# Networking Basics Behind the World Wide Web

**Jeff Offutt**
**http://www.cs.gmu.edu/~offutt/**

**SWE 642**
**Software Engineering for the World Wide Web**

Adapted from chapter 1 slides for :
Web Technologies : A Computer Science Perspective
by
JEFFREY C. JACKSON



## Topics

1. The Internet
2. Internet Protocols
3. World Wide Protocols
4. WWW Requests and Responses
5. Client Caching
6. Character Sets
7. Web Clients and Browsers
8. Web Servers
9. Security on the Web

# The Internet

- Technical origin: ARPANET (late 1960's)
  - One of earliest attempts to network heterogeneous, geographically dispersed computers
  - Email first available on ARPANET in 1972 (and quickly very popular!)
- ARPANET access was limited to select DoD-funded organizations
  - Mostly universities and military bases

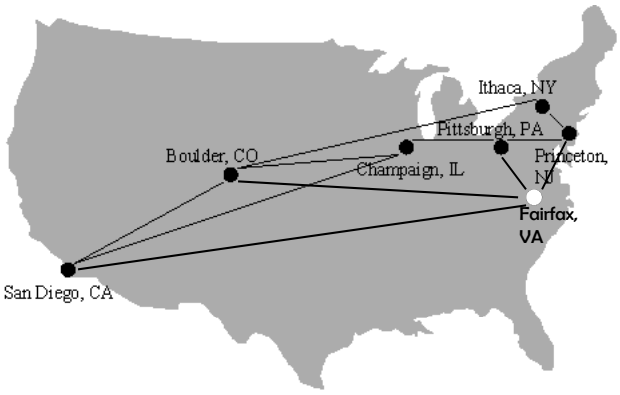




# The Internet

- Open-access networks
  - Regional university networks (e.g., SURAnet)
  - CSNET for CS departments not on ARPANET
- NSFNET (1985-1995)
  - Primary purpose: connect supercomputer centers
  - Secondary purpose: provide backbone to connect regional networks

# The Internet








# The Internet

- Internet : the network of networks connected via the public backbone and communicating using TCP/IP communication protocol
  - Backbone initially supplied by NSFNET, privately funded (ISP fees) beginning in 1995

# Topics

1. The Internet
2. Internet Protocols
3. World Wide Protocols
4. WWW Requests and Responses
5. Client Caching
6. Character Sets
7. Web Clients and Browsers
8. Web Servers
9. Security on the Web





# Internet Protocols

- Communication protocol : how computers talk
  - For example : Telephone “protocol”: how you answer and end call, what language you speak, etc.
- Internet protocols developed as part of ARPANET research
  - ARPANET began using TCP/IP in 1982
- Designed for use both within local area networks (LANs) and between networks

# Internet Protocol (IP)

- IP is the fundamental protocol defining the Internet
- Original IP definition from 1981 :
  - ftp://ftp.rfc-editor.org/in-notes/std/std5.txt
- IP address :
  - 32-bit number (IPv4) or 128-bit (IPv6)
  - Associated with at most one device at a time on the same network
    - although a device may have more than one
  - Written as four dot-separated bytes, e.g. 192.0.34.166

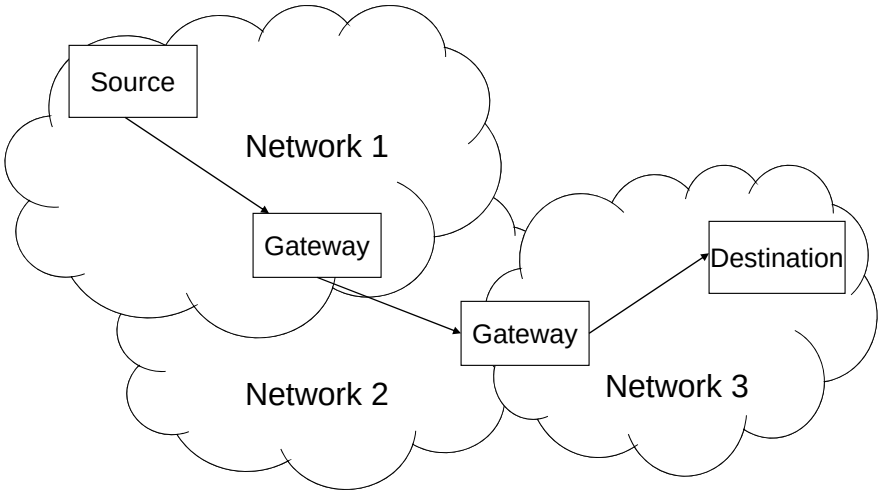




# IP

- IP function : Transfer data from source device to destination device
- IP source software creates a packet representing the data
  - Header : source and destination IP addresses, length of data, etc.
  - Data itself
- If destination is on another LAN, packet is sent to a gateway that connects to more than one network

