


The HTTP Protocol

WAVV 2006
Chattanooga, TN

Understanding the HTTP Protocol

WAVV 2006


Chuck Arney
illustro Systems International LLC
carney@illustro.com

 Copyright © 2006 illustro Systems International, LLC

Handouts


- Download a copy of this presentation

<http://www.illustro.com/conferences>

 Copyright © 2006 illustro Systems International, LLC WAVV2006-2

Agenda

- What is the HTTP protocol?
- Uses for HTTP
- Uniform Resource Locators
- MIME
- Structure of an HTTP Transaction
- HTTP Methods

 Copyright © 2006 illustro Systems International, LLC WAVV2006-3

The HTTP Protocol

Agenda

- Persistent Connections
- Content Negotiation
- Conditional Requests
- Proxy Servers
- Caching
- Resources
- Web 2.0



Copyright © 2006 illustro Systems International, LLC

WAVV2006-4

What is the HTTP Protocol?

- HTTP is the acronym for **Hypertext Transfer Protocol**
- Application level protocol
 - Same level as protocols such as FTP & SMTP
- Request/Response processing model
- Stateless protocol



Copyright © 2006 illustro Systems International, LLC

WAVV2006-5

What is the HTTP Protocol?

- Bi-directional data transfer (TCP Sockets)
- Support for negotiation, caching & proxies



Copyright © 2006 illustro Systems International, LLC

WAVV2006-6

The HTTP Protocol

Uses for HTTP

- The World Wide Web, of course
- XML Web Services (SOAP)
- Control network devices such as routers and printers
- **Actually –**
 - Any application requiring a Request/Response processing model



Copyright © 2006 illustro Systems International, LLC

WAVV2006-7

URI, URL, URN

- URI, Universal Resource Identifier (RFC 2396)
 - Identifies a resource by name, location or other characteristic
 - Can be
 - URL, Universal Resource Locator
 - URN, Universal Resource Name



Copyright © 2006 illustro Systems International, LLC

WAVV2006-8

Dissecting a HTTP URL

http://www.domain.com:80/dir/file?a=3&b=1

^----- protocol
^----- host
^----- port
^----- path
^----- query string



Copyright © 2006 illustro Systems International, LLC

WAVV2006-9

The HTTP Protocol

MIME

- Multipurpose Internet Mail Extensions
 - RFC 2045, November 1996 (part 1)
- Defines standards for text message formats and character sets
 - Including many of the message headers used in HTTP messages
- HTTP messages are “MIME-like” but not completely MIME compliant



Copyright © 2006 illustro Systems International, LLC

WAVV2006-10

Structure of an HTTP Transaction

- The client opens a connection and sends a request message to an HTTP server
- The server returns a response message, usually containing the resource that was requested
- The client receives and process the received data



Copyright © 2006 illustro Systems International, LLC

WAVV2006-11

The request/response format

- The format of the request and of the response are very similar
- Both messages are organized as
 - First line
 - Headers (optional)
 - Empty line
 - Body (optional)



Copyright © 2006 illustro Systems International, LLC

WAVV2006-12

The HTTP Protocol

The request/response format

■ More graphically...

<initial line, different for request & response>

Header1: value

Header2: value

Header3: value

<optional message body goes here, like file contents or query data; it can be many lines long, or even binary data>



Copyright © 2006 illustro Systems International, LLC

WAVV2006-13

The request/response format Initial Request Line

■ A request line has three parts, separated by spaces:

- a *method* name
- the local path of the requested resource
- the version of HTTP being used



Copyright © 2006 illustro Systems International, LLC

WAVV2006-14

The request/response format Initial Request Line

■ Example

GET /path/to/file/index.html HTTP/1.1

- **GET** is the most common HTTP method
 - Method names are always uppercase
- The path is part of the URL after the host name
- The HTTP version always takes the form "HTTP/x.x", uppercase



Copyright © 2006 illustro Systems International, LLC

WAVV2006-15

The HTTP Protocol

The request/response format

Initial Response Line

- Initial response line is called the Status line and also has three parts separated by spaces
 - the HTTP version
 - a *response status code* that gives the result of the request
 - an English *reason phrase* describing the status code
 - Typical status lines are:
 - HTTP/1.1 200 OK or
 - HTTP/1.1 404 Not Found



Copyright © 2006 illustro Systems International, LLC

WAVV2006-16

The request/response format

Initial Response Line

- The status code is meant to be computer-readable
 - It is a three-digit integer, and the first digit identifies the general category of response:
 - **1xx** indicates an informational message only
 - **2xx** indicates success of some kind
 - **3xx** redirects the client to another URL
 - **4xx** indicates an error on the client's part
 - **5xx** indicates an error on the server's part
 - The most common status codes are:
200 OK
404 Not Found
- The reason phrase is meant to be human-readable, and may vary by server



