19.1 Recap of Fault Tolerance

19.1.1 Two Replication techniques to handle fault tolerance

19.1.1.1 Technique 1

Requests are distributed amongst k replicas. In case of failure of a replica, the requests are redistributed. This technique handles crash fault tolerance. It is a simple and the commonly used technique to handle faults.

19.1.1.2 Technique 2

Each request is sent to all the replicas. All replicas process requests and produce results. Then the replicas vote to make a decision. This handles Crash and Byzantine Faults. However it is much harder and more expensive (3k replicas) to implement.

19.1.2 Two Phase Commit (2PC) and Three Phase Commit (3PC)

In Two Phase Commit, the first phase includes the Coordinator querying all the database replicas on whether a transaction has to be committed or aborted. In the decision phase, the results from the replicas are used to make a decision. Even if a single replica wants to abort, the transaction is aborted. This ensures the safety property.

However the failure of the Coordinator blocks the system. In order to handle this blocking a technique called Three phase commit is used. The first two phase is similar to the 2PC. In the third phase, the coordinator tabulates the results and sends them to all the replicas. The replicas send an acknowledgement on receiving this message. Only after receiving the acknowledgement does the coordinator send a commit message.

If the coordinator crashes, the replicas ask each other if they have heard from the coordinator and even if a single replica has a tabulated result (in the precommit stage) the replicas can go ahead and commit. If none of the replicas have a result then the abort the transaction.

The state diagram in the slides shows the various messages passed. The 3PC is always safe irrespective of the failure of the coordinator or the replicas.
19.1.3 Paxos

Paxos is another technique for handling fault tolerance. Unlike the 2PC or the 3PC, Paxos looks at the results of the transaction and decides based on the result. A comparison of the results obtained by the different replicas make sure that the Byzantine Faults are taken care of. This is a complex technique and much harder to implement.

19.2 Recovery

Once the crashed server recovers, operations must be performed for it to recover to the current state. This is done by resynchronizing with the other servers. In case of databases, all replicas maintain logs and the replica which has just recovered can ask another replica for its log and do a log replay to get itself updated. Checkpointing is another means of recovery. The replicas maintain checkpoints and upon a crash it rolls back to its previous checkpoint and then does a log replay from that point.

Independent Checkpointing can lead to inconsistencies in that case the replicas has to rollback until the last consistent state. This cascading rollback can lead to a domino effect. However use of techniques like distributed snapshot solves the problem.