# IP



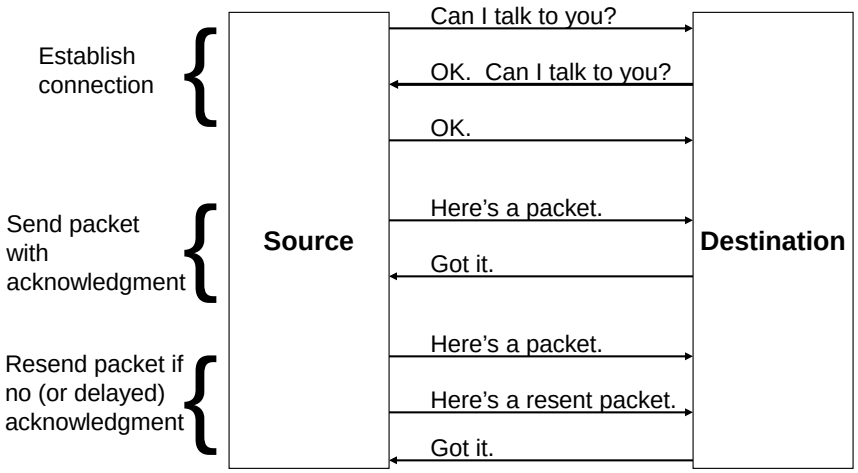





# Transmission Control Protocol (TCP)

- Limitations of IP :
  - No guarantee of packet delivery (packets can be dropped)
  - Communication is one-way (source to destination)
- TCP adds concept of a connection on top of IP
  - Provides guarantee that packets delivered
  - Provide two-way (full duplex) communication
- The original TCP definition :
  - ftp://ftp.rfc-editor.org/in-notes/std/std7.txt

# TCP



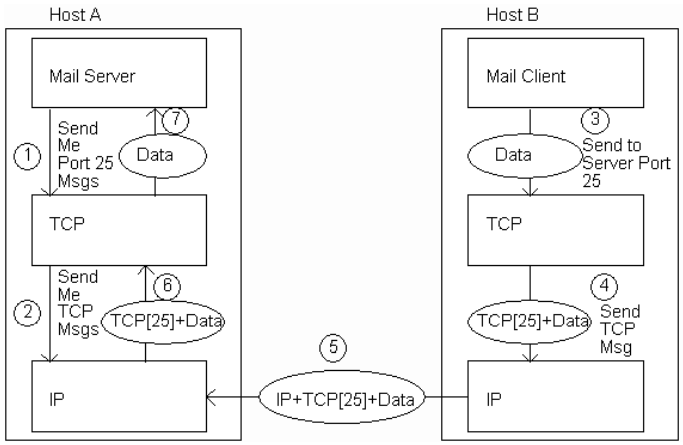





# TCP

- TCP adds concept of a port
  - TCP header contains port number representing an application program on the destination computer
  - Some port numbers have standard uses
    - Example: port 25 is normally used for email transmitted using the Simple Mail Transfer Protocol (SMTP)
    - Definitions : http://www.iana.org/assignments/port-numbers
  - Other port numbers are available first-come-first-served to any application

# TCP








# Domain Name Service (DNS)

- DNS is the “phone book” for the Internet
  - Map between host names and IP addresses
- Host names
  - Labels separated by dots, e.g., *www.example.org*
  - Final label is *top-level domain*
    - Generic: .com, .org, .edu, …
    - Country-code: .us, .il, .cn, …
    - http://www.icann.org/en/tlds/
- Original definition of DNS :
  - ftp://ftp.rfc-editor.org/in-notes/std/std13.txt

# DNS

- Domains are divided into second-level domains, which can be further divided into subdomains, etc.
  - E.g., in *www.example.com*, "example" is a second-level domain
- A host name plus domain name information is called the fully qualified domain name (FQDN) of the computer
  - Above, *www* is the host name, *www.example.com* is the FQDN





# DNS

- *nslookup* is a program that provides command-line access to DNS (on most systems)
- Looking up a host name given an IP address is known as a reverse lookup
  - Recall that a single host may have multiple IP addresses
  - Address returned is the canonical IP address specified in the DNS system

## Analogy to Telephone Network

- IP ~ the telephone network
- TCP ~ calling someone who answers, having a conversation, and hanging up
- UDP ~ calling someone and leaving a message
- DNS ~ directory assistance





