Distributed Algorithms (UAI/503): Berkeley Algorithm

Eldhose Poulose

January 7, 2023

Abstract

In this report, the Berkeley Algorithm and its implementation using Java RMI technology is explained. The source file can be viewed under Github.

1 Introduction

In the Distributed Systems (DS) the nodes are communicating with each other using message passing. To achieve real time applications working in ordered manner the reference parameter used is time. Therefore synchronisation of time is essential for allocating the available resources. Synchronization in DS can be achieved by using physical clock of the node. For synchronization purpose, each node in the system needs to share their local clock time with another node in the system.

2 Berkeley Algorithm

The Berkeley Algorithm follows a master slave communication operation. There is a central computer that serves as the master or a time server. The master master periodically sends a request message to all other slaves or nodes which asks the time of the destination nodes. The master will receive the round trip time value (RTT) from the slaves, and the master will average the time values including its own clock value and readjusts its own clock accordingly. The master will also eliminate readings from faulty clocks that is values far outside the range. It does this by taking a subset of the returned times with a small variance. After this, the master will then send the amount by which each individual's clock requires adjustment to each individual clock. This value can be positive or negative. If the master fails at any point, then a new master is elected to take over and function exactly like its predecessor.

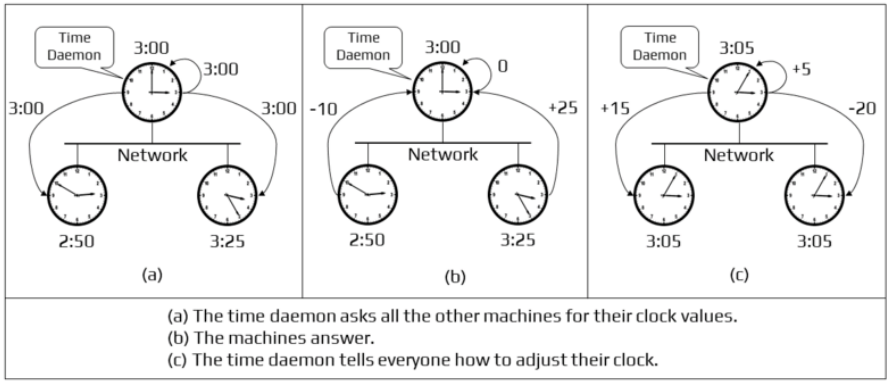

Figure 1: Berkeley Algorithm Workflow[source: see below]

3 Methods

- *client.py*: serial script
- *server.py*: File reading option 1 parallel version C script

4 Results

To test the script run server.py on the machine that will receive the connections and client.py on the machine that will send the connections.

Libraries used in the script:

- time - time library
- socket - library for making connections to other hosts

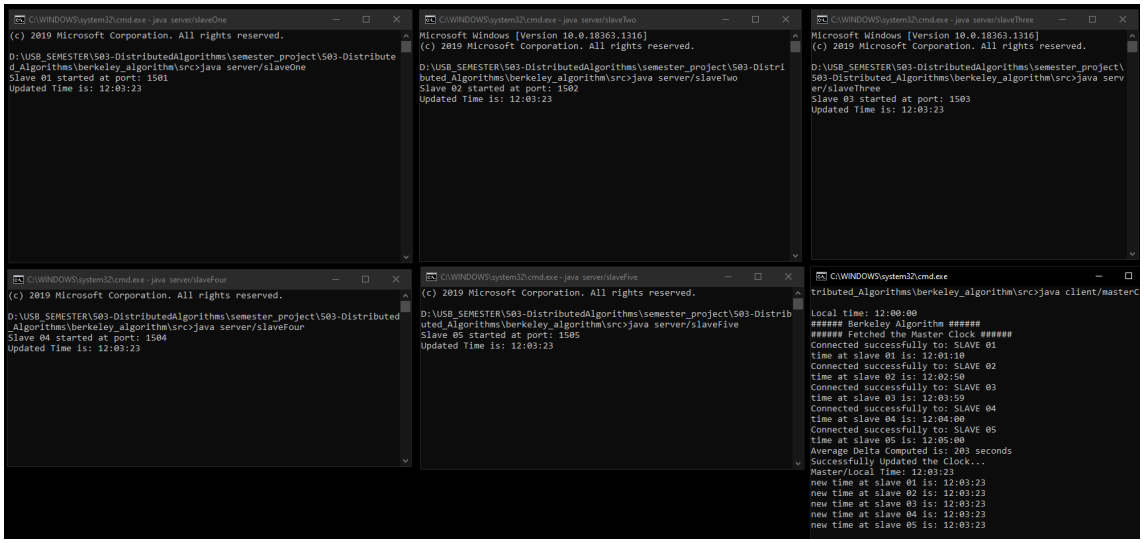

Figure 2: Updating 5 Slaves and a master clock time using Berkeley Algorithm

Link to Github

A Serial Code

```
import os
import io
os.chdir("C:/jithis_drive/zhaw/ACLS/Genomics_track2/miniJATI/batch_J/params")


#f = io.open('file.txt', 'w', newline='\n')

files = []
for i in range(1,601):
    files.append("params_" + str(i))


for i in range(0,6):
    for j in range(0,100):
        x = (i*100) + (j)
        f = io.open(files[x],"w",newline='\n')
        f.write("alphabet=DNA\nalignment=true\n")
        f.write("init.tree=user\ninput.sequence.sites_to_use=all\n")
        if i == 0 or i == 3:
            f.write("model=PIP(model=K80(kappa=2.0),lambda=100.0,mu=0.1)\n")
        if i == 1 or i == 4:
            f.write("model=PIP(model=K80(kappa=2.0),lambda=316.22776601683796,"
                    +"mu=0.31622776601683794)\n")
        if i == 2 or i == 5:
            f.write("model=PIP(model=K80(kappa=2.0),lambda=447.21359549995793"
                    +",mu=0.4472135954999579)\n")
        f.write("rate_distribution=Constant\n")
        f.write("optimization=ND-BFGS(derivatives=BFGS)\n")
        f.write("optimization.ignore_parameters=BrLen,Model\n")
        f.write("optimization.reparametrization=false\n")
        f.write("optimization.alignment=false\n")
        f.write("optimization.topology=false\n")
        f.write("input.sequence.file=/scratch/IAS/AnisGroup/peechjit/execs/"+str(i)+
                ".exec/output/sim-"+str(j)+"_sequence.txt\n")
        if i==0 or i==1 or i==2:
                f.write("input.tree.file=/scratch/IAS/AnisGroup/common/data/trees/"
                        +"ultrameric_tree_b0.01.nwk\n")
        else:
                f.write("input.tree.file=/scratch/IAS/AnisGroup/common/data/trees/"
                        +"unb_ultrameric_tree_b0.01.nwk\n")


        f.write("output.msa.file=/scratch/IAS/AnisGroup/peechjit/mJATI_output/"
                +"msa_out"+str(i)+"_"+str(j)+".fa\n")
        f.write("output.tree.file=/scratch/IAS/AnisGroup/peechjit/mJATI_output/"
                +"trees/tree_out"+str(i)+"_"+str(j)+".nwk\n")
        f.write("output.estimates=/scratch/IAS/AnisGroup/peechjit/mJATI_output/log"
                +str(i)+"_"+str(j)+"\n")
        f.close()
```