

# CSCE 515: Computer Network Programming ----- Web & HTTP

Wenyuan Xu

Department of Computer Science and  
Engineering  
University of South Carolina

CSCE515 – Computer Network Programming

## WWW History

- What is WWW?
  - an architecture framework for accessing linked document spread over millions of machines
- 1989-1990 – Tim Berners-Lee invents the World Wide Web at CERN
  - CERN: European center for nuclear research.
  - Means for distributing high-energy physics data
  - Means for transferring text and graphics simultaneously
- Client/Server data transfer protocol
  - Communication via application level protocol
  - System ran on top of standard networking infrastructure
- Established a common language for sharing information on computers
  - Text mark up language
    - Simple and easy to use
    - Requires a client application to render text/graphics

CSCE515 – Computer Network Programming

## WWW History contd.

- 1994 – Mark Andreessen invents MOSAIC at National Center for Super Computing Applications (NCSA)
  - First graphical browser
  - Internet's first "killer app"
  - Freely distributed
  - Became Netscape Inc.
- 1995 (approx.) – Web traffic becomes dominant
  - Exponential growth
  - E-commerce
  - Web infrastructure companies
  - World Wide Web Consortium

CSCE515 – Computer Network Programming

## How the web works?

- User input:
  - URL
  - Hypertext link/ Hyperlink
- Web browser
  - Gets the *IP address* of the server (via DNS)
  - Makes a *TCP connection* to port 80 on the server
  - Sends an *HTTP request* to the web server
  - Receives the required files from the web server
  - *Releases* the TCP connection.
  - Renders the page onto the screen as specified by its HTML or other web languages

CSCE515 – Computer Network Programming

## WWW Components

- Structural Components
  - Clients/browsers – to dominant implementations
  - Servers – run on sophisticated hardware
  - Caches – many interesting implementations
  - Internet – the global infrastructure which facilitates data transfer
- Semantic Components
  - Hyper Text Transfer Protocol (**HTTP**)
  - Hyper Text Markup Language (**HTML**)
    - eXtensible Markup Language (XML)
  - Uniform Resource Identifiers (**URIs**)

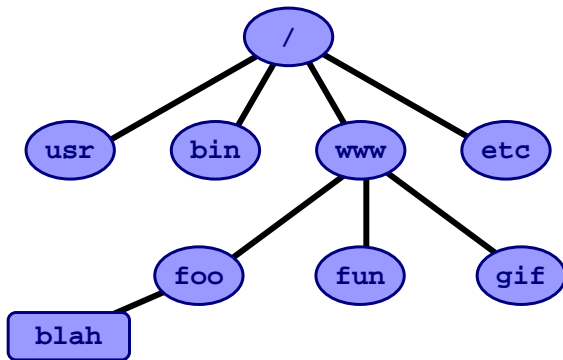
CSCE515 – Computer Network Programming

## URI: Uniform Resource Identifiers

- URIs defined in RFC 2396.
- provide a simple and extensible means for identifying a resource
- Absolute URI: `scheme://hostname[:port]/path`
  - `http://www.cse.sc.edu:80/foo/blah`
  - `ftp://ftp.is.co.za/rfc/rfc1808.txt`
  - `mailto:mduerst@ifi.unizh.ch`
- Relative URI: `/path`  
**No server mentioned** → `/foo/blah`

CSCE515 – Computer Network Programming

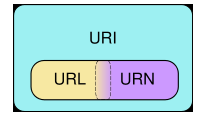
/foo/blah



CSCE515 – Computer Network Programming

## URL vs. URI?

- Most popular form of a URI is the Uniform Resource Locator (URL)
- What is the difference between URL and URI?
- URI = URL+URN
- URN: Uniform Resource Name
  - `urn:isbn:0-395-36341-1`



CSCE515 – Computer Network Programming

## HTTP Hypertext Transfer Protocol

Refs:

RFC 1945 (HTTP 1.0)

RFC 2616 (HTTP 1.1)

## HTTP Basic

- HTTP is the protocol that supports communication between web browsers and web servers.
- A “Web Server” is a HTTP server
- Most clients/servers today speak version 1.1, but 1.0 is also in use.

