Experiment 11

AIM: Write a program to implement 2-Phase Commit client-server.

THEORY:

In a local database system, for committing a transaction, the transaction manager has to only convey the decision to commit to the recovery manager. However, in a distributed system, the transaction manager should convey the decision to commit to all the servers in the various sites where the transaction is being executed and uniformly enforce the decision. When processing is complete at each site, it reaches the partially committed transaction state and waits for all other transactions to reach their partially committed states. When it receives the message that all the sites are ready to commit, it starts to commit. In a distributed system, either all sites commit or none of them does. Distributed two-phase commit reduces the vulnerability of one-phase commit protocols.

ALGORITHM

Phase 1:

Prepare Phase

- After each slave has locally completed its transaction, it sends a "DONE" message to the controlling site. When the controlling site has received "DONE" message from all slaves, it sends a "Prepare" message to the slaves.
- The slaves vote on whether they still want to commit or not. If a slave wants to commit, it sends a "Ready" message.
- A slave that does not want to commit sends a "Not Ready" message. This may happen when the slave has conflicting concurrent transactions or there is a timeout.

Phase 2:

Commit/Abort Phase

- After the controlling site has received "Ready" message from all the slaves −
  - The controlling site sends a "Global Commit" message to the slaves.
  - The slaves apply the transaction and send a "Commit ACK" message to the controlling site.
  - When the controlling site receives "Commit ACK" message from all the slaves, it considers the transaction as committed.
- After the controlling site has received the first "Not Ready" message from any slave
  - The controlling site sends a "Global Abort" message to the slaves.
  - The slaves abort the transaction and send a "Abort ACK" message to the controlling site.

- When the controlling site receives “Abort ACK” message from all the slaves, it considers the transaction as aborted.

SOURCE CODE:

Server

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include <string.h>
#define TRUE 1
#define FALSE 0
#define ML 1024
#define MPROC 32
typedef struct wireless_node
{
 int priority;
 int parent;
}wireless_node;
wireless_node w;
int max(int a, int b)
{
 return a >= b? a:b;
}
int connect_to_port(int connect_to)
{
 int sock_id;
 int opt = 1;
 struct sockaddr_in server;
 if ((sock_id = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
 {
 perror("unable to create a socket");
 exit(EXIT_FAILURE);
 }
 setsockopt(sock_id, SOL_SOCKET, SO_REUSEADDR, (const void*)&opt, sizeof(int));
 memset(&server, 0, sizeof(server));
 server.sin_family = AF_INET;
 server.sin_addr.s_addr = INADDR_ANY;
 server.sin_port = htons(connect_to);
 if (bind(sock_id, (const struct sockaddr *)&server,sizeof(server)) < 0)
 {
 perror("unable to bind to port");
 exit(EXIT_FAILURE);
 }
 return sock_id;
}
void send_to_id(int to, int from, char message[ML])
{
 struct sockaddr_in cl;
 memset(&cl, 0, sizeof(cl));
 cl.sin_family = AF_INET;
 cl.sin_addr.s_addr = INADDR_ANY;
 cl.sin_port = htons(to);
 sendto(from, (const char *)message, strlen(message), MSG_CONFIRM, (const struct sockaddr
*)&cl, sizeof(cl));
}
void begin_commit(int id, int *procs, int num_procs)
{
 int itr;
 char message[ML];
 sprintf(message, "%s", "SCMT");
 for (itr = 0; itr < num_procs; itr++)
 {
 printf("Sending begin commit to: %d\n", procs[itr]);
 send_to_id(procs[itr], id, message);
 }
}
void announce_action(int self, int *procs, int num_procs, char msg[ML])
{
 int itr;
 for (itr = 0; itr < num_procs; itr++)
 send_to_id(procs[itr], self, msg);
}
int main(int argc, char* argv[])
{
 int self = atoi(argv[1]), n_procs = atoi(argv[2]), procs[MPROC];
 int sender, okcnt = 0, nocnt = 0, dncnt = 0;
 int sock_id, coord_id, itr, len, n, start, ix;
 char buffer[ML], flag[ML], p_id[ML], msg[256];
 struct sockaddr_in from;
 for(itr = 0; itr < n_procs; itr += 1)
 procs[itr] = atoi(argv[3 + itr]);
 printf("Creating node at %d\n", self);
 sock_id = connect_to_port(self);
 begin_commit(sock_id, procs, n_procs);
 while(TRUE)
 {
 sleep(2);
 memset(&from, 0, sizeof(from));
 n = recvfrom(sock_id, (char *)buffer, ML, MSG_WAITALL, (struct sockaddr *)&from, &len);
 buffer[n] = '\0';
 printf("Recieved: %s\n", buffer);
 if (strcmp(buffer, "CMOK") == 0)
 okcnt += 1;
 else if (strcmp(buffer, "CMNO") == 0)
 nocnt += 1;
 if ((nocnt + okcnt) == n_procs)
 {
 printf("Recieved replies from all clients\n");
 if (okcnt == n_procs)
 { printf("Announcing complete commit\n");
 announce_action(sock_id, procs, n_procs, "CDON");
 }
 else
 { printf("Announcing abort commit\n");
 announce_action(sock_id, procs, n_procs, "CABT");
 }
 }
 if (strcmp(buffer, "DONE") == 0)
 {
```

