Zebo Xiong | A04907051

CS5352.751/752, Distributed Computing, Summer I, 2020
Assignment 2

Zebo Xiong, A04907051

Issued: 07/16/2020
Due: 07/23/2020

1. (5 + 5 + 10 = 20 pts) This problem is pertaining to the NTP protocol.

(1) Explain the meaning of offset o_i between two NTP servers.

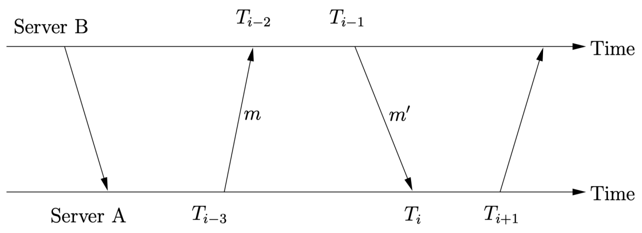
We cannot get the real "o" between two servers' clock (such as NTP server A and NTP server B, no matter which stratums they exist in).

However, an offset "o-i", which is an estimate of the actual offset between the two clocks, can be calculated with other information such as T(i-3), T(i-2), T(i-1) and T(i)

(2) Why the values of t and $t^{'}$ cannot be measured? Why we can obtain the value of t + $t^{'}$ accurately?

We cannot measured the t and t prime because we do not know the two servers' offset in advance. Thus, we cannot capture the real send and receiving time point (Otherwise the t and t prime can be easily gotten by the server time difference).

However, we can get the time points on each server (such as server A and sever B).



Thus, we get the four time points during the procedure-call or symmetric mode. Then we have below equation:

$$T_{i-2} = T_{i-3} + t + o \quad \text{and} \quad T_i = T_{i-1} + t' - o$$

After we combine the two equations we get: t+t' = T(i) + T(i-2) - T(i-3) - T(i-1), or, $t + t' = T_{i-2} - T_{i-3} + T_i - T_{i-1}$

Zebo Xiong | A04907051

(3) The propagation delay d_i plays a significant role in the accuracy of NTP protocol. Explain using your own words why.

Without calculating the propagation delay “d_i”, we cannot start estimating the offset between two server.

Once we confirm the “d_i”, we can get “o_i” which is the half of “d_i”. Afterward, we can find the range of the real “o”.

It shows that “o_i” is an estimate of o, and d_i” is a measure of the accuracy of this estimate on “o_i”.

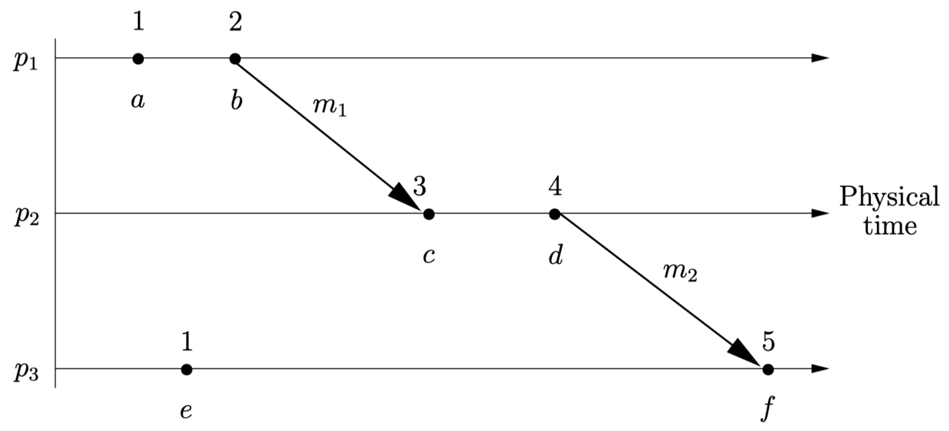
Additionally, NTP employed “data filtering algorithm” which use eight continuous pairs $< o_i , d_i >$. The value o_i of that corresponds to the minimum value d_i is chosen as “o_i” → The whole process is working with propagation delay “d_i”

Also, NTP use peer-selection to synchronize peer severs. Without the propagation delay information for each sever, this step will not be processed properly.

