

CSCI 241

The HTTP Protocol and The Web Server

*Department of CS & QM
Winthrop University
Fall 2011*

Prepared by X. Wang

The HTTP Protocol

- The HyperText Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems
- HTTP has been in use by the World-Wide Web global information initiative since 1990
- HTTP is a stateless protocol that can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers

Prepared by X. Wang

2

The HTTP Protocol - Versions

- The first version of HTTP, referred to as HTTP/0.9, was a **simple** protocol for raw data transfer across the Internet
- HTTP/1.0, as defined by RFC1945, improved the HTTP/0.9 by allowing messages to be in the format of **MIME-like** messages
- The current version is **HTTP/1.1** that includes more stringent requirements than HTTP/1.0 in order to ensure reliable implementation of its features
- HTTP/1.1 specification can be found at <http://www.w3.org/Protocols/rfc2616/rfc2616.html>

Prepared by X. Wang

3

The HTTP Protocol - Principle

- An **HTTP conversation** between client and server consists of a **client request** and a **server response** over a single **TCP connection**
- HTTP uses the default port **TCP 80**, but other ports can also be specified
- The client initiates the conversation with an HTTP request to the server
- The server then fulfills the request, for example by returning an HTML document or sending an error message as the response, and then closes the connection

Prepared by X. Wang

4

The HTTP Protocol – Persistent Connection

- A significant difference between HTTP/1.1 and earlier versions of HTTP is **persistent connections** that are the default behavior of any HTTP connection
- Prior to persistent connections, a separate TCP connection was established to fetch each URL
- This increases the load on HTTP servers and causes congestion on the Internet
- The use of inline images and other associated data often require a client to make multiple requests of the same server in a short amount of time

Prepared by X. Wang

5

The HTTP Protocol – Persistent Connection

- In HTTP/1.1, unless indicated, the client should assume that the server will maintain a persistent connection
- Persistent connections provide a mechanism by which a client and a server can signal the close of a TCP connection
- With persistent connection, HTTP requests and responses can be pipelined on a connection, network congestion is reduced, and latency on subsequent requests is reduced

Prepared by X. Wang

6

HTTP Requests

- An HTTP request message contains information about a resource on the server and the action the client wishes the server to perform on the resource
- An HTTP request message is from a client to a server
- It includes a **Request-line**, **optional headers**, a **CRLF** and an **optional body**:

```
Request-line CRLF
[general-header | request-header | entity-header CRLF]
CRLF
[body]
```

Prepared by X. Wang

7

HTTP Requests

- The first line of a request message is a Request-line
- A Request-line has the following format:
`Method Request-URI HTTP-Version CRLF`
- The Method indicates the method to be performed on the resource identified by the Request-URI. The Method is case-sensitive
- HTTP/1.1 defined eight standard methods: **GET**, **POST**, **HEAD**, **OPTIONS**, **PUT**, **DELETE**, **TRACE**, **CONNECT**

Prepared by X. Wang

8

HTTP Requests

- A **Uniform Resource Identifier (URI)** consists of both a virtual path and an optional query string, separated by a query character (?)
`/cgi-bin/code.cgi?query-string`
- The virtual path is a string to identify a resource (document or service) being requested on the server
- The virtual path is *virtual* because it always uses a `/`, independent of the client or server operating system
- **HTTP-Version** indicates the version of HTTP understood by the client. For example: **HTTP/1.1**

Prepared by X. Wang

9

HTTP Requests

- The **request-header** fields allow the client to pass additional information about the request, and about the client itself, to the server
- These fields act as request modifiers, with semantics equivalent to the parameters on a programming language method invocation
- The **body** is used by the **POST** method to indicate a significant body of **data** in the request

Prepared by X. Wang

10

HTTP Requests - GET

- The **GET** method means retrieve whatever information is identified by the Request-URI
- An example

```
GET /assignment.html HTTP/1.1 [CRLF]
If-Modified-Since: Fri, 5 August 2011 8:00:00 GMT [CRLF]
[CRLF]
```

Prepared by X. Wang

11

HTTP Requests - POST

- The **POST** method allows the client to include a significant body of data in a request
- The **Content-length** header is mandatory with a post request; the request headers must be followed by a blank line
- A POST method is usually used to start a server-side application, like CGI, PHP, ASP or JSP

```
POST /php/register.php HTTP/1.1 [CRLF]
Content-Type: application/octet-stream [CRLF]
Content-Length: 2048 [CRLF]
[CRLF]
body
```