## Higher-level Protocols

- Many protocols build on TCP
  - Telephone analogy : TCP specifies how we initiate and terminate the phone call, but another protocol needs to specify how we carry on the actual conversation
- Examples :
  - SMTP (email)
  - FTP (file transfer)
  - HTTP (transfer of Web documents)

# Topics

1. The Internet
2. Internet Protocols
3. World Wide Protocols
4. WWW Requests and Responses
5. Client Caching
6. Character Sets
7. Web Clients and Browsers
8. Web Servers
9. Security on the Web





# World Wide Web

- Originally, one of several systems for organizing information on the Internet
  - Competitors : WAIS, Gopher, ARCHIE
- Distinctive feature of Web : support for hypertext
  - Text containing links
  - Communication via Hypertext Transport Protocol (HTTP)
  - Document representation using Hypertext Markup Language (HTML)

# World Wide Web

- The Web is the collection of machines (Web servers) on the Internet that provide information, particularly HTML documents, via HTTP
- Machines that access information on the Web are known as Web clients
- A Web browser is software that lets an end user access the Web





# Hypertext Transport Protocol (HTTP)

- HTTP is based on the request-response communication model :
  - Client sends a request
  - Server sends a response
- HTTP is a stateless protocol :
  - The protocol does not require the server to remember anything about the client between requests
- The original standards proposal for HTTP :
  - ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt

# HTTP

- Normally implemented over a TCP connection
  - 80 is standard port number for HTTP
- Typical browser-server interaction:
  - User enters Web address in browser
  - Browser uses DNS to locate IP address
  - Browser opens TCP connection to server
  - Browser sends HTTP request over connection
  - Server sends HTTP response to browser over connection
  - Browser displays body of response in the client area of the browser window





# HTTP

- The information transmitted using HTTP is often entirely text
- Can use the Internet's Telnet protocol to simulate browser request and view server response

# HTTP

Connect

```
$ telnet www.example.org 80
Trying 192.0.34.166...
Connected to www.example.org
(192.0.34.166).
Escape character is '^]'.
```

Send Request

```
GET / HTTP/1.1
Host: www.example.org
```

Receive Response

```
HTTP/1.1 200 OK
Date: Thu, 09 Oct 2003 20:30:49
GMT
...
```





# Topics

1. The Internet
2. Internet Protocols
3. World Wide Protocols
4. WWW Requests and Responses
5. Client Caching
6. Character Sets
7. Web Clients and Browsers
8. Web Servers
9. Security on the Web

# HTTP Request

Clients send requests to servers to ask for a resource (usually a file or to run a program)

Structure of the request :

1. start line
2. header field (s)
3. blank line
4. optional body





1. ***start line***
2. header field (s)
3. blank line
4. optional body

# HTTP Request

- Start line
  - Example: GET / HTTP/1.1
- Three space-separated parts:
  - HTTP request method
  - Request-URI
  - HTTP version
    - We will cover 1.1, in which version part of start line must be exactly as shown

1. ***start line***
   - ***Request-URI***
2. header field (s)
3. blank line
4. optional body

# HTTP Request

- Uniform Resource Identifier (URI)
  - Syntax: *scheme* : *scheme-depend-part*
    - Ex: In *http://www.example.com/* the scheme is *http*
    - *URI specification:*
    - *ftp://ftp.rfc-editor.org/in-notes/rfc2396.txt*
  - Request-URI is the portion of the requested URI that follows the host name
    - supplied by the required Host header field
    - Ex : "/" is Request-URI portion of http://www.example.com/
    - *"~offutt/" is Request-URI of* http://www.cs.gmu.edu/~offutt/





1. ***start line***
   - ***Request-URI***
2. header field (s)
3. blank line
4. optional body

# URI

URIs come in two types :

- Uniform Resource Name ( URN )
  - Used to identify resources with unique names, such as books (which have unique ISBNs)
  - Scheme is *urn*
  - *URN specification:* ftp://ftp.rfc-editor.org/in-notes/rfc2141.txt
- Uniform Resource Locator ( URL )
  - Specifies location where a resource can be found
  - In addition to *http*, some other URL schemes are *https*, *ftp*, *mailto*, and *file*
  - *URL specification:* ftp://ftp.rfc-editor.org/in-notes/rfc2396.txt

# HTTP Request

1. ***start line***
   - ***http request method***
2. header field (s)
3. blank line
4. optional body

Common request methods :

- GET
  - Used if link is clicked or address typed in browser
  - No body in request with GET method
- POST
  - Used when submit button is clicked on a form
  - Form information contained in body of request
- HEAD
  - Requests that only header fields (no body) be returned in the response
- We will use GET and POST in this class