Copyright © 2006 illustro Systems International, LLC

WAVV2006-17

The request/response format

Header Lines

- Header lines provide information about the request or response, or about the object sent in the message body
- Header lines are in the text header format
 - one line per header
 - of the form "**Header-Name: value**"
 - ending with CRLF



Copyright © 2006 illustro Systems International, LLC

WAVV2006-18

The HTTP Protocol

The request/response format

The Message Body

- An HTTP message may have a body of data sent after the header lines
- In a request, this is where user-entered data or uploaded files are sent to the server
- In a response, this is where the requested resource is returned to the client



Copyright © 2006 illustro Systems International, LLC

WAVV2006-19

The request/response format

The Message Body

- If an HTTP message includes a body, there are usually header lines in the message that describe the body
 - **Content-Type:** header gives MIME-type of the data in the body, such as **text/html** or **image/gif**
 - **Content-Length:** header gives the number of bytes in the body



Copyright © 2006 illustro Systems International, LLC

WAVV2006-20

The request/response format

Sample HTTP Exchange

- To retrieve the file at the URL
<http://www.somehost.com/path/file.html>
- Send a request to www.somehost.com like:
GET /path/file.html HTTP/1.1
User-Agent: HTTPTool/1.1
[empty line here]



Copyright © 2006 illustro Systems International, LLC

WAVV2006-21

The HTTP Protocol

The request/response format

Sample HTTP Exchange

- The server should respond on the same connection with something like

```
HTTP/1.1 200 OK
Date: Sat, 08 Apr 2006 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354

<html> <body> <h1>Web Page</h1>
(more file contents) . . .
</body> </html>
```
- After sending the response, the server may close the connection



Copyright © 2006 illustro Systems International, LLC

WAVV2006-22

HTTP Methods

- The methods that can be used on a request are
 - GET, request a document be returned
 - POST, sends data to server and receives result
 - HEAD, requests a document headers only
 - PUT, upload new document
 - DELETE, delete an object
 - Plus a few others



Copyright © 2006 illustro Systems International, LLC

WAVV2006-23

HTTP Methods

POST Method

- A POST request is used to send data to the server to be processed in some way, like by a CGI script
- Different from a GET request in that
 - a block of data sent with the request, in the message body
 - Extra headers to describe message body
 - **Content-Type:**
 - **Content-Length:**



Copyright © 2006 illustro Systems International, LLC

WAVV2006-24

The HTTP Protocol

HTTP Methods

POST Method

- Different from a GET request in that...
 - *request URI* is not a resource to retrieve; it's usually a program to handle the data you're sending
 - HTTP response is normally program output, not a static file
- The most common use of POST is to submit HTML form data to CGI scripts



Copyright © 2006 illustro Systems International, LLC

WAV/2006-25

Persistent Connections

- HTTP 1.0 used a connection for only one request/response
- HTTP 1.1 uses persistent connections by default
 - **Connection:** header can be **Keep-Alive** or **Close**
 - Multiple requests/responses can be handled over one connection
 - Requests can be *pipelined*



Copyright © 2006 illustro Systems International, LLC

WAV/2006-26

Content Negotiation

- Used to select "Best" response for a request
 - Server-driven negotiation
 - Agent-driven negotiation
 - Transparent negotiation
- Server-driven negotiation normally used



Copyright © 2006 illustro Systems International, LLC

WAV/2006-27

The HTTP Protocol

Content Negotiation Server-Driven

- User agent provides a list of preferences
 - Accept Media types
 - Accept-Language Text languages
 - Accept-Encoding Content encoding
 - Accept-Charset Character sets
- Server decides "Best" resource from those available
- Vary header may be returned to tell caches what request headers used to make decision



Copyright © 2006 illustro Systems International, LLC

WAVV2006-28

Content Negotiation Agent-Driven

- Server returns status code **300**
Multiple Choices and list of available resources
- User-agent decides which one is "Best" and issues new request for it
- Requires multiple trips to the server



Copyright © 2006 illustro Systems International, LLC

WAVV2006-29

Content Negotiation Transparent

- Combination of server-driven and agent-driven when the resource is cached
- When a cache contains the possible resources and is aware of the variances, it can provide the server-driven selection
- Offloads work from the server



Copyright © 2006 illustro Systems International, LLC

WAVV2006-30

The HTTP Protocol

Conditional Requests

- Optional request headers can be included to make a request conditional
 - If-Modified-Since/If-Unmodified-Since
 - Compares date and time (Date: header)
 - If-Match/If-None-Match
 - Compares entity tags (ETAG: header)



Copyright © 2006 illustro Systems International, LLC

WAVV2006-31

Conditional Requests

- When specified condition is true, server returns requested resource
- When condition is not true, a status code is returned with no message-body
 - 304 Not Modified
 - 412 Precondition Failed



