**Department of Computing**

**Course Code: CS332**

**Class: BSCS-10ABC**

**Lab 05: Understanding the Physical clock algorithms**

**CLO1: Design Distributed Protocols**

**Date: March 6, 2023 and March 9, 2023**

**Instructors: Dr. Khurram Shahzad & Dr.Shah Khalid**

# Lab 05: Understanding the Physical clock algorithms

**Introduction**

This lab focuses on the giving overview of physical clock algorithms. An implementation will be required to demonstrate the Cristian and Berkeley Algorithms.

**Objectives**

To understand the concept of Cristian Algorithm and Berkeley Algorithm.

**Tools/Software Requirement**

MS Visual Studio 2013

Or any editor of your choice

**Language**

Any of your choice, C++/Java/Python etc.

**Description**

Cristian’s Algorithm

Cristian’s Algorithm is a clock synchronization algorithm is used to synchronize time with a time server by client processes. This algorithm works well with low-latency networks where Round Trip Time is short as compared to accuracy while redundancy prone distributed systems/applications do not go hand in hand with this algorithm. Here Round Trip Time refers to the time duration between start of a Request and end of corresponding Response.

Algorithm:

1) The process on the client machine sends the request for fetching clock time(time at server) to the Clock Server at time T\_0.

2) The Clock Server listens to the request made by the client process and returns the response in form of clock server time.

3) The client process fetches the response from the Clock Server at time T\_1 and calculates the synchronised client clock time using the formula given below.

\[ T\_{CLIENT} = T\_{SERVER} + (T\_1 - T\_0)/2 \]

where T\_{CLIENT} refers to the synchronized clock time,

T\_{SERVER} refers to the clock time returned by the server,

T\_0 refers to the time at which request was sent by the client process,

T\_1 refers to the time at which response was received by the client process

**Berkeley’s Algorithm**

Berkeley’s Algorithm is a clock synchronization technique used in distributed systems. The algorithm assumes that each machine node in the network either doesn’t have an accurate time source or doesn’t possess an UTC server.

**Algorithm**  
1) An individual node is chosen as the master node from a pool nodes in the network. This node is the main node in the network which acts as a master and rest of the nodes act as slaves. Master node is chosen using a election process/leader election algorithm.

2) Master node periodically pings slaves nodes and fetches clock time at them using [Cristian’s algorithm](https://www.geeksforgeeks.org/cristians-algorithm/).

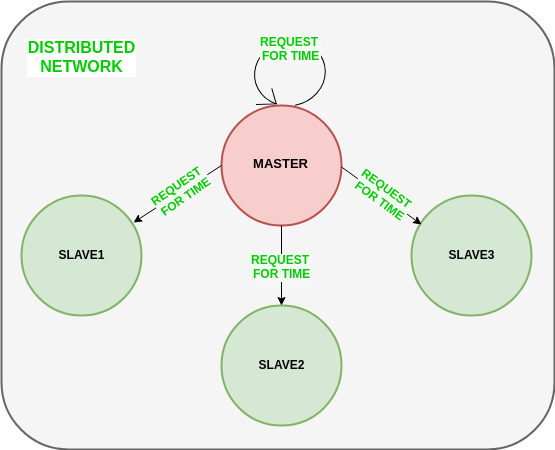
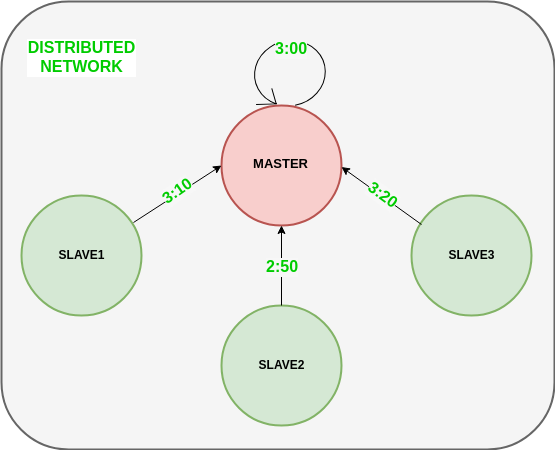
Diagram below illustrates how the master sends request to slave nodes.  


Diagram below illustrates how slave nodes send back time given by their system clock.  


3) Master node calculates average time difference between all the clock times received and the clock time given by master’s system clock itself. This average time difference is added to the current time at master’s system clock and broadcasted over the network.

Psuedocode for above step:

# receiving time from all slave nodes

repeat\_for\_all\_slaves:

time\_at\_slave\_node = receive\_time\_at\_slave()

# calculating time difference

time\_difference = time\_at\_master\_node - time\_at\_slave\_node

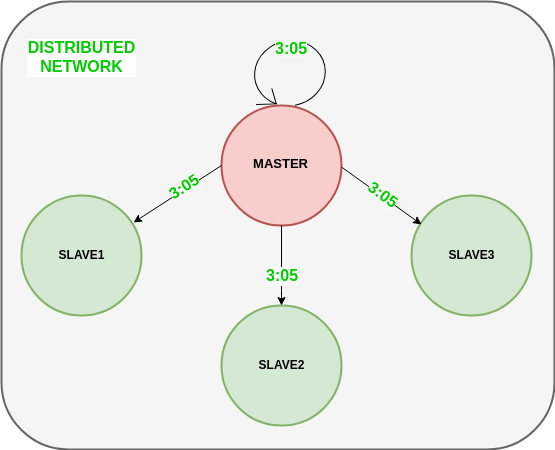
# average time difference calculation

average\_time\_difference = sum(all\_time\_differences) / number\_of\_slaves

synchronized\_time = current\_master\_time + average\_time\_difference

# broadcasting synchronized to whole network

broadcast\_time\_to\_all\_slaves(synchronized\_time)

Diagram below illustrates the last step of Berkeley’s algorithm.  


**Lab Tasks**

1. Implement the Cristian Algorithm to get the time returned by server,

Process Delay latency, Actual clock time at client side, and Synchronized process client time.

1. Implement the Berkeley Algorithm add 6 clients that need to be synchronized and print receiving time from all slave nodes, time difference, average\_time\_difference, synchronized\_time and finally print the synchronized\_time of all the client nodes.

.

**Deliverables**

Submit a single word file with code and screenshots of your output on LMS.