ECE 448/528
Application Software Design

Lecture 8. The Hypertext Transfer Protocol
Spring 2025

Won-Jae Yi, Ph.D.

Department of Electrical and Computer Engineering
Illinois Institute of Technology

The Hypertext Transfer Protocol (HTTP)

Standard Network Services and Well-Known Ports

- Many different services are available from the Internet.
  - Standardization efforts, e.g., the RFC (Request for Comments) documentation, provide details of such services.
  - Each service will appoint one or more well-known ports that clients may connect.
  - Well-known ports are usually below 1,024 and OS requires root privilege to open those ports.
- Over the years, many such services become obsolete due to security or usability concerns while a few remain.
- An incomplete list of services we may use daily:
  - DNS (Domain Name System): port 53 TCP/UDP
  - NTP (Network Time Protocol): port 123 TCP/UDP
  - HTTP (Hypertext Transfer Protocol): port 80 TCP
  - HTTPS (HTTP over TLS/SSL): port 443 TCP
  - SSH (Secure Shell): port 22 TCP
  - TFTP (Trivial File Transfer Protocol): port 69 UDP
  - RDP (Remote Desktop Protocol): port 3389 TCP

IP Addresses and Domain Names

- It is not convenient to use IP addresses directly.
  - Difficult to memorize.
  - Not flexible: what if the server needs to be moved to a new IP address?
- DNS services: translate domain names that humans can remember into Internet addresses.
  - e.g. “www.iit.edu” to 50.19.226.237
  - Most tools like ping and the browser will automatically utilize DNS before connecting to the actual server.
- Domain names are managed and serviced hierarchically.
  - Individuals can buy them from providers and rely on them to provide DNS services.
  - Companies may maintain their own DNS servers to provide additional services like load balancing.

Uniform Resource Identifier (URI)

- A string to identify a resource.
  - We'll be clearer on what is a resource later in the lecture.
- **scheme://authority/path?query**
  - http://ecasp.ece.iit.edu/testxe/index.php?mid=ecasp_summer_2024&document_srl=9901
- Consist of ASCII alphanumeric characters and punctuations: -_.!~.
  - Other characters are encoded in UTF-8 using % followed by the hexadecimal codes.

Uniform Resource Locator (URL)

- A URI that also provides a specific network location for the resource.
- **`protocol://user@host:port/path?query#fragment`**
  - Usual **protocol**: file, http, https
  - **user@** may be omitted when there is no authentication
  - **host** can be IP address or domain name
  - **port** may be omitted under default settings protocol
  - **path** like /forum/index.php may look like a filesystem path, it may or may not map to an actual file.
  - query provides additional arguments to the server, in the format of **key=value pairs separated by &.**
  - **fragment** refers to a particular part of the resource.

Hypertext Markup Language (HTML)

- A type of resource representation that allows to hyperlink to other resources.
  - e.g. other HTML pages and images.
- A tree-like structure represented as contents surrounded by start tags (<tagname>) and end tags (</tagname>).
  - The root of the tree is the <html> element.
  - The <body> element directly under <html> defines the content of the web page.
  - <p> elements are used for paragraphs.
  - <a> elements are used for hyperlinks.
- Each element can have attributes inside the start tag as `key="value"` pairs separated by spaces.
  - e.g., `<a href="javafaq.html">` where the href attribute gives the hyperlink.
- **Please refer to https://www.w3schools.com/html/ for more thorough tutorial.**

Relative Links

- **href attribute is relative.** Consider the web page

http://www.ibiblio.org/javafaq/javatutorial.html

  - For <a href="javafaq.html">, it refers to **http://www.ibiblio.org/javafaq**/javafaq.html

  - For <a href="/projects/ipv6/">, it refers to **http://www.ibiblio.org**/projects/ipv6/

