Csci 4131

HTTP wrapped up
Introduction to Node.js

Lecture 13, February 28th

Spring 2024

Dr. Dan Challou

Logistics – Csci 4131 Lecture 13, February 28th

- Exam 1 Results are available on GradeScope – Regrade request window closes Monday (March 4th at 11:59pm).

- My virtual office hours for this Thursday (and this Thursday only) have been moved to Today from 4:30-5:30 pm – same Zoom connection information as on Thursdays from 1 to 2 pm

- **HW 3 is due this Sunday, March 3rd at 11:59 pm**. Special late submission policies and penalties apply – see the assignment write-up for details. There will be no instructional support over break (we are on break too)

- zyBooks HW 6 is available in your zyBook and due on March 10th

- HW 4 will be posted on the class Canvas site (in the assignments section) over the break

- Exam 1 Results are available on GradeScope – Regrade request window closes Monday (March 4st at 11:59pm).

Note

- Solutions to HW assignments are available and can be reviewed at any office hour, BUT
  - **you will have to take notes.**
- We don't make electronic copies available, and you aren't allowed to take pictures or screen shots without our permission

Reading - HTTP Protocol (HW 4 Refs)

- Foundation of the WWW
- Target – impart a deeper understanding of how the HTTP protocol works, along with a better understanding of browser and web server function
- **Reading:**
  - https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

(The file named: In Introduction to HTTP Basics.pdf in the Resources section of the class Canvas site)

- RFC 2616 (HTTP 1.1)

https://tools.ietf.org/html/rfc2616

http://www.w3c.org/Protocols/

https://www.jmarshall.com/easy/http/ (This is a nice site too)

Upcoming Reading and Tutorials: Node.js, JSON, Ajax

- Node.js
  - https://www.w3schools.com/nodejs/default.asp
  - https://www.tutorialspoint.com/nodejs/
  - https://nodejs.org/en/docs/guides
- JSON
  - https://www.w3schools.com/js/js_json_intro.asp
  - https://www.json.org/
  - Optional: Sebesta – Chapters 10, Section 3.3
- AJAX
  - https://www.w3schools.com/xml/ajax_intro.asp
  - Optional: Sebesta – Chapter 10

Last Time

- HTTP Details Request and Response Messages Revisited
  - Reviewed the HTTP Post Message
  - HTTP Error Codes / Error Messages
- Started to review a Presentation Layer Example (EchoClient and EchoServer)

Today -

- Review Lecture 12 Exercises
- Review Example of building an HTTP server on top of the Presentation Layer (Building and testing a Get request
- Introduction to Node.js?

Questions?

Lecture 12, Exercise 1 Review

Assume (DO NOT ACTUALLY DO THIS): You type the following into your browser's address bar:
http://www.csci4131.edu/test.html (<- *this **NOT** a real url*) and hit the enter key

Assume: On the server (www.csci4131.edu), the file **test.html** exists, has read permissions and contains the following:

<html><body> Hello World </body><html>

Manually (in writing – do not use your computing device)

Specify:

a) The exact contents of the request line in the http request message sent by the browser to the server

b) The exact contents of the status line of the http response message sent by the server back to the browser making the request

c) The content-type of the response message body

Lecture 12, Exercise 2 Review:

An HTTP 1.1. Compliant Python webserver is running on the host computer:

```
csel-kh1262-11.cselabs.umn.edu.
(note,linux runs on CSELABs computers)
```

The server was executed (i.e., run) from the `/webserver` folder (directory) (which has world executable permissions) and is listening on port `9004`. The contents of `the /webserver` folder (directory) are as follows:

```
-rw------- 1 x500user CSEL-student 3275 Oct 3 07:25 server.py
-rw-r--r-- 1 x500user CSEL-student 1261 Oct 4 17:42 schedule.html
-rw-r--r-- 1 x500user CSEL-student  368 Oct 4 21:40 main.js
-rw-r--r-- 1 x500user CSEL-student 2561 Oct 4 17:42 my schedule.html
```

What HTTP 1.1 message response code will be sent by the server to the client/browser after the following requests are sent by the client/browser:

```
GET /webserver/schedule.html HTTP/1.1                200
Host: csel-kh1262-11.cselabs.umn.edu
Accept: */*

DELETE /webserver/main.js HTTP/1.1                   405 (1)
Host: csel-kh1262-11.cselabs.umn.edu                 401 (2)
Accept: */*                                          200 (3)
                                                     403  (4)

HEAD /webserver/my schedule.html HTTP/1.1            400 (1)
Host: csel-kh1262-11.cselabs.umn.edu                 200 (windows only)
```

Unix / Linux: File Permission/Access

- https://www.tutorialspoint.com/unix/unix-file-permission.htm
- https://www.geeksforgeeks.org/permissions-in-linux/

Questions?