CSCE515 – Computer Network Programming

## From the RFC

- “HTTP is an **application**-level protocol with the **lightness** and **speed** necessary for distributed, hypermedia information systems.”
- Transport Independence
  - The RFC states that the HTTP protocol generally takes place over a TCP connection, but the protocol itself is **not dependent** on a specific transport layer.

CSCE515 – Computer Network Programming

## Request - Response

- HTTP has a simple structure:
  - client sends a request
  - server returns a reply.
- HTTP can support multiple request-reply exchanges over a single TCP connection.

CSCE515 – Computer Network Programming

## Well Known Address

- The “well known” TCP port for HTTP servers is port 80.
- Other ports can be used as well...

CSCE515 – Computer Network Programming

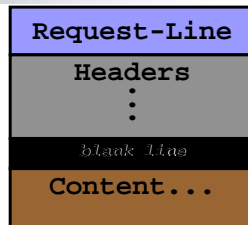
## HTTP Versions

- The original version now goes by the name “HTTP Version 0.9”
  - HTTP 0.9 was used for many years.
- Starting with HTTP 1.0 the version number is part of every request.
  - tells the server what version the client can talk (what options are supported, etc).

CSCE515 – Computer Network Programming

## HTTP 1.0+ Request

- Lines of text (ASCII).
- Lines end with CRLF “\r\n”
- First line is called “Request-Line”

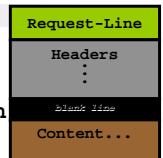


CSCE515 – Computer Network Programming

## Request Line

*Method URI HTTP-Version\r\n*

- The request line contains 3 *tokens* (words).
- space characters “ ” separate the tokens.
- Newline (\n) seems to work by itself (but the protocol requires CRLF)
- Typical HTTP request:  
`GET /index.html HTTP/1.0`



CSCE515 – Computer Network Programming

## Request Method

- The Request Method can be:

GET            HEAD            PUT  
POST          DELETE          TRACE  
OPTIONS

*future expansion is supported*

CSCE515 – Computer Network Programming

## Methods

- GET: retrieve information identified by the URI.
- HEAD: retrieve meta-information about the URI.
- PUT: Store information in location named by URI.
- POST: send information to a URI and retrieve result.
- DELETE: remove *entity* identified by URI.

CSCE515 – Computer Network Programming

## More Methods

- TRACE: used to trace HTTP forwarding through proxies, tunnels, etc.
- OPTIONS: used to determine the capabilities of the server, or characteristics of a named resource.

CSCE515 – Computer Network Programming

## Common Usage

- GET, HEAD and POST are supported everywhere.
- HTTP 1.1 servers often support PUT, DELETE, OPTIONS & TRACE.

CSCE515 – Computer Network Programming

## HTTP Version Number

“HTTP/1.0” or “HTTP/1.1”

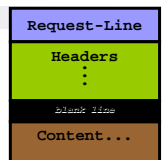
HTTP 0.9 did not include a version number in a request line.

If a server gets a request line with no HTTP version number, it assumes 0.9

CSCE515 – Computer Network Programming

## The Header Lines

- After the *Request-Line* come a number (possibly zero) of HTTP *header lines*.



- Each header line contains an attribute name followed by a ":" followed by a space and the attribute value.
  - The Name and Value are just text.
  - Host: **www.sc.edu**
- Request Headers provide information to the server about the client
  - what kind of client
  - what kind of content will be accepted
  - who is making the request
- There can be 0 headers (HTTP 1.0)
- HTTP 1.1 requires a **Host:** header

CSCE515 – Computer Network Programming

## Example HTTP Headers

**Accept:** text/html

**Host:** www.sc.edu

**From:** neytmann@cybersurg.com

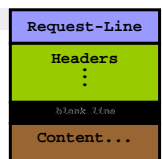
**User-Agent:** Mozilla/4.0

**Referer:** http://foo.com/blah

CSCE515 – Computer Network Programming

## End of the Headers

- Each header ends with a CRLF ( `\r\n` )
- The end of the header section is marked with a blank line.
  - just CRLF
- For GET and HEAD requests, the end of the headers is the end of the request!



