# Assignment 2: CS3530

## 1. Emulating an HTTP Server (Basic Topology)

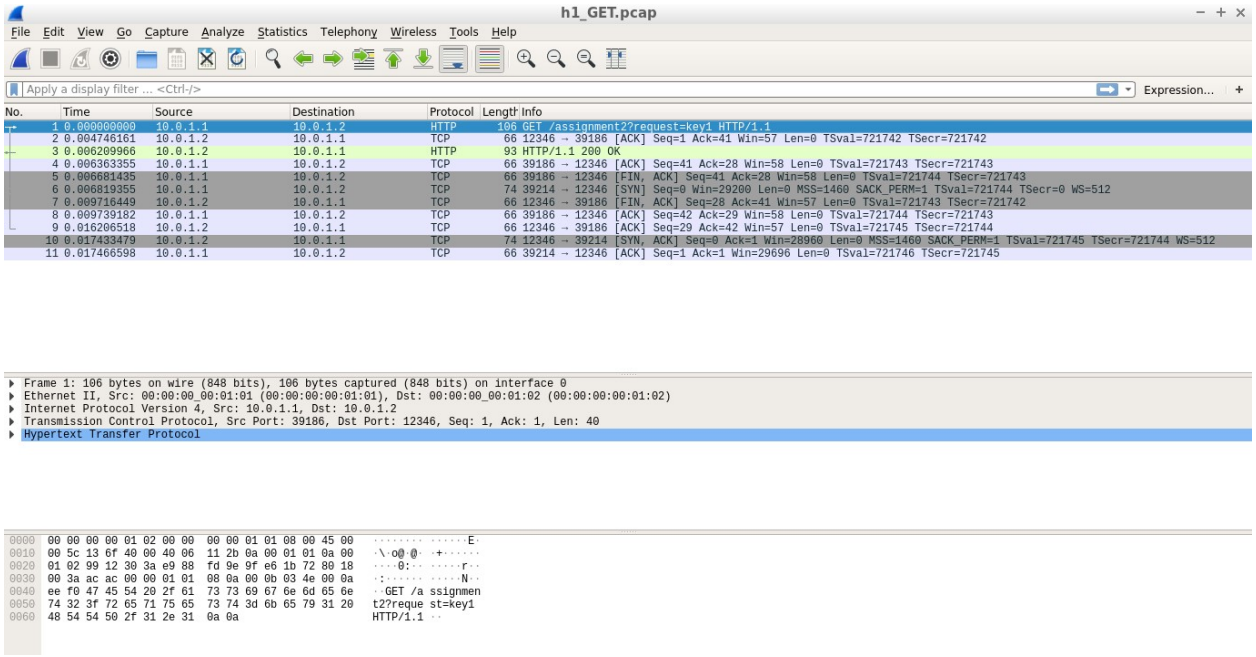
The topology contains 2 hosts - h1 and h2, both of which act as a client and a server respectively connected to the same switch s1.