Copyright © 2006 illustro Systems International, LLC

WAVV2006-32

Proxy Servers

- The HTTP protocol specification provides specific support for proxy servers
- Proxy servers are both a client and a server
 - Accept requests from other clients
 - Forward the request to a server (or another proxy)
 - Cache the response
 - Respond to the original client



Copyright © 2006 illustro Systems International, LLC

WAVV2006-33

The HTTP Protocol

Proxy Servers

- Or, they respond immediately with a cached page instead of sending the request over the Internet
- Clients must be configured to use a proxy
 - The client connects to the proxy instead of the host specified in the URL
 - The request must specify the Absolute-URI instead of the Request-URI



Copyright © 2006 illustro Systems International, LLC

WAVV2006-34

Proxy Servers

- Headers specifically for proxies
 - Proxy-Authorization:
 - Proxy-Authenticate:
 - Max-Forwards:
 - Via:



Copyright © 2006 illustro Systems International, LLC

WAVV2006-35

Caching

- Caching of resources can be done by a cache server (proxy), the HTTP client, or both
- The goal of caching in HTTP is to:
 - Eliminate the need to send requests
 - Reduce the number of trips to the server
 - Uses an expiration mechanism
 - Eliminate the need to send full responses
 - Reduce network bandwidth requirements
 - Uses a validation mechanism



Copyright © 2006 illustro Systems International, LLC

WAVV2006-36

The HTTP Protocol

Caching

- Cached resources are either "Fresh" or "Stale"
 - Based on Age and freshness lifetime
- The HTTP specification defines formulas to calculate age and freshness
- Stale documents must be "validated" with the server using:
 - Last-modified date (If-Modified-Since)
 - ETag (If-None-Match)



Copyright © 2006 illustro Systems International, LLC

WAVV2006-37

Caching

- Stale resources returned to a client must have the Warning: header added
- Caching is controlled by
 - Cache-Control: header
 - Expires: header
 - Date: header
 - ETag: header



Copyright © 2006 illustro Systems International, LLC

WAVV2006-38

Caching Cache-Control: Header

- For requests
 - no-cache
 - no-store
 - max-age = *seconds*
 - max-stale = *seconds*
 - min-fresh = *seconds*
 - only-if-cached



Copyright © 2006 illustro Systems International, LLC

WAVV2006-39

The HTTP Protocol

Caching

Cache-Control: Header

- For responses
 - public
 - private
 - no-cache
 - no-store
 - no-transform
 - must-revalidate
 - proxy-revalidate
 - max-age = seconds



Copyright © 2006 illustro Systems International, LLC

WAVV2006-40

Resources

- HTTP/1.1 RFC 2616, June 1999
 - Replaces RFC 2068, January 1997
- HTTP/1.0 RFC 1945, May 1996
- MIME RFC 2045, November 1996
- World Wide Web Consortium
 - www.w3c.org
 - News, updates, drafts, reports
- HTTP Made Really Easy, tutorial
 - www.jmarshall.com/easy/http/



Copyright © 2006 illustro Systems International, LLC

WAVV2006-41

Web 2.0

- Future of the World Wide Web
 - Assumes everything up to now was 1.0
- It's all about performance and the user experience
 - 1.0
 - For every server interaction we wait for an entirely new web page to load
 - Nothing can be done with the current page while we wait for the new page



Copyright © 2006 illustro Systems International, LLC

WAVV2006-42

The HTTP Protocol

Web 2.0

- It's all about performance and the user experience...
 - 2.0
 - Asynchronous server requests made in background
 - User interface on current page continues to function
- Much more of the application code resides in the web page in the form of scripting instead of being on the server



Copyright © 2006 illustro Systems International, LLC

WAVV2006-43

Web 2.0

- Uses Asynchronous JavaScript and XML (Ajax)
 - Phrase coined by Jesse James Garrett of Adaptive Path
- JavaScript XMLHttpRequest method used to make asynchronous requests to same server from which web page was retrieved
- Call back JavaScript function handles result data when it arrives



Copyright © 2006 illustro Systems International, LLC

WAVV2006-44

Web 2.0

- Dynamic HTML used to update existing web page instead of entirely replacing it with a new page
- User interface in existing page continues to be active while background requests are made
- User experience is more like a desktop application than a web application



Copyright © 2006 illustro Systems International, LLC

WAVV2006-45

The HTTP Protocol

Web 2.0

- In spite of the method name, the received data can be any format the JavaScript is prepared to handle
 - HTML
 - Plain text with delimiters
 - XML documents processed with DOM



Copyright © 2006 illustro Systems International, LLC

WAVV2006-46

Web 2.0 Resources

- IBM Developerworks web site, multi-part Ajax article
 - <http://www-128.ibm.com/developerworks/web/library/wa-ajaxintro1.html?ca=dgr-Inxw01MasterAJAX>
- AJAX World Magazine
 - <http://ajax.sys-con.com>
- Google for AJAX or WEB 2.0 for additional information



Copyright © 2006 illustro Systems International, LLC

WAVV2006-47
