**Name: Shivam Tiwari**

**RollNo: 5117060**

# ****Aim:**** Implementation of Clock Synchronization Algorithms.

**Cristian’s Algorithm:**

**It** 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.

Below is an illustration imitating working of Cristian’s algorithm:

Lightbox

**Algorithm:**

1) The process on the client machine sends the request for fetching clock time(time at server) to the Clock Server at time *T0*.

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 *T1* and calculates the synchronised client clock time using the formula given below.

*TCLIENT = TSERVER + (T1-T0)/2*

where *TCLIENT*refers to the synchronised clock time,  
*TSERVER* refers to the clock time returned by the server,  
T0 refers to the time at which request was sent by the client process,  
T1 refers to the time at which response was received by the client process.

**Working/Reliability of above formula:**

*T0-T1*refers to the combined time taken by the network and the server to transfer request to the server, process the request and returning the response back to the client process, assuming that the network latency *T0*and *T1* are approximately equal.

The time at client side differs from actual time by at most *(T1-T0)/2* seconds. Using above statement we can draw a conclusion that the error in synchronisation can be at most *(T1-T0)/2* seconds.  
Hence,

  error

**PROGRAM:**

**Server :**

import socket

import datetime

# function used to initiate the Clock Server

def initiateClockServer():

s = socket.socket()

print("Socket successfully created")

# Server port

port = 8000

s.bind(('', port))

# Start listening to requests

s.listen(5)

print("Socket is listening...")

# Clock Server Running forever

while True:

# Establish connection with client

connection, address = s.accept()

print('Server connected to', address)

# Respond the client with server clock time

connection.send(str(

datetime.datetime.now()).encode())

# Close the connection with the client process

connection.close()

# Driver function

if \_\_name\_\_ == '\_\_main\_\_':

# Trigger the Clock Server

initiateClockServer()

**Client :**

import socket

import datetime

from dateutil import parser

from timeit import default\_timer as timer

# function used to Synchronize client process time

def synchronizeTime():

s = socket.socket()

# Server port

port = 8000

# connect to the clock server on local computer

s.connect(('127.0.0.1', port))

request\_time = timer()

# receive data from the server

server\_time = parser.parse(s.recv(1024).decode())

response\_time = timer()

actual\_time = datetime.datetime.now()

print("Time returned by server: " + str(server\_time))

process\_delay\_latency = response\_time - request\_time

print("Process Delay latency: " \

+ str(process\_delay\_latency) \

+ " seconds")

print("Actual clock time at client side: " \

+ str(actual\_time))

# synchronize process client clock time

client\_time = server\_time \

+ datetime.timedelta(seconds = \

(process\_delay\_latency) / 2)

print("Synchronized process client time: " \

+ str(client\_time))

# calculate synchronization error

error = actual\_time - client\_time

print("Synchronization error : "

+ str(error.total\_seconds()) + " seconds")

s.close()

# Driver function

if \_\_name\_\_ == '\_\_main\_\_':

# synchronize time using clock server

synchronizeTime()

**Output:**

**Sever:**

D:\Academic\Sem8 doc\DC>python clock\_server.py

Socket successfully created

Socket is listening...

Server connected to ('127.0.0.1', 49453)

**Client:**

D:\Academic\Sem8 doc\DC>python clock\_client.py

Time returned by server: 2021-04-01 18:19:17.736409

Process Delay latency: 0.0022155999999999842 seconds

Actual clock time at client side: 2021-04-01 18:19:17.738407

Synchronized process client time: 2021-04-01 18:19:17.737517

Synchronization error : 0.00089 seconds