CAP Theorem:

In order to achieve several things like Consistency, Availability and Partitioning for a distributed system all of them have tradeoffs with each other and are dependent on each other. Thus, trying to improve one characteristic would weaken the role to be played, thus we have to come to a common ground for achieving good throughputs and efficient solutions. It the same scenario with the distributed operating systems too.

Whenever the node is trying to read the context from other nodes or trying to fetch the information, we need to make a reliable and consistent connection such that the writes are correct and doesn't cause any errors. Any of the read operation that is requested form the node has to start after making the read operation. For a consistent system, the sender or producer has to make write operation before responding to the request that is made.

Availability, nodes that are the part of the distributed network should be up and running and should be available for the consumer nodes for making the requests to be processed. Similarly, with the partitioning tolerance, we need to maintain the tolerant system even if some of the messages sent by the node are lost and that can be recovered back and sent to the consumer node. All of this makes the distributed system and which impossible to be constructed in the real world.

Reliable remote procedure:

Whenever a program in the node is trying to make remote procedure call to another set of instructions for another machine thus mimicking the server-client model architecture where the client makes the procedure call, and the server does the work for the client and reports back to the client. Many advantages of RPC are that we can operate multiple RPCs parallelly and can speed up the extraction of the information. With the basic understanding, we had dived deep in to understand the various types of RPCs available as: Callback RPC, Broadcasting the RPC requests, Sending the requests in batch are utilized based on the scenario we wanted to work with.

Some of the benefits of using RPC, it helps us to develop the high-level language communication process which is easy to troubleshoot when an unexpected result occurs, and these are confined to the individual system thus makes to have less dependency with the other nodes, It supports threads for making it to derived parallelism and add an extra level of computation.

Redundancy:

With the above discussed features, we need to work on coming up with a solution for achieving the stable distributed system thus one such way is to maintain the redundant information shared across the network such that it can handle the effects discussed above. Some of the sample redundant techniques followed here are Information Redundancy, Time redundancy, Physical Redundancy. Each one of them deals with the specific requests that is dealt. With the Information Redundancy, we tend to capture things that are replicated and stored in several places such that the nodes can access the information when requested thus we can deal with the fault tolerant system such that if any node is down and the request will be redirected to another node that has information and we will be able to fetch it and send it across to the client. Time redundancy is where we will work on maintaining the time sensed information for each capture and it holds true for answering the requests if the entire server is down and the backup helps to recover the information and transmit across the distributed network.