Web Servers

Topics

- Web Server Definition
- Web Site versus Web Server
- Steps in Handling a Client Request
- Access Control
- Dynamically Generated Responses
- Passing Data to/from the Script
- Creating and Using Cookies
- Sharing Information Across Requests
- Server Architecture
- Server Hosting
- Case Study of the Apache Web Server

Web Server Definition

- A Web server is a program that generates and transmits responses to client requests for Web resources.
- Handling a client request consists of several key steps:
  - Parsing the request message
  - Checking that the request is authorized
  - Associating the URL in the request with a file name
  - Constructing the response message
  - Transmitting the response message to the requesting client

Web Server Definition

- The server can generate the response message in a variety of ways:
  - The server simply retrieves the file associated with the URL and returns the contents to the client.
  - The server may invoke a script that communicates with other servers or a back-end database to construct the response message.

Web Site versus Web Server

- ***Web site*** and ***Web server*** are different:
  - A ***Web site*** consists of a collection of Web pages associated with a particular hostname.
  - A ***Web server*** is a program to satisfy client requests for Web resources.

Steps in Handling a Client Request

- A Web server proceeds through the following steps in handling an HTTP request:
  - Read and parse the HTTP request message

    for example GET the resource /foo.htm
  - Translate the URL to a file name

    for example the resource be located in the base directory such as /www, where the URL

    http://www.bar.com/foo/index.html corresponds to

    the file of www/foo/index.html
  - Determine whether the request is authorized
  - Generate and transmit the response that includes header to show the status information

Access Control

- A Web server may limit which users can access certain resources. Access control requires a combination of ***authentication*** and ***authorization***.
  - Authentication identifies the user who originated the request.
  - Authorization determines which users have access to a particular resource.

AUTHENTICATION

- Most client-server systems authenticate a user by asking for a name and password.
- Web server must perform authentication for ***every*** request for a resource that has access restrictions.
- The server returns an HTTP response that indicates that the request requires authorization.
- The response also identifies what kind of authentication is required.
- The response also identifies the ***realm***
  - a string that associates a collection of resources at the server

AUTHORIZATION

- To control access to Web resources, the server must employ an authorization policy.
- A policy typically expressed in terms of an access control list that enumerates the users who are granted or denied access to the resources.
- In addition to checking the user name, the server may allow or deny access to the resource based on other information associated with the HTTP request, such as the host name or IP address of the requesting client.
- Authenticating HTTP requests can impose a heavy load on the Web server.

Dynamically Generated Responses

- This feature differentiates the Web from earlier file transfer services on the Internet.
- Dynamically generated responses are created in a variety of ways:
  - Server-side include
  - Server script

Server-Side Include

- A ***server-side include*** instructs the Web server to customize a static resource based on directives in an HTML-like file.

Server Script

- A ***server script*** is a separate program that generates the request resource.
- The program may run as
  - Part of the server
  - A separate process
- The main role of the Web server is
  - To associate the requested URL with the appropriate script
  - To pass data to/from the script
- The main role of the script is
  - To process the input from the server
  - To generate the content to the client

Server Script

- The server can interact with the script in several different ways:
  - Separate process invoked by the server
  - Software module in the same process
  - Persistent process contacted by the server

Passing Data to/from the Script

- Decoupling the scripts from the Web server requires a well-defined interface for passing data between the two pieces of software.
- Common Gateway Interface (CGI) defines interfaces for a variety of operating system platforms.

Table 4.1

Table 4.1. Example CGI environment variables

Type	Variable	Example
Server	SERVER_NAME SERVER_SOFTWARE SERVER_PROTOCOL SERVER_PORT DOCUMENT_ROOT GATEWAY_INTERFACE	www.bar.com Apache/10206 HTTP/1.0 80 /www CGI/2.0
Client	REMOTE_ADDR REMOTE_HOST	10.9.57.188 users.berkeley.edu
Request	CONTENT_TYPE CONTENT_LENGTH REQUEST_METHOD QUERY_STRING ACCEPT_LANGUAGE HTTP_USER_AGENT	Text/html 158 GET NAME=Noam+Chomsky De-CH Mozilla/2.0

