CS 348
Computer Networks
Lec 12

Spring 2020 IIT Goa

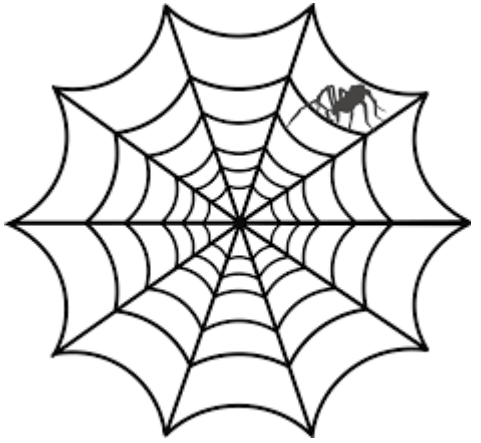
Course Instructor: Dr. Neha Karanjkar

Disclaimer: These slides are based on the content in “Computer Networking: A Top-down Approach by Kurose & Ross, 7th ed” and some specific topics are referenced from Wikipedia.

Questions

- What is the “Interface” between applications and the Internet? How can applications use services of the layers below? **The Sockets API**
- Some popular applications, how they work, protocols they use:
  - **The Web** and HTTP, Email, Peer-to-peer applications
- How can “names” be translated to IP addresses? **DNS**

The Web and HTTP



The World Wide Web (WWW) also known as **The Web**

- Is an **Information-sharing** application that uses the Internet

  How is Knowledge/Information structured?

  - As a graph or “web”. One topic connects/links to others.

- The Web has a **Client-server architecture:**
  - A Client process (web broswer) requests web-pages from a Server and displays them to the user.

History of the Web

- Invented by **Sir Tim Berners-Lee in 1989 at CERN**
- **1990:** Berners-Lee **w**rote the first web browser and server program

Interesting Ref:
https://webfoundation.org/about/vision/history-of-the-web/

The World Wide Web Consortium (**W3C**) maintains open standards related to the Web

Components of The Web

- **Documents** (and other resources such as Images/videos)
- **Links** between these documents/objects
- **An addressing scheme:**
  - For uniquely identifying each document/object

Components of The Web

- **Documents** (and other resources such as Images/videos): **Web Resources**
- **Links** between these documents/objects: **Hyperlinks**
- **An addressing scheme: Uniform Resource Locator (URL)**
  - For uniquely identifying each document/object

Components of The Web

- **Documents** (and other resources such as Images/videos): **Web Resources**
- **Links** between these documents/objects: **Hyperlinks**
- **An addressing scheme: Uniform Resource Locator (URL)**
  - For uniquely identifying each document/object
- **The sending/receiving programs: Web Browser** and **Web Server**
- **A Protocol** for commmunicating between the sending/receiving programs:
  - **Hyper Text Transfer Protocol (HTTP)**
- **A standard for formatting/displaying the Information:**
  - **Hyper Text Markup Language (HTML)**

The Web

- **Web Page**: A single document

- **Website:** Multiple web resources with a common theme, a common domain name, or both, make up a website.

- **Web Server:** Websites are stored in computers that run a program called a web server that responds to requests made over the Internet from web browsers running on client computers.

HTML

- **Need some way of expressing how to display/format the information**
- **Hypertext Markup Language (HTML)**
  - Allows creators to express formatting info such as:
    - Bold/italic text, font size, style and color, images, image position ... and much more.

- **How to Create Webpages? HTML/CSS Tutorials:** https://www.w3schools.com/html/default.asp
- Demo: view template files index.html and page2.html in a text editor