```
 dncnt += 1;
 printf("clients confirmed commit\n");
 if (dncnt == n_procs)
 {
 printf("All process announced commit action\n");
 exit(EXIT_SUCCESS);
 } }}
 return 0;
}
```

Client

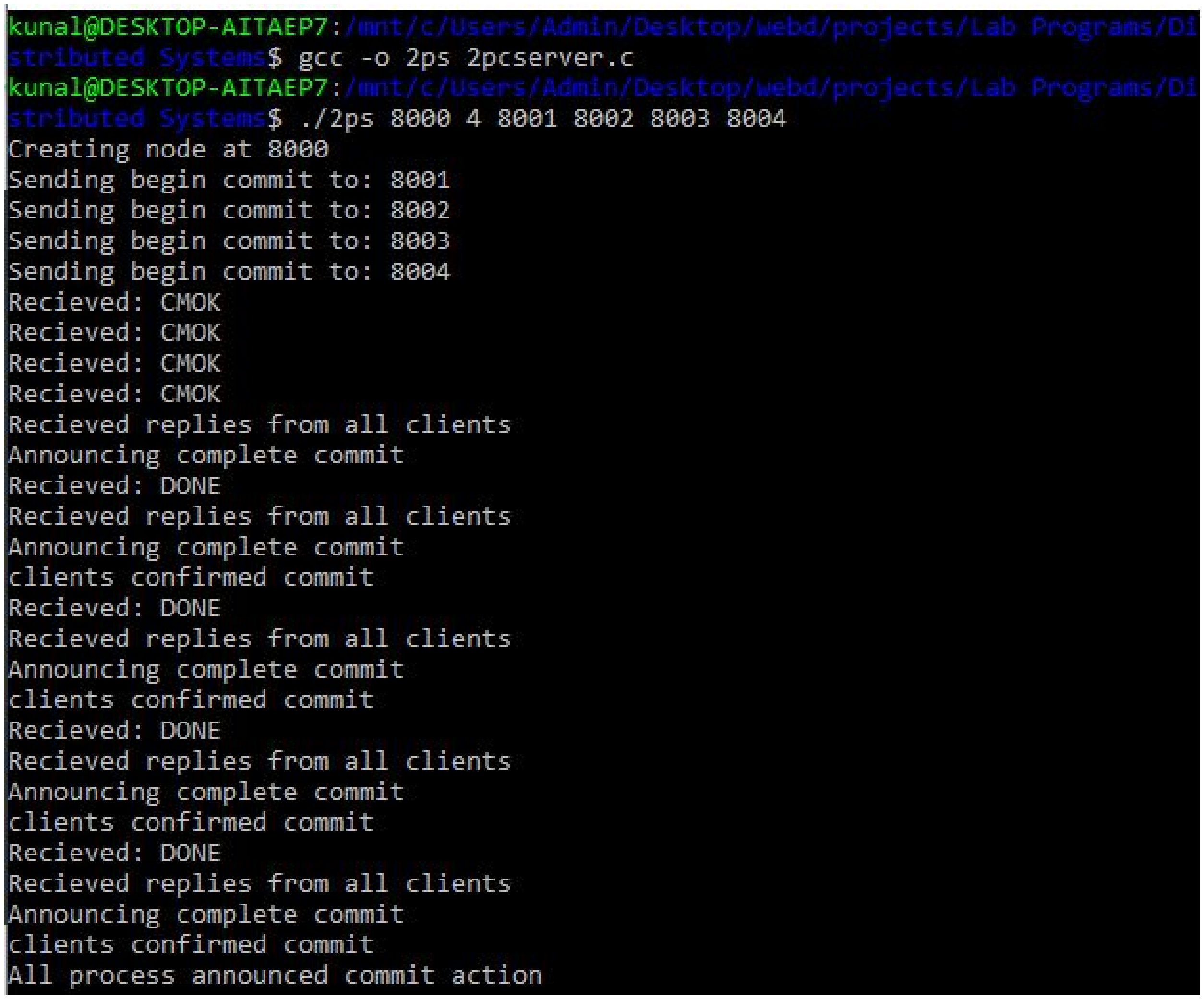
```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include <string.h>
#define TRUE 1
#define FALSE 0
#define ML 1024
#define MPROC 32
typedef struct wireless_node
{
 int priority;
 int parent;
}wireless_node;
wireless_node w;
int max(int a, int b)
{
 return a >= b? a:b;
}
int connect_to_port(int connect_to)
{
 int sock_id;
 int opt = 1;
 struct sockaddr_in server;
 if ((sock_id = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
 { perror("unable to create a socket");
 exit(EXIT_FAILURE);
 }
 setsockopt(sock_id, SOL_SOCKET, SO_REUSEADDR, (const void*)&opt, sizeof(int));
 memset(&server, 0, sizeof(server));
 server.sin_family = AF_INET;
 server.sin_addr.s_addr = INADDR_ANY;
 server.sin_port = htons(connect_to);
 if (bind(sock_id, (const struct sockaddr *)&server, sizeof(server)) < 0)
 {
 perror("unable to bind to port");
 exit(EXIT_FAILURE);
 }
 return sock_id;
}
void send_to_id(int to, int from, char message[ML])
{
 struct sockaddr_in cl;
 memset(&cl, 0, sizeof(cl));
 cl.sin_family = AF_INET;
 cl.sin_addr.s_addr = INADDR_ANY;
 cl.sin_port = htons(to);
 sendto(from, (const char *)message, strlen(message), MSG_CONFIRM, (const struct sockaddr
*)&cl, sizeof(cl));
}
void begin_commit(int id, int *procs, int num_procs)
{
 int itr;
 char message[ML];
 sprintf(message, "%s", "SCMT");
 for (itr = 0; itr < num_procs; itr++)
 {printf("Sending begin commit to: %d\n", procs[itr]);
 send_to_id(procs[itr], id, message);
 }
}
```