CSCE515 – Computer Network Programming

## POST

- A POST request includes some *content* (some data) after the headers (after the blank line).
- There is no format for the data (just raw bytes).
- A POST request must include a Content-Length line in the headers:

**Content-length: 267**

CSCE515 – Computer Network Programming

## Example GET Request

```
GET /~wyxu/index.html HTTP/1.1
Accept: */*
Host: www.cse.se.edu
User-Agent: Internet Explorer
From: cheater@cheaters.org
Referer: http://foo.com/
```

← There is a blank line here!

CSCE515 – Computer Network Programming

## Example POST Request

```
POST /~wxy/changegrade.cgi HTTP/1.1
Accept: */*
Host: www.cse.sc.edu
User-Agent: SecretAgent V2.3
Content-Length: 35
Referer: http://monte.cs.rpi.edu/blah

stuid=6660182722&item=test1&grade=99
```

CSCE515 – Computer Network Programming

## Typical Method Usage

GET used to retrieve an HTML document.

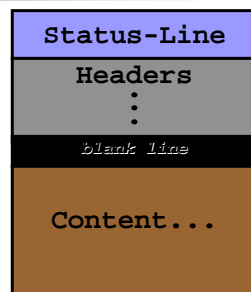
HEAD used to find out if a document has changed.

POST used to submit a form.

CSCE515 – Computer Network Programming

## HTTP Response

- ASCII Status Line
- Headers Section
- Content can be anything (not just text)
  - typically an HTML document or some kind of image.



CSCE515 – Computer Network Programming

## Response Status Line

*HTTP-Version Status-Code Message*

- Status Code is 3 digit number (for computers)
- Message is text (for humans)

CSCE515 – Computer Network Programming

## Status Codes

- 1xx Informational
- 2xx Success
- 3xx Redirection
- 4xx Client Error
- 5xx Server Error

CSCE515 – Computer Network Programming

## Example Status Lines

HTTP/1.0 200 OK

HTTP/1.0 301 Moved Permanently

HTTP/1.0 400 Bad Request

HTTP/1.0 500 Internal Server Error

CSCE515 – Computer Network Programming

## Response Headers

- Provide the client with information about the returned *entity* (document).
  - what kind of document
  - how big the document is
  - how the document is encoded
  - when the document was last modified
- Response headers end with blank line

CSCE515 – Computer Network Programming

## Response Header Examples

Date: Wed, 30 Jan 2002 12:48:17 EST

Server: Apache/1.17

Content-Type: text/html

Content-Length: 1756

Content-Encoding: gzip

CSCE515 – Computer Network Programming

## Content

- Content can be anything (sequence of raw bytes).
- **Content-Length** header is required for any response that includes content.
- **Content-Type** header also required.

CSCE515 – Computer Network Programming

## Content type

- Also known as:
  - Multipurpose Internet Mail Extensions (MIME) type
  - Internet media type
- A two-part identifier for file format on Internet
  - text/css: Cascading Style Sheets;
  - text/html: HTML;
  - text/plain: Textual data;
  - text/xml: Extensible Markup Language;

CSCE515 – Computer Network Programming

## Content type - More

- application/pdf: PDF files;
- application/msword: word files
- audio/mpeg: MP3 or other MPEG audio;
- audio/x-wav: WAV audio
- image/gif: GIF image;
- image/jpeg: JPEG JFIF image;
- image/tiff: Tag Image File Format;
- video/mpeg: MPEG-1 video with multiplexed audio;

CSCE515 – Computer Network Programming

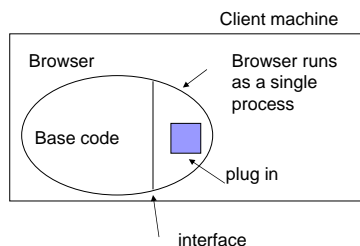
## Web browser

- text/html: display directly
- MIME type is not one of the build-in ones:
  - consults its table of MIME types
  - table associate a MIME type with a viewer
- To interpret a rapidly growing collection of file types:
  - Plug-in
  - helper application

