Assignment #4

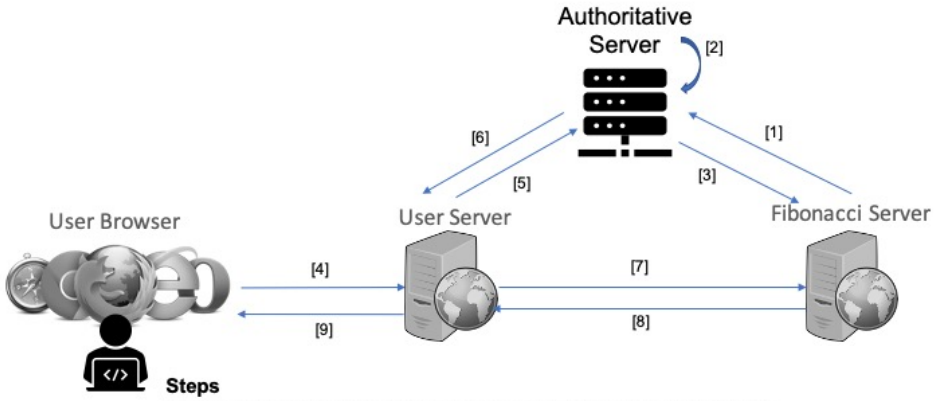
Due Date: March 27, 2020 10pm EST

Objectives: Understand the top layer of Internet: Application layer. Develop a simple network application, explore and understand DNS protocol.

Deliverables:

- Submit all your code to NYU Classes (as a .zip file).
- Also make sure to commit your programming question to Github. We will use the last commit time for grading.
    - In Github create a folder for the programming exercise and name it "**dns_app**". Make sure to upload your code in there.
- Late submission policy (no exception)
    - 1 day – 50% off
    - 2 day – 75% off
    - 3 day – 100% off

1) **Programming Assignment**: In this part, you will implement a simplified Authoritative Server for your network of applications. Figure below shows an overview of the system that you need to develop.



Steps
[1] Fibonacci Server registers its hostname with Authoritative Server
[2] Authoritative Server creates a DNS record for the Fibonacci server.
[3] Authoritative Server a response back indicating the success or failure.
[4] User visit:
http://IP_HTTP_SERVER:PORT/fibonacci?hostname=fibonacci.com&number=10&as_ip=3.4.5.6
[5] User server parse the hostname from the query and query the DNS Authoritative server via DNS query
[6] Authoritative Server returns back the IP address of the Fibonacci HTTP Server
[7] User server request http://FIBONACCI_SERVER_IP/fibonacci?number=8
[8] Fibonacci Server returns back the answer with 200 code.
[9] User server returns the result to the user

The system that you will implement has 3 components:

a. **User server (US)** is a simple HTTP web server (e.g Flask), running in port **8080**, that accepts a GET HTTP requests in path "/fibonacci?hostname=fibonacci.com&fs_port=K&number=X&as_ip=Y&as_port=Z". The path accepts five parameters: **hostname and fs_port** which contains the hostname and port number of the server which will be queried to get a response for Fibonacci number for a given **sequence number X**. And **as_ip, as_port** which are the IP address and port number of the Authoritative Server (AS). If any of the parameters are missing, server should return HTTP code **400** indicating the bad request. If the request is successful it should return HTTP code **200** with the Fibonacci number for the sequence number X. The only problem is that US does not know the IP address of the given hostname and therefore need to query its Authoritative DNS server to learn about it.

b. **Fibonacci Server (FS)** is an HTTP web server, running in port **9090**, that provides the Fibonacci value for a given sequence number X. In order to achieve this objective, FS performs the following:

   1. **Hostname Specification**: FS accepts a HTTP PUT request at path "/**register**" where the body contains the name and IP address of the server (FS) and IP address of the Authoritative Server (AS). You can choose any name of your choice, let's say "fibonacci.com". The body should contain **a json object** as below. FS should parse the body and noted that hostname and IP and IP, port of the AS and moved to step 2 for registration to authoritative server.

