Homework Assignment 3 Solutions
ECE 151 / CMPSC 171 - Spring 2013

MARKING SCHEME:
Problems 1,2,3,5,6 - 15 points
Problems 4 - 25 points

1. (a) Give an example of a name that is not address, and explain why it is a name but not an address.
(b) Give an example of a name that is not an identifier, and explain why it is a name but not an identifier.
(c) Give an example of an address that is not an identifier and explain why it is an address but not an identifier.

(a) An example that is a name but not an address is the URL of a webpage such as 'https://www.google.com'. This URL only refers to the name of the entity which has to be accessed but does contain any information on how this entity can be located in the network. Hence it is not an address.
(b) An example that is a name but not an identifier is the name of a computer such as 'ECI_sys1'. This refers to the name of the entity in the network. Since this name is not unique for that entity alone and in fact the same name could be assigned to other entities in the network as well, this name cannot be counted as an identifier.
(c) An example that is an address but not an identifier is dynamically assigned IP address. In this case, when an entity moves into a network an IP address is assigned for the machine using DHCP (Dynamic Host Configuration Protocol) to facilitate the communication of that entity with other entities in the network. This IP address assigned is not static and unique for that entity but assigned dynamically whenever the entity becomes part of the network. Hence it cannot be an identifier.

2. Consider the Chord system shown in Lecture 5, Slide 18, and assume that node 23 has just joined the system.
(a) Draw the finger table of the new node 23.
(b) Draw the finger tables of all the other nodes, after node 23 joined the system.

(a) Finger Table of Node 23:

i	***Succ. Node***
1	28
2	28
3	28
4	1
5	9

(b) Finger Tables of other nodes:

i	*Node 1*	*Node 4*	*Node 9*	*Node 11*	*Node 14*	*Node 18*	*Node 20*	*Node 21*	*Node 28*
1	4	9	11	14	18	20	21	23	1
2	4	9	11	14	18	20	23	23	1
3	9	9	14	18	18	23	28	28	1
4	9	14	18	20	23	28	28	1	4
5	18	20	28	28	1	4	4	9	14

3. (a) Explain what it means to *resolve* a name.
(b) Explain the difference between *iterative* and *recursive* name resolution.
(c) Give an advantage and a disadvantage of recursive name resolution compared to iterative name resolution.

(a) 'Name Resolution' refers to the process to obtain the IP address of an entity in order to communicate with it, whose domain name is already known.

(b) Iterative name resolution:

In this case the domain name is resolved iteratively in steps where in each step the IP address of the root server of the underlying sub-domain is known. The client sends the entire domain name to the root server initially and gets the IP address of the name server of the outermost domain in the first step. In the next step the client contacts the name server of the outermost domain and gets the IP address of the name server of the next outermost sub-domain. This process continues iteratively till the client gets the IP address of the name server of the innermost sub-domain. Finally the client gets the IP address of the server which holds the requested information. In this case there are multiple name resolution requests sent from the client side. An example for iterative name resolution is shown in the slide 23 in chapter 5.

Recursive name resolution:

In this case the domain name is resolved recursively with the name server in each domain requesting the name server of the subsequent inner domain for name resolution. The client sends the domain name to the root server which in turn contacts the name server of the outermost domain for name resolution. The name server of the outermost domain in turn sends the resolution request to the name server of the next outermost sub-domain and this process continues till the IP address of the name server of the innermost domain is known. The root server sends this requested address to the client. Thus the domain name is resolved with a single request from the client side. An example for recursive name resolution is shown in the slide 24 in chapter 5.

(c) Advantages of recursive name resolution in comparison to iterative name resolution are,

- Since there is only a single request from the client side for name resolution the communication costs are reduced. This factor is very significant particularly in the case when the client is located geographically very apart from the name servers.
- Caching the intermediate results in the name servers for subsequent look-ups can be of huge benefit in recursive name resolution.

Disadvantages of recursive name resolution are,

- This method puts higher work load on each name server as each name server has to handle the complete resolution of the part of the name in its domain. This could be a critical factor while scaling the system as each name server will have to handle a larger number of requests.

4. In Cristian's algorithm for clock synchronization, a processor makes multiple requests for the time from a remote clock on a time server, and chooses the reply having the smallest round-trip delay. It then uses the time in that reply to synchronize its clock with the remote clock on the time server. A smaller round-trip delay results in a more accurate clock synchronization.
Using discrete probabilities and assuming that the round-trip delay is uniformly distributed between 0 and 1 seconds, calculate the probabilities of various values of the smallest round-trip delay, such as 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0 seconds, for 1, 2, 4, 8 and 16 requests to the time server, and show the resulting graphs where the horizontal axis represents the round-trip delay and the vertical axis represents the probabilities of those round-trip delays.
To estimate the probabilities, create a simple program that makes many (say 100) trials. Thus, for example, for 4 requests, your program will consider 100 batches each with 4 requests. For each request, it will determine the round-trip delay using a random number generator. It will then choose the request with the smallest round-trip delay of the 4 requests, for each of the 100 batches. Thus, for the 100 data points it obtains, your program will then find the number of data points with a particular round-trip delay and then divide by 100 to find the probability of that round-trip delay.

