Commit Protocols and their Issues in Distributed Databases

Vinay Sharma¹, Prof. Vijay B. Aggarwal²

¹MCA-III yr, Jagan Institute of Management Studies, GGSIP University, Delhi
²Professor, Jagan Institute of Management Studies, GGSIP University, Delhi

ABSTRACT
A number of protocols for complete and reliable execution of Distributed Transactions have been devised so far out of which the 2 phase commit protocol, 3 phase commit protocol and Quorum based protocols are quite famous. In this paper we study the performance issues of these commit protocols.

The 2-Phase commit protocol is a blocking protocol i.e. the issue of Coordinator failure is not addressed by the protocol in a dignified and efficient manner. In the case of Coordinator failure, in 2-Phase commit protocol, each participant is waiting for the coordinator to recover and then tell the participants about the global decision being taken by it about the ultimate fate of the transaction which was in process when the coordinator failed [3].

Though some circumstances permit the protocol to elect a new participant and proceed with the protocol, but a new coordinator can be elected only in case when all participants know clearly that only the coordinator site has failed and no participant site has failed. [1]

On the contrary, the 3-Phase commit protocol is known as non-blocking protocol as the issue of Coordinator failure is being handled by the protocol in a much reliable manner. Though the statement “Non-blocking” confines to the scenario where only the site failure is happening and not the coordinator failure i.e. the 3-Phase commit protocol is non-blocking when failures are restricted to site failures only.

The Quorum based protocol, however, is more reliable in serious failures like Network Partitioning (simple partitions) in distributed databases. This Protocol works similar to 3-Phase commit protocol in normal processing, whereas in case of Network Partitioning the protocol determines the ultimate fate of the transaction by voting i.e. quorums. The major problem being faced by this protocol is that the protocol only works for simple partitioning of the Network i.e. the Network is divided in to two components.

Apart from all of the above discussion, the three protocols have various issues with their execution. We give a brief overview of these issues in commit protocols.

KEYWORDS
Transaction, Commit, Protocols, 2-Phase commit, 3-Phase commit, Quorum, Coordinator, Participant, site, failure, performance, network partitioning, 2PC, 3PC.

1. INTRODUCTION
A Transaction, as defined by [Skeen], is an atomic operation on distributed database system. Either all changes by the transaction are permanently installed in the database, in which case the transaction is said to be committed, or no changes persist, in which case the transaction is said to be aborted.

It is the task of a commit protocol to ensure that a Transaction is atomically executed.

Transaction Processing is a core and integral part of Transaction Management in any Database Management System, but it attracts more interest and becomes more complex and essential when we discuss this topic in Distributed databases environment. Transaction Management includes the most important and inevitable part which is known as RELIABILITY of Transactions in Distributed databases.

A ‘RELIABLE’ Distributed DBMS can be defined as one which should be able to continue executing user-requests without violating database consistency even if the components of the distributed computing environment fail.

Reliability refers to the ‘Atomicity’ & ‘Durability’ characteristics of a Transaction. Two specific aspects of Reliability are ‘COMMIT’ & ‘RECOVERY’ Protocols.

In our study, we primarily focus over various commit protocols already available for the fulfillment of reliability feature of a distributed database system.

The various commit protocols available and employed in today’s distributed database networks are:
- 2 Phase commit protocol
- 3 Phase commit protocol
- 3 Phase Quorum based protocol
- Replica Control protocol
- Lazy Replication protocol

In this article we discuss various merits and demerits of these protocols.

The above mentioned protocols can be divided in 2 major categories as:
1. Commit protocols for Non-replicated databases
2. Replicated database commit protocols

The last two of the mentioned protocols fall under the second category, i.e.
- Replica Control protocol
- Lazy Replication protocol

In our study we focus only on Non-Replicated databases in general. We tend to figure out the problem areas or the issues with these commit protocols and also find out the situations where a specific commit protocol would fail to execute or would not be appropriate to execute in some distributed databases access scenarios in Real-time.

In this article we not only try to find the problems or issues of these commit protocols in distributed scenarios but we
also tend to formulate some key points or features which can become the basis for further research to find out some new and better commit protocols which could probably be resilient to the issues and problems faced by the currently available protocols and come up with some valid and acceptable solutions to the problems of real-time distributed environments.

