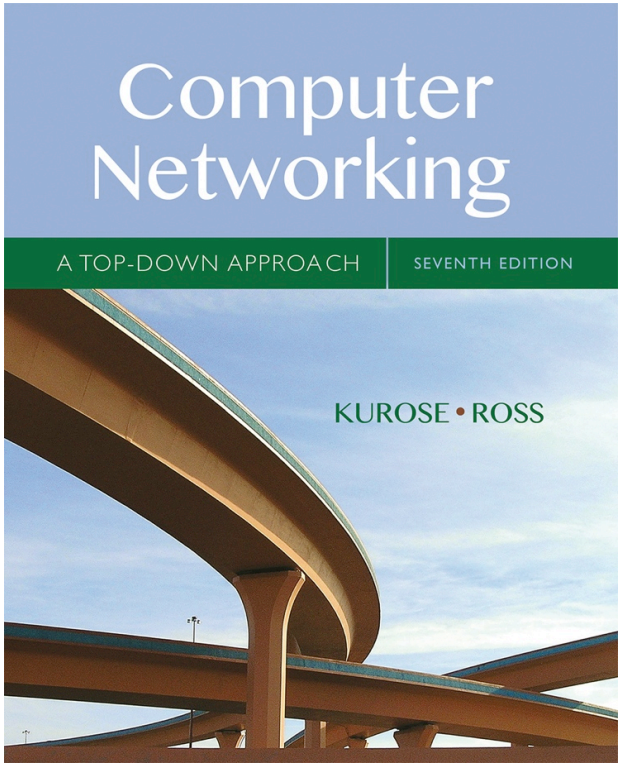
CN-Basic
L14

HTTP Overview

Dr. Ram P Rustagi
rprustagi@ksit.edu.in
http://www.rprustagi.com
https://www.youtube.com/rprustagi

Resources Acknowledgement

Chapter 2
Application Layer



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Resources

- RFC 1935: What is internet
- RFC 2616: HTTP Protocol
- http://www.iana.org/assignments/media-types/index.html

The Web

- Different from other TV, Radio
  - –Operates on demand
    - •Does not force users to tune in
  - –Enables every one to become the producer
  - –Channels (URLs) are way too many
  - –Search engines enables easy navigation
- RIAs make it exciting
  - –Users become engross

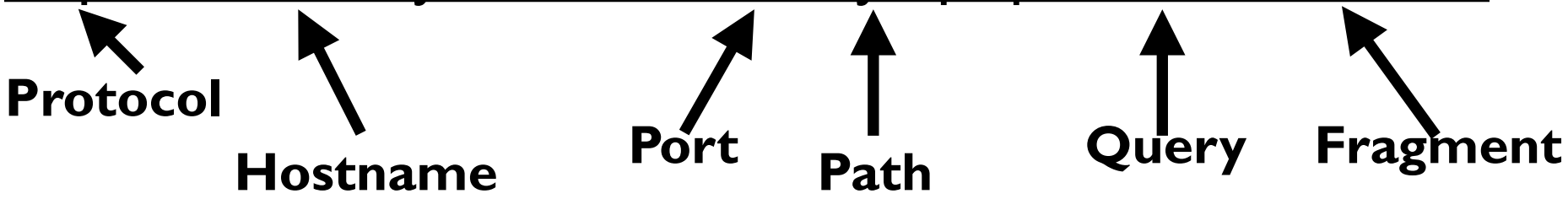
Overview of Web

- Web Server
  - –Apache, Nginx, IIS, Embedded WS
- Web Clients
  - –Display: GUI browsers, lynx
  - –Fetch: wget, curl
- Communication protocols
  - –HTTP, HTTPS
  - –Extension of HTTP
    - •WebDAV, CardDAV, CalDAV…

Web and HTTP

- *First, a review…*
- What is a *web page*
  - It consists of *objects*
  - Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of *base HTML-file* which includes *several referenced objects*
- Each object is addressable by a *URL,* e.g.,