### a. PCAP traces at H1 for all three types of requests

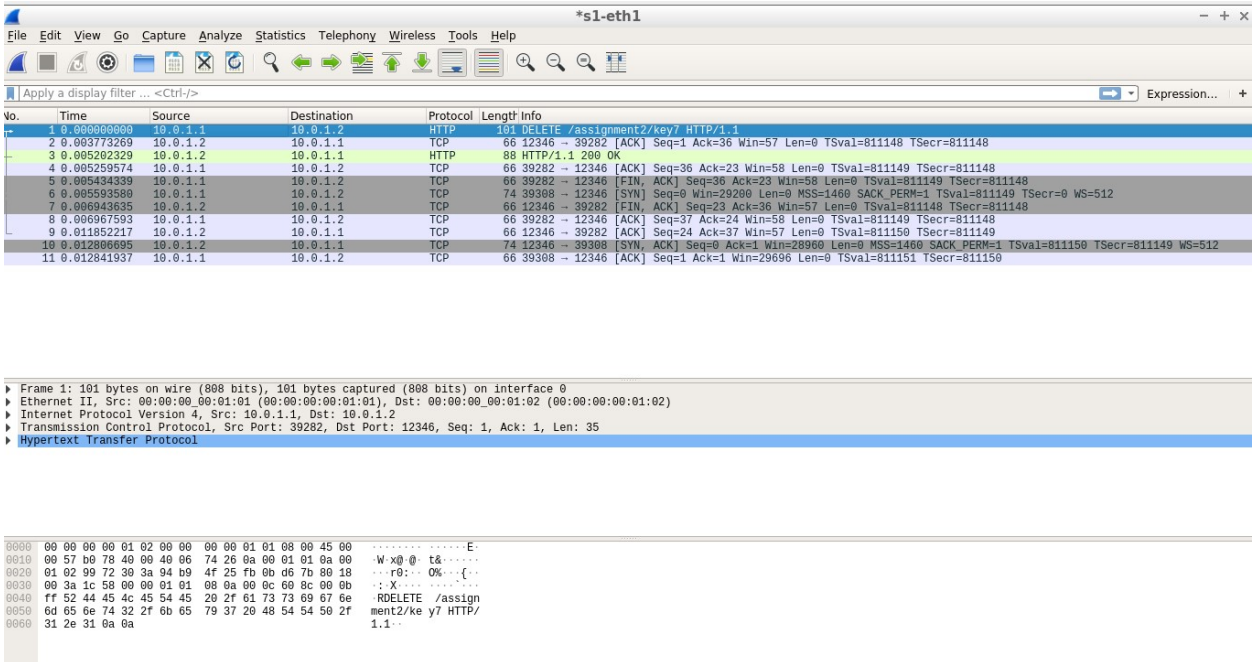
Following are three snapshots of the pcap traces for all the types of requests:
(1 request/response for each of GET, DELETE, PUT)

Note that the individual pcap files are available in the folder basic/pcaps.

