Networking and Communication Protocol
A network consists of 4 components:
1. Computers (or hosts)
2. Transmission lines
3. Network adaptors
4. Network software

Types of networks:
1. Local Area Network (LAN)
   - Ethernet
   - IBM's Token Ring
   - FDDI (Fiber Distributed Data Interface)
2. Wide Area Network (WAN)
   - Leased telephone lines
   - T-carrier (T-1, T-4)
   - broadband ISDN
A network must provide connectivity among a set of computers.
A network can consist of two or more computers directly connected by some physical medium, such as a coaxial cable or an optical fibre.

- **Link**

The computers connected are called **Nodes**.

A node can be a more specialized piece of hardware rather than a general-purpose computer.

Physical links are sometimes limited to a pair of nodes.

- **Point-to-point Links**
• Different network topologies
  – Star
  – Ring
  – Irregular

• In other cases, more than two nodes may share a single physical link
  – *Multiple-access Links*
Broadcast Channel

- Packets/messages sent by one host can be received by all the others
- An address field within the packet species for which host it is intended
- When a packet is received, the host checks the address field. If the packet is intended for some other host, it is just ignored
- Allows addressing a packet to all destinations by using a special code in the address field
- Multicasting: transmit to a subset of hosts
- Connectivity between two nodes does not necessarily imply a direct physical connection between them
Internet or Internetworking

- No single network can connect all hosts/users
- Connecting networks using routers
  - Switch packets from one network to another
- Routers, Gateways, Repeaters, Bridges
- The networks may be built using different hardware technologies
- A host is said to be multihomed if it is connected to more than one networks (i.e. more than one network interfaces)
- It is impossible to connect every pair of hosts
  - Hosts should share communication lines
  - Communicate by packet-switching
- Advantage: Highly Scalable
- Distinguish between the Internet and an internet
Communication Protocol

- Network Software to achieve communication
- Users concentrate on application software
- If two parties have to communicate, they should agree on the same set of standards, rules and conventions
- The same is true for network communication
  - Communication Protocol
- Objectives:
  1. Allows heterogeneous networks to communicate
     - e.g. Different technologies (Ethernets or TokenRings), different transmission media
  2. Open standards, and publicly available
  3. Hide details of physical networks from application programs
  4. Global (common) addressing scheme
  5. Allow the user to view the internet as a single, virtual network to which all hosts connect despite their actual physical connections
Peer-to-peer relationship is a virtual relationship between identical layers across various sites.
Open Systems Interconnection Model

Recognized by the International Standards Organization (ISO).

- Why are protocols layered?
  1. Network software is complicated
  2. Details related to communication and networking are hidden from application programs
  3. Network independence (technology, topology)
  4. Promote re-usability

- Example:
  - TCP/IP
  - Xerox Networking Systems (XNS)
  - IBM's Systems Network Architecture (SNA)
  - Novell IPX
OSI Model

• Layer 0: The physical media

• Layer 1: Physical:
  • Specifications related to transfer of data bits.
    • Modulation techniques, voltages, etc.

• Layer 2, Data Link:
  • Specs related to delivery of frames across two adjacent nodes.
    • A frame carries information such as address of sender and recipient, codes relating to error checking
  • Data link layer specifies how a node gain access to the media, how it picks up the frames from the media, and the structure of the frame
The OSI Model

- Layer 3, Network:
  - Specs relating to sending message over the entire network
    - routing, congestion control
    - Data frame is now called data packet in Layer 3.
The OSI Model

Layer 4, Transport:
- Ensure end-to-end delivery
  - function executed at the sender and recipient end
  - Packets are in the right order and no packet has been lost
The OSI Model

- **Layer 5, Session:**
  - Establishment and maintenance of the connection
  - Session rules specifies
    - Order of conversation, topic of negotiation, pace at which the data are sent, control of data flow

- **Layer 6, Presentation:**
  - Specifies format of data being transferred
    - Changing of data codes, encryption, data compression

- **Layer 7, Application:**
The Internet

- Internet originated in 60s as a result of research supported by Advanced Research Project Agency (ARPA) of the U.S. Department of Defense (DOD)
- First version of the network was called APRANET.
- Features of ARPANET were adopted and became Internet.
- Characteristics of Internet:
  - Data Centric network (voice not included)
  - Separation of Communication and data processing between 2 computers, one called router or gateway and the other host
- Packet Switched Network
  - Packets travel independently over the network instead of dedicated line between two points.
The Internet: Packet Switching

Packets travel from sending host to destination host. Along the way the packets are routed from one router to another.