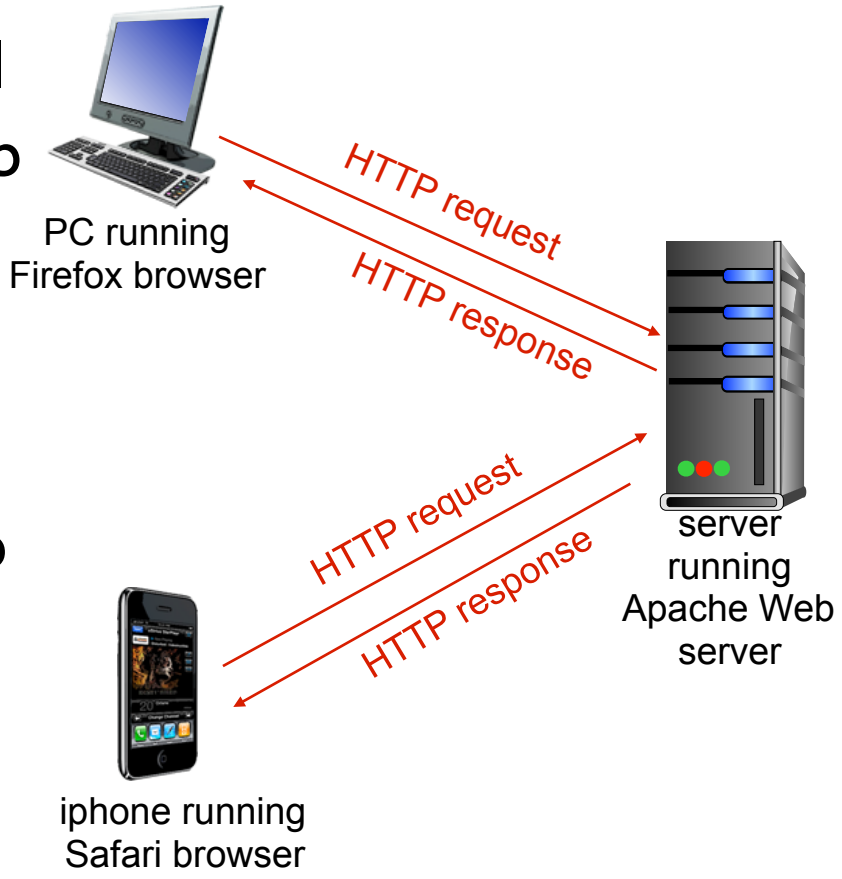
https://www.myweb.com:80/tryit.php?fn=first#Name



HTTP overview

- HTTP: hypertext transfer protocol

- Web's application layer protocol
- Defines how client requests web page from server
- client/server model
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests



HTTP overview (continued)

- *uses TCP:*
- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

- *HTTP is "stateless"*
- server maintains no information about past client requests

aside

- protocols that maintain "state" are complex!
- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

Stateless protocol

- Server has no information about previous request
- When client re-requests the web page
    - Few seconds after the previous request
  - Server will send the page again
  - Will not say that page was served just now
- Explain Stateless (in non-technical context)
  - E.g. Re-answering the question again and again

HTTP Protocol

- First interaction/implementation
  - A subset of intended protocol
  - (unofficially) labeled as HTTP 0.9
- HTTP 0.9
  - Client-server, request-response protocol
  - ASCII protocol, running on TCP/IP
  - Design to xfer HTML document
  - Connection is closed after each request
  - No meta data (HTTP headers)

HTTP 1.0

- Key protocol changes
  - Request has multiple header lines
  - Response is prefixed with status line
  - Response has its own header lines
  - Response can be non-HTML
    - A plain text file, image, other contents
  - TCP connection closed after response served
  - Other supports
    - Content encoding, character set, multi-part
    - Authentication, caching, proxy behaviours,
    - Date formats …

