

PROJECT 2

BHANU PHANINDRA GORANTLA(QJ82419)

Assignment 1:

In this assignment we are asked to implement Berkley algorithm synchronization and it has been done as shown in presentation slides. This is split in to two programs namely berk.cpp and node2.cpp

Command line argument format for bank.cpp

./filename [port] [nodes in network excluding daemon]

Command line argument format for bank.cpp

./filename [server address] [port]

Nodes are randomly initialized using random function with in interval of 1 to 10 including daemon process. By the end of final step of this algorithm nodes are synchronized that is they will have same logical clock. Since there is no need of communication with other nodes, for this program only one port is used for time daemon.

Issue faced: while trying to implement this algorithm, identifying which clock belongs to which node when no other data except for time difference is sent. This can be fixed by mapping (own way using array) messages with socket descriptors but I used local copy to use difference.

```
Machine [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminal
fanix@fanix-VirtualBox: ~/Desktop/Berkley
fanix@fanix-VirtualBox:~/Desktop/Berkley$ ./node2 127.0.0.1 1060
my clock is :
3
Offset Received::3
New time after synchronisation is:6
fanix@fanix-VirtualBox:~/Desktop/Berkley$

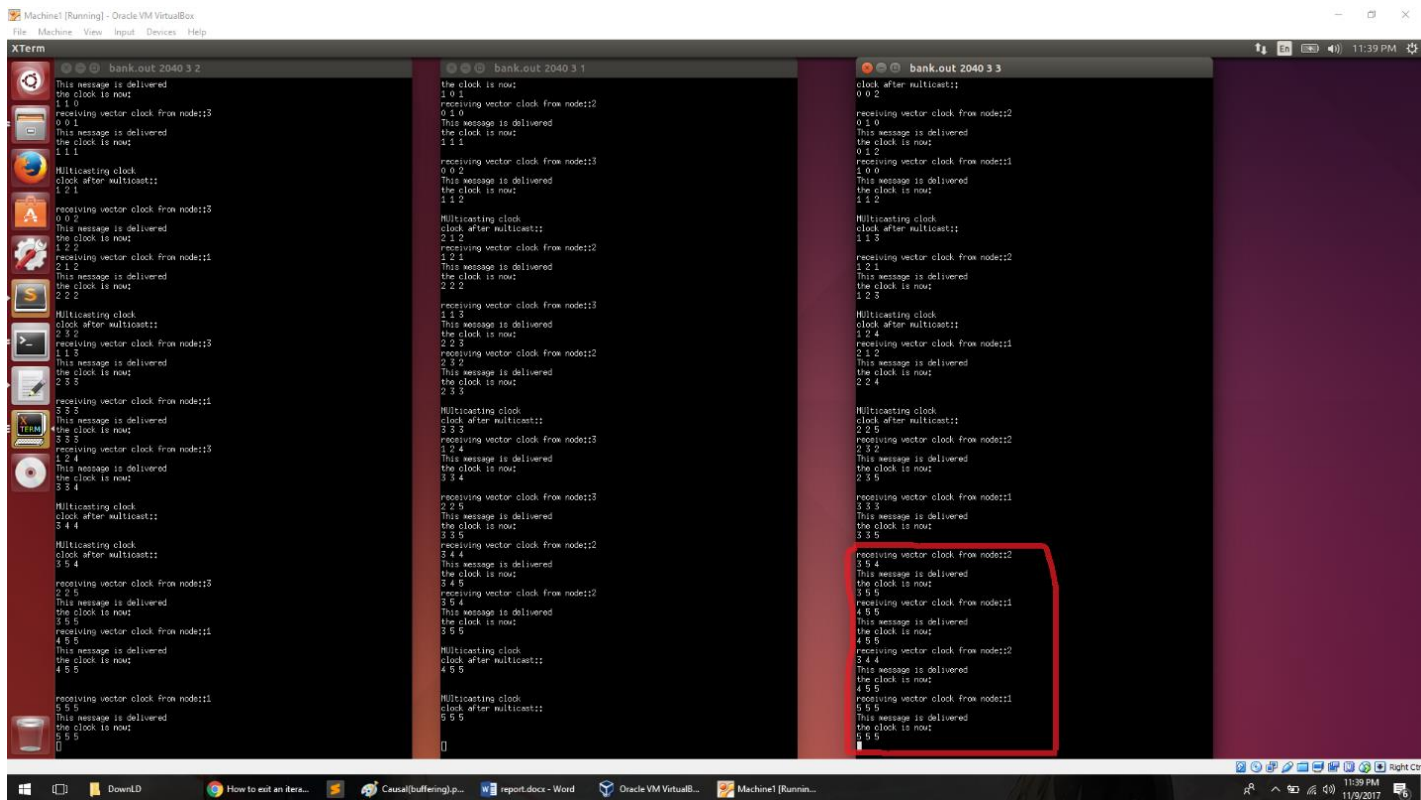
fanix@fanix-VirtualBox:~/Desktop/Berkley$ ./node2 127.0.0.1 1060
my clock is :
6
Offset Received::9
New time after synchronisation is:6
fanix@fanix-VirtualBox:~/Desktop/Berkley$

fanix@fanix-VirtualBox:~/Desktop/Berkley$ g++ bank.cpp -o bank -std=c++11 -pthread
fanix@fanix-VirtualBox:~/Desktop/Berkley$ ./bank 1060 2
Daemon clock is :7
average::1
new Daemon clock is:6
fanix@fanix-VirtualBox:~/Desktop/Berkley$
```