CSCE515 – Computer Network Programming

## Plug-in

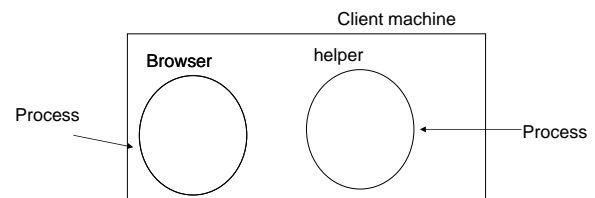
- A code module that the browser fetches from a special directory on the disk and installs as an extension to itself.



CSCE515 – Computer Network Programming

## Helper application

- A complete program, running as a separate process.



CSCE515 – Computer Network Programming

## Examples

- application/pdf: PDF files;
  - → Acrobat is automatically started
- application/msword: word files
  - → Microsoft word

CSCE515 – Computer Network Programming

## Single Request/Reply

- The client sends a complete request.
- The server sends back the entire reply.
- The server closes it's socket.
- If the client needs another document it must open a new connection.

This was the default for HTTP 1.0

CSCE515 – Computer Network Programming

## Persistent Connections

- HTTP 1.1 supports persistent connections (this is the default).
- Multiple requests can be handled over a single TCP connection.
- The **Connection:** header is used to exchange information about persistence (HTTP/1.1)
- 1.0 Clients used a **Keep-alive:** header

CSCE515 – Computer Network Programming

## Example

- wget [www.google.com](http://www.google.com)
- Using wireshark capture the file
- Homework,
  - Using browser to open [www.google.com](http://www.google.com), check whether the header is different from “wget”

CSCE515 – Computer Network Programming

## HTML Hyper-Text Markup Language

## HTML Basics

- I assume everyone knows something about HTML.
  - If not: check the home page for some links.
- Documents use elements to “mark up” or identify sections of text for different purposes or display characteristics
- Mark up elements are not seen by the user when page is displayed
- NOTE: Not all documents in the Web are HTML!

CSCE515 – Computer Network Programming

## HTML Tables:

- `<TABLE>` , `</TABLE>` start/end a table
- `<TR>` , `</TR>` start/end a table row
- `<TD>` , `</TD>` start/end a table cell
- `<TH>` , `</TH>` start/end table header cell

CSCE515 – Computer Network Programming

## timedate.com Hit Table

```
<TABLE>
<TR>
  <TH>hour</TH>
  <TH>number of hits</TH>
</TR>
<TR>
  <TD>12-1AM</TD>
  <TD>4,320</TD>
</TR>
<TR>
  <TD>1-2AM</TD>
  <TD>18,986</TD>
</TR>
</TABLE>
```

hour	number of hits
12-1AM	4,320
1-2AM	18,986
2-3AM	246

CSCE515 – Computer Network Programming

## HTML Example

```
<HTML>
<HEAD>
<TITLE> PB's HomePage </TITLE>
</HEAD>
<BODY>
<CENTER><IMG SRC = "bad_picture.gif" ALT = " " ><BR></CENTER>
<P><CENTER><H1>USC Computer Science and Engineering
Department</H1></CENTER>
Welcome to my goofy HomePage!
...
<A HREF = http://www.cse.sc.edu/~pb/mydogs\_page.html> Spot's Page </A>
</BODY>
</HTML>
```

CSCE515 – Computer Network Programming

## Concurrent Server Design Alternatives

## Concurrent Server Design Alternatives

- One child process per client
- Spawn one thread per client
- Preforking multiple processes
- Prethreaded Server

CSCE515 – Computer Network Programming

## One child per client

- Traditional Unix server:
  - TCP: after call to `accept()`, call `fork()`.
  - UDP: after `recvfrom()`, call `fork()`.
  - Each process needs only a few sockets.
  - Small requests can be serviced in a small amount of time.
- Parent process needs to clean up after children!!!! (call `wait()`).