# HTTP Request

1. start line
2. ***header field (s)***
3. blank line
4. optional body

- Header field structure:
  - *field name* : *field value*
- Syntax
  - Field name is not case sensitive
  - Field value may continue on multiple lines by starting continuation lines with white space
  - Field values may contain MIME types, quality values, and wildcard characters (*)

# Multipurpose Internet Mail Extensions (MIME)

1. start line
2. ***header field (s)***
3. blank line
4. optional body

- Convention for specifying content type of a message
  - In HTTP, typically used to specify content type of the body of the response
- MIME content type syntax:
  - *top-level type / subtype*
- Examples: text / html, image / jpeg
- MIME Standard :

  ftp://ftp.rfc-editor.org/in-notes/rfc2046.txt





# HTTP Quality Values and Wildcards

1. start line
2. ***header field (s)***
3. blank line
4. optional body

- Example header field with quality values :

```
accept:
    text/xml,text/html;q=0.9,
  text/plain;q=0.8, image/jpeg,
  image/gif;q=0.2,*/*;q=0.1
```

- Quality value applies to all preceding items
- Higher values mean higher preferences
- Wildcards specify quality 0.1 for any MIME type not specified earlier

# HTTP Request

1. start line
2. ***header field (s)***
3. blank line
4. optional body

Common header fields :

- Host : host name from URL (required)
- User-Agent : type of browser sending request
- Accept : MIME types of acceptable documents
- Connection : value `close` tells server to close connection after single request/response
- Content-Type : MIME type of (POST) body, normally application/x-www-form-urlencoded
- Content-Length : bytes in body
- Referer : URL of document containing link that supplied URI for this HTTP request





# HTTP Response

Servers send responses to clients with result of request (error code, a file output of a program)

Structure of the response :

1. status line
2. header field (s)
3. blank line
4. optional body

1. ***status line***
2. header field (s)
3. blank line
4. optional body

# HTTP Response

- Status line
  - Example: HTTP/1.1 200 OK
- Three space-separated parts :
  - HTTP version
  - status code
  - reason phrase (intended for human use)





1. ***status line***
2. header field (s)
3. blank line
4. optional body

# HTTP Response

Status code :

- Three-digit number
- First digit is class of the status code :
  - 1 = Informational
  - 2 = Success
  - 3 = Redirection (alternate URL is supplied)
  - 4 = Client Error
  - 5 = Server Error
- Other two digits provide additional information

1. status line
2. ***header field (s)***
3. blank line
4. optional body

# HTTP Response

Common header fields :

- Connection, Content-Type, Content-Length
- Date : date and time at which response was generated (required)
- Location : alternate URI if status is redirection
- Last-Modified : date and time the requested resource was last modified on the server
- Expires : date and time after which the client's copy of the resource will be out-of-date
- ETag : a unique identifier for this version of the requested resource





# Topics

1. The Internet
2. Internet Protocols
3. World Wide Protocols
4. WWW Requests and Responses
5. Client Caching
6. Character Sets
7. Web Clients and Browsers
8. Web Servers
9. Security on the Web

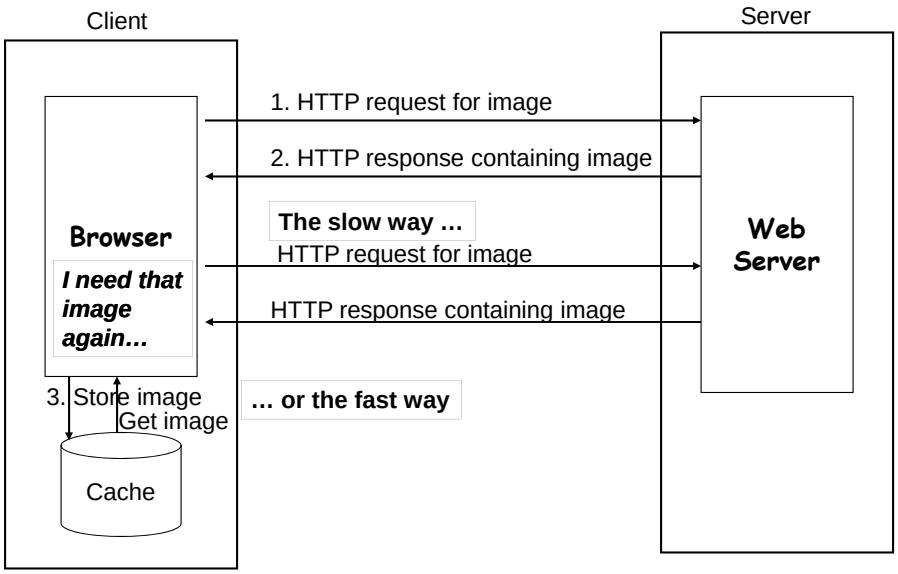
# Client Caching

- A *cache* is a local copy of information obtained from some other source
- Most web browsers store requested resources so subsequent requests to the same resource will not require an HTTP request / response
  - Ex: icon appearing multiple times in a Web page
- This works fine for static files (icons, HTML), but not so well for outputs from programs (web applications)
  - Most browsers have a way for users to override cache (*shift-reload in Firefox*)