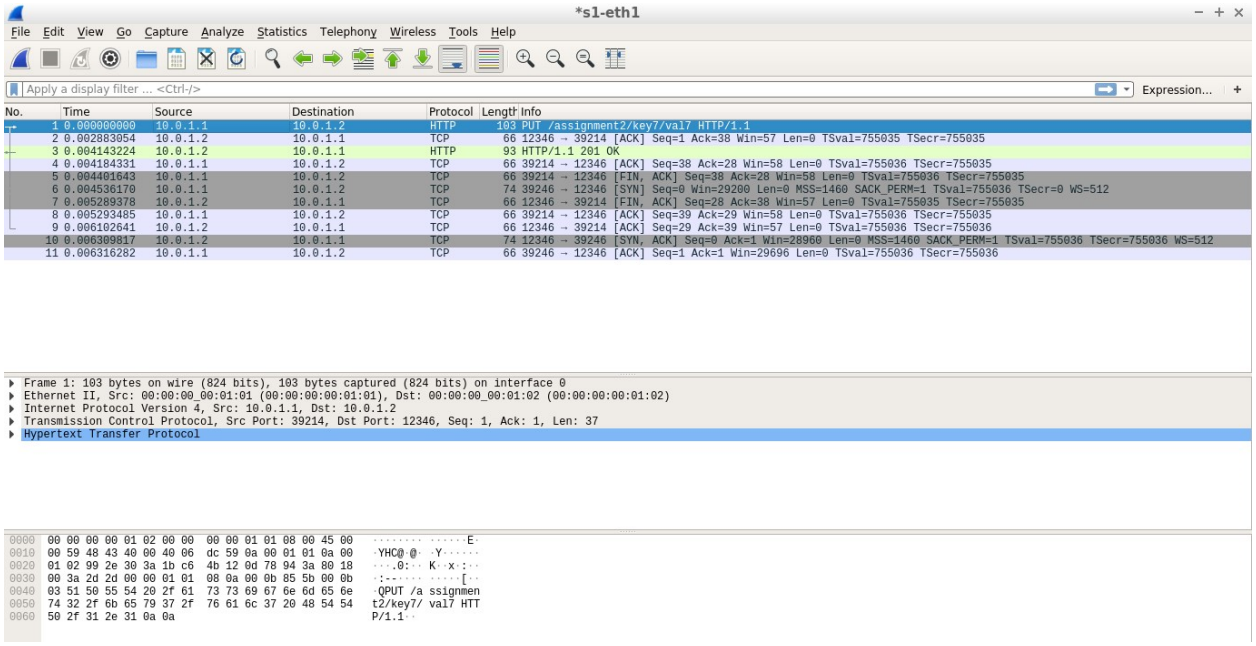
*GET request trace at H1*



*DELETE request trace at H1*




*PUT request trace at H1*




**b. End to End times**

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 | Request 1 (first time) | Request 2 (second time) | Request 3 (third time) | Average Time |
|---|---|---|---|---|
| key1 | 0.005174862 | 0.005140435 | 0.004806779 | 0.005040692 |
| key2 | 0.001287872 | 0.003804495 | 0.004518082 | 0.003203483 |
| key3 | 0.005198547 | 0.004181163 | 0.0045346 | 0.0046381033 |
| key4 | 0.004011415 | 0.003938796 | 0.004262324 | 0.0040708450 |
| key5 | 0.004838614 | 0.004963489 | 0.004757225 | 0.0048531093 |
| key6 | 0.004875395 | 0.004828673 | 0.003964222 | 0.0045560966 |

## 2. Web Cache Development (Star Topology)

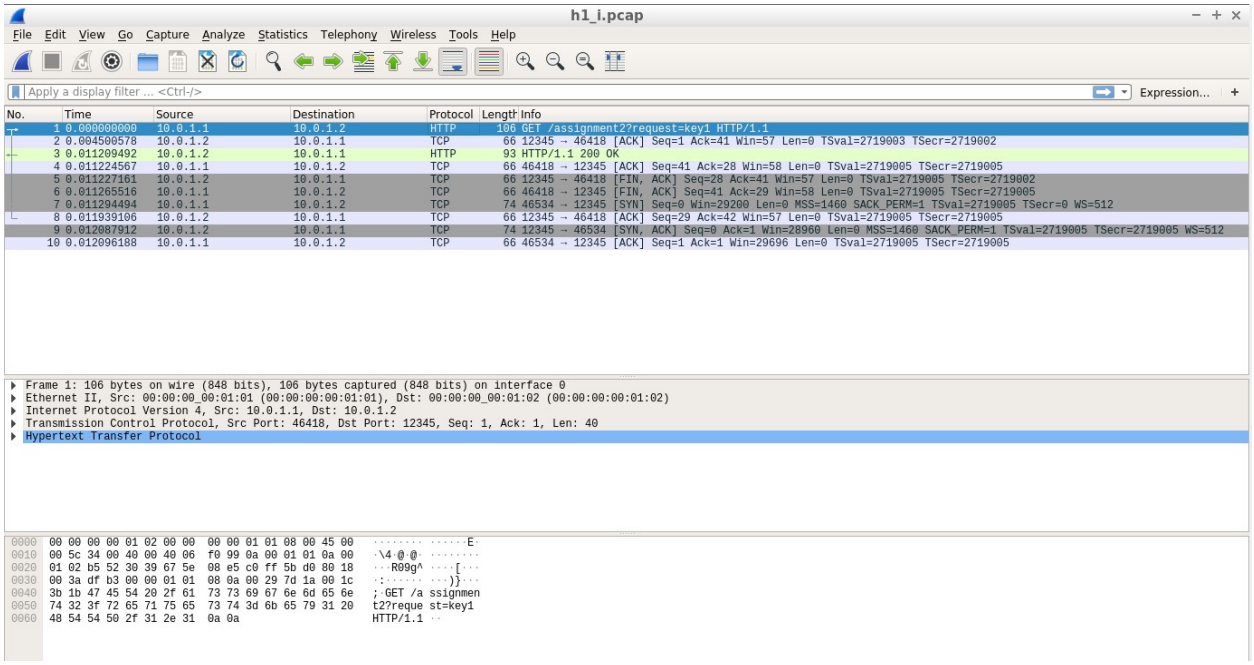
The topology contains 3 hosts - h1, h2 and h3, acting as a client, cache and a server respectively and are connected to the same switch in a star fashion.

