Assignment 1

Marks: 100

Deadline: 15/10/2022 23:59

Penalty: 10% marks will be deducted for each day after the due date.

1. Follow the instructions in this document to get the mininet environment setup and get acquainted with the commands.
https://docs.google.com/document/d/1kautMdV5rYyFuJD7FoxY8JMk06xXKuC3iMwduEIVIS s/edit

Once you have followed the above document, you will be in a position to understand mininet and how to xterm into hosts and capture packets you send on Wireshark.

Next, download the following VM. This contains the necessary files for you to get started on the assignment. That is, you will use this VM for all further assignment coding.

- Link to download the VM: https://drive.google.com/file/d/1maiWDRqWK213oLYxGXQyY3AJXdc4ldGi/view?usp =sharing

2. Download the folder [link given below] and paste it inside the exercises folder of your VM.

- Initial setup requirements for Assignment_1: (Extract this tar file and and copy/replace the folders **basic** and **star** " inside "/home/p4/tutorials/exercises" directory in the VM)
https://drive.google.com/file/d/1xdbOCXHWMSHvmYL9ZKrVbUZ6J4IZKn1r/view?us p=sharing

3. Run sample code:

1. Open terminal, go to "/home/p4/tutorials/exercises/basic"
2. Run "make clean"
3. Run "make run"
4. You are now on the mininet prompt.
5. Run below commands to open the Host terminals:
    a. "xterm h1"
    b. "xterm h2"
6. Commands to run on h2's terminal
    a. bash h2-arp.sh (run once every time you run "make" above)
    b. python server.py (here, enter the IP address as 10.0.1.2)
7. Command to run on h1's terminal
    a. bash h1-arp.sh (run once every time you run "make" above)
    b. python client.py (here, enter the IP address as 10.0.1.2)
8. The code does the following: H1 opens a connection, sends a hello message to H2 on the connection, and receives a hello message from H2, and closes the connection.

4. Write a program to emulate an **HTTP server** in host H2 that listens and responds to three requests from H1: GET, PUT and DELETE **(Marks: 40)**

A. Folder: "/home/p4/tutorials/exercises/basic"

B. Topology:
    a. H1 – Client
    b. H2 – Server
C. Write code in client.py that generates GET, PUT and DELETE requests. Run this code on H1.
D. Write code in server.py that listens, parses, and sends responses to three requests: GET, PUT, and DELETE. Run this code on H2.
E. Create the following key-value store in H2 by sending HTTP PUT requests from host H1. The key value store can be stored in a file or memory.
Example request: PUT /assignment1/key1/val1 HTTP/1.1

Key1	Val1
Key2	Val2
Key3	Val3
Key4	Val4
Key5	Val5
Key6	Val6

F. Test 1: Run GET request for Key1, and server implementation should return Val1
Example request: GET /assignment1?request=key1 HTTP/1.1

G. Test 2: Send a sample DELETE request to delete any key-value pair
Example request: DELETE /assignment1/key1 HTTP/1.1

H. Report to submit:
    a. Capture pcap traces at H1 for all three types of requests (1 request/response for each)
    b. From H1, GET all 6 keys 3 times each and note down the end-to-end time taken to finish the GET request. That is, capture time before issuing a request and after receiving the response and report the difference in the table below.

Key	**Req1 (first time)**	**Req2 (second time)**	**Req3 (third time)**	**Average Time**
Key1				
Key2				
Key3				
Key4				
Key5				
Key6				

5. Webcache development **(Marks: 50)**

A. Folder: “/home/p4/tutorials/exercises/star”

B. Topology:
   a. H1 – Client
   b. H2 – Cache
   c. H3 – Server

C. Write code for:
   a. Client.py: Client(H1) issues GET key requests to Cache (H2).
   b. Cache.py: Cache(H2) listens to GET key requests from Client (H1). If the key is present in the cache, then it responds to the GET key request. Otherwise, it reaches Server(H3), gets the value, caches the requested key-value pair, and returns the value to Client (H1).
   c. Server.py: Server (H3) stores six key-value pairs in the table (same as above) and listens to GET requests from Cache (H2).