{
“hostname”: “fibonacci.com”,
“ip”: “172.18.0.2”,
“as_ip”: “10.9.10.2”,
“as_port”: “30001”
}

2. **Registration to Authoritative Server** Once the request is retrieved at path “/register”, the hostname needs be registered with Authoritative server (AS) via UDP on port **53533**. Your DNS message should include (Name, Value, Type, TTL) where the type in this case is “A” and TTL in **seconds**. Below is a simplified DNS registration request (please follow the format as below -- without the dashes--, each line ends with a new line):

   TYPE=A
   NAME=fibonacci.com
   VALUE=IP_ADDRESS
   TTL=10
   ----
3. Once registration is successful, returns back a HTTP response with code **201**.
4. Serve a path in “**/fibonacci?number=X**” which accepts HTTP GET request and returns Fibonacci number for the sequence number X. Return **200** as a status code if successful. If X is not an integer (for example a string) return **400** indicating the bad format.

c. **Authoritative Server (AS)** is the authoritative server for US. It has two duties. First is to handle the registration requests to pair hostnames to IP, second is to be able to respond to DNS queries from clients.
   i. **Registration**: Accept a UDP connection on port 53533 and register Type A DNS record into the database. Hint: You need to store the value in somewhere persistent, for example in a file so that you can later respond to DNS queries.
   ii. **DNS Query**: Respond to DNS Query on port 53533. Query should include: (Name, Type). If the message conforms to this, server will return the IP address in a DNS message of form: (Name, Value, Type, TTL), retrieved from the file (note that AS can distinguish registration requests from dns queries by simply looking at the fields provided). Here is a sample request and response:
   Request:
   TYPE=A
   NAME=fibonacci.com
   Response:
   TYPE=A
   NAME=fibonacci.com
   VALUE=IP_ADDRESS
   TTL=10

The programming assignment will be graded based on the following. Please follow the instructions carefully for including the port numbers and message formats. Any deviation from the specification above will result losing points in the assignment.

- US responds to requests in path /fibonacci as specified above.
- FS accepts a registration request in /register as specified above.
- FS responds to GET requests in /fibonacci.
- AS performs a registration requests as specified above.
- AS provides DNS record for a given query as specified above.

Please submit the following:

- Create **dns_app** folder in Github, and create three separate folders within it: "**US**", "**FS**" and "**AS**".
- For each folder provide **Dockerfile** and all the files that Dockerfile needs in order to run the specific server.
- Finally make the zip of "**dns_app**" folder and upload to NYU Classes.
- Your final commit date to the folder in Github will be used to grade the submission.

Extra Credit: Run your application in IBM Cloud K8s cluster. Provide a single file (containing your deployment and service) that can be used to deploy your whole application with: "kubectl apply -f deploy_dns.yml". Be sure to expose your ports with nodePort. Name the single file as "**deploy_dns.yml"**. Also note that K8s does not allow nodePort except for the range of 30000-32767. Therefore, make sure that in your .yaml file, specify AS nodePort number as **30001**, FS port number as **30002** and US port number as **30003**.

You can take a look at this link to learn more about how to create this deployment file: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#creating-a-deployment

Tips:

To build an image from Dockerfile, use the following command (change the tag accordingly and notice the . at the end which specifies the context of the build):

```
docker build -t bulutmf/fs:latest .
```

To get servers running within Docker containers communicate with each other, you can create a Docker network with the following command:

```
docker network create N_NAME
```

Once created run your containers by specifying the network name with the following command (change the parameters accordingly):

```
docker run --network N_NAME --name C_NAME -p 53533:53533/udp -it bulutmf/as:latest
```

Containers that are running within the same network, should be able to communicate with each other. You can learn the IP address of your container by inspecting the network that you created with the following command:

```
docker inspect N_NAME
```