```
void announce_action(int self, int *procs, int num_procs, char msg[ML])
{ int itr;
 for (itr = 0; itr < num_procs; itr++)
 send_to_id(procs[itr], self, msg);
}
int main(int argc, char* argv[])
{
 int self = atoi(argv[1]), server = atoi(argv[2]);
 char *action = argv[3];
 int sender, okcnt = 0, nocnt = 0, dncnt = 0, sock_id, coord_id, itr, len, n, start, ix;
 char buffer[ML], flag[ML], p_id[ML], msg[256];
 struct sockaddr_in from;
 printf("Creating node at %d\n", self);
 sock_id = connect_to_port(self);
 while(TRUE)
 {
 sleep(2);
 memset(&from, 0, sizeof(from));
 n = recvfrom(sock_id, (char *)buffer, ML, MSG_WAITALL, (struct sockaddr *)&from, &len);
 buffer[4] = '\0';
 printf("Recieved: %s\n", buffer);
 if (strcmp(buffer, "SCMT") == 0)
 { printf("Sending %s to server\n", action);
 send_to_id(server, sock_id, action);
 }
 else if (strcmp(buffer, "CDON") == 0)
 { printf("Got complete commit, committing to logs\n");
 send_to_id(server, sock_id, "DONE");
 exit(EXIT_FAILURE);
 }
 else if (strcmp(buffer, "CABT") == 0)
 {
 printf("Got abort commit, deleting updates\n");
 send_to_id(server, sock_id, "DONE");
 exit(EXIT_FAILURE);
 }
 }
 return 0;
}
```

OUTPUT:

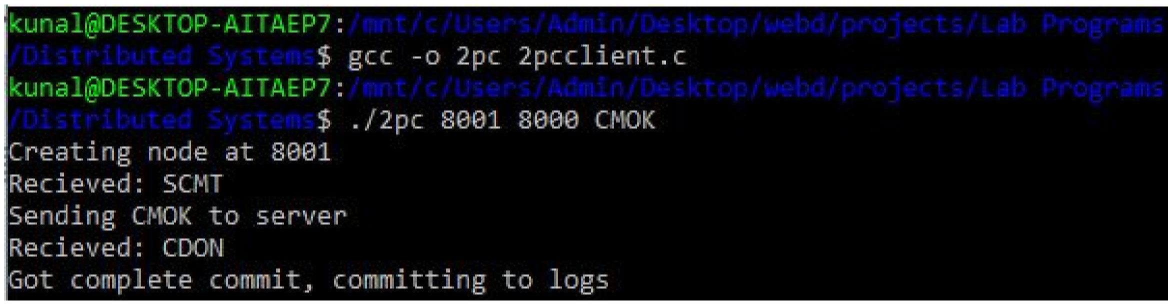
COMPLETE COMMIT

Server

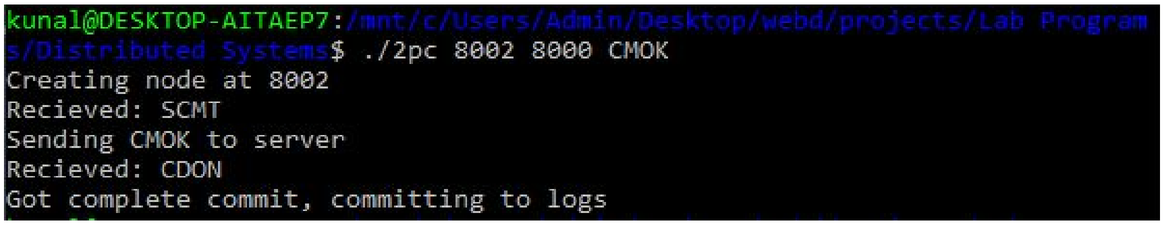