1.1 TERMINOLGY:  
In the text some of the terms are frequently used, we first define these terms (Fig.1):  

**Site:** In a distributed database system, the database is fragmented over a number of places which work together by means of network communication. These places where a part/fragment of database is stored is called site.  

**Participant site:** Out of the available sites, there may be a case when a Transaction need the data only from certain sites and not from all of them, then in that case the sites executing the sub-Transaction are known as Participating sites.  

**Coordinator site:** Out of the Participant sites one of the sites from where the Transaction initiated is usually termed as the Coordinator site. This is the site from where the user actually fired a Query for data retrieval or updation.  

![Fig.1 A distributed database environment](image)

1.2 APPROACH:  
A commit protocol can be conveniently described and understood by a set of state-transition-diagrams, one for each participating site as well as the coordinator site. The diagram for Site $i$ describe the processing of sub-transaction $T_i$.  

2. THE 2-PHASE COMMIT PROTOCOL  
The two phase commit protocol exhibit a simple protocol which ease the transaction processing in distributed database system.  

Initially the coordinator writes a “begin-commit” record in its log and sends a PREPARE message to all the participant sites, and enters a WAIT state. When a participant receives a PREPARE message, it checks whether it can commit the transaction. If yes, then it writes a ready record in its log and sends VOTE-COMMIT message to the coordinator and enters READY state; otherwise the participant writes an abort record and sends a VOTE-ABORT message. This abort decision serves as the GLOBAL-ABORT decision and is known as UNILATERAL-ABORT. After the coordinator has received responses from each of the participants, it then decides to COMMIT or ABORT the transaction based upon the responses of the participants. If even one of the participants has registered a negative vote (VOTE-ABORT), the coordinator has to abort the transaction globally. So it writes an abort record in log and sends a GLOBAL-ABORT signal to all the participants and enters the ABORT state; otherwise, it writes a commit record and sends a GLOBAL-COMMIT signal to all the participants and enters COMMIT state. Upon receiving the global decision from the coordinator the participants either commit or abort the sub-transaction as directed by the coordinator and send back an acknowledgement.  

![Fig. 2 State Transition Diagram for 2-Phase commit protocol](image)

The **THE GLOBAL COMMIT RULE:** This is a set of rules which govern the final decision by the coordinator: [1]  
1. If any (even one) of the participant has voted VOTE-ABORT then the transaction has to be globally aborted.  
2. If all the participants have voted VOTE-COMMIT then the transaction has to be globally committed.  

The communication structure being used in this Centralized 2-Phase commit protocol is known as Centralized 2PC Communication Structure, since the communication is only between the coordinator and the participants; the participants do not communicate among themselves.  

2.1 Termination in 2-Phase commit protocol  
The termination protocol caters to the timeout situations for both the participant as well as coordinator processes. Dealing of specific timeout condition depends upon two factors:  
1. Timings of failures.  
2. Types of failures.  

We discuss the Timeout of coordinator as well as participants in brief:
2.1.1 Coordinator Timeouts
In case of the coordinator, the timeout can occur in three states of Fig. 2. These states are WAIT, COMMIT, and ABORT.

Coordinator Time out in WAIT state: Since here the coordinator is waiting for participants to respond with their vote. The coordinator cannot itself commit the transaction as it would violate Global Commit Rule, but it can decide to globally abort the transaction in which case it writes an “abort” record in the log and signals all the participants to GLOBAL-ABORT.

Coordinator Time out in COMMIT state: Since this is the final state in State Transition Diagram (Fig.2) and the coordinator is not sure that all local participants have finished with their commit procedure or not, therefore the coordinator repeatedly sends GLOBAL-COMMIT message to participants who have not responded to gather their acknowledgement.

Coordinator Time out in ABORT state: Since this is also the final state in State Transition Diagram (Fig.2) therefore it would be handled in same manner as above case.

2.1.2 Participant Timeouts
Participants can either Timeout in INITIAL or in READY state as in Fig. 2.

Participant Timeout in INITIAL state: In this case the participants were waiting for PREPARE message from coordinator therefore the participant can unilaterally abort the transaction. If it receives PREPARE message after this then either it sends VOTE-ABORT or it can simply ignore the message.

