Experiment 3

AIM: Write a program to implement Berkeley clock synchronization algorithm

THEORY:

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.

1. A master is chosen via an election process (Bully algorithm)
2. The master polls the slave who reply with their time (like cristian's algorithm)
3. The master observes the round-trip time of the messages and estimates the time of each slave
4. The master then averages the clock time ignoring any value it receives.
5. Instead of sending the updated current coming back to the other process, the master then sends the amount that each must have adjusted its clock.

ALGORITHM

1. An individual node is chosen as the master node from a pool node in the network. This node is the main node in the network which acts as a master and the rest of the nodes act as slaves. Master node is chosen using an election process/leader election algorithm.
2. Master node periodically pings slaves' nodes and fetches clock time at them using Cristian's algorithm.
3. Master node calculates the 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.

SOURCE CODE:

Server

Time Daemon sends it's time to all connected nodes to adjust average time

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <pthread.h>
#include <fstream>
```

```
#include <sstream>
#include <ctime>
using namespace std;
void error(const char *msg)
{
 perror(msg);
 exit(1);
}
char buffer[256];
int n;
int l_clock = 0, tot = 0, avg = 0, cnt = 0, t = 0;
void Berkeley(long newsockfd)
{
 bzero(buffer, 256);
 stringstream ss, ss1, ss2;
 ss << l_clock;
 string tmpstr1 = ss.str();
 strcpy(buffer, tmpstr1.c_str());
 n = write((long)newsockfd, buffer, strlen(buffer));
 if (n < 0)
 error("ERROR writing to socket");
 bzero(buffer, 256);
 read((long)newsockfd, buffer, 255);
 printf( "Time Difference of Machine %d : %s \n“ , newsockfd , buffer);
 ss1 << buffer;
 bzero(buffer, 256);
 printf( "buffer: %s \n",  buffer );
 string tmpstr2 = ss1.str();
 cout << "tmpstr2:" << tmpstr2 << endl;
 int diff = atoi(tmpstr2.c_str());
 cout << "diff:" << diff << endl;
 tot = tot + diff;
 sleep(2);
 avg = tot / (cnt + 1);
 cout << "avg:" << avg << endl;
 int adj_clock = avg - diff;
 cout << "adj_clock:" << adj_clock << endl;
 bzero(buffer, 256);
 ss2 << adj_clock;
```

```
 string tmpstr3 = ss2.str();
 strcpy(buffer, tmpstr3.c_str());
 n = write((long)newsockfd, buffer, strlen(buffer));
 if (n < 0)
 error("ERROR writing to socket");
}
void *NewConnection(void *newsockfd)
{
 if ((long)newsockfd < 0)
 error("ERROR on accept");
 cout << "Connected to Machine Number: " << (long)newsockfd << endl;
 cnt++;
 while (cnt != t)
 {
 continue;
 }
 Berkeley((long)newsockfd);
 close((long)newsockfd);
 pthread_exit(NULL);
}
int main(int argc, char *argv[])
{
 long sockfd, newsockfd[10], portno;
 socklen_t clilen;
 struct sockaddr_in serv_addr, cli_addr;
 pthread_t threads[10];
 srand(time(0));
 l_clock = (rand() % 10) + 5; //genrate a random number to be taken as the clock time
 if (argc < 2)
 {fprintf(stderr, "ERROR, no port provided\n");
 exit(1);
 }
 sockfd = socket(AF_INET, SOCK_STREAM, 0);
 if (sockfd < 0)
 error("ERROR opening socket");
 bzero((char *)&serv_addr, sizeof(serv_addr));
 portno = atoi(argv[1]);
 serv_addr.sin_family = AF_INET;
 serv_addr.sin_addr.s_addr = INADDR_ANY;
```

```
 serv_addr.sin_port = htons(portno);
 int reuse = 1;
 if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (const char *)&reuse,
sizeof(reuse)) < 0) //To reuse socket address in case of crashes and failures
 perror("setsockopt(SO_REUSEADDR) failed");
#ifdef SO_REUSEPORT
 if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEPORT, (const char *)&reuse,
sizeof(reuse)) < 0)
 perror("setsockopt(SO_REUSEPORT) failed");
#endif
 if (bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
 error("ERROR on binding");
 listen(sockfd, 10);
 clilen = sizeof(cli_addr);
 cout << "Enter the total number of machines to connect: ";
 cin >> t;
 cout << "Time Daemon's Logical Clock: " << l_clock << endl;
 int sock_index = 0;
 for (int i = 0; i < t; i++)
 {newsockfd[sock_index] = accept(sockfd, (struct sockaddr *)&cli_addr, &clilen);
 pthread_create(&threads[i], NULL, NewConnection, (void *)newsockfd[sock_index]);
 sock_index = (sock_index + 1) % 100;
 }
 for (int i = 0; i < t; i++)
 {int rc = pthread_join(threads[i], NULL);
 if (rc)
 { printf("Error in joining thread :\n");
 cout << "Error: " << rc << endl;
 exit(-1);
 }
 }
 cout << "Clock adjustment: " << avg << endl;
 l_clock = l_clock + avg;
 cout << "My Adjusted clock: " << l_clock << endl;
 close(sockfd);
 pthread_exit(NULL);
 return 0;
}
```

Client

Client queries the Time Daemon and adjusts its clock according to Berkeley's