- No fixed routing is maintained between hosts.
- Routers have their own routing tables that change according to the network state.
- As the network state changes due to congestion or link failures, routes followed by the packets change as well.
- Destination host has to get the packets in the right sequence before passing them on to the application.
The Internet: Network connecting other networks

A simplified view of the Internet
The Internet: The Network

- Networks owned by different governments and organizations
- Consists of different levels of backbone: national level, regional level
- The backbones are owned and operated by different Internet service providers and governments
- Local Internet service provider (ISP) leases connections from a national or regional ISP, then provide dial-up access to users.
- The backbone meet at network access points and metropolitan area exchanges
Internet Service Provider

MAE = Metropolitan Area Exchange
NAP = Network Access Point

Point of Presence
TCP/IP Protocol Suite

• Why study TCP/IP?
  – Not vendor-specific
  – Popular, implemented on everything from PC to supercomputers, and networks of all sizes
  – Supported by UNIX

• Simplify the OSI model to 4 layers only:

<table>
<thead>
<tr>
<th>Layer</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Layer</td>
<td>e.g. Telnet, FTP</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>e.g. TCP, UDP</td>
</tr>
<tr>
<td>Network Layer</td>
<td>Internet</td>
</tr>
<tr>
<td>Link Layer</td>
<td>Data Link, Network Access</td>
</tr>
</tbody>
</table>
1. **Application Layer (OSI Layers 5-7)**
   - Users invoke application programs that access services across a TCP/IP internet
   - Represent the data in required form
   - Interact with the Transport layer protocol(s) to send or receive data
2. **Transport Layer (OSI Layer 4)**
   
a) **TCP (Transmission Control Protocol)**
   - A connection-oriented protocol
     - Processes communicating using TCP have to establish a virtual circuit first, which is hung up when interaction is completed
   - Processes communicate via a reliable, full-duplex, byte stream
   - Reliable
     - Error detection
   - Full-duplex
     - Each process can send and receive via the same virtual circuit
   - Byte stream
   - No message boundary

b) **UDP (User Datagram Protocol)**
   - A connectionless protocol
   - Reliability is not guaranteed
   - Data sent as messages
3. **Network Layer (OSI Layer 3)**
   - Internet Protocol (IP)
   - Implements a connectionless and unreliable delivery system supporting both TCP and UDP
   - Each *IP datagram* is delivered and routed independently
   - *Routing* of IP datagrams through internet

4. **Link Layer (OSI Layers 1-2)**
   - Actual interface to the hardware/network
   - Deliver frames to unique hardware addresses
Header Encapsulation

- As data moves down the protocol stack from application to physical network, each layer adds control information to ensure proper delivery
  - e.g. Address, checksum
- When data moves up the protocol stack, the control information will be removed successively
- There is trailer encapsulation also, as in an Ethernet frame (4 bytes, for checksum)
- Length: UDP (8 bytes), TCP (20 bytes), IP (20 bytes), Ethernet (14 bytes)
Internet Addresses

- To communicate with a certain host, we need to know its address
- Internet addresses or IP addresses
  - Global, unique
- A 32-bit integer value
- To make routing efficient, each IP address consists of two parts:
  1. netid: identify the network to which a host attaches
  2. hostid: identify a unique host on that network
- It is easy to extract the hostid or netid portions quickly
  - Routing can be based on netid
  - Routers don't have to keep track of all possible IP addresses
  - Only need to keep track of possible netid's
Three primary classes of IP addresses

1. Class A
   - A small number of large networks (netid 1 to 126), each of more than 65,536 hosts

2. Class B
   - Intermediate size networks: from 256 hosts to $2^{16}$ hosts

3. Class C
   - Small networks of fewer than $2^8$ hosts
Dotted Decimal Notation

- Four decimal integers separated by decimal points, each integer gives the value of one octet/byte of the IP address

```
10000000 00001010 00000010 00011110
```

can be written as

```
128.10.2.30
```

- IP Next Generation initiative was started by the Internet Engineering Task Force. A newly defined standard called IP version 6 (IPv6) is proposed.
Routing

- Each IP datagram contains the IP address of the source host and the IP address of the destination host in its 20-byte IP header
- When fragmentation occurs, each IP datagram fragment contains the above two IP addresses
- When a router receives an IP datagram, the netid portion of the destination IP address
- If the netid suggests that the destination is not a directly connected host of the local network
  - Router determines the next router to forward the IP datagram (called next hop)
  - Routing table
- When the IP datagram finally reaches the network where the destination host resides
  - Direct delivery
- Address Resolution Protocol (ARP) is used for mapping from high-level IP address to hardware address
Port Numbers

- It is possible that more than one user processes are using TCP or UDP at a time
- Need a way to identify to whom a piece of data belongs to
  - Port number
- 16-bit integer

**Well-known Ports**
- In a client/server system, the client must have a way to identify the server it wants
- IP address only tells the host in which the server resides
- Still need a way to identify the particular server process in the host
- Define a set of well-known ports
  - e.g. FTP has the TCP port 21, TFTP has the UDP port 69

