

# CSE:5306 - DISTRIBUTED SYSTEMS

## Project 3 Report

Team:

Name	ID
Chinmay Mandale	1001995246
Rohit Padwal	1001833828

I have neither given nor received unauthorized assistance on this work.

Chinmay Mandale  
Rohit Padwal

**Date:**  
October 31, 2022

### **Work Distribution:**

Chinmay Mandale: Worked on Two-phase commit protocol and Documentation  
Rohit Padwal: Worked on Two-phase commit protocol and Documentation

# Two-phase distributed commit protocol

To ensure correctness and fault tolerance in a distributed database system, the two-phase commit protocol divides a database commit into two phases. Even in the event of site failures and message losses, the protocol results in either all nodes committing the transaction or aborting it.

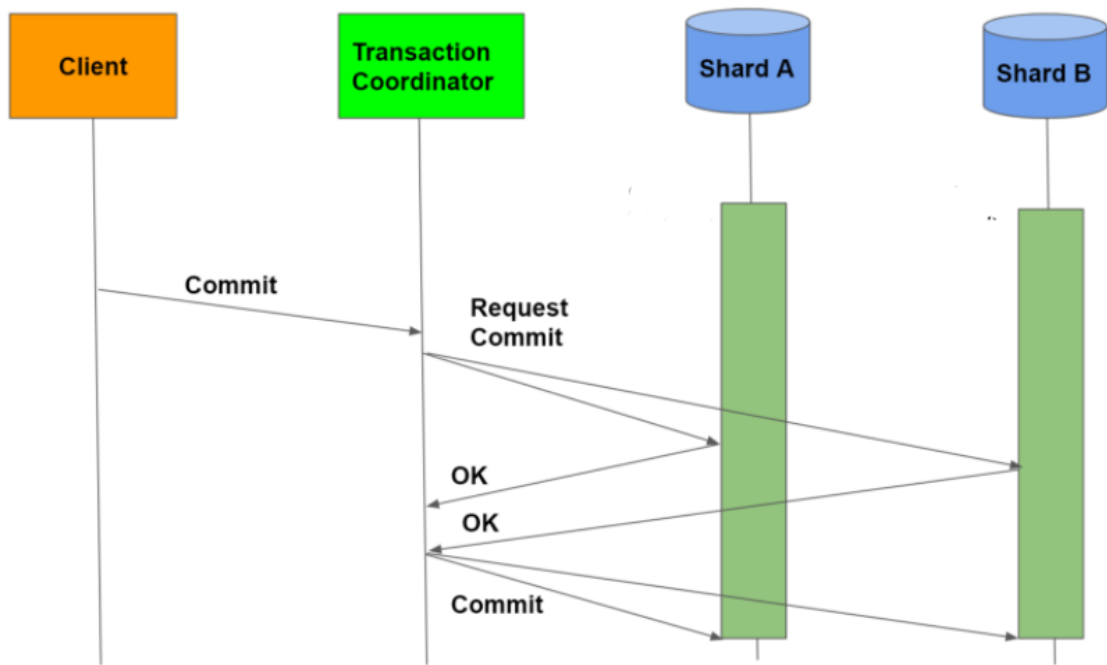
In a distributed transaction, a special object known as a coordinator is required. The coordinator, as the name implies, organizes activities and synchronization between distributed servers. The two-phase commit is carried out as follows:

## Phase 1:

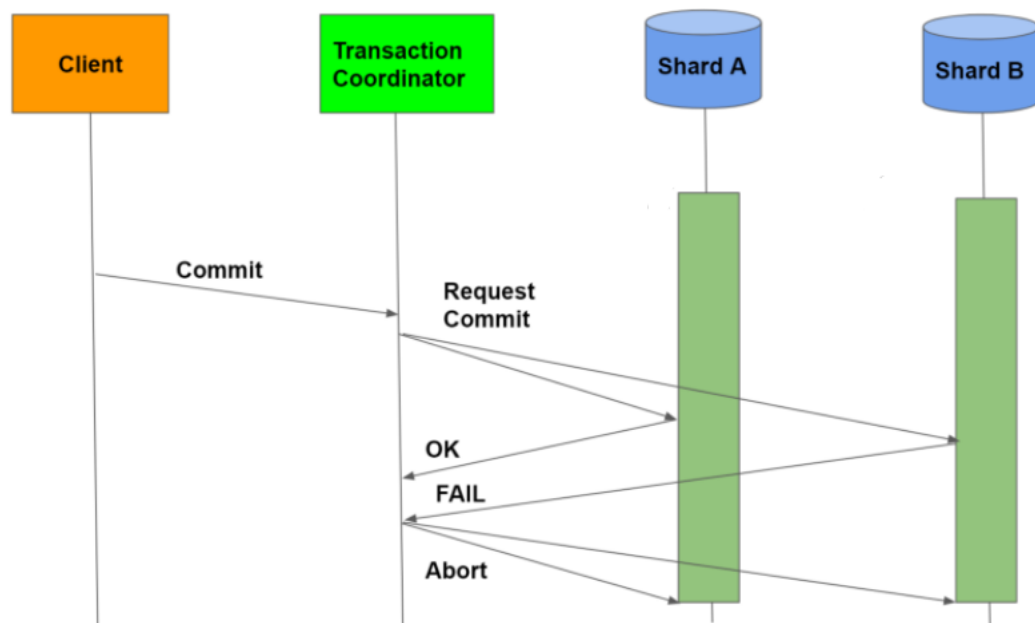
Each server that needs to commit data logs its data records. If a server fails, it responds with a failure message. If the request is successful, the server responds with an OK message.