```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Programs/Di
stributed Systems$ gcc -o 2ps 2pcserver.c
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Programs/Di
stributed Systems$ ./2ps 8000 4 8001 8002 8003 8004
Creating node at 8000
Sending begin commit to: 8001
Sending begin commit to: 8002
Sending begin commit to: 8003
Sending begin commit to: 8004
Recieved: CMOK
Recieved: CMOK
Recieved: CMOK
Recieved: CMOK
Recieved replies from all clients
Announcing complete commit
Recieved: DONE
Recieved replies from all clients
Announcing complete commit
clients confirmed commit
Recieved: DONE
Recieved replies from all clients
Announcing complete commit
clients confirmed commit
Recieved: DONE
Recieved replies from all clients
Announcing complete commit
clients confirmed commit
Recieved: DONE
Recieved replies from all clients
Announcing complete commit
clients confirmed commit
All process announced commit action
```

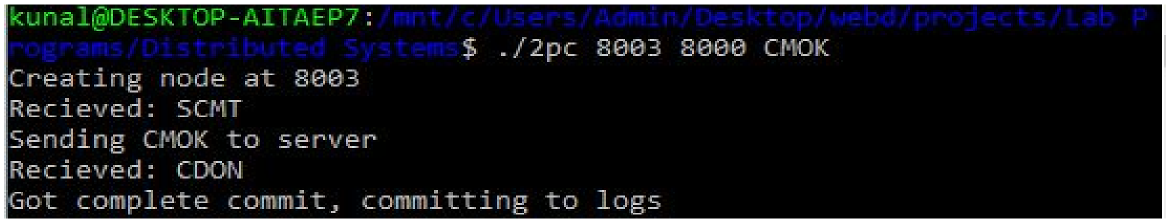
Clients



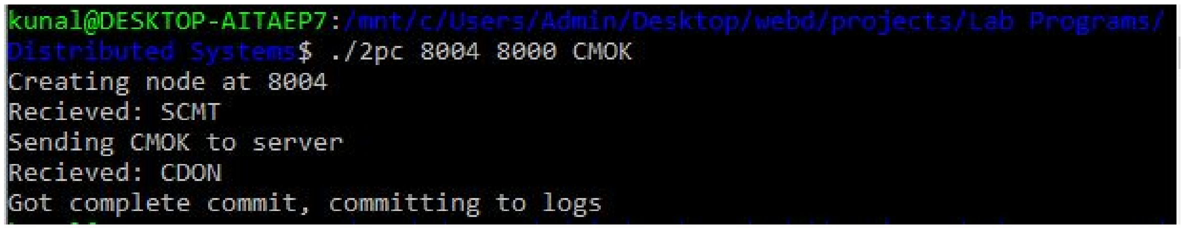
```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Programs
/Distributed Systems$ gcc -o 2pc 2pcclient.c
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Programs
/Distributed Systems$ ./2pc 8001 8000 CMOK
Creating node at 8001
Recieved: SCMT
Sending CMOK to server
Recieved: CDON
Got complete commit, committing to logs
```