  Also hosted (temporarily) at: https://www.iitgoa.ac.in/~nehak/example.html

Structure of a URL

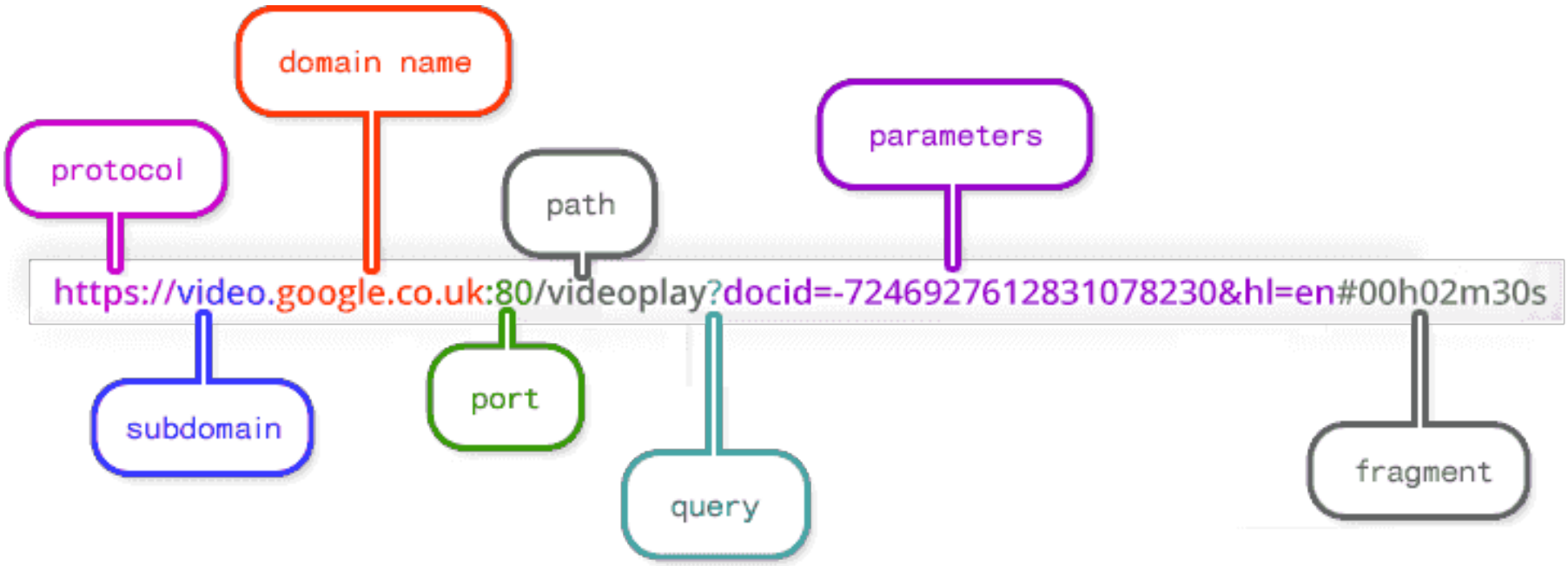


Image Source: https://doepud.co.uk/blog/anatomy-of-a-url

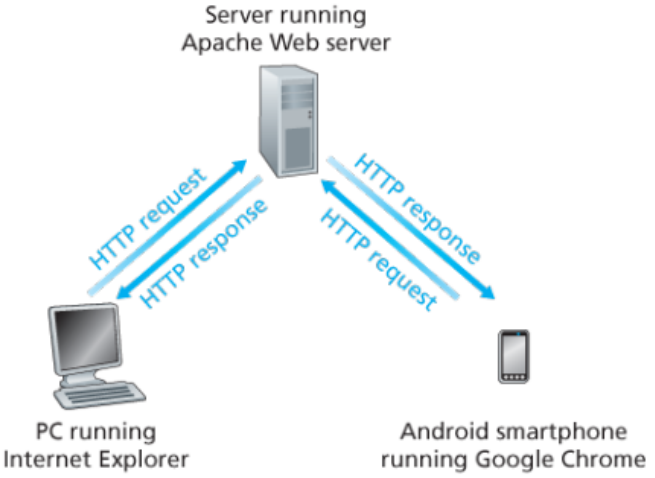
HTTP

- Defines two kinds of messages: **REQUEST** and **RESPONSE**
  - Client sends a REQUEST , server sends back a RESPONSE.
  - Both consist of simple, ASCII encoded text
- Originally: Just one type of Request

  “GET <url>”

  and one type of Response

  “OK <..contents of the webpage...>”