Participant Timeout in READY state: A participant would be in this state only if it has voted VOTE-COMMIT in earlier state, and now the participant is waiting for the global decision taken by the coordinator. The participant neither can unilaterally abort the transaction as it has first voted to commit, nor it can unilaterally commit the transaction as some other participant may have voted VOTE-ABORT. Thus the participant remains blocked until it gets to know the ultimate fate of the transaction.

2.2 Inherent Blocking nature of 2-Phase commit protocol
As seen in above case when participant Timeouts in READY state the protocol shows the blocking nature. This problem basically occurs when the participant is waiting for coordinator’s final verdict about the fate of the transaction but the coordinator fails. One of the legitimate solutions to this problem seems to be ‘Selecting a new Coordinator’. This can be done by a simple process of voting between the sites and the coordinator. Thus this will be same as participant timeout in READY state.

2.3 Recovery Protocols in 2-Phase commit protocol
Recovery protocols refer to the set of actions performed by the sites, to recover their state, when these sites restart after a failure. We discuss Recovery protocols for coordinator and participant sites in brief:

2.3.1 Coordinator Failure
A coordinator may fail in one of the following states as referred in the Fig.2.

IN INITIAL STATE: Upon recovery the coordinator can only restart the commit process over the transaction.

IN WAIT STATE: Here again the coordinator will restart the commit process after recovery by sending PREPARE message to all the participants.

IN COMMIT/ABORT STATE: Since the commit process was almost completed prior to failure, thus after recovery the coordinator does not need to do anything if all the acknowledgments have been received.

2.3.2 Participant Site Failures
A participant may fail in one of the following states as referred in the Fig.2.

IN INITIAL STATE: Upon recovery the participant must unilaterally abort the transaction.

IN READY STATE: Here the participant has already affirmed the coordinator about its decision, therefore upon recovery the participant needs to know the global decision being taken by the coordinator. Thus this will be same as participant timeout in READY state.

IN COMMIT/ABORT STATE: Upon recovery the participant need not do any special action as these are the final states in the state transition diagram.

2.4 Summarizing the Issues in 2-Phase commit protocol
From the above discussion we figure out following problem/issues with the 2-Phase commit protocol:

1. Blocking Problem.
2. Assumptions:
   2.1 Combination of writing a record in the log and sending a message is assumed to be atomic
   2.2 The state transition is assumed to occur after the transmission of response message.

3. THE 3-PHASE COMMIT PROTOCOL
The major drawback of 2-Phase commit protocol, i.e. the Blocking problem, gave rise to invention of a new, reliable and modest commit protocol which exhibit non-blocking nature - The 3-Phase commit protocol.

Before we discuss the 3-Phase commit protocol we define the conditions for a protocol to be Non-Blocking in nature.
A commit protocol is non-blocking if and only if its state transition diagram contains neither of the following:

1. No state that is “adjacent” to both a commit and an abort state.
2. No non-committable state that is “adjacent” to a commit state.

To employ the above said conditions in 3-Phase protocol we need to add a new state in the state transition diagram (Fig.3). This new state would depict the situation when the process is ready to commit but has not yet committed.

Initially the coordinator writes a “begin-commit” record in its log and sends a PREPARE message to all the participant sites, and enters a WAIT state. When a participant receives a PREPARE message, it checks whether it can commit the transaction. If yes, then it writes a ready record in its log and sends a VOTE-COMMIT message to the coordinator and enters READY state; otherwise the participant writes an abort record and sends a VOTE-ABORT message (UNILATERAL-ABORT). After the coordinator has received responses from each of the participants, it then decides to COMMIT or ABORT the transaction based upon the responses of the participants. If even one of the participants has registered a negative vote (VOTE-ABORT), the coordinator has to abort the transaction globally. So it writes an abort record in log and signals all the participants to GLOBAL-ABORT.

3.1 Termination in 3-Phase commit protocol

We discuss the Timeout of coordinator as well as participants in brief:

3.1.1 Coordinator Timeouts

In case of the coordinator, the timeout can occur in three states of Fig. e. These states are WAIT, PRE-COMMIT, COMMIT, and ABORT.

Coordinator Time out in WAIT state: Since here the coordinator is waiting for participants to respond with their vote. The coordinator cannot itself commit the transaction as it would violate Global Commit Rule, but it can decide to globally abort the transaction in which case it writes an “abort” record in the log and signals all the participants to GLOBAL-ABORT.