# Client Caching



# Client Caching

- Cache advantages
  - Faster than HTTP request/response
  - Less network traffic
  - Less load on server
- Cache disadvantage
  - Cached copy of resource may be invalid (inconsistent with remote version)





# Client Caching

- Validating cached resource :
  - Send HTTP HEAD request and check Last-Modified or ETag header in response
  - Compare current date / time with Expires header sent in response containing resource
  - If no Expires header was sent, use heuristic algorithm to estimate value for Expires
    - Ex: Expires = 0.01 * (Date – Last-Modified) + Date
- Sometimes routers have a separate cache
  - Ex: A router that serves all clients in the same company

# Topics

1. The Internet
2. Internet Protocols
3. World Wide Protocols
4. WWW Requests and Responses
5. Client Caching
6. Character Sets
7. Web Clients and Browsers
8. Web Servers
9. Security on the Web





# Character Sets

- Every document is represented by a string of integer values (code points)
- The mapping from code points to characters is defined by a character set
- Some header fields have character set values :
  - Accept-Charset : request header listing character sets that the client can recognize
    - Ex: `accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7`
  - Content-Type : can include character set used to represent the body of the HTTP message
    - Ex: `Content-Type: text/html; charset=UTF-8`

# Character Sets

- Technically, many “character sets” are actually character encodings
  - An encoding represents code points using variable-length byte strings
  - Most common examples are Unicode-based encodings UTF-8 and UTF-16
- IANA maintains complete list of Internet-recognized character sets / encodings
  - http://www.iana.org/assignments/character-sets





# Character Sets

- Typical US PCs produces ASCII documents
- US-ASCII character set can be used for such documents, but is not recommended
  - English has one of the smallest character sets of any human language
- UTF-8 and ISO-8859-1 are supersets of US-ASCII and provide international compatibility
  - Unicode has a code for all characters in all human languages
  - UTF-8 represents all ASCII characters with a single byte each and arbitrary Unicode characters with up to 4 bytes
  - ISO-8859-1 is 1-byte code that includes many characters common in Western European languages, such as é

# Topics

1. The Internet
2. Internet Protocols
3. World Wide Protocols
4. WWW Requests and Responses
5. Client Caching
6. Character Sets
7. Web Clients and Browsers
8. Web Servers
9. Security on the Web





# Web Clients

- Many possible web clients :
  - Text-only “browser” (lynx)
  - Mobile phones
  - Robots (software-only clients, e.g., search engine “crawlers”)
  - etc.
- We will focus on traditional web browsers
  - Firefox, Explorer, Opera, Chrome, …

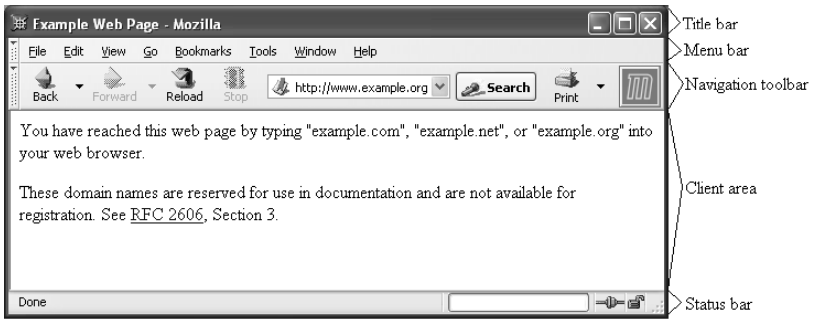
# Web Browsers

First graphical browser running on general-purpose platforms : Mosaic (1993)






# Web Browsers



# Web Browsers

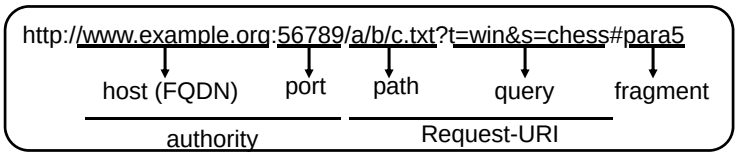
Primary tasks :

- Convert web addresses (URLs) to HTTP requests
- Communicate with web servers via HTTP
- Render (appropriately display) documents returned by a server