Following should be observed:

1) As number of requests increases, the probability of having smaller value for round trip delay increases. This trend should be observed from graphs.

2) For any particular number of requests, the sum of probabilities of all round trip delays (for that request) should be 1.

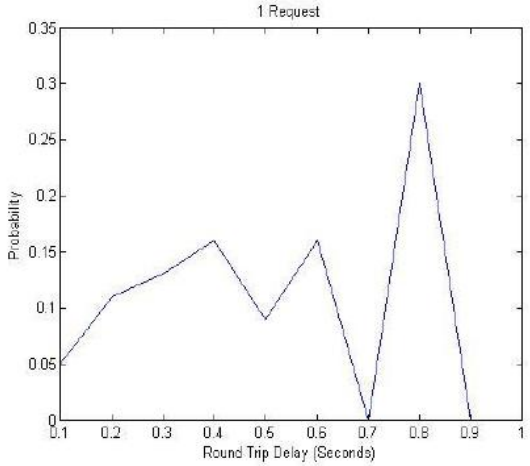


Figure 1. (1 Request to Time Server)

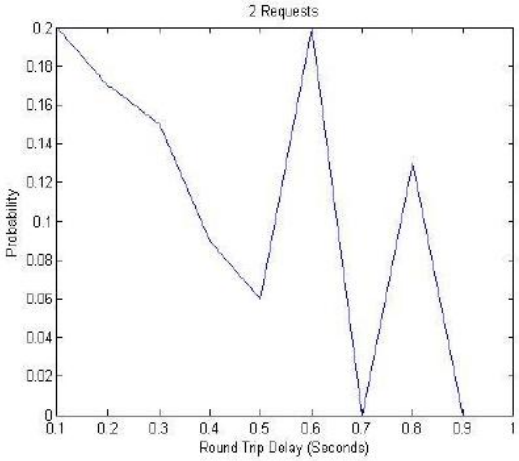


Figure 2. (2 Requests to Time Server)

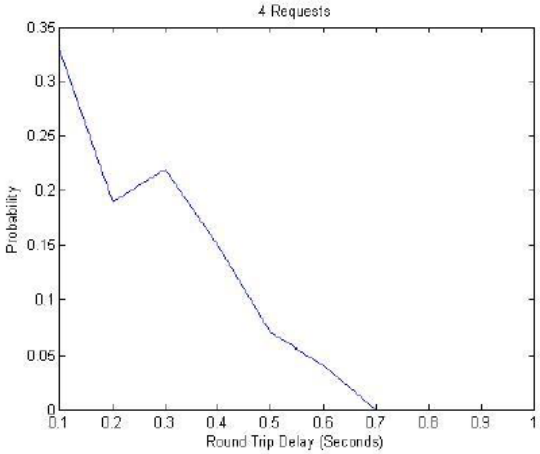


Figure 3. (4 Requests to Time Server)

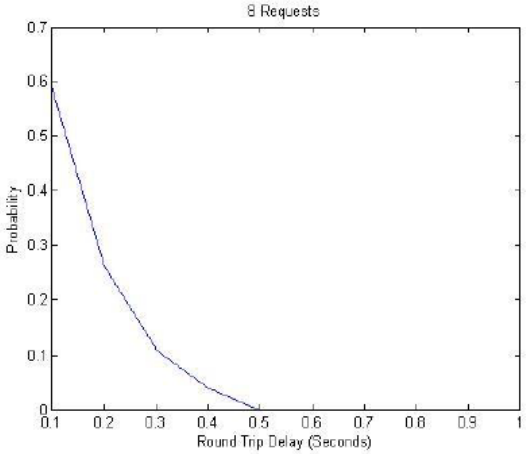


Figure 4. (8 Requests to Time Server)