Prepared by X. Wang

12

HTTP Requests - HEAD

- The **HEAD** method is identical to **GET** except that the server **MUST NOT** return a message-body in the response
- This method can be used for obtaining meta information about the entity implied by the request without transferring the entity-body itself

```
HEAD /index.html HTTP/1.1 [CRLF]
[CRLF]
```

Prepared by X. Wang

13

HTTP Requests - Methods

- The **OPTIONS** method represents a request for information about the communication options available on the request/response chain identified by the Request-URI
- The **PUT** method requests that the enclosed entity be stored under the supplied Request-URI
- The **DELETE** method requests that the origin server delete the resource identified by the Request-URI
- The **TRACE** method is used to invoke a remote, application-layer loop-back of the request message
- This specification reserves the method name **CONNECT** for use with a proxy

Prepared by X. Wang

14

HTTP URI Encoding/Decoding

- In a client request, to allow arbitrary textual characters to be transmitted in unambiguous ASCII format, the request URI may encode some special characters
- A space is encoded to a + (or %20); certain characters can be encoded to a hexadecimal ASCII value, denoted by a preceding
- For example, `/My Documents/ReadMe!` will be encoded to `/My+Documents/ReadMe%21`

Prepared by X. Wang

15

HTTP URI Encoding/Decoding

- An encoded request-URI will be decoded by the server before the server can process it
- A web server will use reverse rule to decode the URI
- For example, a web server will decode the encoded URI `/My+Documents/ReadMe%21` into `/My Documents/ReadMe!`

Prepared by X. Wang

16

HTTP Responses

- After receiving and interpreting a request message, a server responds with an HTTP **response** message
- An HTTP response message is from a server to a client
- It includes a **Status-line**, **optional headers**, a **CRLF** and an **optional body**:

```
Status-line CRLF
[general-header | request-header | entity-header CRLF]
CRLF
[body]
```

Prepared by X. Wang

17

HTTP Responses

- The first line of a Response message is the **Status-line**
- It consists of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by Space characters
- It is also terminated with a CR and LF and has the following format

```
HTTP-Version Status-Code Reason-Phrase CRLF
```

Prepared by X. Wang

18

HTTP Responses

- The **Status-Code** element is a 3-digit integer result code of the attempt to understand and satisfy the request
- The **Reason-Phrase** is intended to give a short textual description of the Status-Code
- The **Status-Code** is intended for use by automata and the **Reason-Phrase** is intended for the human user
- The client is not required to examine or display the **Reason-Phrase**

Prepared by X. Wang

19

HTTP Responses

- The **first digit** of the Status-Code defines the class of response. The last two digits do not have any categorization role.
- There are 5 values for the first digit:
 - 1xx: Informational - Request received, continuing process
 - 2xx: Success - The action was successfully received, understood, and accepted
 - 3xx: Redirection - Further action must be taken in order to complete the request
 - 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
 - 5xx: Server Error - The server failed to fulfill an apparently valid request

Prepared by X. Wang

20

HTTP Responses – Status Code

```

100: Continue
101: Switching Protocols

200: OK
201: Created
202: Accepted
203: Non-Authoritative Information
204: No Content
205: Reset Content
206: Partial Content

300: Multiple Choices
301: Moved Permanently
302: Found
303: See Other
304: Not Modified
305: Use Proxy
307: Temporary Redirect

```

Prepared by X. Wang

21

HTTP Responses – Status Code

```

400: Bad Request
401: Unauthorized
402: Payment Required
403: Forbidden
404: Not Found
405: Method Not Allowed
406: Not Acceptable
407: Proxy Authentication Required
408: Request Time-out
409: Conflict
410: Gone
411: Length Required
412: Precondition Failed
413: Request Entity Too Large
414: Request-URI Too Large
415: Unsupported Media Type
416: Requested range not satisfiable
417: Expectation Failed

```

Prepared by X. Wang

22

HTTP Responses – Status Code

```

500: Internal Server Error
501: Not Implemented
502: Bad Gateway
503: Service Unavailable
504: Gateway Time-out
505: HTTP Version not supported

```

Prepared by X. Wang

23

HTTP Responses

- The following is an HTTP response message

```

HTTP/1.1 400 Bad Request
Date: Fri, 5 August 2011 02:46:21 GMT
Server: Apache/2.2.8 (Linux/SUSE)
Content-Length: 316
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not
understand.<br /></p>
<hr />
<address>Apache/2.2.8 (Linux/SUSE) Server at www.cs.abc.edu
Port 80</address>
</body></html>