# HTTP URLs




- Browser uses authority to connect via TCP
- Request-URI included in start line (/ used for path if none supplied)
- Fragment identifier not sent to server (used to scroll browser client area)

# Web Browsers

Standard features

- Save web page to disk
- Find string in page
- Fill forms automatically (passwords, CC numbers, …)
- Set preferences (language, character set, cache and HTTP parameters)
- Modify display style (e.g., increase font sizes)
- Display raw HTML and HTTP header info (e.g., Last-Modified)
- Choose browser themes (skins)
- View history of web addresses visited
- Bookmark favorite pages for easy return





# Web Browsers

Additional functionality :

- Execution of scripts (drop-down menus)
- Event handling (mouse clicks)
- GUI for controls (buttons)
- Secure communication with servers
- Display of non-HTML documents (PDF) via plug-ins

# Topics

1. The Internet
2. Internet Protocols
3. World Wide Protocols
4. WWW Requests and Responses
5. Client Caching
6. Character Sets
7. Web Clients and Browsers
8. Web Servers
9. Security on the Web





# Web Servers

Basic functionality :

- Receive HTTP request via TCP
- Map Host header to specific virtual host
  - one of possibly many host names sharing an IP address
- Map Request-URI to specific resource associated with the virtual host
  - File : Return file in HTTP response
  - Program : Run program and return output in HTTP response
- Map type of resource to appropriate MIME type and use to set Content-Type header in HTTP response
- Log information about the request and response

# Web Servers

- httpd : UIUC, primary Web server c. 1995
- Apache : “A patchy” version of httpd, now the most popular server (esp. on Linux platforms)
- IIS : Microsoft Internet Information Server
- Tomcat :
  - Java-based
  - Provides container (Catalina) for running Java servlets (HTML-generating programs) as back-end to Apache or IIS
  - Can run stand-alone using Coyote HTTP front-end
  - http://tomcat.apache.org/





# Web Servers

- Some Coyote communication parameters :
  - Allowed / blocked IP addresses
  - Max. simultaneous active TCP connections
  - Max. queued TCP connection requests
  - “Keep-alive” time for inactive TCP connections
- Modify parameters to tune server performance

# Web Servers

Some Catalina container parameters :

- Virtual host names and associated ports
- Logging preferences
- Mapping from Request-URI's to server resources
- Password protection of resources
- Use of server-side caching

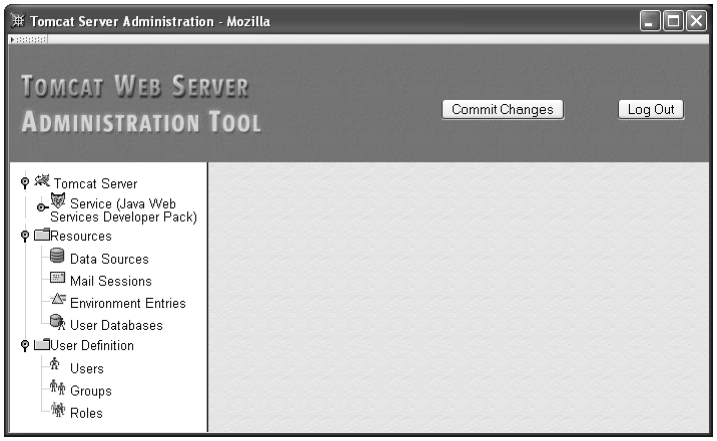




# Tomcat Web Server

- HTML-based server administration
- Browse to
  http://localhost:8080
  and click on Server Administration link
  - localhost is a special host name that means "this machine"

