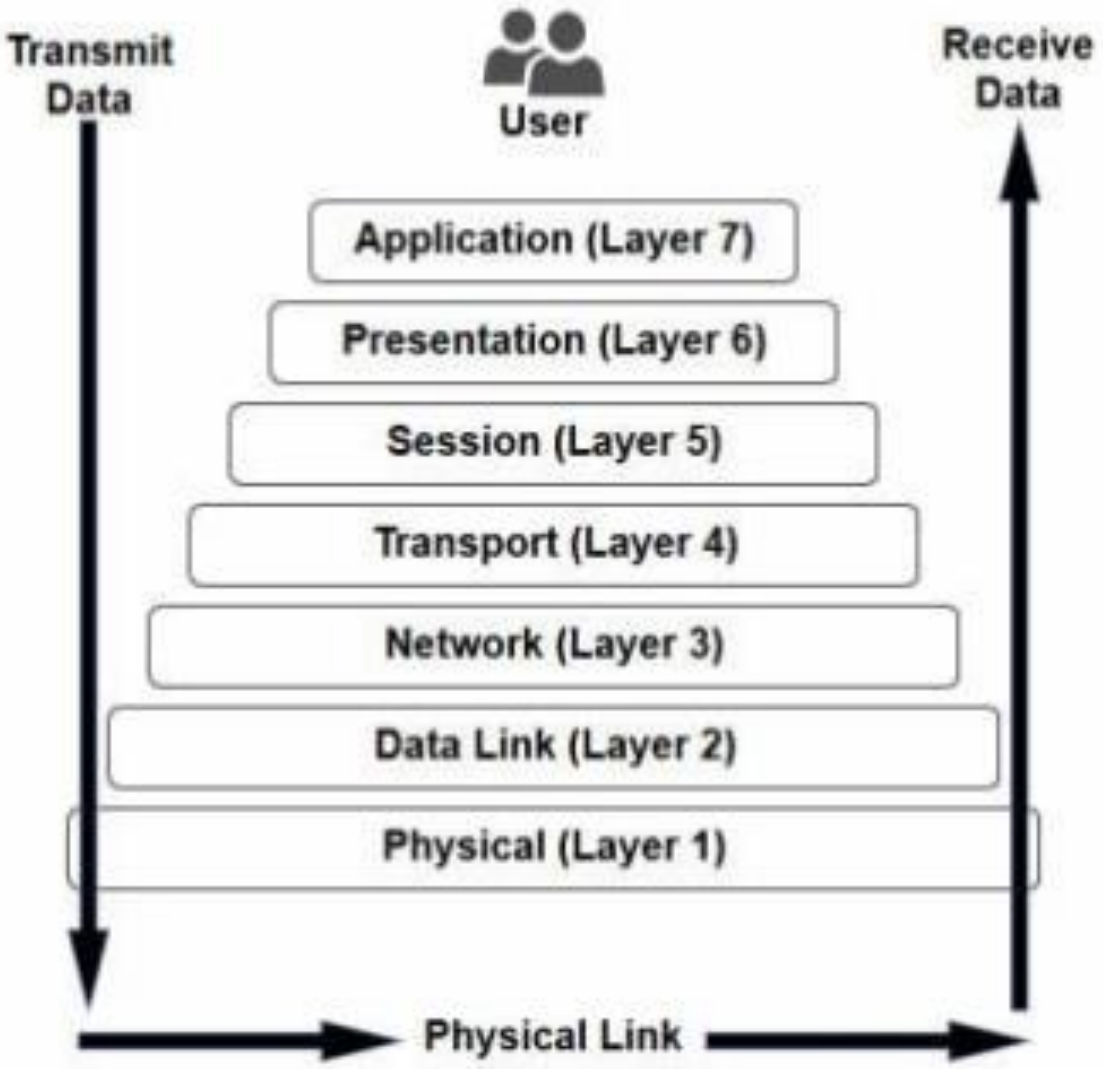
Recitation 6

HTTP Networking & Intro to Flask

The 7 Layers of OSI



https://www.webopedia.com/quick_ref/OSI_Layers.asp

HTTP

- Application Layer Protocol
- Request-Response Model
- Stateless
  - Cookies, sessions etc used to maintain state

https://www.youtube.com/watch?v=SzSXHv8RKdM

Uniform Resource Identifiers (URIs)

- String to identify a resource like a website, web service etc
- Case-insensitive
- **URI = "http:" "//" host [":" port] [abs_path ["?" query]]**

https://www.tutorialspoint.com/http/http_parameters.htm

Example

- http://abc.com:80/~smith/home.html
- http://ABC.com/%7Esmith/home.html
- http://ABC.com:/%7esmith/home.html

- All the three URIs are same. How?

