Homework 2

Introduction to Networks and Their Applications
Due: March 4th, 2018
Version 2.0

Note: To get *full* credit for an answer you must show all your work. Guessing the right answer will earn you no points.

Question 1

Create your own website using the University of Iowa's web servers. Please mention your website address. A simple HTML file uploaded to your homepage is acceptable.

Question 2

With respect to HTTP:

(I) Explain how Web caching can reduce the delay in receiving a requested object.

(II) Consider an HTTP client that wants to retrieve a Web document at a given URL. The IP address of the HTTP server is initially unknown. What transport and application-layer protocols besides HTTP are needed in this scenario?

(III) Assume a Web browser wants to access a Web page, which in turn includes 8 small objects. Also assume that a TCP connection requires a request and a response (one RTT). With respect to Round Trip Time (RTT), how much it takes to fully load the page with:

(a) Non-persistent HTTP with no parallel TCP connection?

(b) Non-persistent HTTP with the browser configured for 5 parallel connections?

(c) Persistent HTTP?

(IV) Assume that there are u users connected to the University of Iowa's wireless network to connect to the Internet. $u = 800$ during the day, and $r = 300$ during the night. These users want to browse some Web sites, and the average number of HTTP requests generated by all the users is $0.5 \times u$ requests per second. Assume that the size of HTTP request messages is negligible, but each HTTP response message returns an object of size 500 Kbits. If the university is connected to the Internet through a 700 Mbps access link,

(a) What is the percentage of access link's capacity consumed by the HTTP response messages during the day and night?

(b) What is the percentage of access link's capacity consumed by the HTTP response messages during the day if we use a Web cache which can directly respond to HTTP requests with probability h?

■

Question 3

With respect to Domain Name System (DNS):

(I) Why DNS is required? (Describe with respect to the Internet protocol stack and Web servers)

(II) Why DNS is operated in a distributed and hierarchical manner?

(III) Run the following command in linux terminal: `host -v uiowa.edu`.
Include the result and discuss about that.

(IV) How can you modify your operating system configurations so that when a DNS query is made, that is sent to a specific DNS sever (e.g., Google's)?

(V) Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that n DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT of $RTT_1 + RTT_2 + ... + RTT_n$. Further suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Let RTT_0 denote the RTT between the local host and the server containing the object. Assuming zero transmission time of the object, how much time elapses from when the client clicks on the link until the client receives the object?

Question 4

As part of this problem, you will implement a simple messaging the system that is implemented in Python. The system has two basic components: a *chat client* that will be used to send messages and *topic server* that will be used to map a user identifier to an IP address. The chat clients will talk with the *chat server* using TCP with the topic server. The code for the chat client will be include in `chat.py` and may run either in "direct mode" or "topic mode". In the direct mode, the client will directly talk with another client. The command line arguments are the following:

- Argument 1: The first argument is "–direct" to indicate that the client will run in "direct mode"
- Argument 2: The second number is number indicating the port number the chat client will be waiting for incoming connections. If the number is zero, the program will act as a client and not open the listening port.
- Argument 3: The second number is a string that include the IP address and port number for the client that you will be talking to. For example, the string *192.168.1.5:2000* indicates that the IP address is 192.168.1.5 and the port number is 2000.

In indirect mode, the client will talk with multiple other clients that are interested in the same topic. Note that a topic is just a user-provided identifier. In this mode, the command line arguments are the following.

- Argument 1: The first argument is "–topic" to indicate that the client will run in "topic mode"
- Argument 2: The second number is a string that include the IP address and port number for the topic server. The string is formatted as above.
- Argument 3: The topic that the client is interested in.

The operation of the chat client has two phases: registration/lookup and chat.

- **Registration:** The client will register with the directory server that the user is using the IP address of the host on which the chat client runs
- **Chat:** The client will send a chat message to the topic server and receive message from the topic server when other clients send their data.

In both modes, the client will read input from the keyboard. Each provided line will be sent in a single chat message whose format is detailed below. When a chat client receives data from a peer chat client or the topic server, it should display the received message. The display should have the following format using the following statement: `print("%s:%s n" % (topic, message)`. You should use `select` to multiplex the reading from the keyboard and the reception of bytes over TCP. The clients will exchange messages that conform to the following format:

In both direct and topic modes, the messages exchanged by the clients will be valid JSON using the following format:

```
{
        "source" : {
                "ip": "192.168.1.1",
                "port" : "2000"
        },
        "destination" : {
                "ip": "192.168.1.5",
                "port" : "2032"
        },
        "message" : {
          "topic" : "lame-conversation",
          "text" : "This is such a lame message"
        }
}
```

The service will include as `topicserver.py` and will maintain the mapping between topic identifiers and a list of sockets from all the clients currently connected in a hash table. The registration message will include information about the topics that the chat client is interested and have the following structure.

```
{
        "source" : {
                "ip": "192.168.1.1",
                "port" : "2000"
        },
        "topics" : ["lame-conversation", "other-conversation"]
}
```

The topic will handle a message from a client as follows. It will first check if the client is registered for the topic included in the message. If this is the case, the message will be broadcast to all the clients that are currently connected to the server. Otherwise, the message will be discarded.
You can obtain partial credit if you get part of the problem correctly implemented:

- Correctly implement the direct mode where the chat clients interact with the topic service (50% of the points).
- Solving the remainder of the problem will earn you the remainder (50% of the points).