### a. PCAP traces for GET request of same key (key1)

#### i. When the key is not present in the cache (H2)

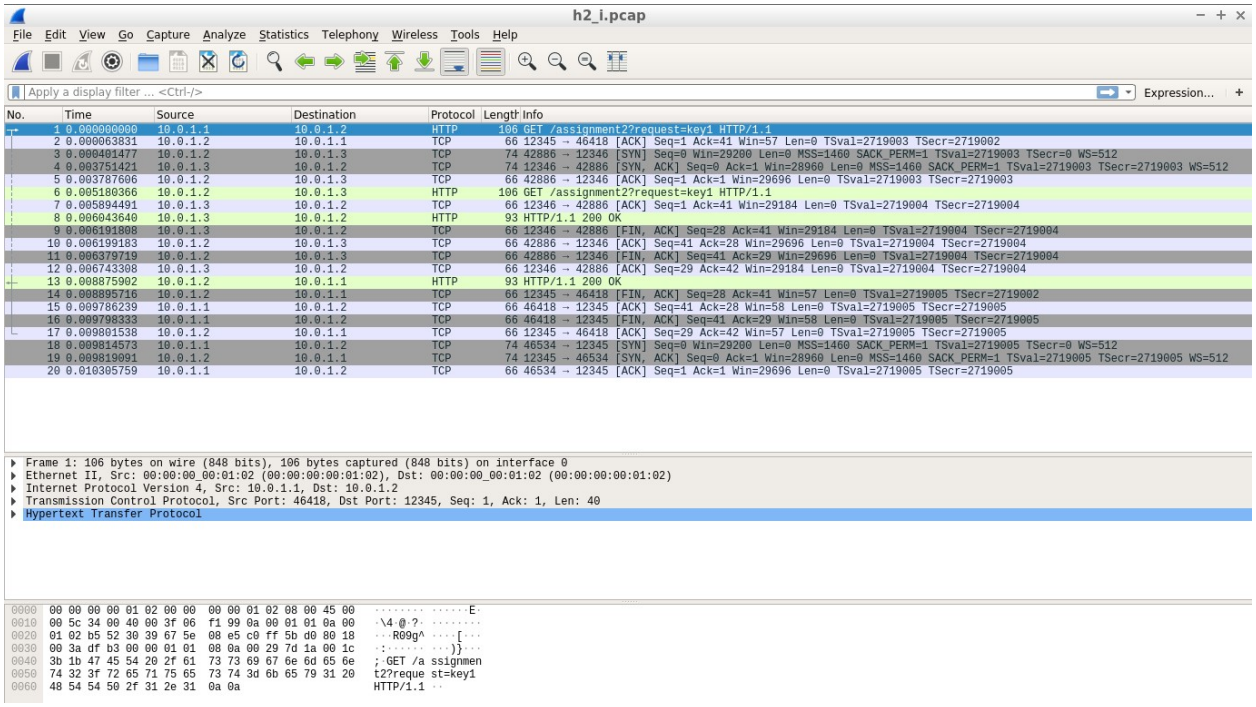
When the client requests the cache for key1's value initially, the key-value pair is not present in the cache. The cache then forwards the request to the server, and the server replies with the value, which the cache will then send to the client. The value of that pair will also be added to the cache's database (this can be tracked in the output of the terminal while running the requests).