2. (12 + 13 = 25 pts) The problem is pertaining to the concepts of logical time and clocks.

(1) It was stated in class that for the vector timestamps, property P_3 (if $e \neq e'$ then $L(e) \neq L(e')$) may not always be true. Do you agree with this statement? Please explain your conclusion.

Perperty P3 may not always be true for “Lamport’s method without process ID”. As shown in below figure:



Zebo Xiong | A04907051

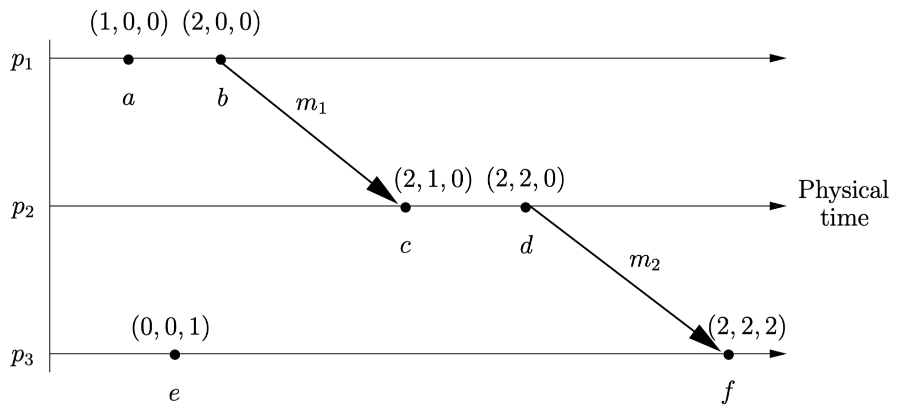
The event "a" and event "e" has the same Lamport timestamp. However the property is always true for vector timestamps.

For the "Vector clocks", each process content the vector with size N – which is the number of total processes.

Each process actually carries all the other processes' timestamp. It means that each process has the whole picture of the global state.

According to the VC mechanism, not only the sender's vector timestamp will be updated, but also the receiver's vector. Thus when the events are different, the Vectors will also be different.

Reference from lecture note:



(2) For the diagram shown in Fig. 1, assume that each process starts with a vector (0, 0, 0). Process p_1 always increases its component by 1, process p_2 always increases its component by 2, and process p_3 always increases its component by 10. Write down the complete vector timestamps for all the events in the diagram.