**Ephemeral Port Numbers**
- When server wants to reply to client, it needs to know which client process to contact also
- Any port number can work, as long as it is unique in the client's host
- Can be re-used when service is completed
- Usually, port numbers from 1 to 255 are for well-known port numbers. Additional port numbers may be reserved
- UNIX 4.3BSD, ephemeral port numbers start from 1024
Summary of TCP/IP

- A hierarchical addressing scheme is defined:
  1. Each IP datagram contains the source and destination IP address in its IP header
     - Identify the two hosts that are communicating
  2. IP header contains a protocol number
     - Tells if an IP datagram is for TCP or UDP (or some other protocol which uses IP)
  3. The TCP header and the UDP header contain the source port number and the destination port number
     - Identify the two user processes which are communicating
Example

- When we type the command:
  ```bash
  % ftp ftp.ics.uci.edu
  ```
  the following steps take place:
  1. The FTP client calls the function `gethostbyname` to convert the hostname `cuse131` into its 32-bit IP address. This function is called a resolver in the Domain Name System (DNS).
  2. The FTP client asks its TCP to establish a connection with that IP address.
  3. TCP sends a connection request segment to the remote host by sending an IP datagram to its IP address.
  4. If the destination host is on a locally attached network, the IP datagram can be sent directly to that host. If the destination host is on a remote network, the IP routing function determines the IP address of a locally attached next-hop router to send the IP datagram to.
  5. A translation is required from the logical IP address to its corresponding physical hardware address via ARP.
  6. The IP datagram is sent to the appropriate host corresponding to `ftp.ics.uci.edu`. 
Domain Name Server (DNS)

- Mapping names to IP addresses
- **Name space**
  - The set of possible names (finite or infinite)
  - Can be flat or hierarchical
- The naming system maintains a collection of **bindings** of names to values
- **Resolution mechanism**
  - A procedure that, when invoked with a name, returns the corresponding value
- **Name server**
  - An implementation of a resolution mechanism that is available on a network and that can be queried by sending it a message
- DNS is the naming system for Internet
- In the past when there were only a few hundred hosts on the Internet, Network Information Center (NIC) maintained a flat table of name-to-address bindings
- The system administrator at each site installed the table on every host at the site
• Name resolution was then simply implemented by a procedure that looked up a host's name in the local copy of the table and returned the corresponding address

• **Scaling problem**
  • DNS employs a hierarchical name space rather than a flat name space
    – A name can be divided into components
  • The “table” of bindings that implements this name space is partitioned into disjoint pieces and distributed throughout the Internet
  • The subtables are made available in name servers that can be queried over the network
Domain Hierarchy

- DNS implements a hierarchical name space for Internet objects
  - e.g. cheltenham.cs.arizona.edu
- Unlike file names, which are processed from left to right, DNS names are processed from right to left and use periods as the separator
- Like the Unix file hierarchy, the DNS hierarchy can be visualized as a tree:
  - Each node in the tree corresponds to a domain
  - The leaves in the tree correspond to the hosts being named
- Domain
  - A context in which additional names can be defined
- Organization of the name space:
  - There are domains for each country
  - “Big six” domains:
    - edu, com, gov, mil, org, net
Name Servers

- The domain hierarchy is actually partitioned into subtrees called zones
- Each zone can be thought of as corresponding to some administrative authority that is responsible for that portion of the hierarchy
- Information contained in each zone is implemented in two or more name servers
• Each name server can be accessed over the Internet based on a client/server model:
  – Clients send queries to name servers, and name servers respond with the requested information
• Sometimes, the response contains the final answer that the client wants
• Sometimes the response contains a pointer to another server that the client should query next
• Thus, it is more accurate to think of DNS as being represented by a hierarchy of name servers rather than by a hierarchy of domains
Each name server implements the zone information as a collection of resource records.
- A resource record is a name-to-value binding:
  - `<Name, Value, Type, Class, TTL>`
- The Type field species how Value should be interpreted:
  - `A`: Value is an IP address
  - `NS`: Value gives the domain name for a host that is running a name server that knows how to resolve names within the specified domain
  - `CNAME`: Value gives the canonical name for a particular host; it is used to define aliases
  - `MX`: Value gives the domain name for a host that is running a mail server that accepts messages for the specified domain
- The root name server contains an NS record for each second-level server.
- It also has an A record that translates this name into the corresponding IP address.
- Together, these two records implement a point from the root name server to each of the second-level servers:
  - `<arizona.edu, telecom.arizona.edu, NS>`
  - `<telecom.arizona.edu, 128.196.128.233, A>`
  - `<bellcome.com, thumper.bellcome.com, NS>`
  - `<thumper.bellcome.com, 128.96.32.20, A>`
Next, the domain arizona.edu has a name server available on host telcom.arizona.edu that contains the following records:

- <cs.arizona.edu, optima.cs.arizona.edu, NS>
- <optima.cs.arizona.edu, 192.12.69.5 A>
- <ece.arizona.edu, helios.ece.arizona.edu, NS>
- <helios.ece.arizona.edu, 128,196,28,166, A>
- <jupiter.physics.arizona.edu, 128,196,4.1, A>
- <saturn.physics.arizona.edu, 128,196,4.2, A>
- <mars.physics.arizona.edu, 128,196,4.3, A>
- <venus.physics.arizona.edu, 128,196,4.4, A>

Finally, a third-level name server, such as the one managed by domain cs.arizona.edu, contains A records for all of its hosts:

- <cs.arizona.edu, optima.cs.arizona.edu, MX>
- <cheltenham.cs.arizona.edu, 192.12.69.60, A>
- <che.cs.arizona.edu, cheltenham.cs.arizona.edu, CNAME>
- <optima.cs.arizona.edu, 192.12.69.5, A>
- <opt.cs.arizona.edu, optima.cs.arizona.edu CNAME>
- <baskerville.cs.arizona.edu, 192.12.69.35, A>
- <bas.cs.arizona.edu, baskerville.cs.arizona.edu, CNAME>
• **MX record:**
  - Maildrop of the whole domain
  - The host which receives mail on behalf of the domain

• **Name Resolution**
  - Suppose the client wants to resolve the name cheltenham.cs.arizona.edu
  - The client first sends a query containing this name to the root server
  - The root server, unable to match the entire name, returns the best match it has
    - NS record for arizona.edu
  - The server also returns all records that are related to this record
    - A record for telcom.arizona.edu
  - The client, having not received the answer, next sends the same query to the name server at IP 128.196.128.233
  - This server also cannot match the whole name, and so returns the NS and corresponding A records for cs.arizona.edu domain
  - Finally, the client sends the same query as before to the server at IP 192.12.69.5
  - This time, the server at IP 192.12.69.5 returns the A record for cheltenham.cs.arizona.edu
• How did the client locate the root server in the first place?
  – The bootstrapping problem is a fundamental to any naming system
• The name-to-address mapping for one or more root servers is well known
• In practice, not all clients know about the root servers
• Instead, the client program running on each Internet host is initialized with the address of a **local name server**
  – e.g. All the hosts in the Department of Computer Science at the University of Arizona know about the server on optima.cs.arizona.edu
• This local name server has resource records for one or more of the root servers
  – < `root', venera.isi.edu, NS >
  – < venera.isi.edu, 128.9.0.32, A >
• Thus the client first queries the local name server, which in turn acts as client to query the remote servers on behalf of the original client
Caching

- The local server will cache the responses from the remote servers
- It may be able to resolve future queries without having to go out over the network again
- How the system works when a user submits a partial name (e.g. cheltenham) rather than a complete domain name (e.g. cheltenham.cs.arizona.edu)?
  - The client program is configured with the local domain in which the host resides (e.g. cs.arizona.edu) and it appends this string to any simple names before sending out a query
• Sample interaction
• The Web is:
  – the world's largest client/server application
    • It has millions of servers.
    • 1.5 millions new pages added daily
  – the first application that brings client/server to the masses
  – the first client/server application of intergalactic proportions

• 3 eras of Web Client/Server:
  – Hypertext Web
  – Interactive Web
  – Object Web
Ethernet Client/Server

• The most popular application of client/server technology
• Multiple clients talking to one single local server over a LAN
• Suitable for small businesses, and departments of large corporations (e.g. branch offices of a bank)

• Advantage:
  – No complex interaction between servers
    • Although there can be loosely-coupled inter-server communication
    • e.g. between the departmental server and the enterprise server
• The middleware tends to be simple
Web Client/Server

- A lot more than surfing through hypertext webs of HTML-tagged information
- Next generation of Internet applications
  - e.g. information highway, electronic bazaar
- Electronic agents of all kinds will be roaming around the network looking for bargains and conducting negotiations with other agents
- Frequent business transactions
- Massive amounts of multimedia data
- Key technologies:
  1. High bandwidth at very low cost
  2. Rich transaction processing
     - Long-lived, secured, nested transactions which span across multiple servers
  3. Roaming agents
  4. Rich data management
     - include active multimedia compound document
  5. Intelligent middleware
     - Provide a single-system image of an infinite number of hybrid servers
     - Heterogeneous servers may reside on Internet, intranet, or corporate backbone network
     - Ensure proper collaboration of heterogeneous servers
     - Frequent inter-server interactions that are transparent to the clients
     - Clients and servers can dynamically join and leave the network, and then discover each other
     - Highly scalable
- The same naming conventions to locate any resource on the global network
Hypertext Era