*PCAP traces snapshot at h1 (client) during the first GET request and response*



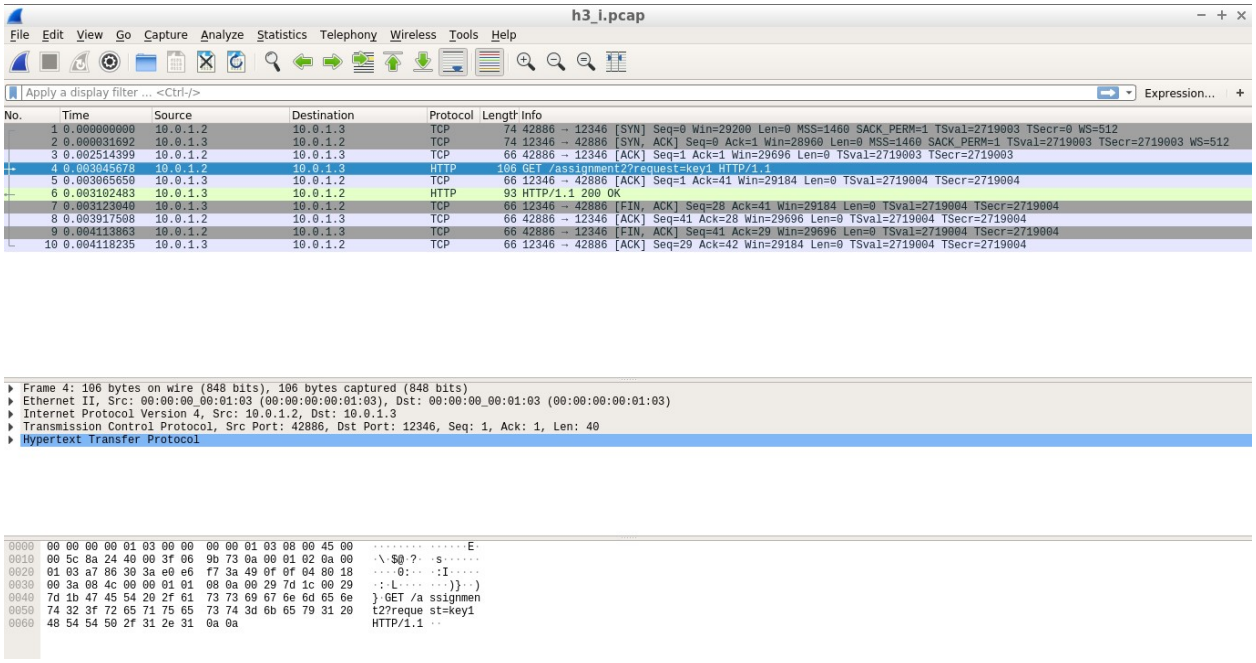
Here we can see the first request/response pair, for when the key is not present in the cache. This is the GET request from the client side to the cache. The overall end-2-end time taken for this request is around 0.012 seconds.

*PCAP traces snapshot at h2 (cache) during the first GET request and response*




In the above trace, there are 2 request/response pairs. This shows the request from the client (No 1 - request from the client), then the cache requesting that key from the server (No 6 - request from cache to the server), getting the response from the server with the key-value pair (No 8 - response from server to cache), and finally the cache replying to the client with the value (No 13 - response from cache to client). The trace for this GET request is long because the cache connects to the server, gets the value, copies the value to its own database and then sends the requested value to the client. Essentially 2 requests/responses are happening.