## Phase 2:

This phase begins once all participants have responded positively. The coordinator then sends a signal to each server containing commit instructions. After committing, each writes the commit to its log record for future reference and sends a message to the coordinator indicating that its commit was successfully implemented. If a server fails, the coordinator notifies all servers and instructs them to roll back the transaction. Following the rollback, each server sends feedback that this has been completed.



Commit Txn after receiving OK from both the servers



Rollback the transaction in case of failure

## Technologies used:

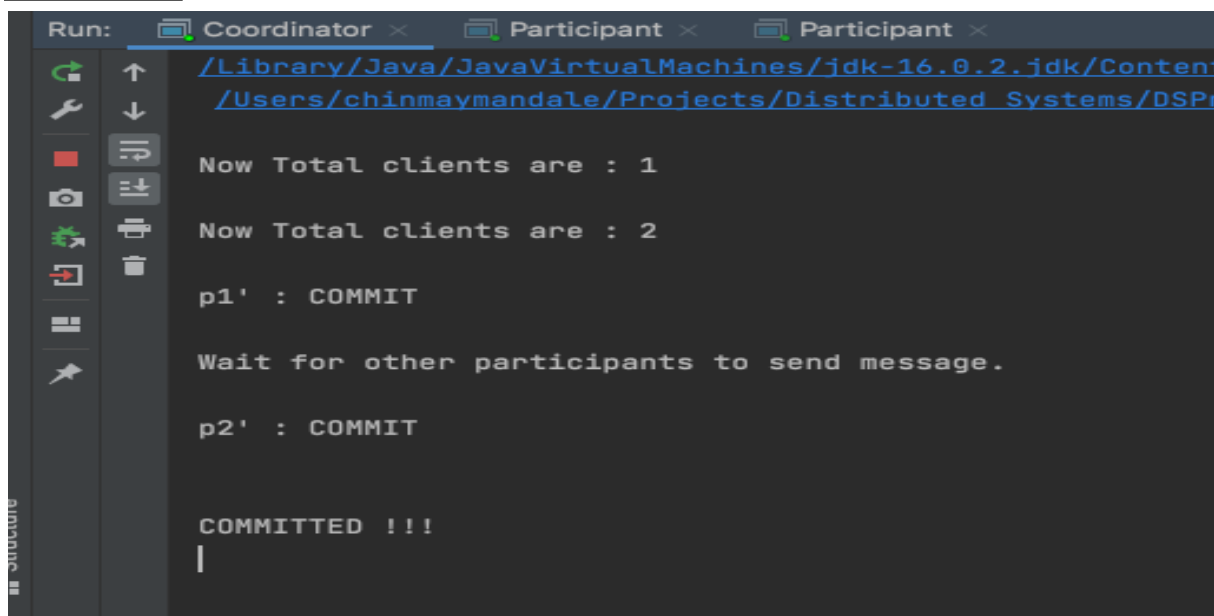
Java 8

## Program Output Snapshots:

1. Run the Coordinator.java file to initiate the coordinator
2. Run Participant.java for creating 1st instance
3. Run Participant.java once again for the second instance
4. We can observe no of instances on server
5. Instance 1 for the Participant.java enters Commit
6. Instance 2 for the Participant.java enters Commit, so it's a global commit
7. Instance 1 for the Participant.java enters Commit
8. Instance 2 for the Participant.java enters Abort, so it's a global abort
9. When one of the instances enters nothing for 30 secs, it will be a global abort.