HTTP

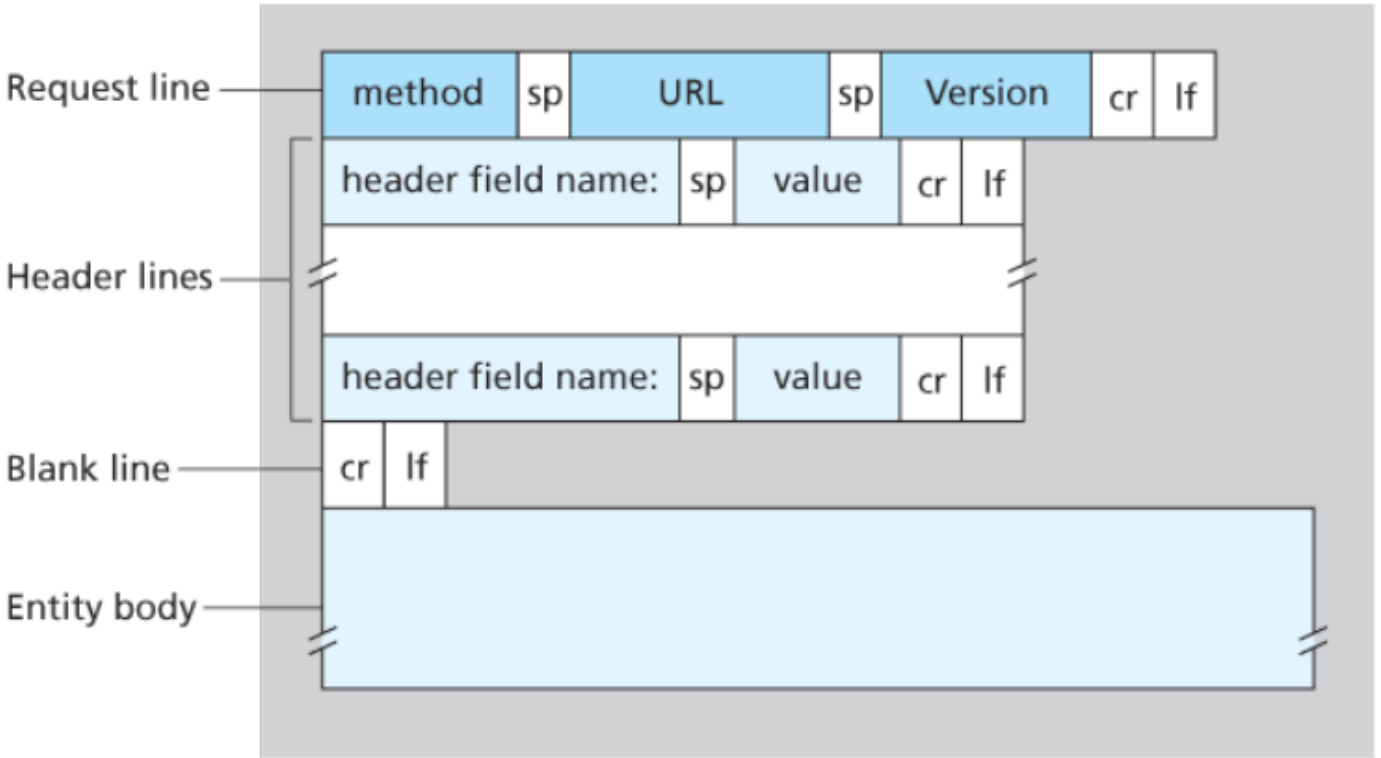
- HTTP uses TCP underneath (for reliable, in-order delivery)
- **An HTTP server is STATELESS**
  - Server treats each request independently, assuming no relation between successive requests
  - Server **does not** keep track of the Client's state

