Title: Implementation of Clock synchronization.

1) Berkeley Algorithm

Code:

ClockServer:

```
package lab_03.RMIBerkeley.Server;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.time.LocalTime;

public interface ClockServer extends Remote {

   LocalTime getTime() throws RemoteException;

   void adjustTime(LocalTime timeClient, long changeInTime) throws
RemoteException;
}
```

ClockServerImpl:

```
package lab_03.RMIBerkeley.Server;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;

public class ClockServerImpl extends UnicastRemoteObject implements
ClockServer {

   /**
    *
    */
   private static final long serialVersionUID = -3844930637761365967L;

   public final DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("HH:mm:ss");
   private LocalTime time;

   public ClockServerImpl(LocalTime time) throws RemoteException {
```

```
        this.time = time;
    }

    @Override
    public LocalTime getTime() throws RemoteException {
        return time;
    }

    @Override
    public void adjustTime(LocalTime timeClient, long changeInTime)
throws RemoteException {
        long localTime = timeClient.toNanoOfDay();
        long thisTime = this.getTime().toNanoOfDay();
        var newTime = thisTime - localTime;
        newTime = newTime * -1 + changeInTime + thisTime;
        LocalTime newLocalTime = LocalTime.ofNanoOfDay(newTime);
        System.out.println("New Time: " + formatter.format(newLocalTime));
        this.time = newLocalTime;
    }

}
```

ServerOne:

```
package lab_03.RMIBerkeley.Server;

import java.time.format.DateTimeFormatter;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.time.LocalTime;

public class ServerOne {
        public static final DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("HH:mm:ss");

        public static void main(String[] args) {
                try {
                        // Server 1
                        ClockServer cs1 = new
ClockServerImpl(LocalTime.parse("07:05:00", formatter));
```

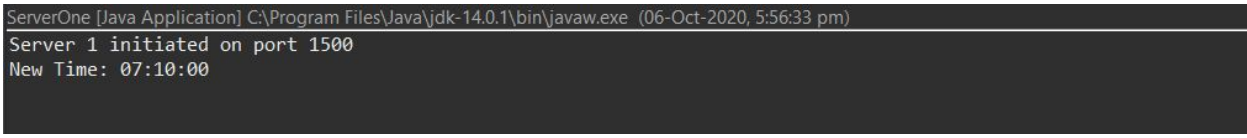
```
                Registry registry1 =
        LocateRegistry.createRegistry(1500);

        registry1.rebind(ClockServerImpl.class.getSimpleName(), cs1);
                System.out.println(String.format("Server 1 initiated
        on port %s", 1500));
            } catch (Exception ex) {
                System.out.println(ex);
            }
        }

    }
    ServerTwo:
    package lab_03.RMIBerkeley.Server;

    import java.time.format.DateTimeFormatter;
    import java.rmi.registry.LocateRegistry;
    import java.rmi.registry.Registry;
    import java.time.LocalTime;

    public class ServerTwo {
        public static final DateTimeFormatter formatter =
        DateTimeFormatter.ofPattern("HH:mm:ss");

        public static void main(String[] args) {
            try {
                // Server Two
                ClockServer cs2 = new
        ClockServerImpl(LocalTime.parse("07:10:00", formatter));
                Registry registry1 =
        LocateRegistry.createRegistry(1501);

        registry1.rebind(ClockServerImpl.class.getSimpleName(), cs2);
                System.out.println(String.format("Server 2 initiated
        on port %s", 1501));
            } catch (Exception ex) {
                System.out.println(ex);
            }
        }
```