### **CASE 1: Global Commit: both participants commit**

#### **Coordinator:**



```
Run: Coordinator x Participant x Participant x
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java
/Users/chinmaymandale/Projects/Distributed Systems/DSP/Coordinator.java

Now Total clients are : 1

Now Total clients are : 2

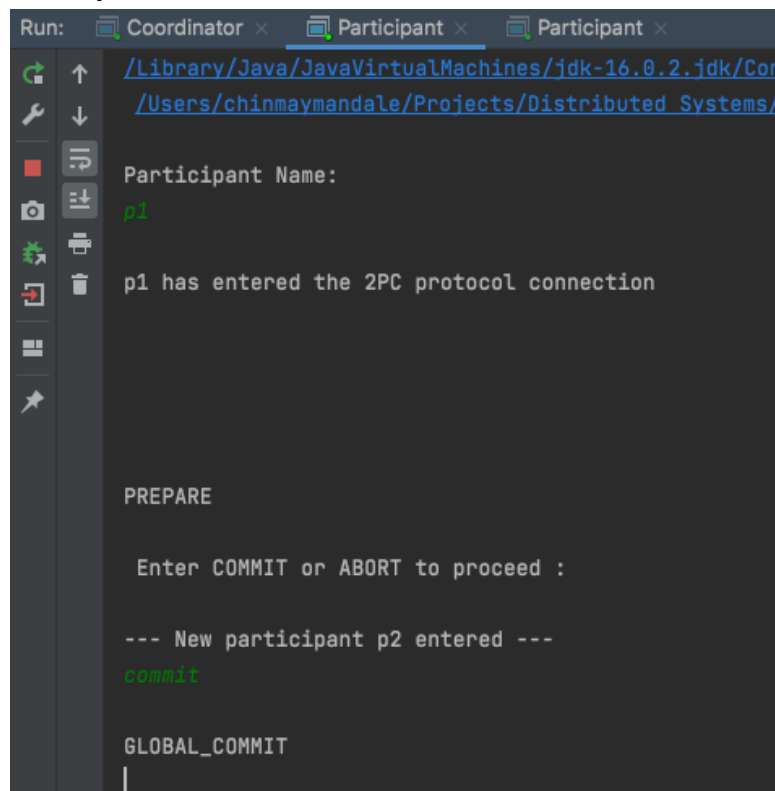
p1' : COMMIT

Wait for other participants to send message.

p2' : COMMIT

COMMITTED !!!
|
```

## Participant 1



The screenshot shows an IDE with three tabs: 'Coordinator', 'Participant', and 'Participant'. The 'Participant' tab is active, displaying the following text:

```
Participant Name:
p1

p1 has entered the 2PC protocol connection

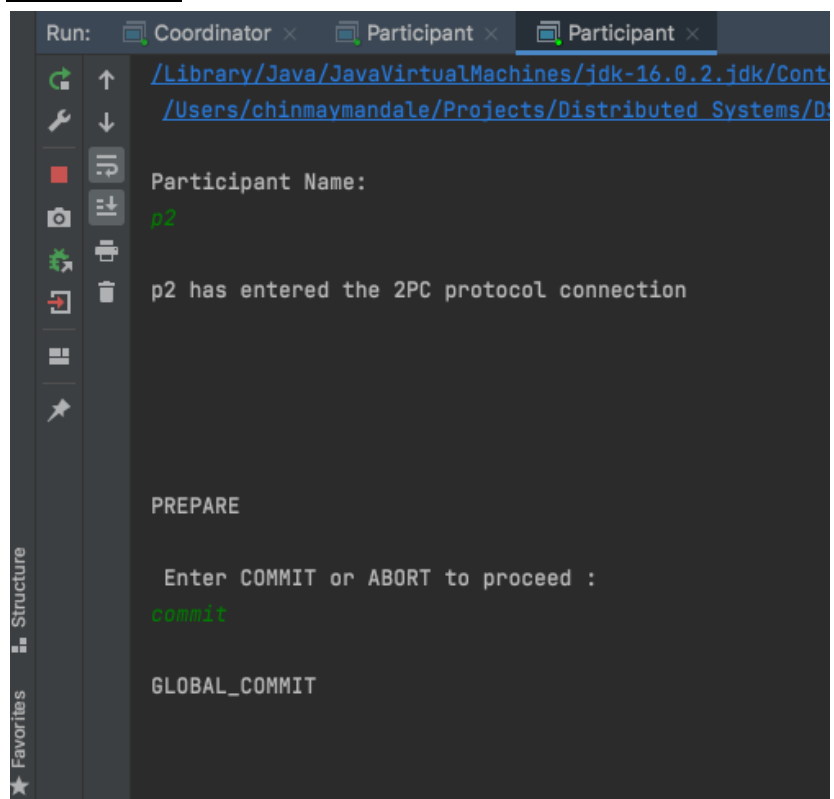
PREPARE

Enter COMMIT or ABORT to proceed :

--- New participant p2 entered ---
commit

GLOBAL_COMMIT
```