- Homepages, hypertext documents
- The Mosaic graphical Web browser (1993) introduced the first true client/server application environment on top of the Internet
- Consists of thin, portable, “universal” clients - web browsers, e.g. Netscape, that talk to superfat servers
- A Web server simply returns documents when clients ask for them by name
• “The world becomes one big hyperlinked document”
• Electronic publishing, personal bulletin boards, document sharing, discussion groups, etc
• Key technologies and protocols:
  – The Internet is the global backbone
    • The application protocols of the Web are built on TCP/IP
  – The Internet is the private backbone
    • Existing Web and Internet technologies, network infrastructure are equally applicable in private corporate networks - the Intranets
  – URLs are used to globally name and access all Web resources
    • The Uniform Resource Locator protocol provides a consistent naming scheme to identify all kinds of Web resources - documents, images, sound clips, and programs
  – HTTP is used to retrieve URL-named resources
    • Hypertext Transfer Protocol
    • A stateless RPC-like protocol for accessing resources
    • Establishes client/server connection, transmits and receives parameters including a returned file, and breaks the connection
  – HTML is used to embed hyperlinks and to describe the logical structure of Web documents
    • Hypertext Markup Language
    • Simple and portable
– Web browsers are universal clients
  • Embedding Images in Documents
    – An image can be embedded in a document with the `<IMG>` tag
    – E.g., `<IMG SRC="zog.gif">` - get the picture file from zog.gif
  – Hyperlinks
    • They provide the hooks that let you transparently navigate from server to server by simply clicking the mouse.
    • You can hyperlink your document using a pair of anchor tags:
      – `<A HREF="Target URL">This is a sample link</A>`
Web Client/Server Interaction

1. Select a target URL
   - e.g. clicking on a hypertext link, explicitly typing in the URL
2. Browser sends an HTTP request to server
3. Server processes the request
   - On the receiving side, the HTTP server is waiting for requests on its well-known port (default is 80 for HTTP)
   - A socket connection is established between the client and the server
   - The server receives client's message, sends the requested HTML file, sends it back to the client in an HTTP response message
   - Close the connection
4. Browser interprets the HTML commands and displays the page contents
   - May need helper application if the file is not HTML
     • E.g. video player for video file
What is URL?

- For the protocol scheme, HTTP is the native protocol Web. Others include
  - Gopher – displays information as on servers as a hierarchy of menus
  - FTP – the oldest Internet protocol for retrieving files
  - News – discussion-group protocol (specify newsgroup or article)
  - Mailto – sending email to a designated email address
  - WAIS – specify the domain name of a target database to be searched and a list of criteria
- The server address field can be a numeric IP address, and optionally include user name or password
- The port number is optional, if so, the well-known port 80 is assumed for HTTP
  - 70 for Gopher, 21 for ftp.
HTTP

- The Web's RPC on top of TCP/IP
- It is stateless, which means that a separate connection is made for every request
  - Simple to implement, yet incur overhead
- Each HTTP client/server interaction consists of
  - a single request/reply interchange
    - HTTP request
    - HTTP response
HTTP request message consists of:

1. request line
   a) method or command to apply to a server resource
      - e.g. GET, POST
   b) URL (without protocol and server domain name)
   c) the protocol version used by the client, e.g. HTTP/1.0

2. request header fields
   - Pass additional information about the request and the client itself to the server
     - much like RPC parameters
   - Each header filed consists of a name, followed by “;” and the field value

3. the entity body (optional)
   - Clients use it to pass bulk information to the server (CGI)

Examples of HTTP methods
- GET - retrieve the specified URL
- POST - send this data to the specified URL

Examples of HTTP header fields
- Accept - lists acceptable MIME type/subtype contents
- User-Agent - provides client browser information

An example of HTTP request message

```
GET /path/file.html HTTP/1.0
Accept: text/html
Accept: audio/x
User-agent: MacWeb
```

Note: crlf: carriage-return/line-feed
• **HTTP response message**
  1. **response header line**
     - HTTP version, the status of the response, and an explanation of the returned status
  2. **response header fields**
     - Information that describes the server's attributes and the returned HTML document to client
  3. **entity body**
     - Contains an HTML document that a client has requested
       - Each HTML document needs a separate request message
         - stateless

An example of HTTP response message:

```
<HTTP version><result code>[<explanation>]<cr><lf>
[<Header> : <value>]<cr><lf>
    ... ...
[<Header> : <value>]<cr><lf>
<cr><lf>
[Entity body]
```

HTTP/1.0 200 OK
Server: NCSA/1.3
Mime_version: 1.0
Content_type: text/html
Content_length: 2000

```
<HTML>
    ...
</HTML>
```

• The result code 200 indicates that the request is successful.
Interaction Era

- Web form
  - An HTML page with one or more data entry fields and a mandatory “Submit” button
- The user clicks on the Submit button to send the form's data contents to a Web server
- The browser collects all the inputs from the form, “packs” them inside an HTTP message, and then invoke either an HTTP GET or POST method on the server side
- On the receiving end, the Web server does not know what to do with a form, as it's not an ordinary HTML document
- So the server invokes the program named in the URL
- The server passes the method request and its parameters to the back-end program using a protocol called the Common Gateway Interface (CGI)
- The back-end program executes the request and returns the results in HTML format to the Web server using the CGI protocol
- The Web server treats the results like a normal HTML document, and returns it to client
- 3-tier architecture
- CGI makes it possible for Internet clients to update databases on back-end servers
• **Web-based Forms**
  