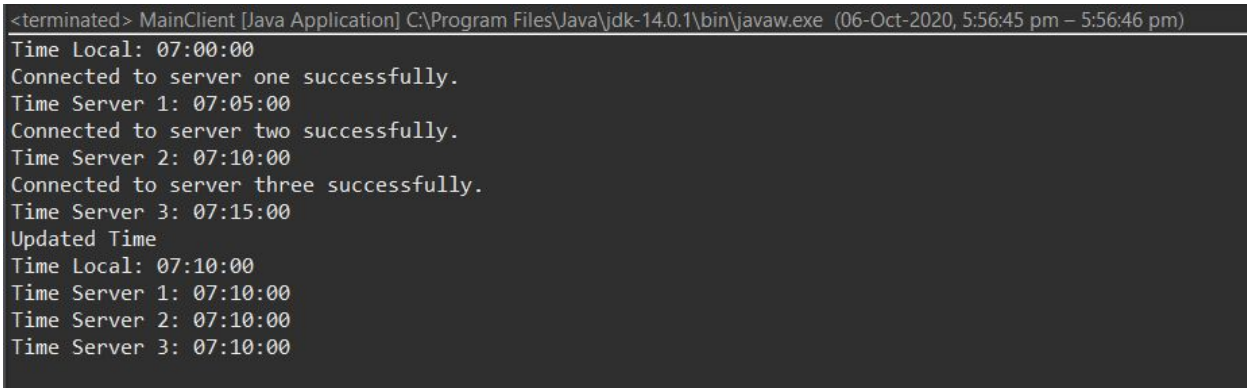
```
}
ServerThree:
package lab_03.RMIBerkeley.Server;

import java.time.format.DateTimeFormatter;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.time.LocalTime;

public class ServerThree {
	public static final DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("HH:mm:ss");

	public static void main(String[] args) {
		try {
			// Server Three
			ClockServer cs3 = new
ClockServerImpl(LocalTime.parse("07:15:00", formatter));
			Registry registry1 =
LocateRegistry.createRegistry(1502);

registry1.rebind(ClockServerImpl.class.getSimpleName(), cs3);
			System.out.println(String.format("Server 3 initiated
on port %s", 1502));
		} catch (Exception ex) {
			System.out.println(ex);
		}
	}
}
MasterClient(MainClient):
package lab_03.RMIBerkeley.Client;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.time.LocalTime;
import lab_03.RMIBerkeley.Server.*;
import java.time.format.DateTimeFormatter;
```

```
public class MainClient {

	public static final DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("HH:mm:ss");

	public static void main(String[] args) {
		try {
			LocalTime timeLocal = LocalTime.parse("07:00:00",
formatter);
			System.out.println("Time Local: " +
formatter.format(timeLocal));

			// Server 1
			Registry registry1 =
LocateRegistry.getRegistry("localhost", 1500);
			ClockServer cs1 = (ClockServer)
registry1.lookup(ClockServerImpl.class.getSimpleName());
			System.out.println("Connected to server one
successfully.");
			LocalTime timeServer1 = cs1.getTime();
			System.out.println("Time Server 1: " +
formatter.format(timeServer1));

			// Server 2
			Registry registry2 =
LocateRegistry.getRegistry("localhost", 1501);
			ClockServer cs2 = (ClockServer)
registry2.lookup(ClockServerImpl.class.getSimpleName());
			System.out.println("Connected to server two
successfully.");
			LocalTime timeServer2 = cs2.getTime();
			System.out.println("Time Server 2: " +
formatter.format(timeServer2));

			// Server 3
			Registry registry3 =
LocateRegistry.getRegistry("localhost", 1502);
			ClockServer cs3 = (ClockServer)
registry3.lookup(ClockServerImpl.class.getSimpleName());
```

```
                System.out.println("Connected to server three
successfully.");
                LocalTime timeServer3 = cs3.getTime();
                System.out.println("Time Server 3: " +
formatter.format(timeServer3));

                var nanoLocal = timeLocal.toNanoOfDay();
                var diffServer1 = timeServer1.toNanoOfDay() -
nanoLocal;
                var diffServer2 = timeServer2.toNanoOfDay() -
nanoLocal;
                var diffServer3 = timeServer3.toNanoOfDay() -
nanoLocal;
                var avgDiff = (diffServer1 + diffServer2 + diffServer3)
/ 3;

                cs1.adjustTime(timeLocal, avgDiff);
                cs2.adjustTime(timeLocal, avgDiff);
                cs3.adjustTime(timeLocal, avgDiff);

                timeLocal = timeLocal.plusNanos(avgDiff);

                System.out.println("Updated Time");

                // New time verification
                System.out.println("Time Local: " +
formatter.format(timeLocal));
                System.out.println("Time Server 1: " +
formatter.format(cs1.getTime()));
                System.out.println("Time Server 2: " +
formatter.format(cs2.getTime()));
                System.out.println("Time Server 3: " +
formatter.format(cs3.getTime()));
            } catch (Exception ex) {
                System.out.println(ex);
            }
        }

    }
```

