

# Distributed Systems

*CSE 5306-003 PROJECT REPORT-3*

---

## TWO-PHASE DISTRIBUTED COMMIT PROTOCOL

---

### **Submitted by**

**Koushik Kondepati (1002069104)**

**Sai Narayana Shanmukh Ayyagari (1002063350)**

### **Instructor:**

Md Hasanuzzaman Noor

## DECLARATION

I hereby declare that I carried out the work reported in this report in the Department of Computer Science and Engineering, The University of Texas at Arlington, under the guidance of Mr. Md Hasanuzzaman Noor. I solemnly declare to the best of my knowledge that I have neither given nor received unauthorized assistance on this work.

Sign1: Koushik Kondepoti

Sign2: Sai Narayana Shanmukh Ayyagari

Date: November 1st, 2022

### **Project Members**

Koushik Kondepoti (1002069104)

Sai Narayana Shanmukh Ayyagari (1002063350)

## Summary:

- 1) Project Brief
- 2) Program Implementation
- 3) Learning Outcomes
- 4) Issues Encountered
- 5) Roles and Responsibilities

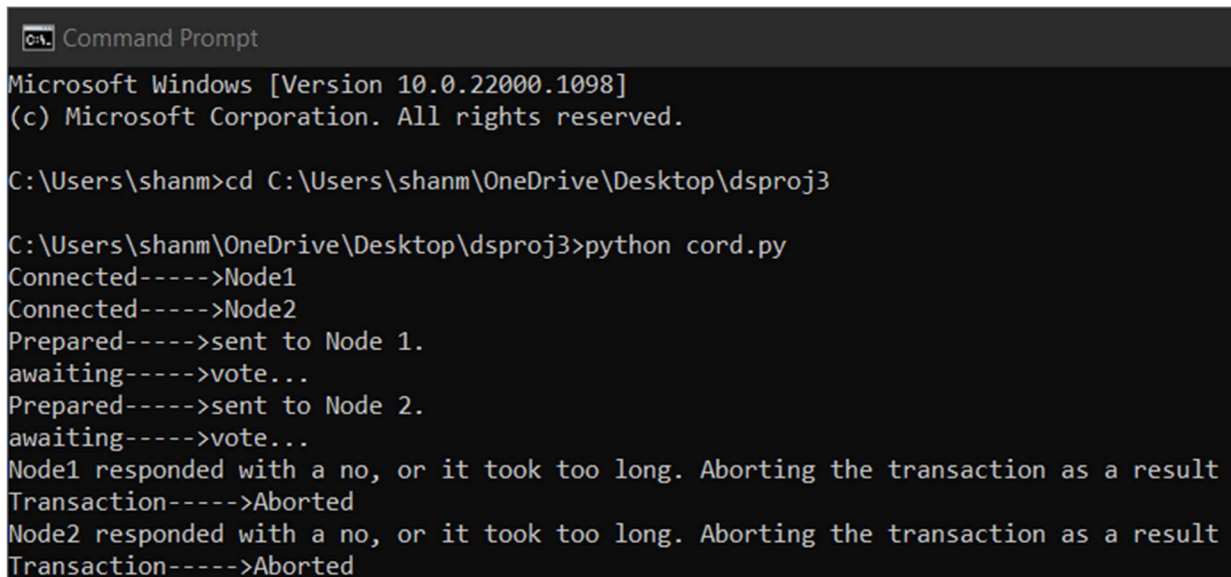
## Project Brief:

- The given project is based on how to establish the Implement a 2-phase distributed commit(2PC) protocol and use controlled and randomly injected failures to study how the 2PC protocol handles node crashes. Assume one transaction coordinator (TC) and at least two participants in the 2PC protocol. Similar to the previous projects, we use multiple processes to emulate multiple nodes. Each node (both the TC and the participants) devises a time-out mechanism when no response is received and transits to either the abort or commit state. For simplicity, you can assume that only one node fails in the controlled test.
- TC failure : If the coordinator fails before sending the "prepare" message, nodes will not receive the "prepare" message until the time-out and will abort. So, they will respond "no" to the "prepare" message after the coordinator comes back up and sends the "prepare" message.
- Node failure : If the transaction coordinator does not receive "yes" from a node, it will abort the transaction.
- TC failure : TC needs to store the transaction information on disk before sending the "commit" message to the nodes. If the TC fails after sending one "commit" message to the nodes, it can't abort. When it comes back up it will send the "commit" message to the nodes that it didn't send the "commit" message to.
- Node failure : A node needs to store the transaction information before replying "yes" to the TC. If it fails (time-out) after replying "yes", after it comes back up, it will fetch the commit information from the TC for that particular transaction.

## Program Implementation:

- The entire project is written by us in PYTHON programming language. This project consists of three programs which are node1 , node2 and co-ordinator.py.
- We created these three separate python files, one for coordinator code and the other two for nodes code implementation. So that the user can run both simultaneously and can check in the terminal.
- To implement the project using RPC we need to download 2-3 additional RPC libraries, for example xmlrpc jar files which are freely available inside PYTHON platform. These rpc dependency libraries play a key role for client server communication.

### 1-TC FAILURE



```
Command Prompt
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shanm>cd C:\Users\shanm\OneDrive\Desktop\dsproj3

C:\Users\shanm\OneDrive\Desktop\dsproj3>python cord.py
Connected----->Node1
Connected----->Node2
Prepared----->sent to Node 1.
awaiting----->vote...
Prepared----->sent to Node 2.
awaiting----->vote...
Node1 responded with a no, or it took too long. Aborting the transaction as a result
Transaction----->Aborted
Node2 responded with a no, or it took too long. Aborting the transaction as a result
Transaction----->Aborted
```

## 2-NODE FAILURE

### NODE1 CODE

```
Command Prompt - python n1.py

C:\Users\shanm\OneDrive\Desktop\dsproj3>python n1.py
Node1----->Active
linked----->coordinator
Preparation----->Awaiting...
127.0.0.1 - - [01/Nov/2022 22:54:00] "POST /RPC2 HTTP/1.1" 200 -
received----->preparation
Sent Yes after----->11.006381750106812 seconds
Voting took far too long... Transaction----->Aborting.
127.0.0.1 - - [01/Nov/2022 22:54:16] "POST /RPC2 HTTP/1.1" 200 -
```

### NODE2 CODE

```
Command Prompt - python n2.py

C:\Users\shanm\OneDrive\Desktop\dsproj3>python n2.py
Node2----->Active
linked----->coordinator
Preparation----->Awaiting...
127.0.0.1 - - [01/Nov/2022 22:54:02] "POST /RPC2 HTTP/1.1" 200 -
received----->preparation
Sent Yes after----->1.0148890018463135 seconds
awaiting----->Commit...
127.0.0.1 - - [01/Nov/2022 22:54:16] "POST /RPC2 HTTP/1.1" 200 -
```

## BONUS-1 CASE:

### TC CHECKING WITH LOG FILES

```
Connected----->Node101-11-2022 22:10:45
Connected----->Node201-11-2022 22:10:45
Prepared----->sent to Node 1.01-11-2022 22:10:50
awaiting----->vote...01-11-2022 22:10:50
Prepared----->sent to Node 2.01-11-2022 22:10:53
awaiting----->vote...01-11-2022 22:10:53
Obtaining----->yes from Node101-11-2022 22:10:56
Obtaining----->yes from Node101-11-2022 22:10:56
Proceeding----->Transaction01-11-2022 22:10:56
FinalizeCommit----->received from node1 01-11-2022 22:10:59
sent commit----->Node1, so sending node201-11-2022 22:11:10
FinalizeCommit----->received from node2 01-11-2022 22:11:13
Transaction----->Successful01-11-2022 22:11:13
```