```

Prepared by X. Wang

24

HTTP General Headers

- HTTP/1.1 accepts the following **general-header** fields
- These provide general control information and may be included in a client request or server response

Cache-Control	Controls caching of a document
Connection	Controls connection keep alive
Date	Indicates the local date
Pragma	Specifies implementation-specific options
Trailer	Indicates the headers included in a chunked trailer
Transfer-Encoding	Specifies the transfer encoding used
Upgrade	Requests a protocol upgrade
Via	Indicates proxies and proxy protocols used
Warning	Warns of a possible problem

Prepared by X. Wang

25

HTTP Request Headers

- HTTP/1.1 accepts the following **request-header** fields
- These headers may be included in a client request

Accept	Specifies the acceptable MIME type
Accept-Charset	Specifies the acceptable character sets
Accept-Encoding	Specifies the acceptable data encodings
Accept-Language	Specifies the acceptable languages
Authorization	Specifies client authentication credentials
Expect	Specifies server behavior required by the client
From	Specifies the client's email address
Host	Specifies the server address being accessed
If-Match	Performs the request if certain criteria are met
If-Modified-Since	Performs the request if the object has been modified
If-None-Match	Performs the request if certain criteria are met
If-Range	Returns range of object if conditionals match
If-Unmodified-Since	Performs the request if the object hasn't been modified

Prepared by X. Wang

26

HTTP Request Headers

Max-Forwards	Specifies a proxy forwarding limit
Proxy-Authorization	Specifies authentication credentials for the proxy
Range	Specifies the part of an object that is required
Referer	Specifies the document that led to this request
TE	Specifies the acceptable transfer encodings
User-Agent	Identifies the client software

Prepared by X. Wang

27

HTTP Response Headers

- HTTP/1.1 accepts the following **response-header** fields
- These headers may be included in a server response

Accept-Range	Specifies which range requests are acceptable
Age	Specifies how old a cached object is
ETag	Specifies the entity tag of the object
Location	Specifies a redirect location
Proxy-Authenticate	Proxy authentication challenge
Retry-After	Specifies how long a service will be unavailable
Server	Identifies the server software
Vary	Identifies which headers affect an object's content
WWW-Authenticate	Client authentication challenge

Prepared by X. Wang

28

HTTP Entity Headers

- HTTP/1.1 accepts the following **entity-header** fields
- These headers may be included in a client request or a server response

Allow	Specifies the request methods supported by the object
Content-Encoding	Specifies the encoding of the object
Content-Language	Specifies the language of the object
Content-Length	Specifies the length of the object
Content-Location	Specifies the actual location of the object
Content-MD5	Specifies an MD5 message digest of the object
Content-Range	Specifies the range of object included
Content-Type	Specifies the MIME type of the object
Expires	Specifies when the returned object expires
Last-Modified	Specifies when the returned object was last modified

Prepared by X. Wang

29

MIME types

- Multipurpose Internet Mail Extensions (MIME) is a mechanism, originally designed for email, to associate a type with a message so that the message receiver can understand how to decode or view it
- MIME allows different types of message content to be sent across the Internet
- MIME is defined in RFC 1521
- The seven top-level MIME types are **text**, **image**, **audio**, **video**, **multipart**, **application**, and **message**

Prepared by X. Wang

30

MIME types

- The following are some of the common content types that are delivered over the HTTP, and include the file extensions that are most commonly associated with such files
- An HTTP server response usually include the content type of the document in a [Content-Type](#) header

Prepared by X. Wang

31

MIME types

MIME Type	File Extension
application/octet-stream	.bin .dms .lha .lzh .exe .class
application/postscript	.ai .eps .ps
application/rft	.rft
application/x-compress	.Z
application/x-gtar	.gtar
application/x-gzip	.gz
application/x-httpd-cgi	.cgi
application/zip	.zip
audio/basic	.au .snd
audio/mpeg	.mpga .mp3
audio/x-aiff	.aif .aiff .aifc
audio/x-pn-realaudio	.ram
audio/x-pn-realaudio-plugin	.rpm
audio/x-realaudio	.ra
audio/x-wav	.wav

Prepared by X. Wang

32

MIME types

MIME Type	File Extension
image/gif	.gif
image/ief	.ief
image/jpeg	.jpeg .jpg .jpe
image/png	.png
image/tiff	.tiff .tif
image/x-cmu-raster	.ras
image/x-portable-anymap	.pnm
image/x-portable-bitmap	.pbm
image/x-portable-graymap	.pgm
image/x-portable-pixmap	.ppm
image/x-rgb	.rgb
image/x-xbitmap	.xbm
image/x-xpixmap	.xpm
image/x-xwindowdump	.xwd