Assignment 2:

In this two versions of program is implemented, one with causal ordering and other without this feature. Since each node must communicate with other nodes each process needs port number. Instead of hardcoding port number of all process, a base port number is provided as command line argument and port number of a particular node is obtained by adding its node number(ID) to base port. Node number is also given as command line argument. In this program all nodes multicast five times(variable hardcoded in loop), therefore output at end of program is vector clock with all entries as five.

The following is screenshot of Non causal multicast.



The screenshot shows three terminal windows running the 'bank.out' program. The windows are titled 'bank.out 2040 3 2', 'bank.out 2040 3 1', and 'bank.out 2040 3 3'. The output in each window shows a sequence of messages being received and sent, with timestamps. The third window (bank.out 2040 3 3) has a red box highlighting a specific message: 'receiving vector clock from node:12' with timestamp '5 5 4'. This message is received before the 'Multicasting clock' message with timestamp '3 4 4' is received, illustrating non-causal ordering.

Three terminals running Non-causal multicast program using bash script.

To simulate delay of packets I intentionally delayed a packet belonging to fourth multicast from process 2 to process 3(Marked that part of output in above message). It could be seen that packet (3,5,4) is delivered to application even before (3,4,4) is delivered.

In causal ordered multicast when same delay is simulated packet (3,5,4) is buffered until correct packet is received. When correct packet is received buffer is checked for next correct packets. Buffer is implemented using Queue data structure.

```

2 2 5
receiving vector clock from node::2
2 3 2
This message is delivered
the clock is now:
2 3 5

receiving vector clock from node::1
3 3 3
This message is delivered
the clock is now:
3 3 5

receiving vector clock from node::2
3 5 4
This message is delivered
the clock is now:
3 5 5

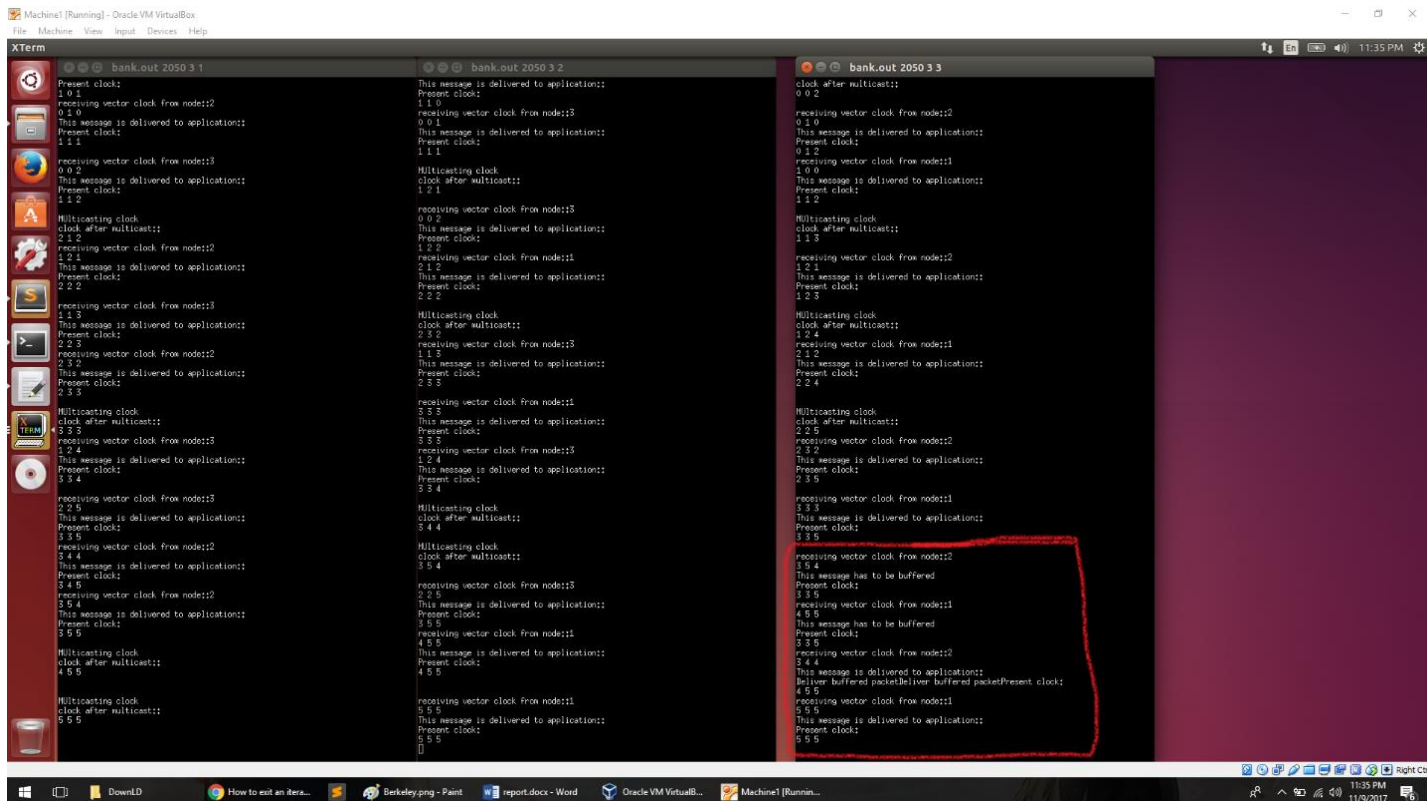
receiving vector clock from node::1
4 5 5
This message is delivered
the clock is now:
4 5 5

receiving vector clock from node::2
3 4 4
This message is delivered
the clock is now:
4 5 5

receiving vector clock from node::1
5 5 5
This message is delivered
the clock is now:
5 5 5