```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Program
s/Distributed Systems$ ./2pc 8002 8000 CMOK
Creating node at 8002
Recieved: SCMT
Sending CMOK to server
Recieved: CDON
Got complete commit, committing to logs
```



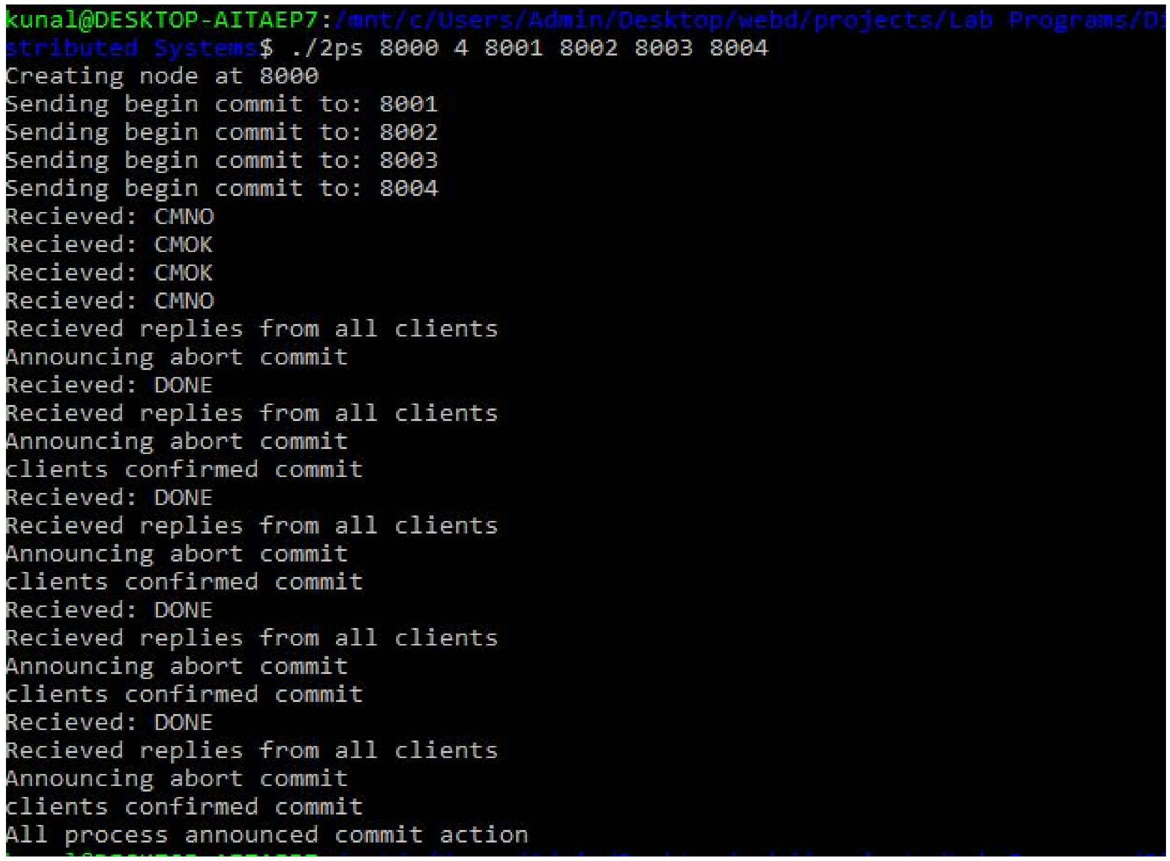
```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab P
rograms/Distributed Systems$ ./2pc 8003 8000 CMOK
Creating node at 8003
Recieved: SCMT
Sending CMOK to server
Recieved: CDON
Got complete commit, committing to logs
```