Multipart/mixed

Prepared by X. Wang

33

MIME types

MIME Type	File Extension
text/html	.html .htm
text/plain	.txt
text/richtext	.rtx
text/tab-separated-values	.tsv
text/sgml	.sgml .sgm
video/mpeg	.mpeg .mpg .mpe
video/quicktime	.qt .mov
video/x-msvideo	.avi
video/x-sgi-movie	.movie

Prepared by X. Wang

34

The World Wide Web

- The World Wide Web is a distributed client and server system for publishing and delivering content over the Internet using HTTP
- A Web Server processes HTTP requests and serves responses
- The term "Web Server" can refer either to web server software or to the particular device or computer dedicated to serving the web pages
- A Web client is a local computer that accesses resources on the Web server through URLs and displays the resources in a Web browser

Prepared by X. Wang

35

Web Servers

- Web servers come in all flavors, shapes, and size
- There are trivial 10-line Perl script web servers, 50-MB secure commerce engines, and tiny servers-on-a-card
- But whatever the functional differences, all web servers receive HTTP requests for resources and serve content back to the clients
- Web servers implement HTTP and the related TCP connection handling
- They also manage the resources served by the web server and provide administrative features to configure, control, and enhance the web server

Prepared by X. Wang

36

Web Servers

- General-Purpose Software Web Servers
- These web servers run on standard, network-enabled computer systems
- Open source software: Apache, W3C's Jigsaw
- Commercial software: Microsoft IIS, Sun iPlanet

Prepared by X. Wang

37

What Real Web Servers Do

- Web servers perform several common tasks:
 1. Set up connection: accept a client connection, or close if the client is unwanted
 2. Receive request: read an HTTP request message from the client connection
 3. Process request: interpret the request message and take action
 4. Access resource: access the resource specified in URI
 5. Construct response: create the HTTP response message with the right headers
 6. Send response: send the response back to the client
 7. Log transaction: place notes about the completed transaction in a log file

Prepared by X. Wang

38

What Real Web Servers Do

- Step1: Accepting Client Connections
- When a client request a TCP connection to the web server, the web server establishes the connection and extracts the client's IP address from the connection
- Once a new connection is established and accepted, the server adds the new connection to the list of existing web server connections and prepares to watch for data on the connection
- The web server is free to reject and immediately close any connection (for unauthorized or malicious client)

Prepared by X. Wang

39

What Real Web Servers Do

- Step2: Receiving Request Messages
- As the data arrives on connections, the web server reads out the data from the client connection
- Parses the request line looking for the request method, the specified URI, and the version number, each separated by a single space, and ending with a CRLF
- Reads the messages headers, each ending in CRLF
- Detects the end-of-headers blank line, ending in CRLF
- Reads the request body if any length specified by the Content-Length header

Prepared by X. Wang

40

What Real Web Servers Do

- Step3: Processing Requests
- Once the web server has received a request, it can process the request using the method, resource, headers, and optional body
- Some methods (e.g., POST) require entity body data
- Other methods (e.g., OPTIONS) allow a request body but don't require one
- A few methods (e.g., GET) forbid entity body data in request messages

Prepared by X. Wang

41

What Real Web Servers Do

- Step4: Mapping and Accessing Resources
- Web servers are resource servers
- They deliver precreated content, such as HTML pages or JPEG images, as well as dynamic content from resource-generating applications running on the servers
- Before the web server can deliver content to the client, it needs to identify the source of the content, by mapping the URI from the request message to the proper content or content generator on the server

Prepared by X. Wang

42

What Real Web Servers Do

- Step4: Mapping and Accessing Resources
- Web servers support different kind of resource mapping
- The simplest form uses the request URI to name a file in the web server's file system
- Typically, a special folder in the web server file system is reserved for web content
- This folder is called the document root, or docroot
- The web server takes the URI from the request message and appends it to the document root

Prepared by X. Wang

43

What Real Web Servers Do

- Step4: Mapping and Accessing Resources
- Virtually hosted docroot:
 - Virtually hosted web servers host multiple web sites on the same web server, giving each site its own distinct document root on the server
 - A virtually hosted web server identifies the correct document root to use from the IP address or hostname in the URI or the Host header
 - Two web sites hosted on the same web server can have completely distinct content, even if the URIs are identical