## Participant 2



The screenshot shows an IDE with three tabs: 'Coordinator', 'Participant', and 'Participant'. The 'Participant' tab is active, displaying the following text:

```
Participant Name:
p2

p2 has entered the 2PC protocol connection

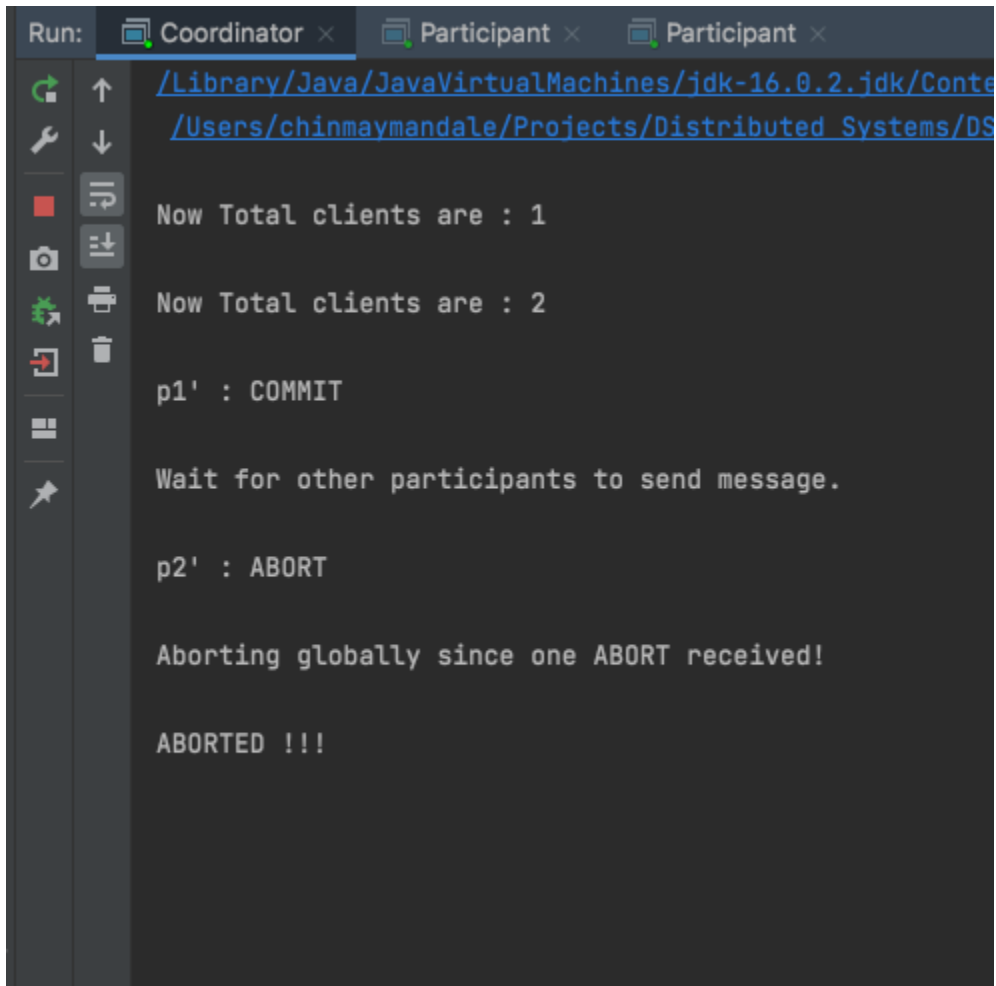
PREPARE

Enter COMMIT or ABORT to proceed :
commit

GLOBAL_COMMIT
```