CSCE515 – Computer Network Programming

## One thread per client

- Almost like using fork - call `pthread_create` instead.
- Using threads makes it easier (less overhead) to have sibling processes share information.
- Sharing information must be done carefully (use `pthread_mutex`)

CSCE515 – Computer Network Programming

## Example

```
main(int argc, char **argv)
{ listenfd=socket(...)
  Bind(listenfd...)
  Listen(listenfd,LISTENQ);
  Signal(SIGCHLD, sig_chld);
  Signal(SIGINT,sig_int);
  For( ; ; ) {
    connfd = Accept(listenfd,
...);
    if ( (pid = Fork())==0) {
      Close(listenfd);
      doit(connfd);
      Close(connfd);
      exit(0);
    }
  }
  Close(connfd);
}
```

**Process version**

```
main(int argc, char **argv)
{ pthread_t tid;

  listenfd=socket(...)
  Bind(listenfd...)
  Listen(listenfd,LISTENQ);
  For( ; ; ) {
    connfd =
    Accept(listenfd, ...);
    Pthread_create(&tid,
NULL, &doit, (void *)
connfd);
  }
  static void doit (void *arg)
  {
    ...
    Close( (int) arg);
    return NULL;
  }
}
```

**Thread version**

CSCE515 – Computer Network Programming

## Prefork()’d Server

- Creating a new process for each client is expensive.
- We can create a bunch of processes, each of which can take care of a client.
- Each child process is an iterative server.

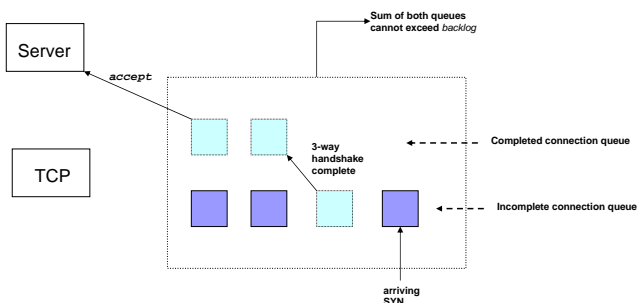
CSCE515 – Computer Network Programming

## Prefork()’d TCP Server

- Initial process creates socket and binds to well known address.
- Process now calls `fork()` a bunch of times.
- All children call `accept()`.
- The next incoming connection will be handed to one child.

CSCE515 – Computer Network Programming

## listen()



CSCE515 – Computer Network Programming

## Preforking

- As the book shows, having too many preforked children can be bad.
- Using dynamic process allocation instead of a hard-coded number of children can avoid problems.
- The parent process just manages the children, doesn't worry about clients.

CSCE515 – Computer Network Programming

## Sockets library vs. system call


- A preforked TCP server won't usually work the way we want if `sockets` is not part of the kernel:
  - calling `accept()` is a library call, not an atomic operation.
- We can get around this by making sure only one child calls `accept()` at a time using some locking scheme.

CSCE515 – Computer Network Programming

## Prethreaded Server

- Same benefits as preforking.
- Can also have the main thread do all the calls to `accept()` and hand off each client to an existing thread.

CSCE515 – Computer Network Programming



## What's the best server design for my application?

- Many factors:
  - expected number of simultaneous clients.
  - Transaction size (time to compute or lookup the answer)
  - Variability in transaction size.
  - Available system resources (perhaps what resources can be required in order to run the service).

CSCE515 – Computer Network Programming



## Server Design

- It is important to understand the issues and options.
- Knowledge of queuing theory can be a big help.
- You might need to test a few alternatives to determine the best design.

CSCE515 – Computer Network Programming



## Assignment & Next time

- Reading:
  - UNP 30
  - RFC 2396: <http://www.ietf.org/rfc/rfc2396.txt>
  - [HTTP 1.1](#)  
[HTTP 1.0](#)
- Next Lecture:
  - Advanced socket programming

CSCE515 – Computer Network Programming