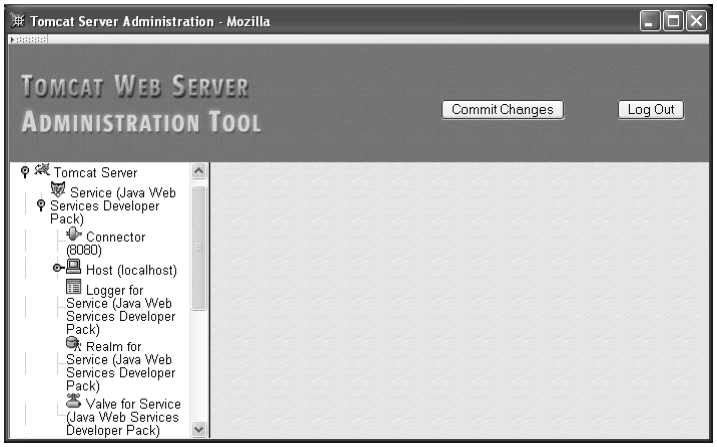
# Tomcat Web Server



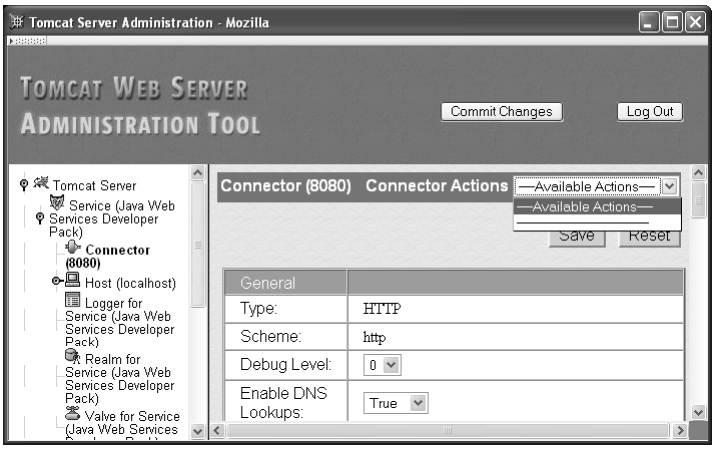





# Tomcat Web Server



# Tomcat Web Server








# Tomcat Web Server

Some connector fields :

- Port Number : port “owned” by this connector
- Max Threads : max connections processed simultaneously
- Connection Timeout : keep-alive time

# Tomcat Web Server

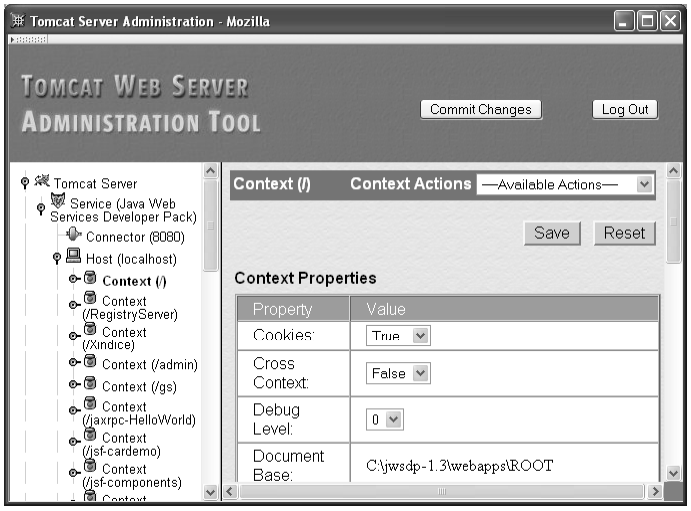





# Tomcat Web Server

- Each host is a virtual host (can have multiple per Connector)
- Some fields :
  - Host: `localhost` or a fully qualified domain name
  - Application Base : directory containing resources associated with this Host
    - May be path relative to JWSDP installation directory

# Tomcat Web Server








# Tomcat Web Server

- Context provides mapping from Request-URI path to a web application
- Document Base field is directory (possibly relative to Application Base) that contains resources for this web application
- For this example, browsing to
  http://localhost:8080/
  returns resource from
  c:\jwsdp-1.3\webapps\ROOT
  - Returns index.html (standard welcome file)

## Tomcat Web Server

- Access log records HTTP requests
- Parameters set using AccessLogValve
- Default location: logs/access_log.* under JWSDP installation directory
- Example “common” log format entry (one line):

```
www.example.org - admin
[20/Jul/2005:08:03:22 -0500]
"GET /admin/frameset.jsp HTTP/1.1"
200 920
```





## Tomcat Web Server

Other logs provided by default in JWSDP :

- Message log messages sent to log service by web applications or Tomcat itself
  - logs/jwsdp_log.* : default message log
  - logs/localhost_admin_log.* : message log for web apps within /admin context
- System.out and System.err output (exception traces often found here) :
  - logs/launcher.server.log

# Tomcat Web Server

Access control :

- Password protection (e.g., admin pages)
  - Users and roles defined in `conf/tomcat-users.xml`
- Deny access to machines
  - Useful to denying access to certain users by denying access from their machines
  - List of denied machines maintained in `RemoteHostValve` (deny by host name) or `RemoteAddressValve` (deny by IP address)





# Topics

1. The Internet
2. Internet Protocols
3. World Wide Protocols
4. WWW Requests and Responses
5. Client Caching
6. Character Sets
7. Web Clients and Browsers
8. Web Servers
9. Security on the Web