HTTP

Browse Through: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

- Types of **REQUESTS**:
  - Most Common: **GET <url>**: Request for a webpage at the given url
  - Some other types of requests:
    - **PUT <data, url>** : store the enclosed data/object at specified url
    - **POST <data, url>**: add data/object to the webpage at the url
      (eg. post message into a bulletin, post data into a form etc)
    - **DELETE <url>**: delete resource at the specified url
- **RESPONSES**:
    - Status (“OK”, “ERROR”, “NOT Found” etc), and Response Body

Format of an HTTP Request



Example of an HTTP Request

```
GET /~nehak/example.html HTTP/1.1
Host: www.iitgoa.ac.in
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
Accept:text/html
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

To view the actual HTTP traffic flowing through your browser (for Google Chrome):

[F12] --> Network Tab --> Request/Response headers (view src)

Format of an HTTP Response

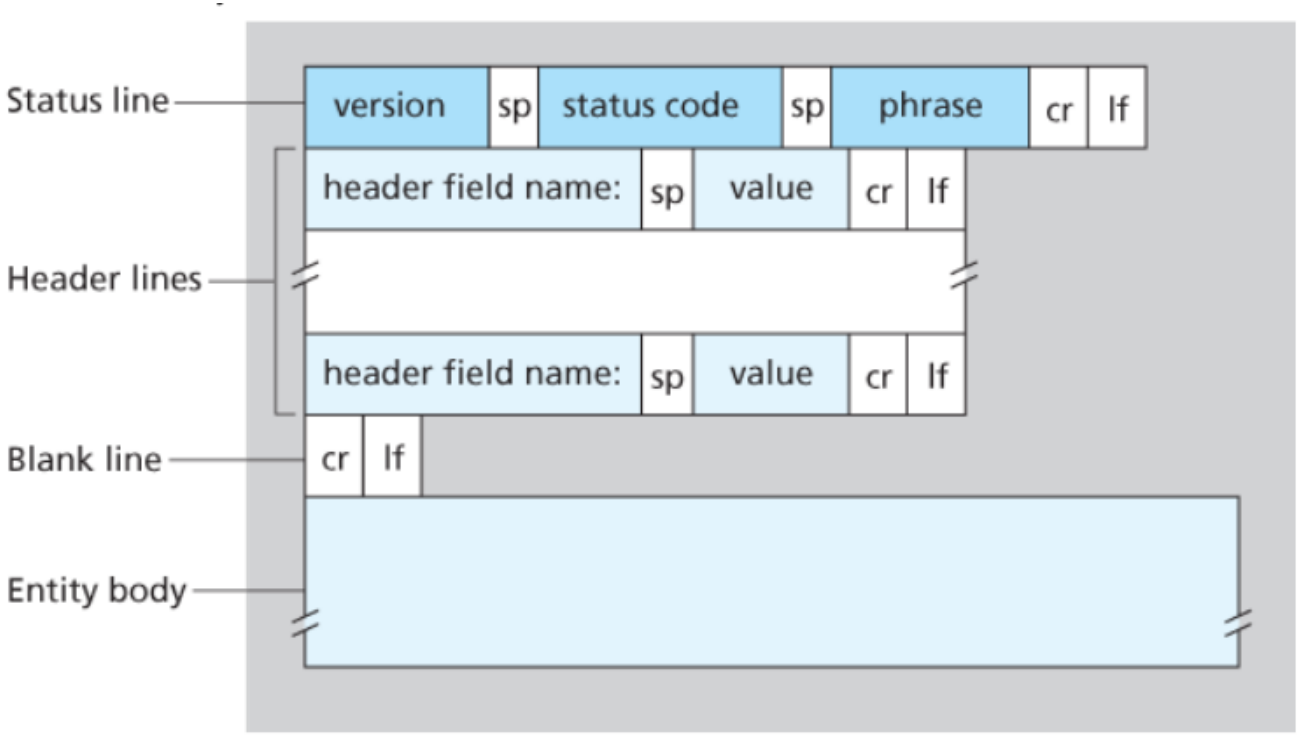
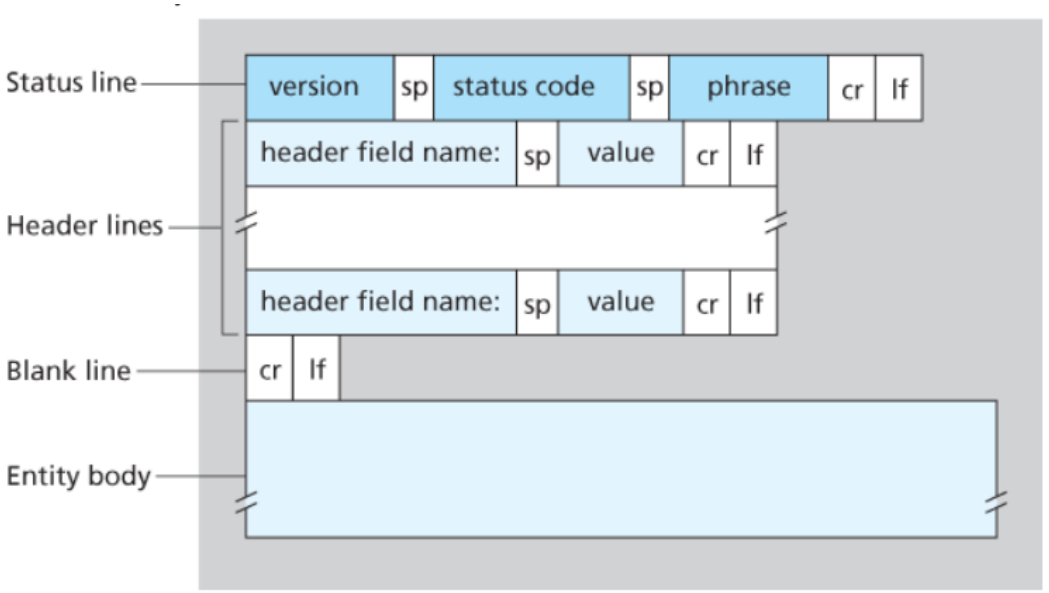


Image Source: Kurose and Ross 7th edition

Format of an HTTP Response



Some Common Response Codes:

- **`200 OK`**
  request succeeded, requested object later in this msg
- **`301 Moved Permanently`**
  requested object moved, new location specified later in this msg (Location:)
- **`400 Bad Request`**
  request msg not understood by server
- **`404 Not Found`**
  requested document not found on this server
- **`505 HTTP Version Not Supported`**

Image Source: Kurose and Ross 7th edition

HTTP

- **Non-Persistent HTTP:**
  - Exactly one request-response pair per TCP connection. Then the connection is closed.
