Two Phase Commit Protocol:
With the help of the two-phase commit technique, several distributed system nodes may commit atomic transactions. All nodes involved in the 2PC must simultaneously commit or abort in order for the transaction to meet the requirements for an atomic one. To start up 2PC, one of the nodes would serve as a coordinator, also known as a transaction manager.

This algorithm may be broken down into two stages prepare and commit. In the preparation phase, you will inquire with other nodes as to whether they can commit the proposed transaction. During the commit phase, other nodes are given the option to either commit to the proposed transaction or to cancel it. The 2PC technique may be used to bring all the nodes in a distributed system to consensus, which means that any given transaction will be accepted by all of the nodes. The processing of messages that are sent exactly once is an application of 2PC that is particularly fascinating.

Blocking atomic commit protocols like 2PC are commonly referred as due to the fact that they require all nodes and members to be online before they can complete their purpose (or a "anti-availability" protocol). If a coordinator were to die, the outcome would need to be sent to all nodes.

Distributed Commit Protocol:
The difficulty of atomic multicasting is illustrative of a larger one, called distributed commit. An operation must be performed by all members of a process group, or else it fails. For reliable multicasting to occur, the action of sending a message must be carried out. Coordinators set up distributed commit. The coordinator chooses whether to conduct the operation locally and informs the other processes (the participants). Using a one-phase commit procedure. The biggest drawback is that the coordinator will not know if a participant cannot complete the operation. Concurrency control can prevent a distributed transaction's local commit.

It's possible that one of the participants will have to block until the coordinator can recover. After all of the participants had received and processed the vote-request message, the coordinator suddenly crashed. Under those conditions, the parties cannot reach a consensus on the appropriate course of action. Because of this, the 2PC protocol is a blocking commit protocol.

Redundancy For Failure Masking:
Adding redundancy to a system's design is one way to make it more resilient to failure. In this approach, the system is replicated, and input is shared across several copies. The total output from all copies is used to determine this. The output generated by many of the copies is what is the actual result if one of the copies is faulty and produces a different output from the rest. This strategy can only address a single problem in the system at a time.

Consensus Under Arbitrary Failure Semantics:
For distributed systems to function, there must be several nodes working together. Having these nodes agree on anything is crucial for the system to work as intended. Distributed system consensus is unique because it addresses one of the most fundamental difficulties in distributed systems: getting everyone on the same page. Consensus is a fundamental principle of distributed system design.

CAP:
No distributed system is free from network failures, and network partitioning generally will be achieved. In the presence of a partition, has two choices as consistency or availability. When choosing consistency over availability, the system will return an error or a time out if particular information cannot be guaranteed to be up to date due to network partitioning**..** Since these two features are the most urgent concerns in distributed systems, the link between availability and consistency has been thoroughly investigated. In

large-scale dispersed contexts, their connection is exclusive, according to the well-known CAP theorem. For this reason, partition tolerance cannot be utilized as a parameter in such configurations.