https://www.tutorialspoint.com/http/http_parameters.htm

HTTP Requests

- A Request-line
- Zero or more header (General|Request|Entity) fields followed by CRLF
- An empty line
- Optionally a message-body

https://www.tutorialspoint.com/http/http_requests.htm

HTTP Requests – Request Line

- **Request-Line = Method SP Request-URI SP HTTP-Version CRLF**

Where SP means a single space character

https://www.tutorialspoint.com/http/http_requests.htm

Idempotence

- “Idempotence is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial application”

https://en.wikipedia.org/wiki/Idempotence

HTTP Request Methods - GET

- Retrieves information from a given server using a given URI
- Read-only method
- Safely Repeatable (Idempotent)
- GET requests can be cached

https://www.youtube.com/watch?v=rhTkRK53XdQ

Exercise

- Write a HTTP GET request for the URI **www.w3.org/pub/WWW/TheProject.html**

GET /pub/WWW/TheProject.html HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.w3.org

https://www.tutorialspoint.com/http/http_requests.htm

HTTP Request Method - POST

- Used to send data to a server to create/update a resource
- Write Method
- Cannot be repeated safely (non-idempotent)
- Example: The result of a web form, new entry to a database etc

https://www.youtube.com/watch?v=rhTkRK53XdQ

HTTP Request Method - PUT

- Used to send data to a server to create/update a resource in a client-specified URI
- Write Method
- Safely Repeatable (Idempotent)

https://www.youtube.com/watch?v=rhTkRK53XdQ

Exercise

- Which of the following gives an error?

This is a path that already exists in the server

- POST /questions/**<existing_question>** HTTP/1.1
- Host: www.example.com/

This is a new path which does not exist

- POST /questions/**<new_question>** HTTP/1.1
- Host: www.example.com/

https://stackoverflow.com/questions/630453/put-vs-post-in-rest

Exercise

- Which of the following gives an error?

- POST /questions/<existing_question> HTTP/1.1
- Host: www.example.com/

- POST /questions/<new_question> HTTP/1.1
- Host: www.example.com/

Gives “Resource Not Found” error because <new_question> does not exist

https://stackoverflow.com/questions/630453/put-vs-post-in-rest

When to use PUT and when to use POST?

- Use PUT when you can update a resource completely through a specific resource path

- Example: if you know that an article resides at http://example.org/article/1234, you can PUT a new resource representation of this article

```
PUT /article/1234 HTTP/1.1
<article>
  <title>red stapler</title>
  <price currency="eur">12.50</price>
</article>
```

http://restcookbook.com/HTTP%20Methods/put-vs-post/

When to use PUT and when to use POST?

- If you do not know the actual resource location,
  - for example, when you add a new article, but do not have any idea where to store it, you can POST it to an URL, and let the server decide the actual URL.

```
POST /articles HTTP/1.1
<article>
   <title>blue stapler</title>
   <price currency="eur">7.50</price>
</article>

HTTP/1.1 201 Created
Location: /articles/63636
```

http://restcookbook.com/HTTP%20Methods/put-vs-post/

HTTP Request Method - DELETE

- Request the server to delete a file at a location specified by the given URL

- Write method

- Safely Repeatable (Idempotent)

https://www.youtube.com/watch?v=rhTkRK53XdQ

Example of DELETE

- Delete the file “index.html” located in the root directory of the server “www.abc.com”

DELETE /index.html HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.abc.com

Accept-Language: en-us

Connection: Keep-Alive

https://www.tutorialspoint.com/http/http_methods.htm

Flask

- Installations
  - pip install Flask==1.0.2
  - pip install Flask-SQLAlchemy==2.3.2

Simple Flask Code

```
from flask import Flask

    app = Flask(__name__)

    @app.route('/')
    def home():
      return "<h1> Hello World </h1>"

    if __name__ =="__main__":
      app.run(debug=True,port=8080)
```

Initializes a new Flask object to variable app

"__name__" is a special python variable of a file that is given the string value "__main__" when that file is executed. Otherwise, it receives its own name.

https://codeburst.io/flask-for-dummies-a-beginners-guide-to-flask-part-uno-53aec6afc5b1

Simple Flask Code

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
  return "<h1> Hello World </h1>"

if __name__ =="__main__":
  app.run(debug=True, port=8080)
```

- Decorator
- The string parameter takes in a route/a URL
- When client requests for the specific URL, the server call the corresponding function (home() in this case)

- Trigger function that runs the web server
- Can specify the port number and/or other parameter like debug

https://codeburst.io/flask-for-dummies-a-beginners-guide-to-flask-part-uno-53aec6afc5b1

Run test.py

- Test the result with different URLs
- Notice the GET requests and corresponding responses that you get when you visit the different URIs

Command Prompt - python test.py

```
C:\>python test.py
 * Serving Flask app "test" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [27/Oct/2018 01:24:42] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [27/Oct/2018 01:24:47] "GET /index HTTP/1.1" 200 -
```

Notice the GET requests being generated when you visit http://127.0.0.1:8080/ and http://127.0.0.1:8080/index respectively. Also, notice the responses you get. Since there is no function bound to http://127.0.0.1:8080/ , so you get the response status code 404 (NOT FOUND). On the contrary, you get response status code 200 (OK) when you visit http://127.0.0.1:8080/index