Creating and Using Cookies

- Cookies are typically created, used, and modified by scripts invoked to generate dynamic responses, rather than by the Web server.
- The browser can be instructed to include a unique cookie in each HTTP request.
- If the request does not include cookie, the script create a new cookie and include the cookie in the header of the response message

  Set-Cookie: Customer="user17"; Version="1"; Path="/book"

  Subsequent requests from the user would include the cookie

  Cookie: Customer="user17"; Version="1"; Path="/book"

Creating and Using Cookies

- Cookies are typically created, used, and modified by scripts invoked to generate dynamic responses, rather than by the Web server.
- The browser can be instructed to include a unique cookie in each HTTP request.
- If the request does not include cookie, the script create a new cookie and include the cookie in the header of the response message

  Set-Cookie: Customer="user17"; Version="1"; Path="/book"

  Subsequent requests from the user would include the cookie

  Cookie: Customer="user17"; Version="1"; Path="/book"

Creating and Using Cookies

- A script can use a cookie as a user identifier in interacting with a back-end database.
- Storing history information in the cookie may obviate the need to retain information about the user in a back-end database.

Sharing Information Across Requests

- A Web server may retain some information to reduce the overhead of handling future requests by:
  - Sharing HTTP responses across requests
  - Sharing metadata across requests

Sharing HTTP Responses Across Requests

- In server-side caching, data from the disk is cached in main memory at the server.
- A Web server cache can be
  - Static files
  - Dynamically generated responses
- the server would need to ensure that the cached result is consistent with the primary copy of the information.

Sharing Metadata Across Requests

- The server could store the information generated in the process such as:
  - Translation of URL to file name
  - Control information about the resource
  - HTTP response headers
- The server could cache certain information across requests for different resources:
  - Current date/time
  - Client name

Sharing Metadata Across Requests

- The server could store the information generated in the process such as:
  - Translation of URL to file name
  - Control information about the resource
  - HTTP response headers
- The server could cache certain information across requests for different resources:
  - Current date/time
  - Client name

Server Architecture

- Some techniques for allocating system resources among competing client requests are :
  - Event-driven server architecture
  - Process-driven server architecture
  - Hybrid server architecture

Event-Driven Server Architecture

- An ***event-driven*** server
  - Has a single process that alternates between servicing different requests
  - Allows the server to serialize operations that modify the same data
  - Performs nonblocking system calls
  - Not used in Most high-end Web servers

Process-Driven Server Architecture

- A ***process-driven*** server
  - Allocates each request to a separate process
    - One master process listens for new connection
    - The master process creates, or forks, a separate process for each new connection
  - Terminates the process after parsing the client request and transmitting the response
    - To prevent memory leak
  - Introduces overhead for switching from one process to another

Hybrid Server Architecture

- In ***Hybrid*** server architectures
  - The strengths of the event-driven and process-driven models are combined
  - Each process would become an event-driven server that alternates between a small collection of requests
  - A single process has multiple independent threads
  - Main process instructs a separate helper process to perform time-consuming operations

Server Hosting

- Multiple Web sites on a single machine
- Multiple machine for a single Web site
  - Figure 4.1.