Zebo Xiong | A04907051

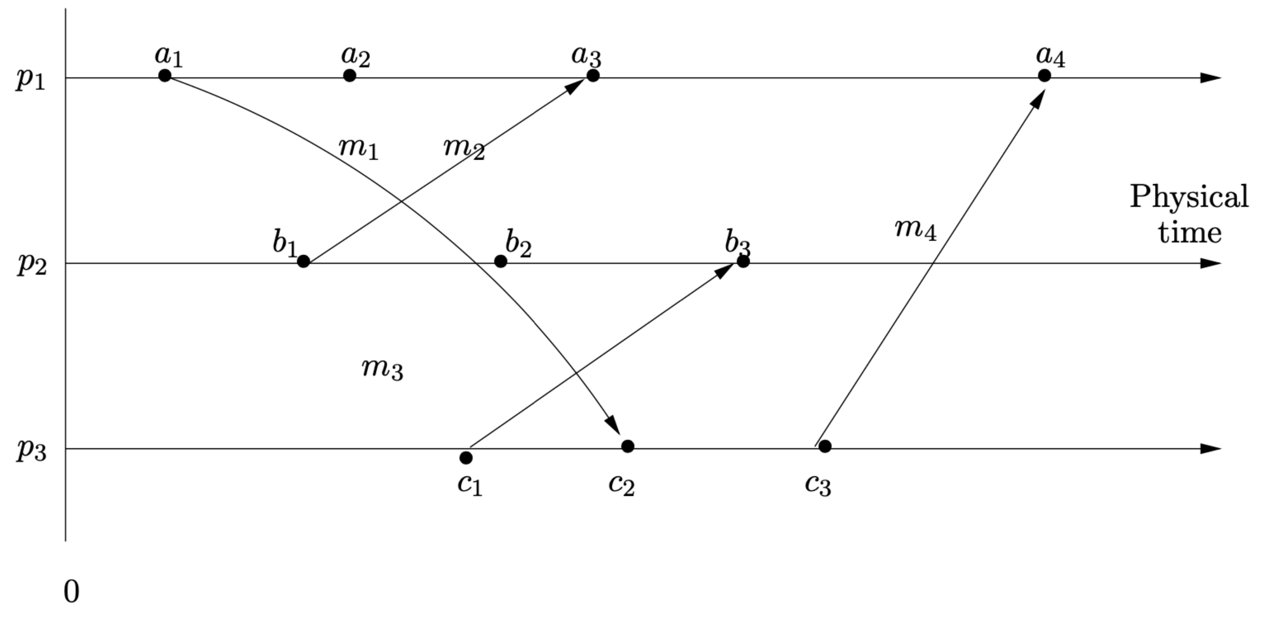


Figure 1: Three processes for Problem 2 and 3

a1 → (1, 0, 0)
b1 → (0, 2, 0)
c1 → (0, 0, 10)

a2 → (2, 0, 0)
b2 → (0, 4, 0)
c2 → (1, 0, 20)

a3 → (3, 2, 0)
b3 → (0, 6, 10)
c3 → (1, 0, 30)

a4 → (4, 2, 30)

Zebo Xiong | A04907051

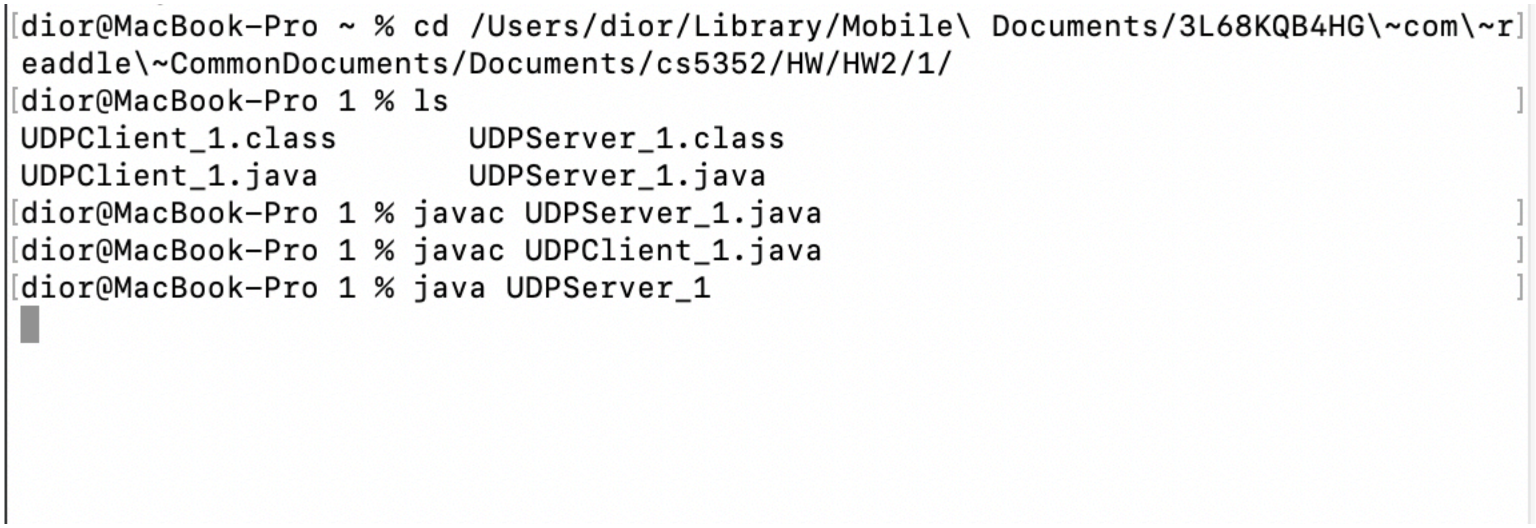
3. (25 + 30 = 55 pts) Modify the Java UDP example given in Chapter 4, p.152-153 as follows.

