Traditional Web-Based Systems

- The Web is a huge distributed system consisting of millions of clients and servers for accessing linked documents
  - Servers maintain collections of documents
  - Clients provide users an interface for accessing and displaying documents

The organization of a traditional Web site.
Web Documents

Most web documents consist of two parts:

• The main part is written in a markup-language such as HTML and XML
• The second part consists of embedded documents
  – Each embedded document has an type
  – Example types: image, audio, video, application (msword, pdf, etc.)

Uniform Resource Locator (URL)

• URLs are used to refer to web documents
• A URL consists of three parts
  – The protocol for transferring the document across the network
  – The DNS name of the server
  – The pathname of the file
• Example URL:
  http://www.cs.iastate.edu/~cs554/news.html
Multitiered Architectures

• Many web sites today support dynamically generated web pages
• These web sites are organized as a three-tiered architecture consisting of a web server, an application server, and a database
  – The web server forwards the client request to the application server
  – The application server accesses the database and generates a web page dynamically using server-side scripting technologies (e.g., Microsoft’s ASP, Sun’s JSP, PHP)

Web Services

• A web service is a service made available over the Internet
  – A web service provides an interface consisting of a collection of operations
  – The operations can be invoked by client programs over the Internet
  – E.g., Amazon web services allow client programs to get information about products, order items, check the status of an order
• Standardization is needed to allow web services to be discovered and accessed over the Internet by client programs
Web Services Architecture

Simple Object Access Protocol (SOAP)

- SOAP is the standard protocol for communication with web services
  - SOAP messages are expressed in XML
  - SOAP messages are transferred using HTTP or SMTP
- Two interaction styles
  - **Conversational-style exchange**: Client sends a document (e.g., a purchase order) to server and gets back a response document (e.g., a confirmation document)
    - Messages are transferred using either HTTP or SMTP
  - **RPC-style exchange**: The client’s SOAP message identifies the procedure to be called, and provides a list of parameter values
    - Messages are transferred using HTTP
Web Server Clusters

• To improve performance and availability, a web service is often implemented using a cluster of servers and a front end
  – Server is replicated on multiple nodes
  – The front end redirects client requests to one of the replicas

Front End Design

• **Transport-layer switch**: Front end performs *TCP handoff*, i.e., it forwards an incoming TCP connection request to one of the servers, considering load on servers

  Server inserts IP address of the switch as the source IP address in its response packet

• **Content-aware request distribution**: Front end reads the content of the HTTP request and then selects the best server
  – E.g., if the front end always forwards requests for the same document to the same server, the server may cache the document resulting in better response times
Round-Robin DNS

• Another way to set up a web server cluster is to use **round-robin DNS** instead of using a front end
  – A single domain name is associated with multiple IP addresses, each corresponding to one of the web servers
  – When resolving the host name of a Web site, the DNS server returns the list of multiple IP addresses
    • DNS server circulates the entries of the list it returns
  – The browser always chooses the first IP address on the list → requests are distributed over the servers in the cluster

Hypertext Transfer Protocol

• Web clients and web servers communicate using the Hypertext Transfer Protocol (HTTP)
  – Servers listen on port 80
  – A client sends a HTTP request to a server and waits for a HTTP response
• HTTP is based on TCP
  – A client first sets up a TCP connection to the server and then sends request message and receives response message using the connection
HTTP Connections

• A web page often contains a collection of embedded documents stored on the same server
  – Each embedded document needs a separate request
• HTTP version 1.0 uses **nonpersistent** connections
  – Each request to a server requires setting up a separate TCP connection, which is costly
• HTTP version 1.1 uses **persistent** connections
  – Several requests can be issued on the same TCP connection

HTTP Operations

• **GET** - request to fetch a document from the server
• **HEAD** - same as GET, except it asks the server to return the response headers only
• **POST** - send data to the server to be processed in some way
• **PUT** - request to store a document on the server
• **DELETE** - request to delete a document on the server
HTTP Request Message Format

(a)

An Example HTTP Request Message

- **Request line**
  - GET /courses/spring2007.html HTTP/1.1
  - Host: www.someschool.edu
  - User-Agent: Mozilla/4.0
  - Connection: close
  - Accept-Language: fr

- **Header lines**
  - (extra carriage-return-line-feed)

- **Blank line**
  - Message body (empty)
HTTP Response Message Format

![Diagram of HTTP Response Message Format]

An Example HTTP Response Message

![Diagram of An Example HTTP Response Message]