- **Persistent HTTP:**
  - Multiple requests sent over a single TCP connection. The connection is kept open by the server and client for a certain amount of time.
  - Reduces TCP connection overhead.
- **Default** behavior for most browsers: Persistent HTTP, Use Pipelining and Parallelism in addition.

Cookies

- **A mechanism to identify a session/user on top of a stateless HTTP server**
- How this works:
  - There is a server-side database
  - Server sends a “Set-cookie:<cookie ID>” header to the Client upon first visit
  - Client stores all cookies until they expire
  - Client includes a header “Cookie:<cookie ID>” in all subsequent requests to the same server
  - Server identifies the user/session from this cookie ID
- Privacy and Security Issues? ....

Web Cache/Proxy Server

- The Caching concept... Cache HIT and Cache MISS
- Web Cache: stores copies of recently accessed pages and returns them to the requester on behalf of the origin web server.
- Purpose:
  - Reduce response times for client requests
  - Reduce traffic on the access link
- What if the webpage is updated in the origin and the cached copy is stale?
  - Freshness: Cached copy invalidated after a certain amount of time
  - Validation: Conditional GET (with a header “If-modified-since:<time>) to check if the cached copy should be updated

Questions

- How are URLs translated into IP addresses?

References and Reading Assignment

- **Kurose and Ross 6th ed**
  - **Section 2.2: The Web and HTTP**

- Wikipedia entry about the WWW: https://en.wikipedia.org/wiki/World_Wide_Web
- [Cold Fusion] A Brief History of the Internet: https://www.youtube.com/watch?v=8sTy8466MoE