  – Can embed special fields in an HTML document:
    
    1. **INPUT fields**
       
       – Text, Password, Hidden, Checkbox, Radio, Reset, Submit, Image
       
       – `<INPUT TYPE="field-type" NAME="Name of field" VALUE="default value">`
    
    2. **SELECT fields**
       
       – Create a dropdown list box from which users pick one or more items
       
       – Selection lists
         
         `<SELECT NAME="Name of field" SIZE="N" MULTIPLE>
         <Option>choice 1
         <Option>choice 2
         :
         <Option>choice N
         </SELECT>`
    
    3. **TEXTAREA fields**
       
       – Multiple lines of input text
    
• The `<FORM>` tag:
  
  `<FORM METHOD="POST" ACTION="HTTP://www.mylab.org/cgi-bin/sampleform">`

  where

  the ACTION attribute specifies an URL of the name of a server CGI program (or script) that can process the form's data
A CGI Scenario

1. User clicks on the form's “submit” button
   - The Web browser collects the data within the form and assembles it into one long string of name/value pairs, each separated by an “&”
2. Web browser invokes a POST HTTP method
   - An ordinary HTTP request that specifies a POST method, the URL of the target program in the “cgi-bin” directory and the typical HTTP headers. The entity body stores the form's data
3. The HTTP server receives the method invocation via a socket connection
   - The server parses the message and discovers that it's a POST for the “cgi-bin” program. So a CGI interaction is started
4. The HTTP server sets up the environment variables
5. The HTTP server starts a CGI program
6. The CGI program reads the environment variables
7. The CGI program receives the message body via the standard input pipe (stdin)
   – The “CONTENT LENGTH” environment variable tells the CGI program how much data is in the message/string. The program parses the string to retrieve the form data
8. The CGI program does some work
   – The CGI program interacts with some back-end resource, e.g. a DBMS or a transaction program, to service the form's request
   – The CGI program then formats the results in HTML format. These results will appear in the entity body of the HTTP response message
9. The CGI program returns the results via the standard output pipe (stdout)
10. The HTTP server returns the results to the client's Web browser
A sample CGI form

A sample CGI output

Edited by Christopher C. Yang
A sample HTML file

```html
<HTML>
<HEAD>
<TITLE>CGI Test Form</TITLE>
</HEAD>

<BODY>

<FORM action="http://www.se.cuhk.edu.hk/~seg4530/cgi-bin/formdump.cgi" method="POST">
<pre>
<SECTION>
Use POST Method
</SECTION>
Name : <INPUT name="NAME" size=30>
<INPUT name="CHKBOX" type="CHECKBOX" value="Checked"> A Checkbox
Textarea : <TEXTAREA align="TOP" name="COMMENT" rows=3 cols=50>
</TEXTAREA>
</PRE>
</FORM>

</BODY>

</HTML>
```
/* formdump.c */
#include <stdio.h>
#include "cgiutil.h"
#define BUFFER_SIZE 1000
#define TITLE "Form Variables"
int main() {
    char input_buf[BUFFER_SIZE], value_buf[BUFFER_SIZE];
    printf( "Content-type: text/html\n\n" );
    printf( "<HTML><HEAD><TITLE>%s</TITLE></HEAD><\n", TITLE );
    printf( "<BODY><H1>%s</H1><\n<HR><P>", TITLE );
    if( cgiGetInput( input_buf, BUFFER_SIZE ) ) {
        printf( "Your name is \" );
        if( cgiFindVar( input_buf, "NAME", value_buf, BUFFER_SIZE ) )
            printf( "<B>%s</B>\", value_buf );
        else
            printf( "<B>missing</B>\" );
        printf( "\nThe checkbox is \" );
        if( cgiFindVar( input_buf, "CHKBOX", value_buf, BUFFER_SIZE ) )
            printf( "<B>checked</B>\" );
        else
            printf( "<B>not checked</B>\" );
        if( cgiFindVar( input_buf, "COMMENT", value_buf, BUFFER_SIZE ) )
            printf( "\n%s: \"<B>%s</B>\n", value_buf );
        else
            printf( "\nNo comments were found. \n" );
    }
    else
        printf( "No CGI variables found. \n" );
    printf( "</PRE></BODY></HTML>\n" );
    return 0;
}
/ *cgiutil.h*/
#define METHOD_POST 2
const char *cgiGetEnv( const char *pname, const char *pdefault );
int cgiGetInput( char *buffer, int max_buffer );
int cgiFindVar( const char *cgi_input, const char *var_name, char *value, int value_size );
/cgiutil.c */
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "cgiutil.h"
#define VALUE_DELIM '='
#define CGIVAR_DELIM '&'