```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Programs/
Distributed Systems$ ./2pc 8004 8000 CMOK
Creating node at 8004
Recieved: SCMT
Sending CMOK to server
Recieved: CDON
Got complete commit, committing to logs
```

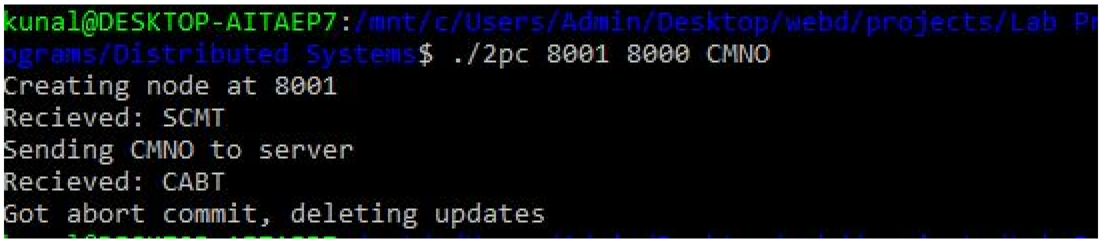
<u>ABORT COMMIT</u>

Server

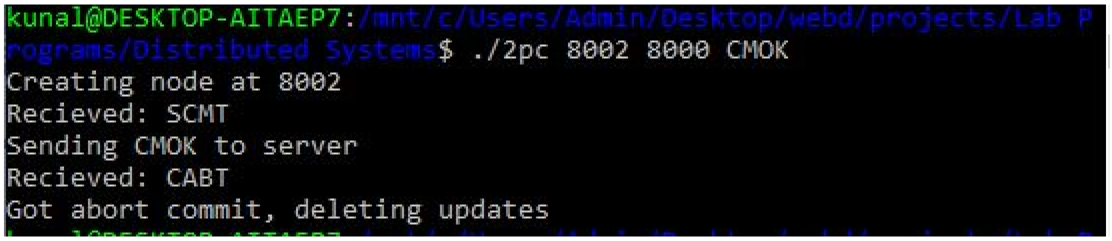