```

The following(below) is screenshot of causal ordering .



Three terminals running causal multicast program using bash script.

```
2 3 2
receiving vector clock from node::2
2 3 2
This message is delivered to application::
Present clock:
2 3 5

receiving vector clock from node::1
3 3 3
This message is delivered to application::
Present clock:
3 3 5

receiving vector clock from node::2
3 5 4
This message has to be buffered
Present clock:
3 3 5
receiving vector clock from node::1
4 5 5
This message has to be buffered
Present clock:
3 3 5
receiving vector clock from node::2
3 4 4
This message is delivered to application::
Deliver buffered packetDeliver buffered packetPresent clock:
4 5 5
receiving vector clock from node::1
5 5 5
This message is delivered to application::
Present clock:
5 5 5
```

NOTE: I hard coded loopcounter for receive function. So changing number of nodes in network might not execute code properly unless loop counter is changed. Bash script file is included for each assignment to run program

Assignment 3:

In this assignment access to shared file is implemented using distributed mutex. If a process wants access to shared file or not given as command line argument. Each process multicasts this information to other process. If a process does not want to access resource it sends ok message to all process. If not it compares with its own Process ID(Since Timestamps are equal) and sends ok signal if it is less than its own process ID. Else it replies back after using the resource. A process access shared text file and write its node number in to file only when it receives ok message from all nodes.

Usage:

./filename [port] [nodes in network] [node number of this program] [request to shared resource]

Request to shared resource can take two values, 0 and 1. 0 indicates access is not required and 1 means shared resource is required.

The following screenshot shows execution of bonus question.

```
bank.out 2040 3 3 0
This message is delivered to application:
Multicasting clock
Present clock:
receiving vector clock from node:1
receiving vector clock from node:2
receiving vector clock from node:3
In bonus assignment:
transmitted node value:1
transmitted node value:2
transmitted node value:3
received OK
my node number is :1

bank.out 2040 3 3 1
receiving vector clock from node:1
This message is delivered to application:
Multicasting clock
Present clock:
receiving vector clock from node:2
receiving vector clock from node:3
In bonus assignment:
transmitted node value:1
transmitted node value:2
transmitted node value:3
received OK
my node number is :1

bank.out 2040 3 3 2
receiving vector clock from node:1
This message is delivered to application:
Multicasting clock
Present clock:
receiving vector clock from node:2
receiving vector clock from node:3
In bonus assignment:
transmitted node value:1
transmitted node value:2
transmitted node value:3
received OK
my node number is :1
```

In above screenshot both process 2 and 1 wants to access shared file. Process 2 is given access first and it writes in to file. Then process 1 updates the contents of the file with its own node number.

This screenshot is the output that is written in to text file.

```
sfile.txt (~/.Desktop/Bonus) - gedit
my node number is :1
```