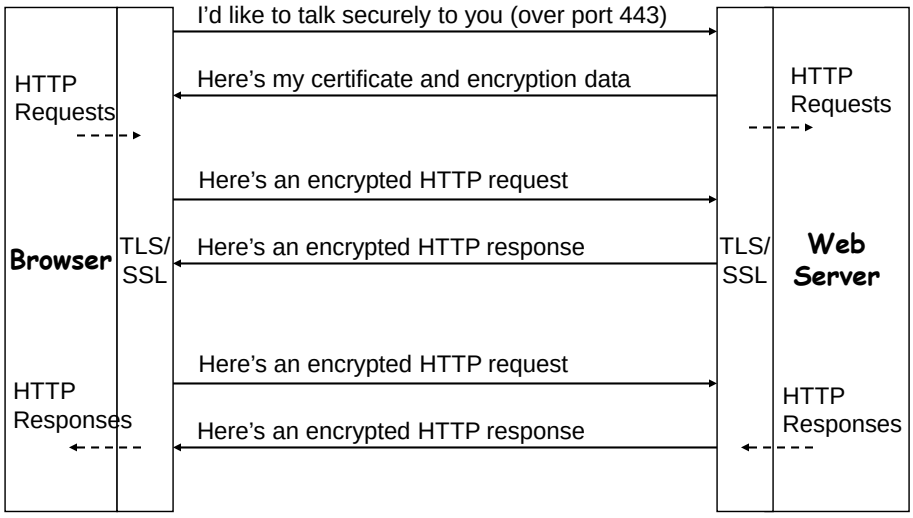
# Secure Servers

- Since HTTP messages typically travel over a public network, private information (such as credit card numbers) should be encrypted to prevent eavesdropping
- https URL scheme tells browser to use encryption
- Common encryption standards:
  - Secure Socket Layer ( SSL )
  - Transport Layer Security ( TLS )
  - TLS specification :
    ftp://ftp.rfc-editor.org/in-notes/rfc2246.txt

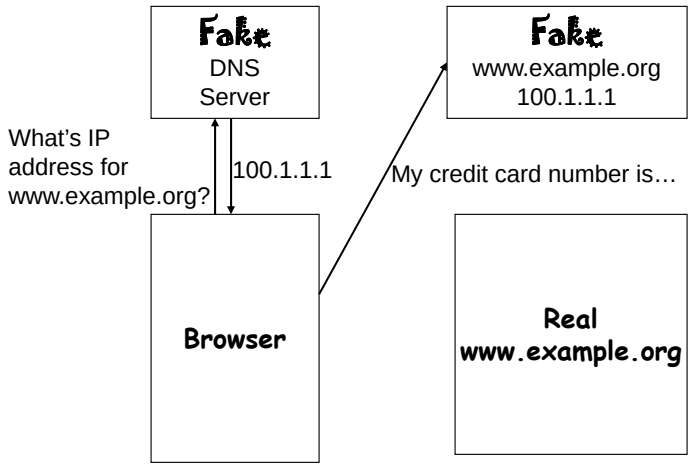




# Secure Servers



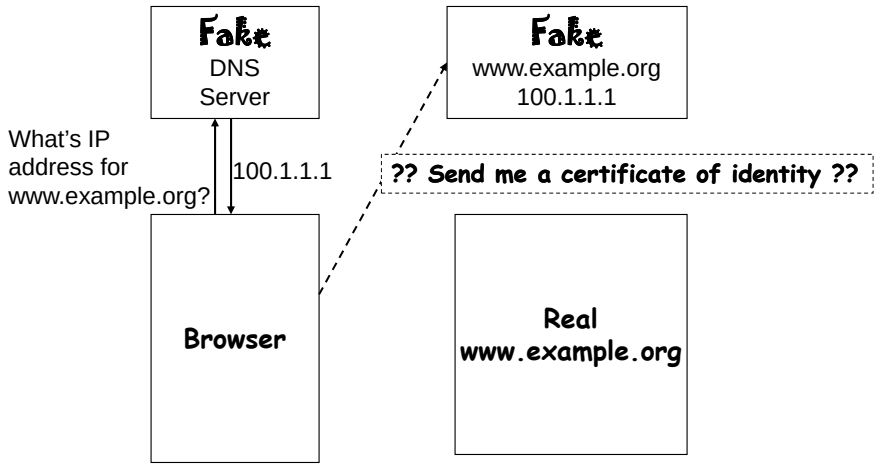
# Secure Servers
# Man-in-the-Middle Attack








# Secure Servers
# Preventing Man-in-the-Middle



# WWW Networking Summary

- The goal of the WWW networking design is to make using the WWW easy
  - Robust, reliable, efficient, simple
- Programmers must have a general idea how the network works

  … but …
- A major goal of the development technologies is to simplify development of distributed, heterogeneous, multi-layer software
  - CGI / Perl, PHP, J2EE, ASP / .NET, …
- J2EE insulates us from most networking details