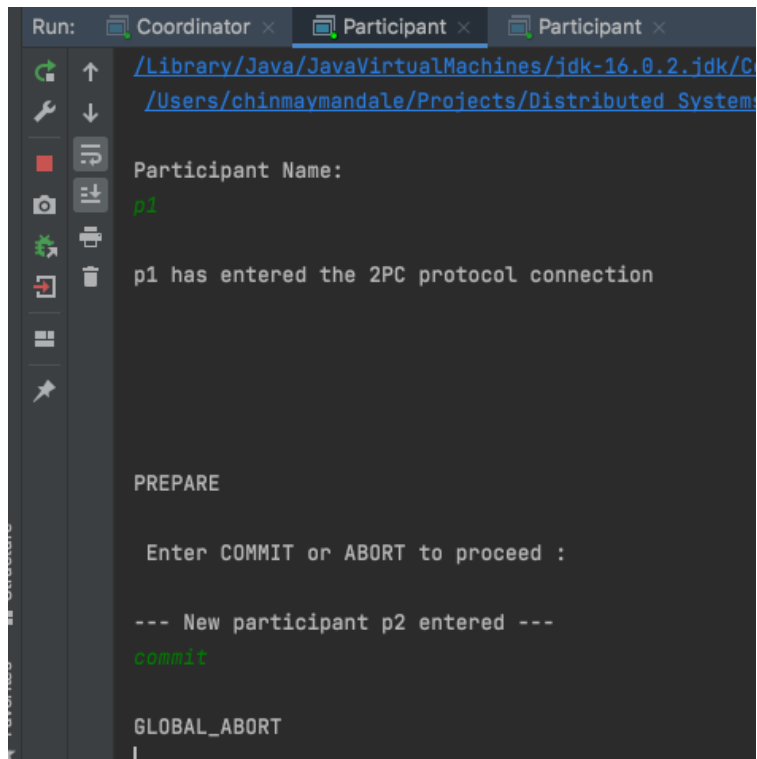
## CASE 2: Global Abort: Participant 1 commits but Participant 2 aborts

### Coordinator:



```
Run: Coordinator x Participant x Participant x
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Conte
/Users/chinmaymandale/Projects/Distributed Systems/DS
Now Total clients are : 1
Now Total clients are : 2
p1' : COMMIT
Wait for other participants to send message.
p2' : ABORT
Aborting globally since one ABORT received!
ABORTED !!!
```

## Participant 1: Commits



```
Run: Coordinator x Participant x Participant x
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/C...
/Users/chinmaymandale/Projects/Distributed Systems...

Participant Name:
p1

p1 has entered the 2PC protocol connection

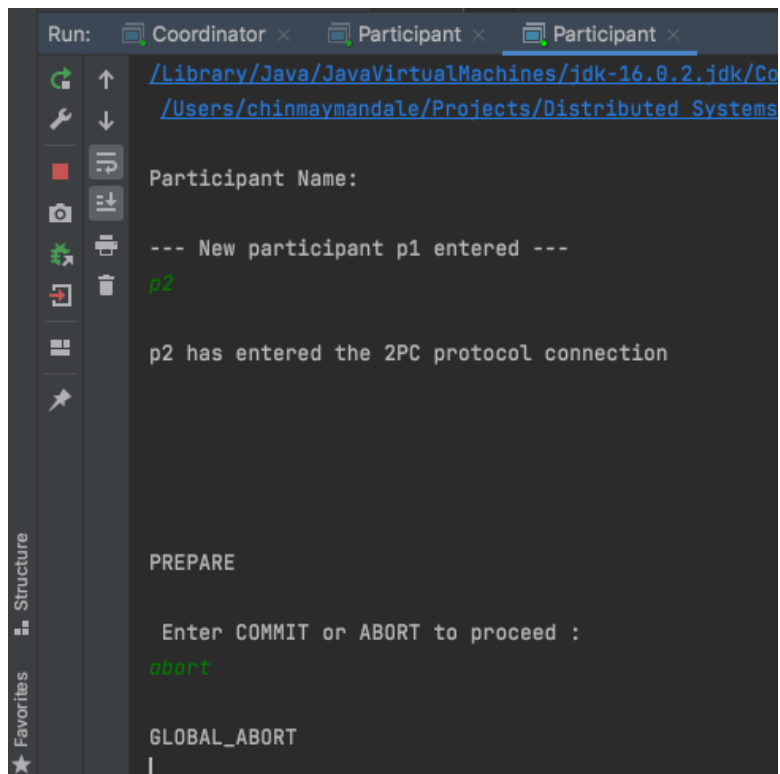
PREPARE

Enter COMMIT or ABORT to proceed :

--- New participant p2 entered ---
commit

GLOBAL_ABORT
```

## Participant 2: Aborts



```
Run: Coordinator x Participant x Participant x
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/C...
/Users/chinmaymandale/Projects/Distributed Systems...

Participant Name:

--- New participant p1 entered ---
p2

p2 has entered the 2PC protocol connection

PREPARE

Enter COMMIT or ABORT to proceed :
abort

GLOBAL_ABORT
```

## Things we learned:

1. Two-phase distributed commit protocol
2. Multithreading in distributed systems
3. This protocol can be used in similar transactions like database

## Problems we faced:

1. Socket failure in client thread
2. Implementation issues for node failure
3. Implementing time out for sockets in Java
4. Managing thread implementation using java.