HTTP 1.1

- RFC 2068 - First official standard (Jan 1977)
- RFC 2616 - Current standard (June 1999)
- A lot of performance optimizations
  - Keep alive connections
  - Chunked encoding transfers
  - Byte range requests
  - Additional caching mechanims
  - Request pipelines
  - Language negotiations
  - Caching directives

HTTP/2

- Goals:
  - Impove transport performance
  - Lower latency and higher thruput
  - No changes in high level semantics
    - All headers, values, use cases are same
  - Any existing HTTP application should work without modification
  - Any server upgrades should be transparent to majority of users

HTTP Messages

- Two types
  - –Request Message
  - –Response Message
- Data is in clear text
  - –Readable by humans
- Structure
  - –Message line
  - –Header lines
  - –Empty lines
  - –Data

HTTP Request Message

- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII (human-readable format)

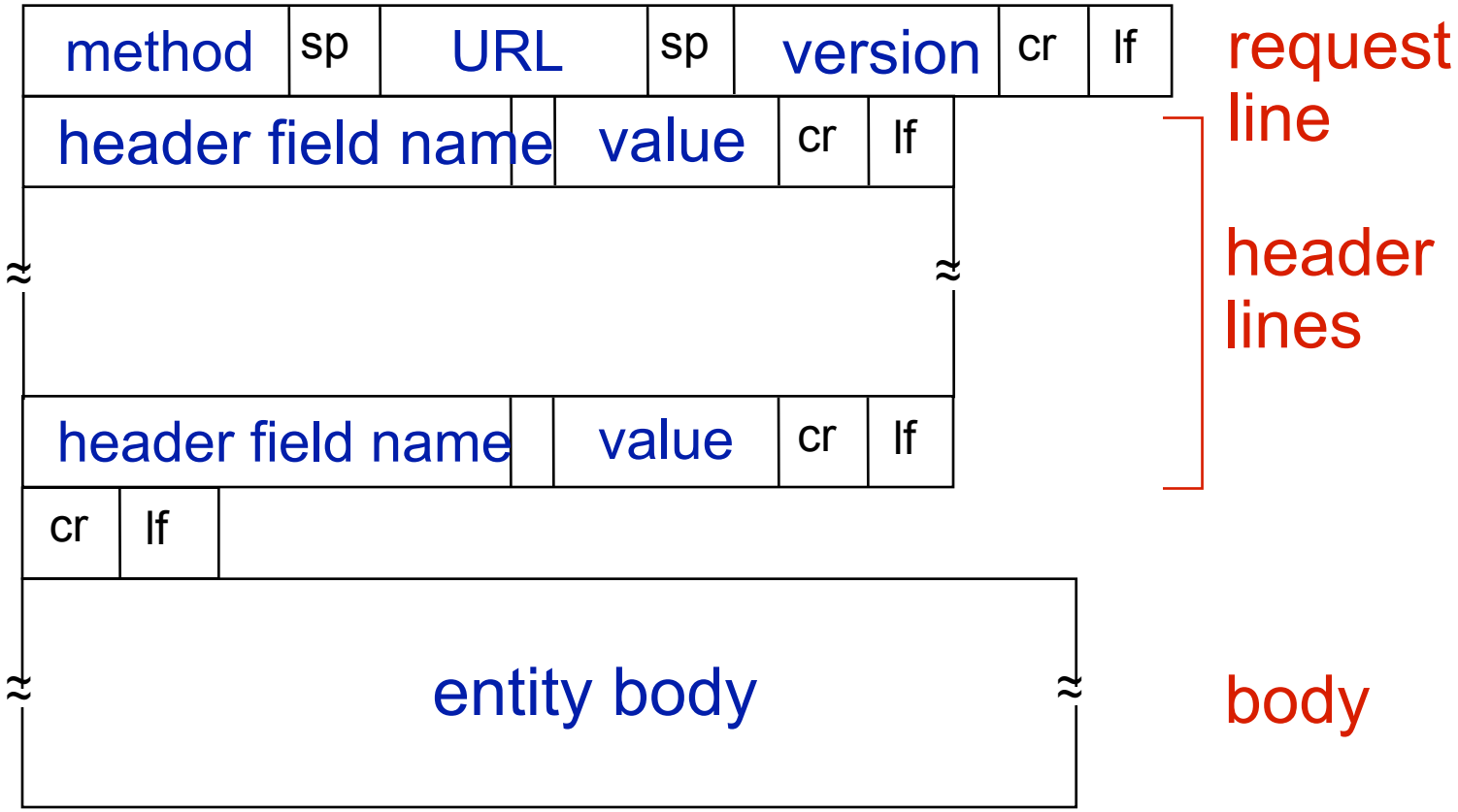
request line (GET, POST, HEAD)