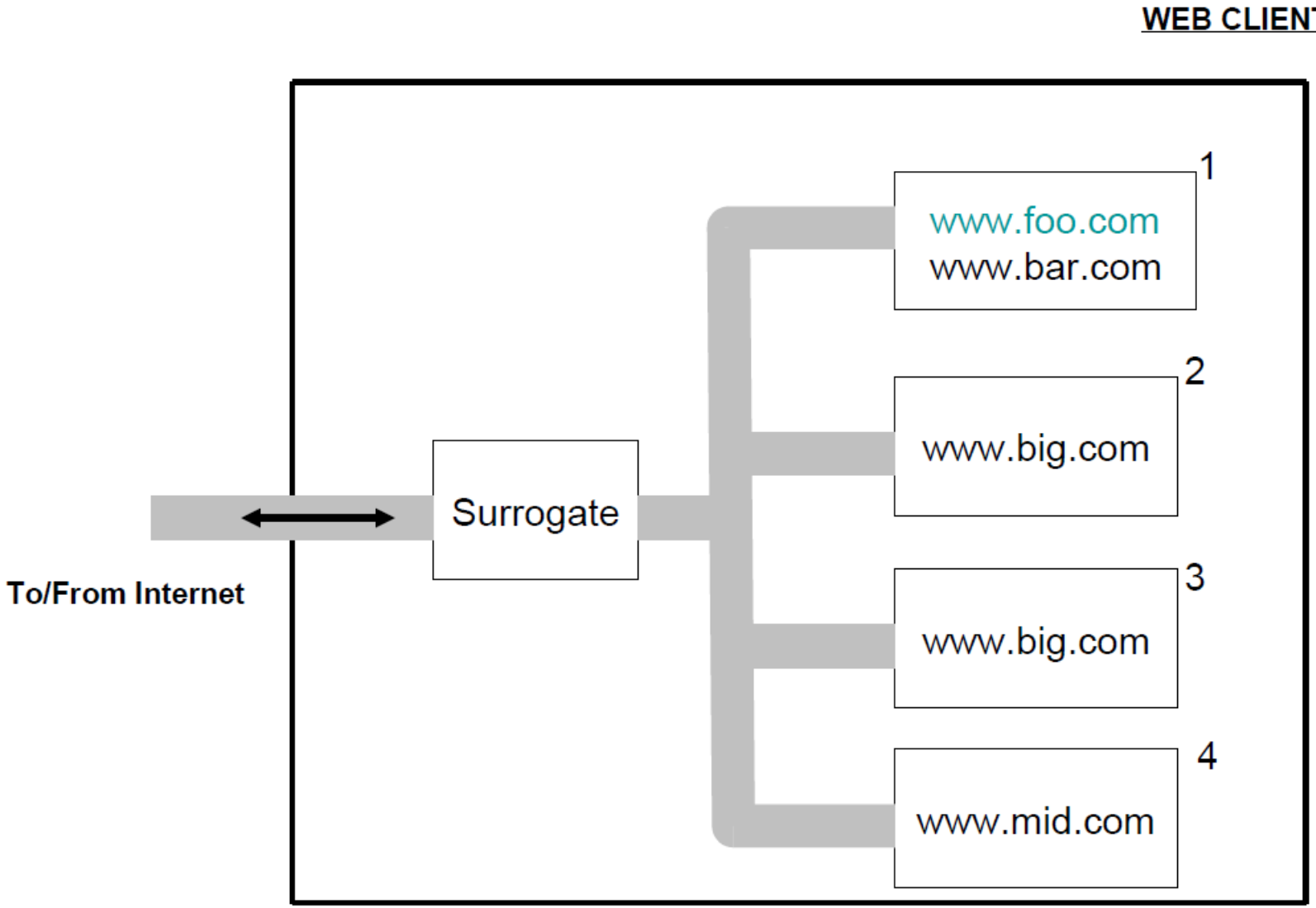

Figure 4.1. Hosting complex with surrogate in front of four server machines

Apache Web Server

- The Apache server follows the process-driven model, with a parent process that assigns a process to each new connection.

Table 4.2 Key configuration directives for child processes and network connection

Directive	(Definition (default value in Apache 1.3.3
StartServers	Initial number of child processes (5)
MaxClients	Maximum number of child processes (256)
MinSpareServers	Target Minimum number of idle children (5)
MaxSpareServers	Target Maximum number of idle children (10)
MaxRequestsPerChild	Maximum number of requests per child (30)
ListenBacklog	Maximum number of pending connections (511)
SendBufferSize	Size of the TCP send buffer (OS default)
MaxKeepAliveRequests	Max number of requests per connection (100)
KeepAliveTimeout	Maximum idle time for connection (15 sec)

Apache Web Server

- Apache has a collection of handlers that perform an action using the requested file, as shown in Table 4.3.

Table 4.3 Built-in handlers in Apache server and default file extension

Handler	(Purpose (file extension
Default-handler	Send the file as static content
Send-as-is	Send the file as an HTTP response message (.asis)
Cgi-script	Invoke the file as a CGI script (.cgi)
Server-parsed	Treat the file as a server-side include (.shtml)
Imap-file	Treat the file as an imagemap rule file (.imap)
Type-map	Treat the file as a map for content negotiation (.var)
Server-info	Get the server's configuration information
Server-status	Get the server's status report