*PCAP traces snapshot at h3 (server) during the first GET request and response*

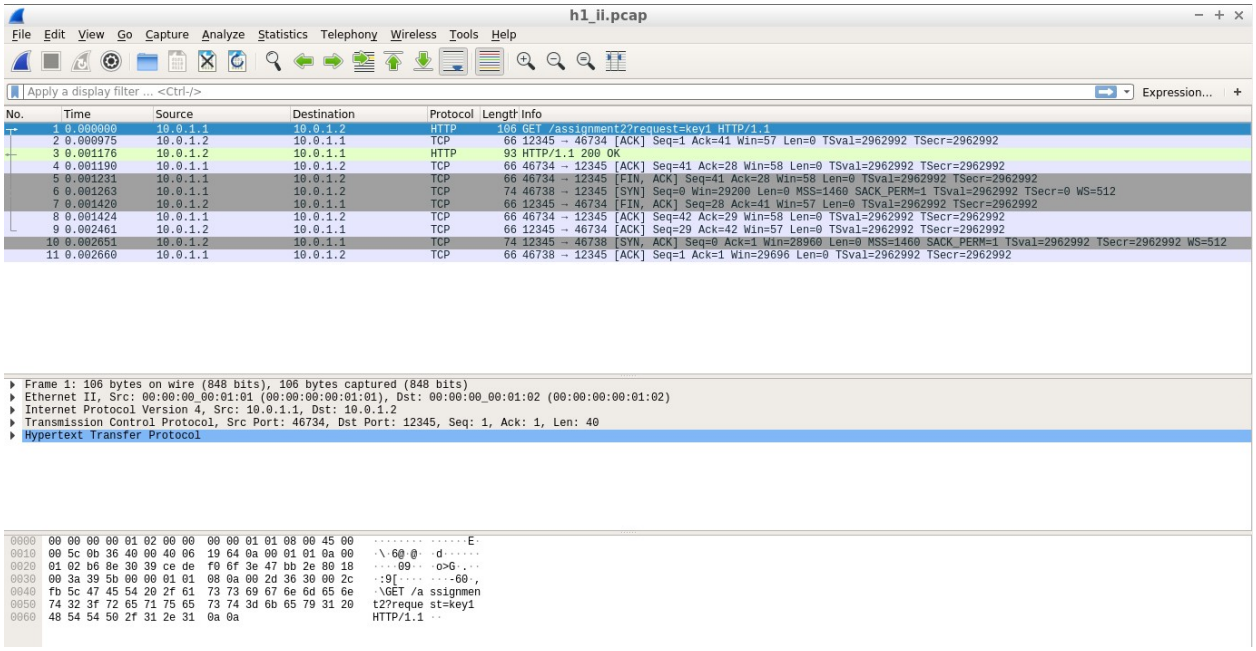



This trace is only that of the cache requesting the server for the same key value pair that was requested by the client to the cache. Interaction with the server is only present in this first GET request. This is also that of a simple client-server connection, where ordinary server-like behavior is observed.

**ii. When the key is present in the cache (H2)**

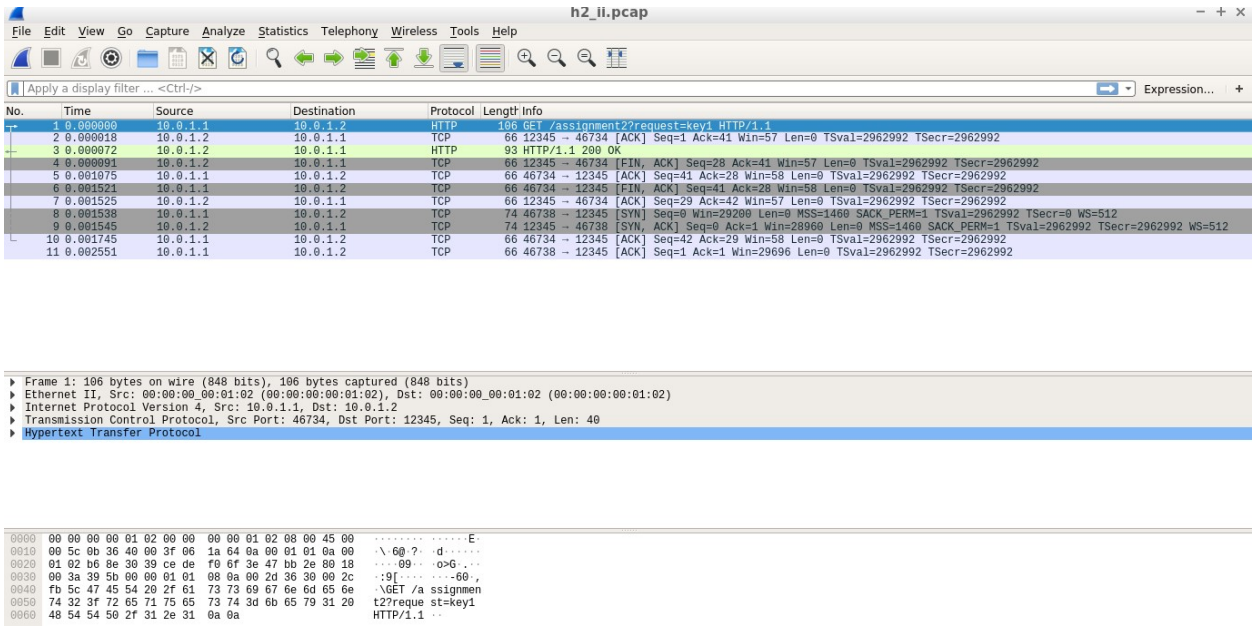
Now, after the first request we know for sure that the key value pair is now stored in the caches database. So, when the client makes a request for this key at this point, we can assume it will be an ordinary GET request behavior at the client and cache side - the cache will simply reply with the requested value.