   (1) The client will iterate in a loop. During each iteration, it will get a character string from standard input and send it to the server. The server will also iterate in a loop. During each iteration it will receive a character string and send it back to the client.

   Compile: javac UDPServer_1.java
   Javac UDPClient_1.java

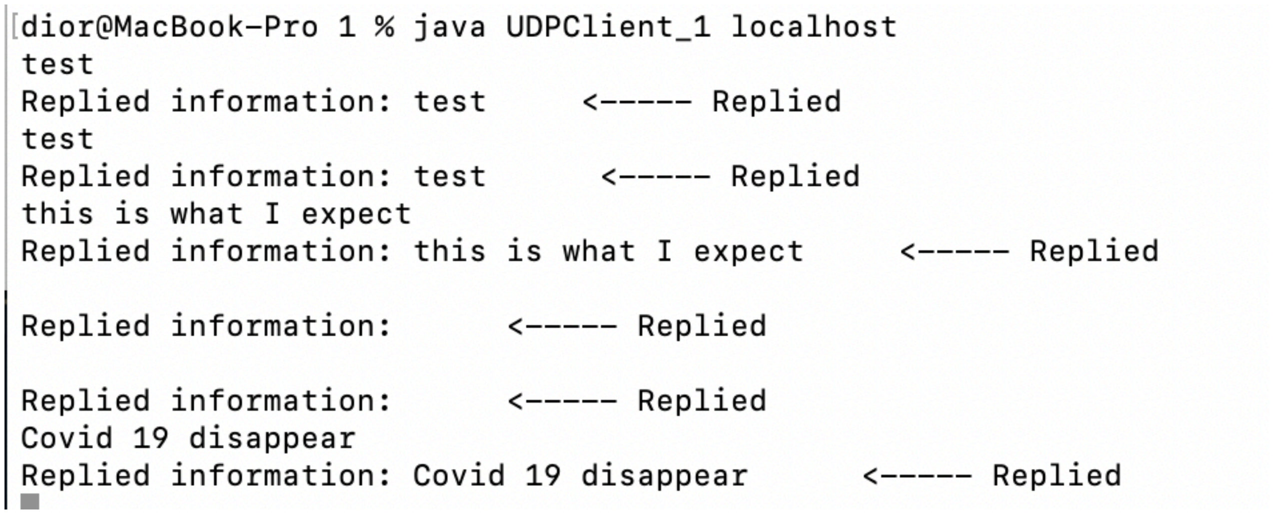
   Run server: java UDPServer_1
   Run client: java UDPClient_1

   The server screen shot:



```
dior@MacBook-Pro ~ % cd /Users/dior/Library/Mobile\ Documents/3L68KQB4HG\~com\~r
eaddle\~CommonDocuments/Documents/cs5352/HW/HW2/1/
dior@MacBook-Pro 1 % ls
UDPClient_1.class       UDPServer_1.class
UDPClient_1.java        UDPServer_1.java
dior@MacBook-Pro 1 % javac UDPServer_1.java
dior@MacBook-Pro 1 % javac UDPClient_1.java
dior@MacBook-Pro 1 % java UDPServer_1
```

The client screen shot:



```
dior@MacBook-Pro 1 % java UDPClient_1 localhost
test
Replied information: test     <----- Replied
test
Replied information: test      <----- Replied
this is what I expect
Replied information: this is what I expect     <----- Replied

Replied information:      <----- Replied

Replied information:      <----- Replied
Covid 19 disappear
Replied information: Covid 19 disappear      <----- Replied
```