header lines

\r\n at start of line indicates end of header lines

```
GET /Workshops/web/welccome.html HTTP/
Accept: text/html,image/*,*/*\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9,hi;q=0
Connection: keep-alive\r\n
Host: rprustagi.com\r\n
Referer: http://rprustagi.com/workshop
   \r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Macintosh; In
   OS X 10_14_6)…\r\n
\r\n
```

HTTP Request Message: General Format



Uploading Form Input

- POST method:
- web page often includes form input
- input is uploaded to server in entity body

- URL method:
- uses GET method
- input is uploaded in URL field of request line:
  - Query String

Example:

- http://rprustagi.com/workshops/web/formsget.html
- http://rprustagi.com/workshops/web/formspost.html
- Use both GET and POST methods and see the difference

HTTP response message

status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

```
HTTP/1.1 200 OK\r\n
Date: Fri, 23 Aug 2019 16:46:27
  GMT\r\n
Server: Apache\r\n
Upgrade: h2,h2c\r\n
Connection: Upgrade, Keep-Alive\r\n
Last-Modified: Tue, 03 Jul 2018
  17:27:18 GMT\r\n
Accept-Ranges: bytes\r\n
Content-Length: 209\r\n
Keep-Alive: timeout=2, max=100\r\n
Content-Type: text/html\r\n
\r\n
data data data data data ...
```

MIME Types

- Originally for email
- Specifies the form of content served
  - –Form: Type/subtype
    - Examples
      - `text/plain, text/html,`
      - `image/gif, image/jpeg,`
      - `audio/mpeg, video/quicktime,`
      - `application/msword,`
        `application/powerpoint`
- Server gets type from file name suffix
  - –pictures.html vs pictures.txt
- Reference: http://www.iana.org/assignments/media-types/index.html

HTTP Status Code

- `1xx` - Informational
  - Request received, continuing process
- `2xx` - Success
  - Action successful, understood and accepted
- `3xx` - Redirection
  - Further action must be taken to complete 4xx
- `4xx` - Client Error
  - Request contains bad syntax or cannot be filled
- `5xx` - Server Error
  - Server failed to fulfil an apparently valid request

HTTP response status codes

- **200 OK**
  - request succeeded, requested object later in this msg
- 301 Move Permanently
  - requested object moved, new location specified later in this msg (Location:)
- 400 Bad Request
  - request msg not understood by server
  - Missing parameters
- 401 Authorization Failed
- 403 Forbidden Access
- 404 Not found
  - requested document not found on this server

HTTP response status codes

- Status codes 5xx
  - **500 Internal Server Error**
    - **http://localhost/cgi-bin/rprustagi/example-die.pl**
  - **505 HTTP Version Not Supported**

4xx Status codes

HTTP Headers

- `Content-Type: text/html`
- `Content-Length: 209`
- `Accept-Language: en-US`
  - Determines your preference
- `Accept-Encoding: gzip, deflate`
  - Determines compressed download
- `Keep-Alive: timeout=2, max=100`
- `User-Agent:`
  - Determines client type
  - Mobile, web, tablet

Summary

- HTTP Protocol
  - URL structure
- HTTP headers
  - Request and Response
- HTTP Methods
  - GET, POST
- Status Codes