int hexvalue( char ch ) {
    return ( ch > 'A' ? ( toupper( ch ) - 'A' ) + 10 : ( ch - '0' ) );
}

char decode( char hex_digit1, char hex_digit2 ) {
    int decoded_char;
    decoded_char = hexvalue( hex_digit1 );
    decoded_char *= 16;
    decoded_char += hexvalue( hex_digit2 );
    return (char)decoded_char;
}

int scanValue( const char src[], char dest[], int dest_size ) {
    int src_pos = 0;
    int dest_pos = 0;
    int done = 0;
    while( !done && src[src_pos] && dest_pos < dest_size - 1 ) {
        switch( src[src_pos] ) {
        case '%':
            if( src_pos < (int)( strlen( src ) - 2 ) ) {
                dest[dest_pos++] = decode( src[src_pos + 1], src[src_pos + 2] );
                src_pos += 3;
            } else {
                dest[dest_pos++] = src[src_pos++];
            }
            break;
        case '+':
            dest[dest_pos++] = ' ';
            src_pos++;
            break;
        case VALUE_DELIM:
        case CGIVAR_DELIM:
            done = 1;
            break;
        default:
        dest[dest_pos++] = src[src_pos++];
        }
    }
    dest[dest_pos] = '\0';
    return src_pos;
}
const char *cgiGetEnv( const char *pname, const char *pdefault ) {
    char *presult = getenv( pname );
    if( presult != NULL )
        return presult;
    else
        return pdefault;
}

int cgiGetInput( char buffer[], int buffer_size ) {
    int method_type = 0;
    int content_len = atoi(cgiGetEnv( "CONTENT_LENGTH", "0" ) );
    buffer[0] = '\0';
    if( ( content_len > 0 ) && ( content_len < buffer_size ) ) {
        fread( buffer, 1, content_len, stdin );
        buffer[content_len] = '\0';
    }
    method_type = METHOD_POST;
    return method_type;
}

int cgiFindVar( const char *cgi_input, const char *var_name, char *value, int value_size ) {
    char *pname_start = strstr( cgi_input, var_name );
    while( pname_start != NULL ) {
        if( VALUE_DELIM == *( pname_start + strlen( var_name ) ) ) {
            if( pname_start != cgi_input ) {
                if( *( pname_start - 1 ) != CGIVAR_DELIM ) {
                    pname_start = strstr(pname_start + strlen( var_name ), var_name );
                    continue;
                }
            }
            scanValue(pname_start + strlen( var_name ) + 1, value, value_size );
            return strlen( value );
        }
        pname_start = strstr( pname_start + strlen( var_name ), var_name );
    }
    return 0;
}
CGI and State

- The CGI protocol is totally stateless, so how does CGI maintain information from one form to the next
- **Hidden fields** in the HTML document returned in HTTP response message
• Solving HTTP's statelessness
  – Cookies - server data held on the client
  – Microsoft's Active Server Pages (ASPs) – extend cookies with session objects on the servers to represent their client

• Solving CGI inefficiency
  – Microsoft's ASP and JavaSoft's Java Server Pages (JSP) - introduce server-side scripts within HTML pages that can invoke objects. The server page typically executes a server-side script that knows how to instantiate and invoke object.

• Dynamic HTML - every element in an HTML page is a scriptable object.

• The problems with these approaches are:
  – They still require HTTP and the Web server to mediate between objects running on client and objects running on the server.
  – There is no way for a client object to directly invoke a server object.
  – The HTTP form is still the basic unit of client/server interaction.
  – It leads to poor scalability
The Java Object Era

- Java lets us write small component-like programs called applets that can be downloaded into a browser that is Java-compatible (so called Java powered)
- Applets allow us to distribute executable code across the Web along with data
- Java applets allow us to create highly interactive Web pages that have locally executable content
- Server becomes a warehouse of programs, data, and HTML pages
1. Request the applet
   - A Web browser requests a Java applet when it encounters an HTML `<APPLET>` tag. The tag's attributes include the name of the program – the class filename.

2. Receive the applet
   - The browser initiates a separate TCP/IP session to download the applet (treated like any other HTML object, e.g. an external image).