*PCAP traces snapshot at h1 (client) during the second GET request and response*



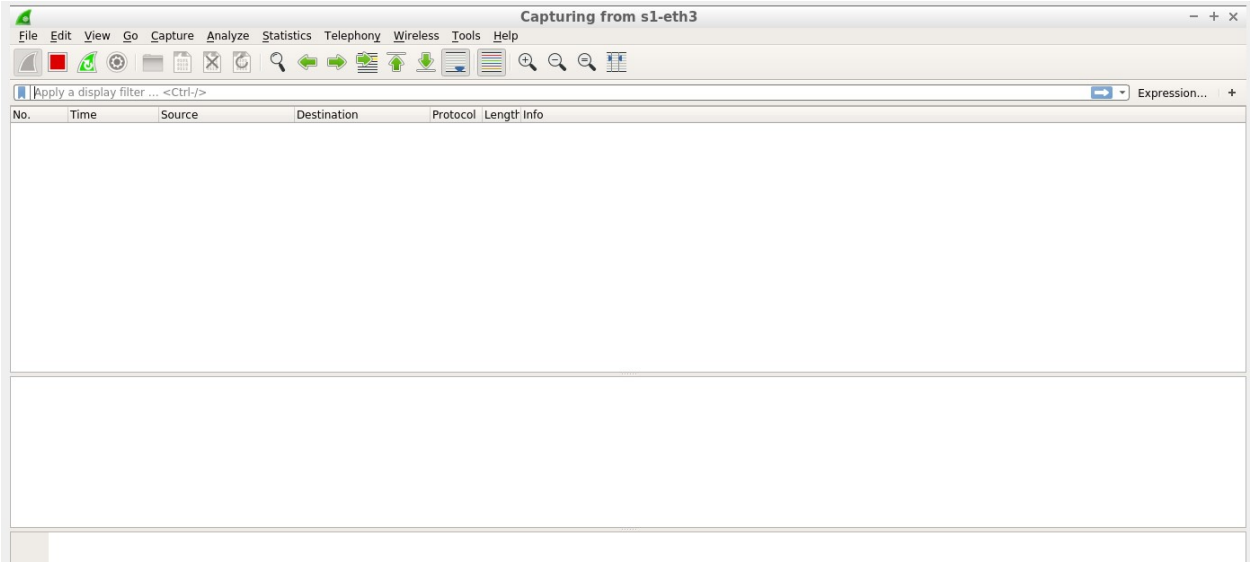

Here we can see the second GET request for the same key. By this point, the key value pair is stored in the cache database. Now, the end-2-end time taken is around 0.001 seconds. So there is a significant reduction in the time taken to give the response to the client compared to the first request.

*PCAP traces snapshot at h2 (cache) during the second GET request and response*




Here as well, we can see the second GET request at the cache side. Over here the cache acts like a simple server, the interaction being a simple client-server type connection (like in question 1 - basic topology).

*PCAP traces snapshot at h3 (server) during the second GET request and response*




As expected, there is no activity or interaction with the server in this case. Since the value is already present with the cache, the cache will be able to satisfy the client's request without involvement of the server.

Note that the individual pcap files are available in the folder star/pcaps.

**b. End to End times**

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 | Request 1 (first time) | Request 2 (second time) | Request 3 (third time) |
|---|---|---|---|
| key1 | 0.017914311 | 0.004772632 | 0.004921236 |
| key2 | 0.021335847 | 0.005533870 | 0.004148218 |
| key3 | 0.017454997 | 0.005572584 | 0.001033328 |
| key4 | 0.013621024 | 0.004087662 | 0.004633021 |
| key5 | 0.016182809 | 0.004299085 | 0.007892530 |
| key6 | 0.017011895 | 0.004356772 | 0.003929093 |
| **Average Time** | 0.017253481 | 0.004770434 | 0.004426237 |

**Comparison of the average time taken for Req1 (all keys), Req2 (all keys) and Req3 (all keys):**

1. **Differences observed:**
   From the table above, we can clearly see that on an average, the 1st request takes a significantly higher time than the subsequent 2 requests, for all keys. Almost more than twice the time is taken.

2. **Justification for why there is a difference**:
   The first request takes more time, because the requested data (key-value pair) was initially not present in the cache, at the time of making the first request. So the cache had to request this pair from the server again, and also store a copy of this in its own database. The next 2 requests to the cache, requesting the same key value pair, will now get a much quicker response because the cache now stores a copy of that value with it. Hence, the time initially taken by the cache to request from the server again will have been cut down.

**PLAGIARISM STATEMENT**

*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: Padmini Palivela, Pushkal Mishra
Date: 26-09-2023
Signature: Padmini Palivela, Pushkal Mishra


## Group Members:

Padmini Palivela - EE20BTECH11038
Pushkal Mishra - EE20BTECH11042