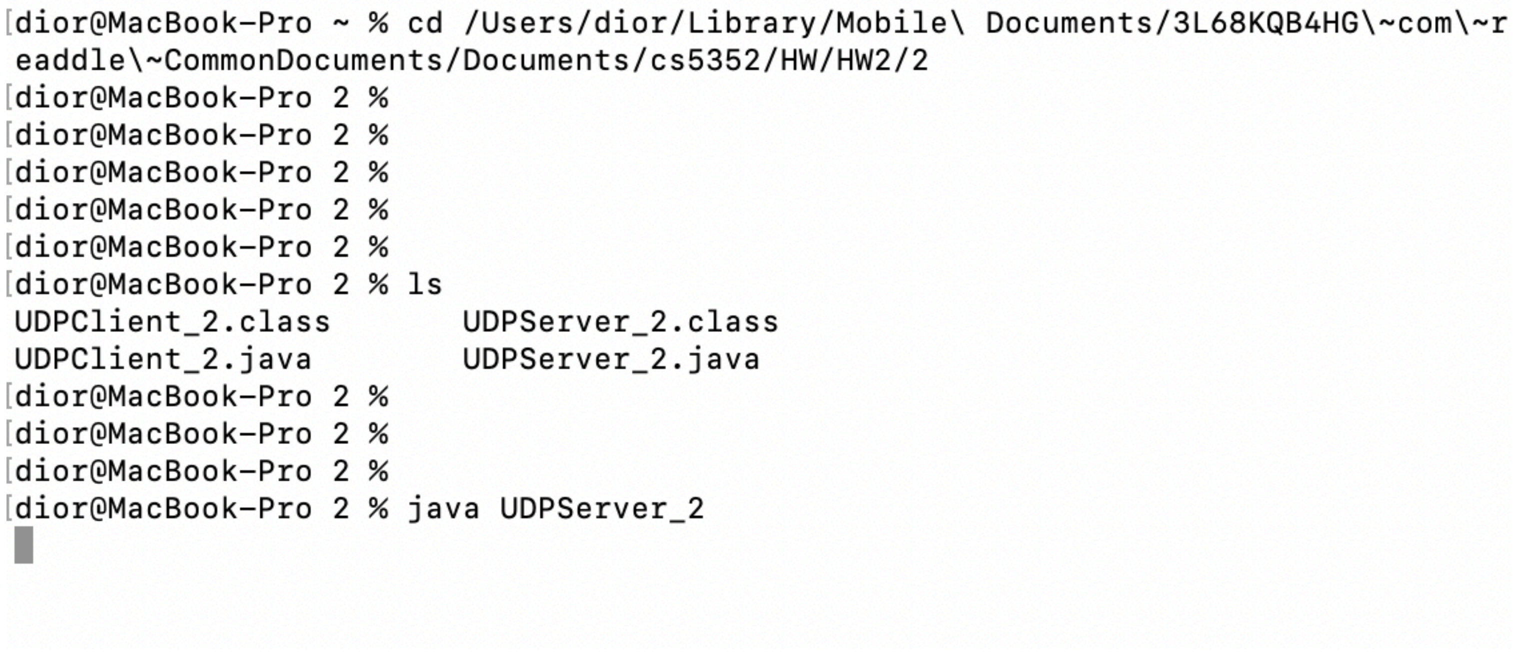
(2) In this second modification, the client will still send a character string to the server. However, that character string is the name of a cammand. After receiving that cammand name, the server will try to excute that command and send output of the command back to the client. If the server cannot excute the command, it will issue an error message back to the client. The client should just print each line of replies, plus the IP number of replying computer on its standard output.