## BONUS-2 CASE:

### NODE CHECKING WITH LOG FILES

#### NODE1 LOG

```
linked----->coordinator01-11-2022 22:16:19
Preparation----->Awaiting...01-11-2022 22:16:19
Commit----->Received01-11-2022 22:16:30
slept too much after the request01-11-2022 22:16:41
Connected----->Node101-11-2022 22:16:17
Connected----->Node201-11-2022 22:16:17
Prepared----->sent to Node 1.01-11-2022 22:16:22
awaiting----->vote...01-11-2022 22:16:22
Prepared----->sent to Node 2.01-11-2022 22:16:25
awaiting----->vote...01-11-2022 22:16:25
Obtaining----->yes from Node101-11-2022 22:16:28
Obtaining----->yes from Node101-11-2022 22:16:28
Proceeding----->Transaction01-11-2022 22:16:28
FinalizeCommit----->received from node1 01-11-2022 22:16:41
sent commit----->Node1, so sending node201-11-2022 22:16:42
FinalizeCommit----->received from node2 01-11-2022 22:16:55
Transaction----->Successful01-11-2022 22:16:55
I waited too long to send the commit, thus I'm asking TC for transaction details.01-11-2022 22:16:57
-----Begin----->transaction information-----01-11-2022 22:16:57
-----End----->transaction information-----01-11-2022 22:16:57
Transaction----->Successful01-11-2022 22:16:57
```

#### NODE2 LOG

```
linked----->coordinator01-11-2022 22:16:21
Preparation----->Awaiting...01-11-2022 22:16:21
Commit----->Received01-11-2022 22:16:44
slept too much after the request01-11-2022 22:16:55
Connected----->Node101-11-2022 22:16:17
Connected----->Node201-11-2022 22:16:17
Prepared----->sent to Node 1.01-11-2022 22:16:22
awaiting----->vote...01-11-2022 22:16:22
Prepared----->sent to Node 2.01-11-2022 22:16:25
awaiting----->vote...01-11-2022 22:16:25
Obtaining----->yes from Node101-11-2022 22:16:28
Obtaining----->yes from Node101-11-2022 22:16:28
Proceeding----->Transaction01-11-2022 22:16:28
FinalizeCommit----->received from node1 01-11-2022 22:16:41
sent commit----->Node1, so sending node201-11-2022 22:16:42
FinalizeCommit----->received from node2 01-11-2022 22:16:55
Transaction----->Successful01-11-2022 22:16:55
I waited too long to send the commit, thus I'm asking TC for transaction details.01-11-2022 22:16:59
-----Begin----->transaction information-----01-11-2022 22:16:59
-----End----->transaction information-----01-11-2022 22:16:59
Transaction----->Successful01-11-2022 22:16:59
```

## Learning Outcomes:

- We understood on how to implement the two-phased distributed protocol by using transaction coordinator and two processes.
- First our transaction coordinator sends the prepare message to the two nodes and if the two nodes receives the message then they will send the yes message which means the transaction is successful.
- If the transaction coordinator fails to send the prepare message then the two nodes will wait for the tc and once the tc sends the prepare message after sleep then the transaction is aborted which results in tc failure.
- If any one of the nodes doesnot receive the prepare message because of one of the node timeout then that means node failure case.
- in the bonus section the tc at one point of time coordinator sleeps for some time and again listens to the log files compares and then sends the commit message to that particular nodes. And the transaction is successful, if not transaction is aborted.
- In the bonus second question node fails to transit to the transaction coordinator because of time-sleepout then it compares with the transaction coordinator log files and then sends the yes message which means the transaction is completed , if not transaction is aborted.
- Our transaction coordinator sends the prepare message to the two nodes and if the two nodes receives the message then they will send the yes message which means the transaction is successful.



## Challenges we encountered:

- We initially ran into some package errors while building the project, but we later achieved our first milestone after installing the correct XMLRPC package.
- The second biggest challenge is dealing with the time-out and implementation of TC and two other processes and which we are calling them as nodes ( node1 and node2 ).

## Roles and Responsibilities:

- We first categorized the project into two parts, first is implementing the crud functionalities such as download, upload, delete and rename including add and sort functionalities
- Secondly, we concentrated on establishing the client server communication using XMLRPC package by downloading some additional jar files