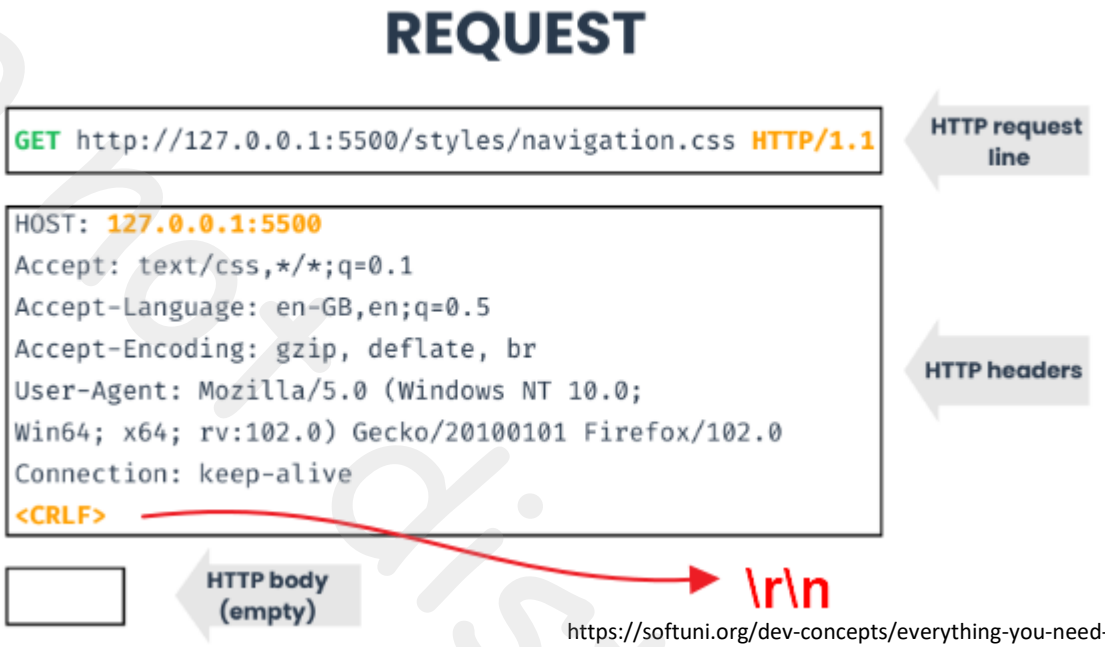
Hypertext Transfer Protocol (HTTP)

- A request-response protocol on top of byte streams.
  - Default to port 80 TCP.
- Both request and response consist of a header line, lines of header fields, an empty line, and an optional message body.
  - Since all lines are delimited by `"\r\n"`, similar to our `ReverseServer`, we can use terminal programs like telnet to explore HTTP.
- The request header line: `METHOD path?query HTTP/ver`
  - e.g.) `GET / HTTP/1.1`
    - **GET**: METHOD; **/**: path; **HTTP/1.1**: HTTP version 1.1
- Request header fields: `Keyword: Value`
  - For example…
    - Host: domain name of the website
    - User-Agent: name of the program sending the request.

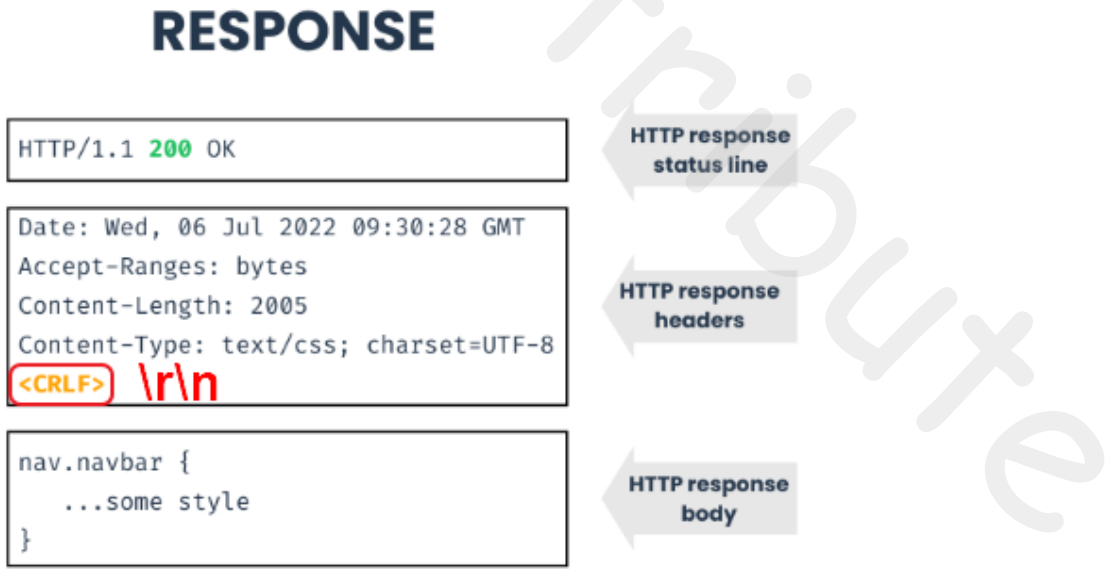
Hypertext Transfer Protocol (HTTP) Cont.

- The response header line: `HTTP/ver StatusCode Msg`
  - e.g.) `HTTP/1.1 200 OK`
  - Status code ranges
    - **2xx**: successful
    - **3xx**: relocation and redirection
    - **4xx**: client error
    - **5xx**: server error
- A few response header fields
  - `Content-Type`: type of the message body
    - `text/html` for HTML pages and `image/png` for PNG images.
  - `Content-Length`: length (in bytes) of the message body.
- Most modern browsers have the Inspect Element functionality that allows you to inspect the details of HTTP sessions.