3. Load and execute the applet
   - The browser loads the applet into the client's memory, and then executes it.

4. Discard the applet
   - The browser deletes the applet from memory when it exits the Web page.
Java’s Mobile Code System

- In addition to exchanging traditional content { e.g. text, graphics, audio, and video - Java lets Web applications exchange mobile code
- Foundation technology for mobile agents
- Mobile agents are called Java applets, which are self-contained pieces of executable code
- Advantages:
  1. Write an application once which can run anywhere
  2. Internet becomes the distribution vehicle for software applications. Many customers can be reached instantly with very low software distribution costs
  3. Minimum effort for porting and end-user installation
• Mobile code must be **portable** and **safe**
  1. A safe environment for executing mobile code
     • Control the execution environment of the mobile code
  2. Platform-independent services
     • Must be executable on a variety of O.S. and hardware platforms
  3. Life cycle control
     • Loading, unloading, executing the mobile code
  4. Applet distribution
     • Need facilities for moving applets across the network. It must guarantee that the code is not tampered with when in transit. Can certify the applet and authenticate the identities of both clients and servers
       - Prevent mobile code to become virus
Java Virtual Machine

- Abstract computing machine
- It has an instruction set
- It is usually interpreted, but it may also be implemented by compiling its instruction set to that of a real CPU, much like a conventional programming language
- It may also be implemented in microcode, or directly in silicon
- The Java Virtual Machine knows nothing of the Java programming language
- A Java program is “compiled” into a set of instructions
  - Java Virtual Machine instructions (byte-codes)
- The bytecodes and a symbol table (as well as other ancillary information) are stored in a “class” file
- Java-powered browser
  - The browser is equipped with a Java Virtual Machine which can understand Java bytecodes
Java's Defense System

- Various levels of protection or mechanisms (perimeters), for example:
  - The Java language defense perimeter
    - Do not support pointers in the traditional C and C++ sense. Java compiled code references memory via symbolic “names” that are resolved to real memory addresses at run-time by the Java interpreter
  - The verifier's defense perimeter
    - Protect against hostile compilers, code tamperers, faulty bytecodes
  - Sun is working on an applet certification scheme that uses public key technology to certify that an applet comes from a reputed source
Ultimate Java Client Machine

- RISC chips that natively execute Java byte-codes
  - Run 50 times faster than current implementations
    - e.g. PicoJAVA, MicroJAVA, UltraJAVA
- Java “operating system”
  1. Java language
  2. Java libraries
  3. Java Component Services
- Java Application Environment
  - Java-powered browsers
WHAT IS ACTIVE SERVER PAGE (ASP)?

- It's a standard HTML file that has been extended with additional features. Therefore, it can contain anything normally in an HTML file
  - (e.g.: Java applet, client-side script, client-side ActiveX control)
- Additional features that make ASP Unique
  - It can contain server-side scripts (Jscript, VBScript) which create Web pages with dynamic content. (e.g.: display different messages at different times of the day)
  - It provides a no. of built-in objects which make the script powerful. These objects allow both retrieving information from/sending information to browser.
    - (e.g.: Request object: retrieve information that a user has posted in an HTML form and respond to that info within a script)
  - ASP can be extended with additional no. of standard server-side ActiveX components. Moreover, you also can create additional ActiveX components
  - ASP can interact with a database. (SQL Server, Access, Oracles)
How ASP works?

- Microsoft introduced ASP with the 3rd release of IIS (Internet Information Server). ASP transformed IIS from being a mere server of static content to being a server of dynamic content.

- Function of IIS (and other Web server)
  - The main function of IIS is to serve static HTML page. (When a person requests a Web page from a Web site using IIS, the server fetches a static HTML file from disk/memory and sends to the browser)
    - It acts as an efficient interface between browsers and a bunch of files sitting on the Web server's hard drive. IIS is no different from any Web server in this respect since the main function of any Web server is to serve HTML files.
• Process of serving an HTML page in a general Web server
  1. User enters Internet address of an HTML file in browser to request a Web page
  2. Browser sends a request for the requested Web page to a Web server such as IIS
  3. Web server receives request and recognizes it as an HTML file (html/htm ext)
  4. Web server retrieves the proper HTML file from disk/memory and sends the file back to browser
  5. Html file is interpreted and displayed in the browser window

• ASP changed all these.
  – While IIS still serves static HTML pages. ASP allows IIS to serve dynamic content as well.
  – Using ASP, pages can response to user requests.
  – The Web server itself becomes active in the process of creating Web page.
• Process of Serving an ASP in IIS
  1. User enters Internet address of an ASP file (http://..../hello.asp)
  2. Browser sends the request for an ASP to IIS
  3. Web server receives request and recognizes it as an ASP file
  4. Web server retrieves the proper ASP file from disk/memory
  5. Web server sends file to a special program named ASP.dll
  6. The ASP file is processed from top to bottom and execute all commands.
     The result of this process is a standard HTML file
     From server side: All the commands in an ASP must be executed to create an HTML page. This allows an ASP to contain dynamic contents
     From client side: An ASP is almost exactly the same as a normal HTML page. This allows an ASP to be compatible with all the browsers.
     The only difference is file extension (.asp/.htm)
  7. The HTML file is sent back to the user's browser
  8. The HTML file is interpreted and displayed in the browser. Moreover, an ASP needs not be recompiled every time it's requested. If page has previously been requested and not be altered, the ASP will be retrieved from the cache instead of being processed again.