Screenshot:





2) Crisitian's Algorithm

Code:

Client:

```
package lab_03.RPCCristian;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.text.SimpleDateFormat;
import java.util.Scanner;
import java.util.logging.Level;
```

```
import java.util.logging.Logger;

public class Client {
	public static SimpleDateFormat SDF = new
SimpleDateFormat("HH:mm:ss");
	public static long currentTime;
	private static String serverName;
	private static int serverPort;
	private static long timeNext;

	public static class Update {
		public void run() throws IOException {
			Socket client = new Socket(serverName, serverPort);
			OutputStream outToServer =
client.getOutputStream();
			DataOutputStream out = new
DataOutputStream(outToServer);
			InputStream inFromServer = client.getInputStream();
			DataInputStream in = new
DataInputStream(inFromServer);
			int i = 10;
			out.writeInt(10);
			for (int j = 0; j < i; j++) {
				timeNext += in.readLong();
			}
			timeNext /= i;
			currentTime = timeNext;
			timeNext = 0;
			System.out.println("New Time is " +
SDF.format(currentTime));
			client.close();
		}
	}

	public static class InacurateClockTime extends Thread {
		private long changeInTime;

		public InacurateClockTime(long inaccuracy) {
			changeInTime = inaccuracy;
```

```
                        currentTime = System.currentTimeMillis();
                    }

                    @Override
                    public void run() {
                        while (true) {
                            try {
                                Thread.sleep(1000 + changeInTime);
                                currentTime += 1000;

System.out.println(SDF.format(currentTime));
                            } catch (InterruptedException ex) {

Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
                            }
                        }
                    }
                }

                public Client(String serverName, int serverPort) {
                    Client.serverName = serverName;
                    Client.serverPort = serverPort;
                }

                public static void main(String[] args) throws InterruptedException,
IOException {
                    Scanner sc = new Scanner(System.in);
                    System.out.println("How inaccurate is the clock");
                    long inaccuracy = sc.nextLong();
                    new Client("localhost", 5000);
                    Client.InacurateClockTime time = new
InacurateClockTime(inaccuracy);
                    Client.Update U = new Update();
                    System.out.println("How often check the clock");
                    long checkTime = sc.nextLong();
                    sc.close();
                    time.start();
                    while (true) {
                        Thread.sleep(checkTime);
```