Zebo Xiong | A04907051

Compile: javac UDPServer_2.java
javac UDPClient_2.java

Run server side: java UDPServer_2
Run client side: java UDPClient_2 ls localhost

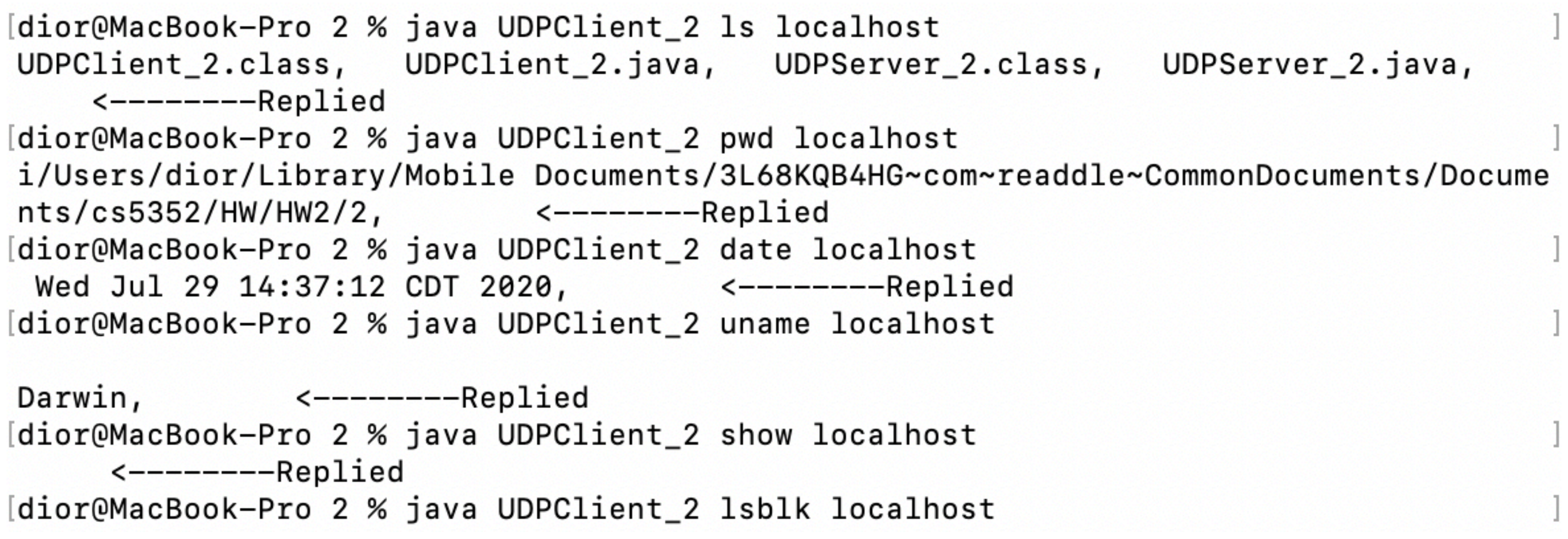
The server side screen shot:



```
[dior@MacBook-Pro ~ % cd /Users/dior/Library/Mobile\ Documents/3L68KQB4HG\~com\~r
eaddle\~CommonDocuments/Documents/cs5352/HW/HW2/2
[dior@MacBook-Pro 2 %
[dior@MacBook-Pro 2 %
[dior@MacBook-Pro 2 %
[dior@MacBook-Pro 2 %
[dior@MacBook-Pro 2 %
[dior@MacBook-Pro 2 % ls
UDPClient_2.class       UDPServer_2.class
UDPClient_2.java        UDPServer_2.java
[dior@MacBook-Pro 2 %
[dior@MacBook-Pro 2 %
[dior@MacBook-Pro 2 %
[dior@MacBook-Pro 2 % java UDPServer_2
```

The client side screen shot:

Zebo Xiong | A04907051



```
[dior@MacBook-Pro 2 % java UDPClient_2 ls localhost                                  ]
 UDPClient_2.class,   UDPClient_2.java,   UDPServer_2.class,   UDPServer_2.java,
     <---------Replied
[dior@MacBook-Pro 2 % java UDPClient_2 pwd localhost                                 ]
 i/Users/dior/Library/Mobile Documents/3L68KQB4HG~com~readdle~CommonDocuments/Docume
 nts/cs5352/HW/HW2/2,        <---------Replied
[dior@MacBook-Pro 2 % java UDPClient_2 date localhost                                ]
  Wed Jul 29 14:37:12 CDT 2020,        <---------Replied
[dior@MacBook-Pro 2 % java UDPClient_2 uname localhost                               ]

 Darwin,        <---------Replied
[dior@MacBook-Pro 2 % java UDPClient_2 show localhost                                ]
      <---------Replied
[dior@MacBook-Pro 2 % java UDPClient_2 lsblk localhost                               ]
```

Notes: You should implement and test the programs on the department Linux machines. The client/server should be able to run on the same computer, or on different computers. For this problem, if you are not already familiar with Java programming language, you have to read a Java textbook or read the JDK1.8.0 (or the latest version) manual about basic control (iteratios) and file I/O. The manual can be found from http://www.oracle.com/technetwork/java/index.html