Coordinator Time out in PRE-COMMIT state: In this state the coordinator can not decide if all the participants are in PRE-COMMIT state. But this is sure that they are at least in the READY state i.e. they must have voted to commit the transaction. Thus the coordinator can move all the participants to PRE-COMMIT state by sending a “prepare-to-commit” signal. Also the coordinator can now safely commit the transaction globally by sending a GLOBAL-COMMIT message to all.

Coordinator Time out in COMMIT state: Since the coordinator does not actually know whether all the participants have performed the commit process. But they are at least in the PRE-COMMIT state. Thus the coordinator can proceed as in the above case.

Coordinator Time out in ABORT state: Since the coordinator does not actually know whether all the participants have performed the abort process. But they are at least in the PRE-COMMIT state. Thus the coordinator can proceed as in the above case.

3.1.2 Participant Timeouts

Participants can either Timeout in INITIAL, READY or PRE-COMMIT state as in Fig. 3

Participant Timeout in INITIAL state: Here the participant was waiting for PREPARE message from coordinator therefore the participant can unilaterally abort the transaction. If it receives PREPARE message after this then either it sends VOTE-ABORT or it can simply ignore the message.

Participant Timeout in READY state: A participant would be in this state only if it has voted VOTE-COMMIT in earlier
state, and now the participant is waiting for the global decision taken by the coordinator. The termination protocol proceeds by choosing (electing) a new coordinator. The new coordinator terminates the transaction.

**Participant Timeout in PRE-COMMIT state:** A participant would be in this state only if it has received PREPARE-TO-COMMIT from the coordinator and is waiting for the final decision from the coordinator thus this case can be addressed same as above case.

### 3.2 Recovery Protocols in 3-Phase commit protocol

Recovery protocols refer to the set of actions performed by the sites, to recover their state, when these sites restart after a failure. We discuss Recovery protocols for coordinator and participant sites in brief:

#### 3.2.1 Coordinator Failure

**IN INITIAL STATE:** Upon recovery the coordinator can only restart the commit process over the transaction.

**IN WAIT STATE:** Here the participants have already terminated the transaction, therefore upon recovery, the coordinator has to ask around the fate of the transaction. [1]

**IN PRECOMMIT STATE:** similarly as in the above case the participants have already terminated their transactions, therefore coordinator has to ask round the fate of the transaction.

**IN COMMIT/ABORT STATE:** Since the commit process was almost completed prior to failure, thus after recovery the coordinator does not need to do anything if all the acknowledgments have been received.

#### 3.2.2 Participant Site Failures

A participant may fail in one of the following states as referred in the Fig.3.

**IN INITIAL STATE:** Upon recovery the participant must unilaterally abort the transaction.

**IN READY STATE:** Here the participant has already affirmed the coordinator about its decision, therefore upon recovery the participant needs to know the global decision being taken by the coordinator. Thus this will be same as participant timeout in READY state.

**IN PRE-COMMIT STATE:** Upon recovery the participant need to ask around to determine how the other participants have terminated the transaction.

**IN COMMIT/ABORT STATE:** Upon recovery the participant need not do any special action as these are the final states in the state transition diagram.

### 3.3 Summarizing the Issues in 3-Phase commit protocol

From the above discussion we figure out following problem/issues with the 3-Phase commit protocol:

1. Non-Blocking property has been achieved over the price that fewer cases of independent recovery are possible.[1]

2. This also results in more messages being exchanged during recovery.[1]

Assumptions: (1) Combination of writing a record in the log and sending a message is assumed to be atomic. (2) The state transition occurs after the transmission of response message.

### 4. NETWORK PARTITIONING

Network Partitioning refers to the splitting of the participant sites in distributed database architecture with formation of clusters of one or more site. These clusters have lost the connection between them i.e. there is no way to communicate from one site in one cluster to some other site in other cluster. The network partitioning can be viewed diagrammatically as:

![Fig. 4. Simple Network Partition](image)

In the case of Network Partitioning, the major concern is with the termination of transactions that were active at the time of partitioning. [1]

### 5. QUORUM BASED 3-PHASE COMMIT PROTOCOL

The fundamental idea is that a transaction is executed if a majority of the sites vote to execute it. This idea of voting has been generalized to voting with quorums [1].