Building a Simple (limited functionality) HTTP Server in Python

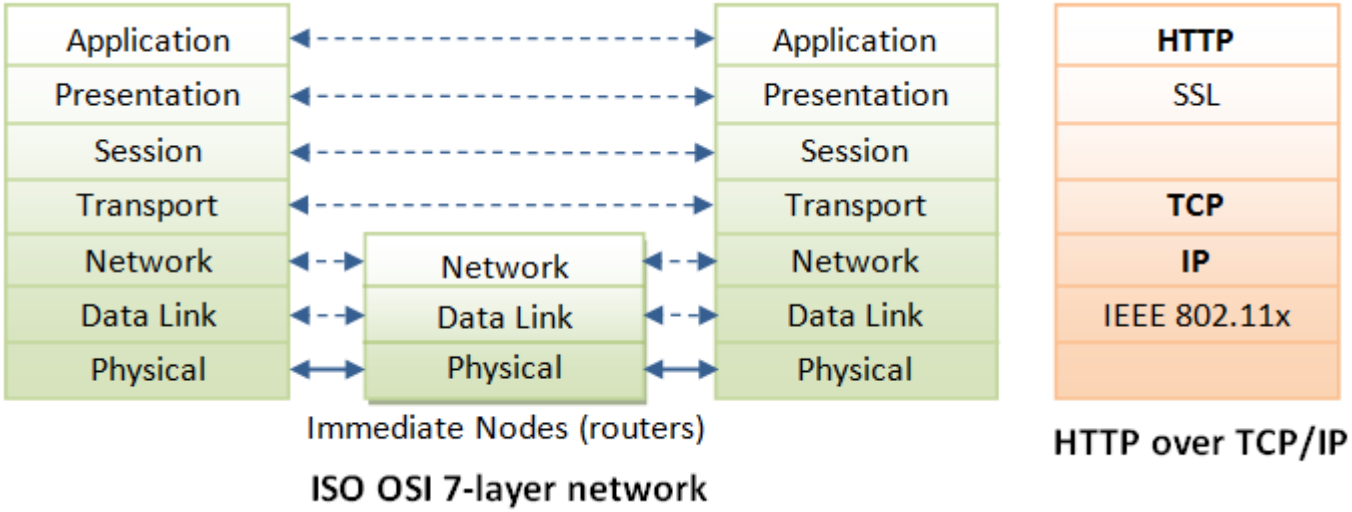
- Build application Layer on top of Presentation Layer (sockets)

Lets have a look at Python Programs: Echo Client and Echo Server

- ISO-OSI Level 6 Programs and below – can be accessed via sockets (ISO level 6 functionality)
- ***HTTP is an ISO-OSI Level 7 protocol –that is an Application-level protocol***
- HTTP is built on level 6 (and below) applications

Recall: HTTP is built on Presentation Layer Protocols, which is built on TCP/IP

- HTTP is a client-server application-level protocol.
- It typically runs over a presentation layer protocol with a TCP/IP connection underneath, as illustrated below.



- (***HTTP need not run on TCP/IP. It only presumes a reliable transport. Any transport protocols that provide such guarantees can be used.***)
- See: https://en.wikipedia.org/wiki/Transport_layer for a nice discussion of other possible protocols

Interface to Layer 6 (sockets) Echo Client & Server Code Review

EchoClient.py and Echo Server.py are Available on Canvas in the Lecture 7 Materials,

Please download them now and we will review them

A Tiny HTTP server in built on EchoSever (GET Requests only)

Get the zip file: **GetDemo.zip**
from the Lecture 13 Materials on Canvas

UnZip it

Let's step through it, then
and run it – test it out with your browser

Toward Building The Back-End of our Full – Stack Web Site

- We'll use **node.js** to build the WebServer for the back end of our website
- Why?
  - Node.js is in JavaScript, and there are lots of nice free frameworks available for it (we will use express).
  - Node.js is more widely used than Python or Python Frameworks.
  - And finally, one of the course objectives is to learn JavaScript (which is the among the most widely used programming languages)
    - https://www.simplilearn.com/best-programming-languages-start-learning-today-article

Node.js:

(info obtained from:
https://www.w3schools.com/nodejs/nodejs_intro.asp
https://www.w3schools.com/nodejs/nodejs_get_started.asp
https://www.w3schools.com/nodejs/nodejs_modules.asp

- Node.js is an open source **server** framework
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript to implement and augment **server** functionality

Node.js can:

- generate dynamic page content
- create, open, read, write, delete, and close files on the server
- can collect form data
- can add, delete, modify data in your database

A Node.js file contains:

- tasks that will be executed when triggered by certain events
  - A typical event is someone trying to access a port on the server
  - Node.js files must be initiated (run/exectred) on the server before having any effect
- Node.js files have extension ".js"

Node.js handles a file request as follows

1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

- Thus node.js is single threaded, non-blocking, and asynchronous
- Here is how Php handles a file request:
  1. Sends the task to the computer's file system.
  2. Waits while the file system opens and reads the file.
  3. Returns the content to the client.
  4. Ready to handle the next request.
- So, for this task, PHP (and ASP) operate synchronously (and block).

Your first node.js web server (courtesy of w3shools) –try it out now (code along)

```
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.end('Hello World!');
}).listen(8080);
```

Log into the CSE lab machines (I'll use vole)

Put the code in the file: **myfirst.js**

You run it from the command line by typing:

node myfirst.js

Then fire up a browser, and in the address bar type:
http:/localhost:8080

And, you will get the response: Hello World – rendered in your browser

Lecture 13, Exercise 1

- Upload your working copy of your first node.js program

- Use the Lecture 13, Exercise 1 Submission Link

Like Python, Node.js has lots of libraries (called modules) that you will want to include in your application

- For example:
  - The http module is used to create an http server

```
var http = require('http');
http.createServer(…
```

More on this in the coming weeks after break

Next Class

- Post Break – March 11th
- HW4 Discussed
- More on Node.js
- Stay well and warm and have a great break!