URL specifies the location of a web resource.
  - For example:
    - http://www.w3.org/Addressing/URL/Overview.html
    - http://www.cs.umn.edu
- URL specifies the following:
  - host location (using DNS)
  - Optional port number (by default it is port 80)
  - File system path for the resource on the server

URL Structure

protocol :// host : port / directory-pathname / file # fragment

- Protocol can be:
  - http, https, telnet, ftp, mailto
- Port number can be omitted if the server is running on the well-known port for that service.
  - For example, HTTP servers (port 80), HTTPS (port 443) FTP (port 21), Telnet (port 23).
- File system pathname for a resource (which may be data file or executable program)
  - Fragment refers to a named "object" within the file (resource).

URL

A URL can be in one of the two forms:

- Absolute or Complete URL
  - It specifies the complete access path for the named resources in the Internet.
- Relative or Partial URL
  - It is meaningful only in the context of some other URL.
  - It is used when the referenced resource is on the same host machine as the referring resource.
  - Typically it only contains a relative filepath for some resource

Absolute URL

The specification of an absolute URL contains the following information:

- Protocol to be used to access the resource: ftp, http, mailto
- DNS name or IP address of the server that contains the resource.
- If needed, specification of the server's port number.
- The directory path within which the resource is contained.
- The name of the file representing that resource.
- Some specific named component (fragment) within the resource, such as a named "anchor" within an HTML document.
- Query-string containing parameters to be passed to the server resource.

Example of URL

- Example of URL with inclusion of single parameter to be passed to the resource.

http:// www.cs.umn.edu /admissions /application.cgi?someValue

Command line parameter to be passed to the CGI program

Example of a URL with Query-String

Query-string

http://www.cs.umn.edu/admissions/application.cgi? param1=v1&param2=v2

Parameters to be passed to the CGI program

In the example URL shown below:

Server host DNS name is www.cs.umn.edu

Default port 80 is to be used for the connection

/admission/application.cgi is the resource filepath name on the server

The *query-string* contains *(param=value) pairs of strings* which will be passed to the CGI program an environment variable..

HTTP Protocol

Hyper-Text Transfer Protocol is a simple protocol for request/reply based interactions between a client and a server.

This serves as the foundation of the World Wide Web.

- Request messages contains a very small number methods that can be invoked on the server.
  - It is extensible so new methods can be added.
  - Parameter data can be included in a request.
- Response messages indicate the status of the request processing and possibly some response data.

Browser Operations

1. Use of application specifies a URL for some web service.
2. Browser opens a TCP connection to port 80 of the host specified in the URL.
3. It sends an HTTP request message to the server running on the other end of the connection.
4. Server sends a response message, which may contain some web resource such as an HTML document.
    1. Brower starts rendering the HTML document
    2. If the document contains other embedded objects specified by URLs, the browser fetches each one of those successively during the rendering phase.

REQUEST MESSAGE: A sample example HTTP request

Suppose that the URL specified is
http://www.cs.umn.edu/~tripathi/index.html

HTTP request message looks as shown below:

```
GET    /~tripathi/index.html    HTTP/1.1
Accept: image/gif
Accept: image/jpeg
User-Agent:  Mozilla/4.05 [en] (X11; I; Linux 1.3.20 i486)
From:  tripathi@cs.umn.edu
    * a blank line *
```

HTTP Request

The first line is the REQUEST LINE, and it contains three items:

1. Name of the requested operation.
2. Request-URI (specifying the resource).
3. HTTP version.

REQUEST LINE

Method-Name   Request-URI   HTTP-VERSION   CRLF

Here CRLF stands for "carriage return" (carriage-return character '\r' (ascii 13) and new line character '\n' (ascii 10).

Method-name can be one of: GET,   POST,   HEAD (case-sensitive).

1. GET to retrieve the contents of a resource.
2. POST to update or modify a resource (e.g. adding item to a database).
3. HEAD to obtain meta-information related to the resource (e.g. last modification time).

General Structure of a Request Message

BNF grammar rule:

```
REQUEST-LINE
  * (   GENERAL-HEADERS
          | REQUEST-HEADER
          | ENTITY-HEADER )
       CRLF
       [ ENTITY-BODY ]
```

General headers are applicable to any kind (request or response) message. For example "Date"

Request headers are applicable to request messages.

For example: "If-Modified-Since"

Entity headers are applicable to the the body of the enclosed entity.

For example: “Content-type” or "Context-length"

The order of the headers is not important.

See Section 10 or RFC 1945.

In general, all HTTP headers (general, request, response, entity) have the form:

```
FIELD-NAME ":" [ FIELD-VALUE ] CRLF
```

For example:

```
If-Modified-Since: Tue, 29 Sep 1998 12:43:49 GMT
Content-Length: 4687
Content-Type: text/html
```

RESPONSE MESSAGE

```
HTTP/1.1   200    OK
Date: Tue, 8 Sep 2015 12:43:49 GMT
Server: Apache/1.3.1 (Unix)
Last-Modified: Tue, 1 Sep 2015 14:43:08 GMT
Connection: close
Content-Type: text/html

<html>
<head>
   < title> University of Minnesota
       -- Department of Computer Science and Engineering </title>
    …..
```

Category 2xx is used to indicate SUCCESS

Code	Significance and Use
200	OK
201	New resource created at the server
202	Accepted for processing. Non-committal.
203	Non-authoritative content
204	No contents

Category 3xx is used for re-direction

Code	Significance and Use
300	Multiple choices for accessing the resource
301	Moved permanently to another location
302	Moved temporarily
303	See other
304	Not modified

Category 4xx is used for indicating client error.

Code	Significance and Use
400	Bad request
401	Unauthorized
402	Payment required. (Return this code to get rich.)
403	Forbidden. Server does not understand what you are trying to do. Authorization would not help.
404	Not found

Category 5xx is used for indicating internal server errors.

Code	Significance and Use
500	Internal server error.
501	Requested method is not implemented.
502	Bad gateway. The server, while acting as a proxy or gateway, received an invalid response from some upstream server.
503	Service unavailable. Overload possibility.