Hypertext Transfer Protocol (HTTP) Cont.



https://softuni.org/dev-concepts/everything-you-need-to-know-about-http-protocol/

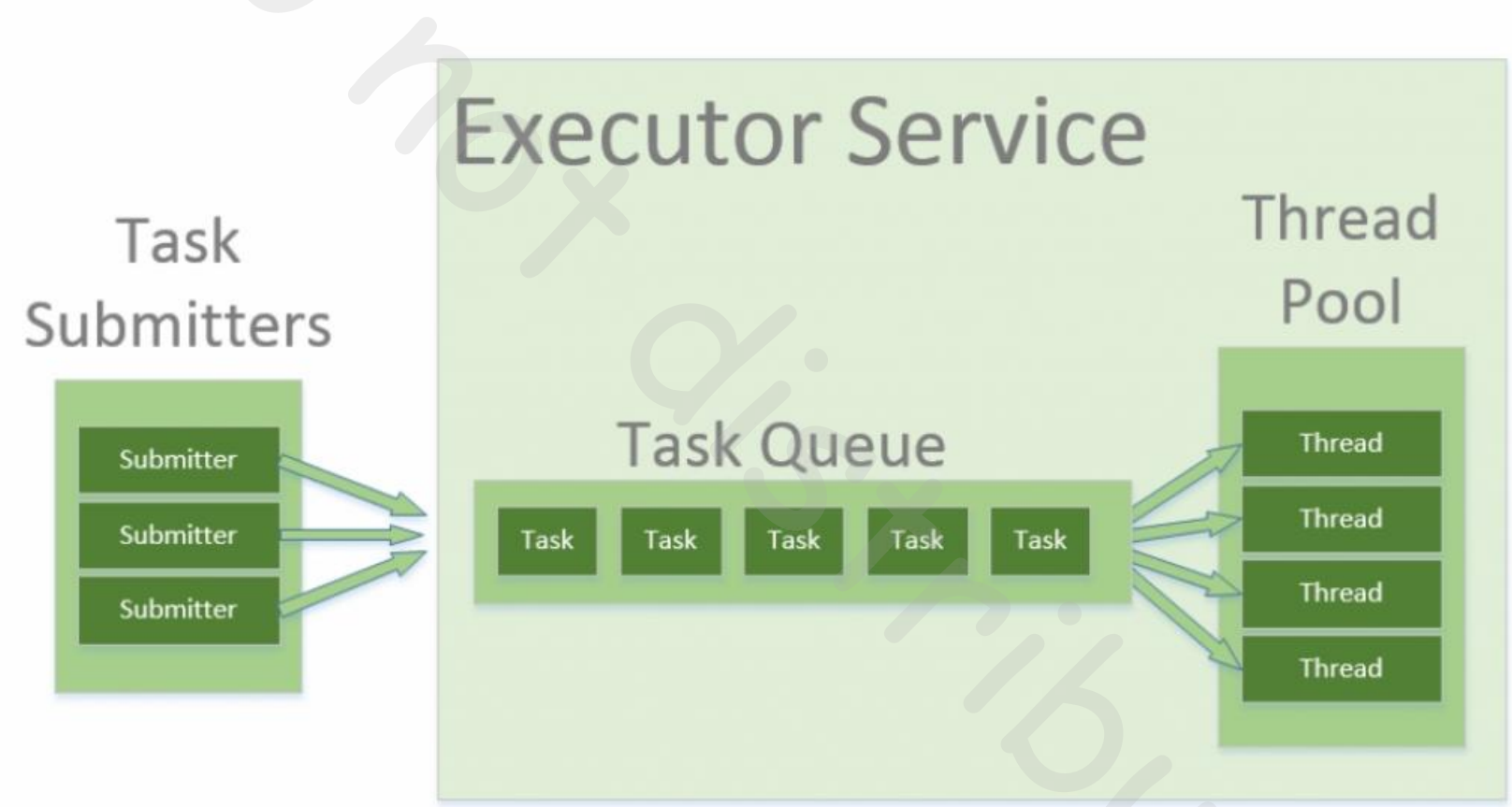


Simple HTTP Server Design

Simple HTTP Server Design

- Adopted from the textbook, in `src/main/ece448/lec08`.
  - Under branch `lec08-simple`.
- `JHTTP` to accept client connections.
  - Use a thread pool to serve clients.
- `RequestProcessor` to serve local files.
  - Extract a single request.
  - Locate and read the local file.
  - Send the response back.
  - Generate error response when necessary.

ThreadPool using ExecutorService



https://www.baeldung.com/thread-pool-java-and-guava