```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Programs/Di
stributed Systems$ ./2ps 8000 4 8001 8002 8003 8004
Creating node at 8000
Sending begin commit to: 8001
Sending begin commit to: 8002
Sending begin commit to: 8003
Sending begin commit to: 8004
Recieved: CMNO
Recieved: CMOK
Recieved: CMOK
Recieved: CMNO
Recieved replies from all clients
Announcing abort commit
Recieved: DONE
Recieved replies from all clients
Announcing abort commit
clients confirmed commit
Recieved: DONE
Recieved replies from all clients
Announcing abort commit
clients confirmed commit
Recieved: DONE
Recieved replies from all clients
Announcing abort commit
clients confirmed commit
Recieved: DONE
Recieved replies from all clients
Announcing abort commit
clients confirmed commit
All process announced commit action
```

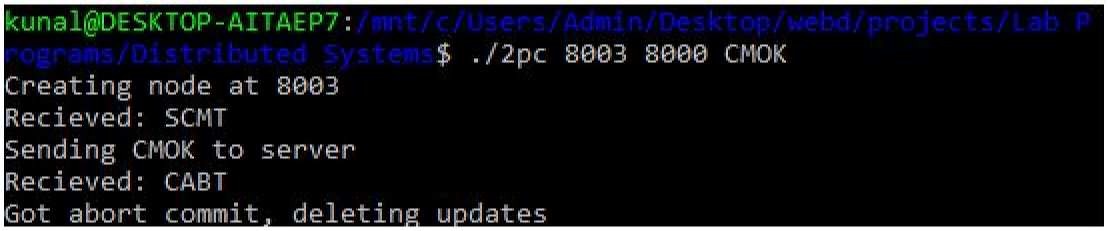
Client



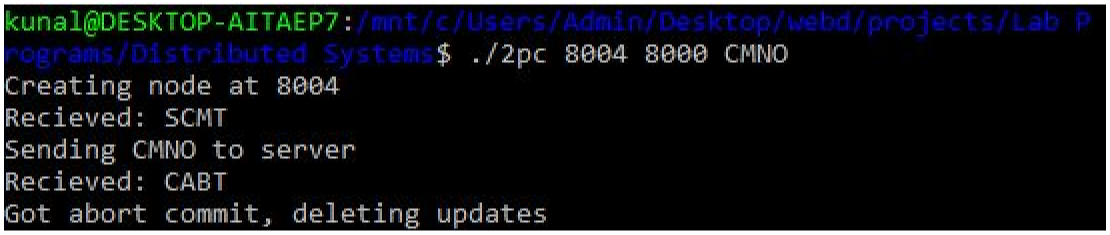
```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Pr
ograms/Distributed Systems$ ./2pc 8001 8000 CMNO
Creating node at 8001
Recieved: SCMT
Sending CMNO to server
Recieved: CABT
Got abort commit, deleting updates
```



```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab P
rograms/Distributed Systems$ ./2pc 8002 8000 CMOK
Creating node at 8002
Recieved: SCMT
Sending CMOK to server
Recieved: CABT
Got abort commit, deleting updates
```



```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab P
rograms/Distributed Systems$ ./2pc 8003 8000 CMOK
Creating node at 8003
Recieved: SCMT
Sending CMOK to server
Recieved: CABT
Got abort commit, deleting updates
```



```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab P
rograms/Distributed Systems$ ./2pc 8004 8000 CMNO
Creating node at 8004
Recieved: SCMT
Sending CMNO to server
Recieved: CABT
Got abort commit, deleting updates
```

LEARNING OUTCOMES:

We successfully implemented a 2-phase commit. In a distributed system, either all sites commit or none of them does. Distributed two-phase commit reduces the vulnerability of one-phase commit protocols.