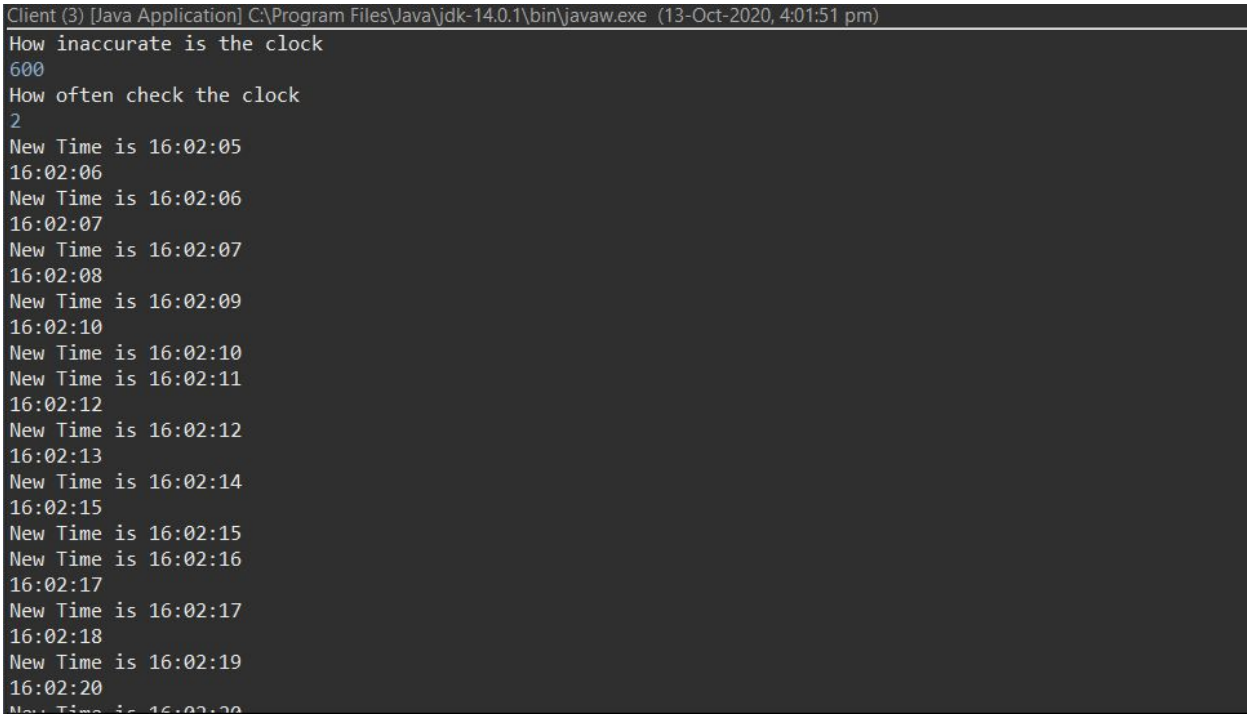
```
                        U.run();
                    }
                }

            }
```

Server:

```
package lab_03.RPCCristian;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Server extends Thread {
    private final ServerSocket serverSocket;

    public Server(int port) throws IOException {
        serverSocket = new ServerSocket(port);
    }

    public void run(){
        while (true) {
            try {
                Socket server = serverSocket.accept();
                DataInputStream in = new
DataInputStream(server.getInputStream());
                DataOutputStream out = new
DataOutputStream(server.getOutputStream());
                int i = in.readInt();
                for (int j = 0; j < i; j++) {
                    Thread.sleep((long) (100 + new
Random().nextInt(51)));
```

```
                out.writeLong(System.currentTimeMillis());
                                }
                                server.close();
                            } catch (UnknownHostException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                            } catch (IOException | InterruptedException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                            }
                        }
                    }

                    public static void main(String[] args) throws IOException {
                        int port = 5000;
                        Thread t = new Server(port);
                        t.start();
                    }

}
```

Screenshot:


Client (3) [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe (13-Oct-2020, 4:01:51 pm)
How inaccurate is the clock
600
How often check the clock
2
New Time is 16:02:05
16:02:06
New Time is 16:02:06
16:02:07
New Time is 16:02:07
16:02:08
New Time is 16:02:09
16:02:10
New Time is 16:02:10
New Time is 16:02:11
16:02:12
New Time is 16:02:12
16:02:13
New Time is 16:02:14
16:02:15
New Time is 16:02:15
New Time is 16:02:16
16:02:17
New Time is 16:02:17
16:02:18
New Time is 16:02:19
16:02:20