Skeen designed a more formal protocol based on this principle known as Quorum based 3-Phase commit protocol. Every site in the system is assigned a vote $V_i$. If the total number of votes in the system is $V$, commit quorum be $V_c$ and abort quorum be $V_a$. Then the following rules must be obeyed in the implementation of the commit protocol:

1. $V_a + V_c > V$, where $V_a \geq 0$, $V_c \leq V$
2. To abort a transaction, the abort quorum must be obtained.
3. To commit a transaction, the commit quorum must be obtained.

To get a clear understanding of the protocol let us take a real time example.

Assume a Distributed network contains 30 sites in all. After a simple network partition 2 partitions we formed one with 10 sites and other with 20 sites. Now we consider the second partition with 20 sites.

According to the formula stated by protocol

\[ V = 20 \]

And lets set the abort and commit quorums as 11 and 12, i.e.

\[ V_a = 11 \]
\[ V_c = 12 \]

Clearly it satisfy the condition (1) i.e. $V_a + V_c > V$ as,

\[ 11 + 12 = 23 > 20 \]

now while collecting votes, if 11 sites out of total of 20 sites voted to ABORT then it is confirmed that only 9 can vote to commit then in that case $V_a$ quorum is obtained and by condition (2) the transaction would be Globally Aborted.
Similarly, if 12 out of 20 sites vote to commit, then only 8 sites may have voted to abort, then $V_c$ quorum is obtained, and by condition (3) the transaction would be Globally committed.

In order to inculcate these rules, the 3-Phase commit protocol must be modified a bit in the third phase. In this protocol it is necessary that the participants make an explicit decision to join either the commit or the abort quorum and not change their votes afterward. Since the READY (or WAIT) state does not satisfy these requirements, Therefore another state, PRE-ABORT need to be included. The transition from the PRE-ABORT state to the ABORT state requires an abort quorum.[1]

5.1 Working of Quorum Protocol
With this modification, the termination protocol works as follows. Once a new coordinator is elected, it requests all participants to report their local states. Depending on the responses, it terminates the transaction as follows: [4]

1. If any one of the participants is in COMMIT state then the new coordinator commits the transaction and sends a Global commit signal to all the participants.
2. If any one of the participants is in ABORT state then the new coordinator decides to abort the transaction and sends a Global Abort signal to all the participants.
3. If the Commit quorum is obtained in PRE-COMMIT state then the new coordinator decides to commit the transaction and sends a Global commit signal to all the participants.
4. If the Abort quorum is obtained in PRE-ABORT state then the new coordinator decides to abort the transaction and sends a Global abort signal to all the participants.
5. If commit quorum is not obtained but sum of the votes of the participants in PRE-COMMIT and READY state are enough to form a commit quorum then new coordinator moves to participants to the PRE-COMMIT state by sending a PREPARE-TO-COMMIT signal. Now the coordinator waits for a commit quorum to be obtained.
6. similarly, If abort quorum is not obtained but sum of the votes of the participants in PRE-ABORT and READY state are enough to form a abort quorum then new coordinator moves to participants to the PRE-ABORT state by sending a PREPARE-TO-ABORT signal. Now the coordinator waits for an abort quorum to be obtained.

5.2 Summarizing the Issues in Quorum based 3-Phase commit protocol
From the above discussion we figure out following problem/issues with the Quorum based 3-Phase commit protocol:

1. The protocol is resilient to simple network portioning but is not resilient to multiple network partitions.
2. It is a Blocking protocol. The coordinator in a partition may not be able to form either abort or commit quorum.
3. If number of failures is very few, then this protocol requires 3 phases, 5 end to end message delays and about 5N messages (where N is number of participants). This cost is substantially higher than cost of 2-Phase commit protocol by approx 50%

CONCLUSION
In this paper we study various commit protocols for distributed databases. Out of which, we primarily focused on 2-Phase, 3-Phase, and Quorum based 3-Phase commit protocols. We discussed their working and find out the issues and problems these protocols are not able to cater to. We summarize, at the end of each section, the issues involved in each of these protocols. We cannot judge as to which protocol is better in all terms. No protocol is ideal to use. Even the best out of these protocols, the quorum based 3-Phase commit protocol, have certain drawbacks and performance issues.

FUTURE SCOPE
This paper would probably be helpful in performance evaluation of the various commit protocols and devising some new protocols which could overcome the problems/issues being faced, but not resolved, by these protocols in distributed database environments.

REFERENCES