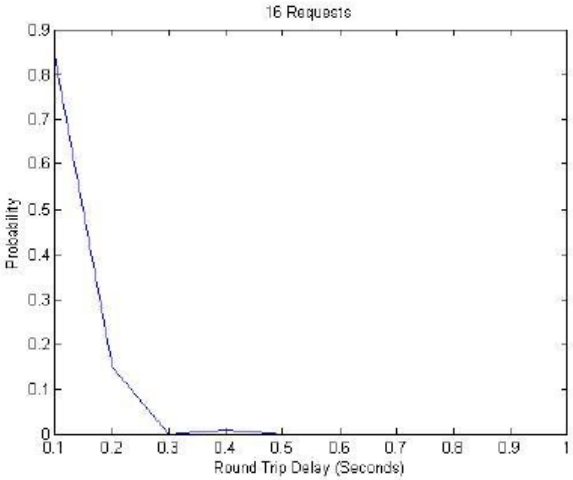
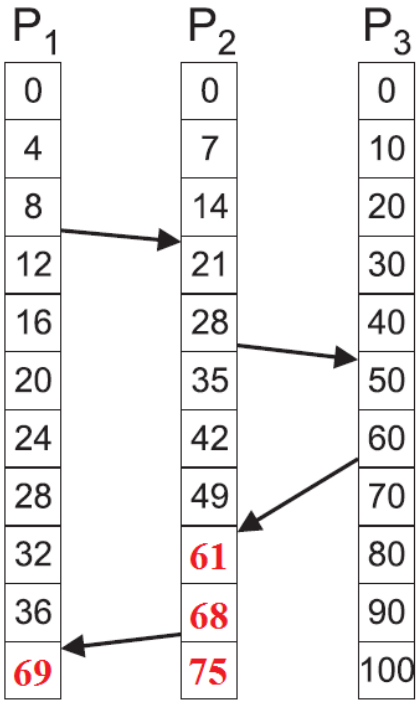
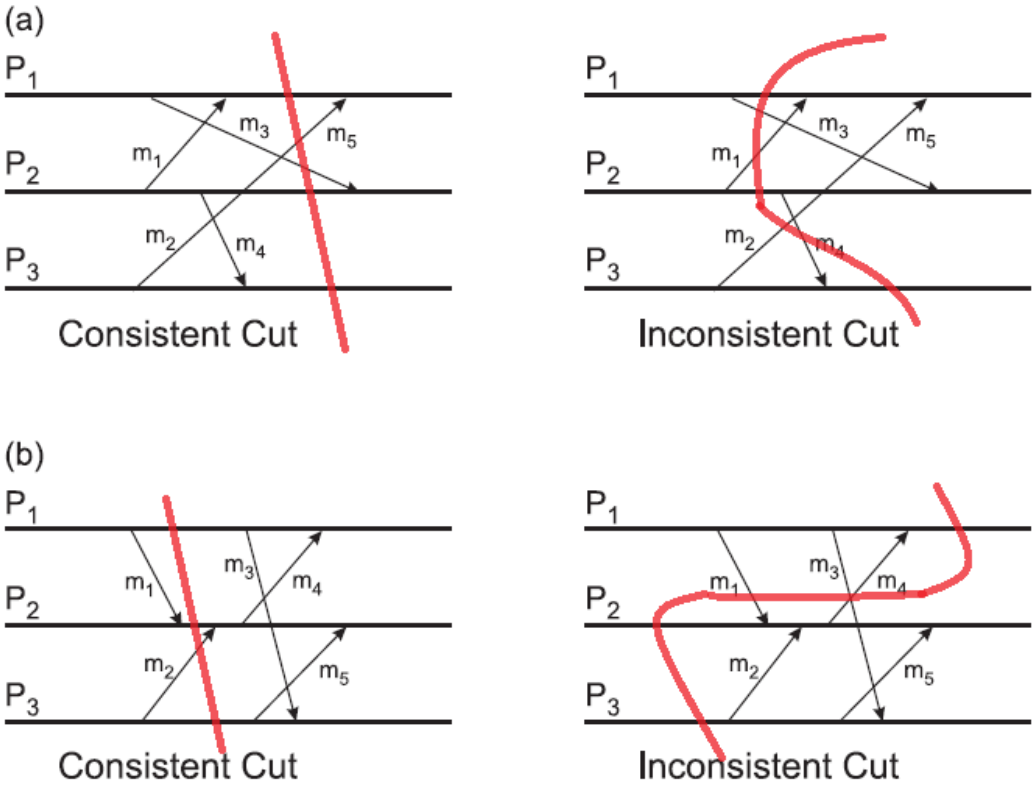


Figure 5. (16 Request to Time Server)

5. Consider the logical clocks of three processes, with events occurring at different rates in the different processes, as shown in the diagram on the left. Using Lamport's algorithm, show how each process adjusts its logical clock by filling in the logical clock values in the diagram on the right.



6. In the following diagrams there are three processes with messages sent and received by those processes. In (a), draw a consistent cut (distributed snapshot) at the left, and an inconsistent cut at the right. Similarly, in (b), draw a consistent cut at the left, and an inconsistent cut at the right.



In (a), for the inconsistent cut, m4 is received before it is sent. In (b), for inconsistent cut, m4 is received before it is sent.
A cut should not intersect any process twice. A cut should intersect all processes.