[**Transaction Management in the R\* Distributed Database Management System**](https://classroom.google.com/u/1/c/Nzc1MzI5NjAyOVpa/a/ODk0MDE3MDAyMVpa/details)

Two phase commit protocol (2PC) is a form of atomic commit that uses a coordinator to decide about committing or aborting a transaction. The main focus of this paper is to compare the 2PC with presumed abort (PA) and presumed commit (PC). They differ in how they handle failure conditions. In the rest of the paper look at the protocols being used and how they are being used and on the commits and aborts.

Two-phase commit protocol divides each transaction into two phases. The first phase is the prepare phase, where each node participating in the transaction is sent a prepare message. Nodes respond to this message with a YES vote or NO vote stating if they are willing to commit the transaction. If all node votes are YES votes, the commit phase of the protocol is started. The coordinator first writes a commit log and then sends a commit message to each node participating in the transaction. Upon receipt of a commit message from the coordinator, each subordinate node commits the transaction by first writing to their log and then updating any data from the transaction, finally the subordinate node sends an ACK to the coordinator. If one or more nodes voted NO, then the coordinator writes an abort log and sends abort messages to each node. Upon receipt of an abort message each node writes an abort log, releases any locks held, and sends an ACK to the coordinator. After the coordinator receives ACK messages from each node, it writes an end log and can remove any references to the transaction from memory.

The coordinator sends a PREPARE message to each subordinate in the transaction to start. Subordinate receiving a prepare message and voting to participate in the transaction. To participate, it writes a prepare log to durable storage and sends a YES message to the coordinator. For a subordinate opting not to participate in the transaction. In this case, the subordinate writes an abort log to durable storage and sends a NO message to the coordinate. The subordinate can to release all locks and forget any state about the transaction, it will never participate in the transaction after sending a NO vote. After the coordinator has received at least one NO vote and must abort the transaction. When all subordinate vote Yes, transaction is committed.

For handling failure, it is assumed that a recovery process is running at each node in the cluster. When a failure occurs the process reads the local write-ahead-log and accumulates information about any running transactions in progress at the time of failure. If the recovery process on a coordinator node finds a transaction in the committing or aborting phase, the coordinator resends COMMIT or ABORT messages to any subordinates that have not acknowledged the transaction. If the recovery process on a subordinate node finds a transaction in the prepare phase, it periodically tries to contact the coordinator to find out how to resolve the transaction. Once the subordinate knows how to resolve the transaction it proceeds as normal.

**Conclusion:**

The system extended two-phase commit protocol by Presumed Abort (PA) and Presumed Commit (PC), to reduce message traffic among data sites and log writes for a shorter response time. Presumed abort outperforms the standard two-phase commit protocol. The paper compare presumed abort and presumed commit but the comparison is not that clear. For transactions with updated presumed commit requires less ACK messages than presumed abort however if transaction is read-only.

**Questions:**

**1 -** How 2PC will handle the situations where multilevel trees of processes occur?

**2 -** How system reacts when the user application decides to commit a transaction but the coordinator receives a commit transaction command?

**3 -** In case of transaction abort situation how the system reacts if coordinator writes the abort log and didn’t sends an ABORT message to the subordinate?

**4 -** Will transaction be considered complete if after receiving all the ACK messages and coordinator doesn’t writes an end log signaling?