```
#include <stdio.h>
#include <cstdlib>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sstream>
#include <ctime>
using namespace std;
void error(const char *msg)
{ perror(msg);
 exit(0);
}
int main(int argc, char *argv[])
{
 long sockfd, portno, n;
 struct sockaddr_in serv_addr;
 struct hostent *server;
 int l_clock = 0;
 char buffer[256];
 if (argc < 2)
 {fprintf(stderr, "usage %s port\n", argv[0]);
 exit(0);
 }
 portno = atoi(argv[1]);
 srand(time(0));
 l_clock = (rand() % 15) + 5; // Defining the clock time range between 5 and 20
 sockfd = socket(AF_INET, SOCK_STREAM, 0);
 if (sockfd < 0)
 error("ERROR opening socket");
 server = gethostbyname("localhost");
 if (server == NULL)
 { fprintf(stderr, "ERROR, no such host\n");
 exit(0);}
```

```
 bzero((char *)&serv_addr, sizeof(serv_addr));
 serv_addr.sin_family = AF_INET;
 bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr, server->h_length);
 serv_addr.sin_port = htons(portno);
 if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
 error("ERROR connecting");
 cout << "My logical Clock: " << l_clock << endl;
 bzero(buffer, 256);
 n = read(sockfd, buffer, 255);
 if (n < 0)
 error("ERROR reading from socket");
 cout << "Time Daemon Initiating Berkeley's Algorithm!\nThis is TD's logical clock: " <<
buffer << endl;
 stringstream ss, ss1;
 ss << buffer;
 string tmpstr1 = ss.str();
 int tmp = atoi(tmpstr1.c_str());
 int diff = l_clock - tmp;
 cout << "My Time Difference from TD: " << diff << endl;
 bzero(buffer, 256);
 ss1 << diff;
 string tmpstr2 = ss1.str();
 strcpy(buffer, tmpstr2.c_str());
 n = write(sockfd, buffer, strlen(buffer)); // Sending this machine's time difference to bmaster
 if (n < 0)
 error("ERROR writing to socket");
 bzero(buffer, 256);
 n = read(sockfd, buffer, 255);
 printf("Clock Adjustment= %s\n", buffer);
 int adj_clock = atoi(buffer);
 l_clock = l_clock + adj_clock;
 printf("My Adjusted clock: %d \n" , l_clock);
 close(sockfd);
 return 0;
}
```

OUTPUT:

Server

```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$ ./berkley 8080
Enter the total number of machines to connect: 2
Time Daemon's Logical Clock: 12
Connected to Machine Number: 4
Connected to Machine Number: 5
Time Difference of Machine '5' : -4
buffer:
tmpstr2:-4
diff:-4
Time Difference of Machine '4' : 2
buffer:
tmpstr2:2
diff:2
avg:0
adj_clock:4
avg:0
adj_clock:-2
Clock adjustment: 0
My Adjusted clock: 12
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$
```

Client

```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$ ./berkleyc 8080
My logical Clock: 14
Time Daemon Initiating Berkeley's Algorithm!
This is TD's logical clock: 12
My Time Difference from TD: 2
Clock Adjustment= -2
My Adjusted clock: 12
```

```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$ ./berkleyc 8080
My logical Clock: 8
Time Daemon Initiating Berkeley's Algorithm!
This is TD's logical clock: 12
My Time Difference from TD: -4
Clock Adjustment= 4
My Adjusted clock: 12
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$
```

LEARNING OUTCOMES:

We learnt how to implement Berkeley's algorithm. We also learnt how clock synchronization technique works by implementing a master and slave node as a distributed system.There are many ways Berkeley could be improved further -

• Ignoring significant outliers in calculation of average time difference

• In case master node fails/corrupts, a secondary leader must be ready/pre chosen to take the place of the master node to reduce downtime caused due to master’s unavailability.