D. Command to run:
   a. Open terminal, go to “/home/p4/tutorials/exercises/star”
   b. Run “make” (this sets the
   c. You are now on the mininet prompt.
   d. xterm h1
   e. xterm h2
   f. xterm h3
   g. Commands to run on h1’s terminal
      i. bash h1-arp.sh (run once every time you run “make” above)
      ii. python client.py
   h. Commands to run on h2’s terminal
      i. bash h2-arp.sh (run once every time you run “make” above)
      ii. python cache.py
   i. Command to run on h3’s terminal
      i. bash h3-arp.sh (run once every time you run “make” above)
      ii. python server.py

E. Report to submit:
   a. Capture pcap traces for two GET requests for the same key (e.g., key1) at H1, H2, and H3:
      i. When the key is not present in Cache (H2)
      ii. When the key is present in Cache (H2)
   b. Send a total of 3 GET requests for each key. Note down the end-to-end time taken to finish GET requests at H1. That is, capture time before issuing a request and after the response and report the difference in the table below.

Key	Req1 (first time)	Req2 (second time)	Req3 (third time)

Key1			
Key2			
Key3			
Key4			
Key5			
Key6			
Average Time			

6. **Report to submit (Marks 10):** Compare the average time taken for Req1 (all keys), Req2 (all keys) and Req3 (all keys)

(1) differences observed.

(2) justification for why there is a difference.

Note: Ignore socket connection establishment times when you calculate the time above.

7. **What to submit:**

- Readme file
- Zip basic and star folders and name the zip file as <ROLLNO>_Assignment1.
- The folders should contain your code (client/cache/server.py), pcap files (h1/h2/h3.pcap), your reports, and all other inbuilt files. Failure to do so will result in difficulty on our side for evaluation and will result in penalties.

PLAGIARISM STATEMENT <Include it in your report>

We certify that this assignment/report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, packages, datasets, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment/project in any other course lab, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, We understand my responsibility to report honor violations by other students if we become aware of it.

Name:
Date:
Signature: <keep your initials here>

FAQs:

Clarifications / FAQs to some common queries concerning star topology:

1. What are all the types of HTTP requests that we should implement in the star topology?
A: Yes. Since you have already developed the code for all three requests inbasic topology, you only have to port it over to the star topology. That being said, it is sufficient in this assignment for you to implement PUT and GET requests only. This means that you enter the keys into the server using PUT requests and then subsequently get them using GET requests as per the requirements of the assignment.

2. Should PUT requests also go through the cache (H2)?
A: Yes. The client only establishes a connection with the cache. The cache depending on the conditions and type of request, will further establish connections with the server (all using HTTP protocol over TCP). The responses will also take the same route to reach the client (through the cache).

3. What about responses?
A: All requests and responses should follow the same route back to the client and follow HTTP protocol specifications.

4. Can we have some example HTTP Requests?
A:
GET /assignment1?key=key1 HTTP/1.1
PUT /assignment1 HTTP/1.1
{ "Key": "key1", "Value":
"val1" }
(or)
PUT /assignment1/key1/val1 HTTP/1.1
Since you are handling the server and its code, you can choose various representations as long as it is a valid HTTP request.
5. Can we have some example HTTP Responses?
A: HTTP/1.1 200 OK
Use appropriate response codes. In your implementation, you should be able to justify during evaluation why a particular code is being returned. If any response data needs to be sent, it can be sent in the next line.

6. How do I know if my HTTP request and responses are valid?
A: If Wireshark (already installed in the assignment VM) recognizes your requests as HTTP, you are good to go.

7. What are all the cache and server scenarios that must be handled?
A: The cache system has to be consistent. That is, for example, during the first GET request, the response needs to be cached. Since you only have to implement two types of requests, it is expected that your code will justify them.

8. Where should we store the key value store?

A: It can either be in memory or a file. Since the evaluation will require you to show the whole process, starting from PUT -> GET, persistence would not matter in this case.

9. At the client, how should we show the response?
A: It is your choice. It is enough to show the raw HTTP response (Just print the string). Additionally, you can parse it and show the reply as well.

10. How should we measure the time between the request and response at the client?
A: You can use either python libraries in the code or use the timestamps from Wireshark captures. It is recommended that you take Wireshark timestamps which are generally more accurate.