Prepared by X. Wang

44

What Real Web Servers Do

- Step4: Mapping and Accessing Resources
- User home directory docroots:
 - Another common use of docroots gives people private web sites on a web server
 - A typical convention maps URIs whose paths begin with a slash and tilde (/~) followed by a username to a private document root for the user
 - The private docroot is often the folder called public_html inside the user's home directory

Prepared by X. Wang

45

What Real Web Servers Do

- Step4: Mapping and Accessing Resources
- Directory Listings:
 - A web server can receive request for directory URLs, where the path resolves to a directory, not a file
 - Most web servers can be configured to take a few different actions when a client requests a directory URL: return an error; return a special, default "index file"; scan the directory and return an HTML containing the directory listing
 - Most web servers look for a file named index.html, index.htm or default.htm inside the specified directory (more default files may be specified in the web server configuration)

Prepared by X. Wang

46

What Real Web Servers Do

- Step4: Mapping and Accessing Resources
- Dynamic Content Resource Mapping:
 - Web servers can also map URIs to dynamic resources – that is, to programs that generate content on demand
 - The web server needs to know when a resource is a dynamic resource, where the dynamic content generator program is located, and how to run the program
 - Most web servers provide basic mechanism to identify and map dynamic resources by indicating a special path in URI, or a special file extension name

Prepared by X. Wang

47

What Real Web Servers Do

- Step5: Constructing Responses
- Once the web server has identified the resource, it performs the action described in the request method and returns the response message
- The response message contains a response status code, response headers, and a response body if one was generated
- For the response body, a Content-Type header is required to describe the MIME type of the body, and a Content-Length header to indicate the size of the body

Prepared by X. Wang

48

What Real Web Servers Do

- Step5: Constructing Responses
- Redirection:
 - Web servers sometimes return redirection responses instead of success messages
 - A web server can redirect the browser to go elsewhere to perform the request
 - A redirection response is indicated by a 3XX status code
 - The `Location` response header contains a URI for the new or preferred location of the content

Prepared by X. Wang

49

What Real Web Servers Do

- Step6: Sending Responses
- Web servers use correct client connection to send out the response messages
- For nonpersistent connections, the server is expected to close its side of the connection when the entire message is sent
- For persistent connections, the connection may stay open, in which case the server needs to be extra cautious (negotiation) to handle the connection

Prepared by X. Wang

50

What Real Web Servers Do

- Step7: Logging
- Finally, when a transaction is complete, the web server notes an entry into a log file, describing the transaction performed

Prepared by X. Wang

51

Apache Web Server

- The Apache HTTP Server is an open source code implementation of an HTTP (Web) server
- The Apache Server works on Unix and Windows platforms
- From <http://httpd.apache.org/download.cgi>, we can download the latest Apache web server
- The current best available version is Apache 2.2.20
- On Windows, we can download *Win32 Binary (MSI Installer)* `apache_2.2.20-win32-x86-no_ssl.msi`

Prepared by X. Wang

52

Apache Web Server

- Run the downloaded file to install the Apache Server
- Click `Next>`, check `I accept the terms in the license agreement`, click `Next>`, `Next>`
- On Server Information page, fill in `Network Domain`, `Server Name`, and `Administrator's Email Address`, check for `All Users`, on `Port 80`, click `Next>`
- Select `Typical`, click `Next>`
- Use the default installation folder, click `Next>`
- Click `Install` to start the installing process
- Click `Finish` to finish the installation

Prepared by X. Wang

53

Apache Web Server

- Apache server is installed under `C:\Program Files\Apache Software Foundation\Apache2.2`
- The docroot is `C:\Program Files\Apache Software Foundation\Apache2.2\htdocs`
- Open a web browser, enter URL as <http://localhost/>
- We can see the installation successful information
- We can `start, stop or configure` the Apache server from `Apache HTTP Server 2.2 in All Programs` of Start menu

Prepared by X. Wang

54

Microsoft IIS

- Microsoft Internet Information Service (IIS) is another web server provided with Windows XP Professional
- Open Start -> Control Panel, select Add or Remove Programs, click Add/Remove Windows Components from the left column, check Internet Information Service (IIS) from the panel, and click Next>
- Then computer will begin to install IIS (The Windows XP Pro CD-ROM maybe required at this step)
- Follow the instruction to finish the IIS installation

Prepared by X. Wang

55

Microsoft IIS

- If installation is successful, in Control Panel, open Administrative Tools, we can see Internet Information Services there
- Through this tool, we can start, stop and configure the IIS
- For IIS, the docroot is `C:\Inetpub\wwwroot`

Prepared by X. Wang

56