Prepared by : Dr. Saurabh Bilgaiyan

Q11 (a) What different classes of failure can occur in RPC systems? [1mark]

Q11 (b) What do you mean by Orphans with respect to failures in RPC? [1mark]

Q11(c) How to deal with Orphans while a client crashes during RPC? [5marks]

Answer 11 (a)Five different classes of failure that can occur in RPC systems:

1. The client is unable to locate the server.
2. The request message from the client to the server is lost.
3. The reply message from the server to the client is lost.
4. The server crashes after receiving a request.
5. The client crashes after sending a request.

Answer 11(b)
if a client sends a request to a server to do some work and crashes before the server replies? At this point a computation is active and no parent is waiting for the result. Such an unwanted computation is called an orphan.

Answer 11 (c) :
In solution 1, before a client stub sends an RPC message, it makes a log entry telling what it is about to do. The log is kept on disk or some other medium that survives crashes. After a reboot, the log is checked and the orphan is explicitly killed off. This solution is called extermination. The disadvantage of this scheme is the horrendous expense of writing a disk record for every RPC. Furthermore, it may not even work, since orphans themselves may do RPCs, thus creating grandorphans or further descendants that are impossible to locate. Finally, the network may be partitioned, due to a failed gateway, making it impossible to kill them, even if they can be located.
In solution 2, called reincarnation, all these problems can be solved without the need to write disk records. The way it works is to divide time up into sequentially numbered epochs. When a client reboots, it broadcasts a message to all machines declaring the start of a new epoch. When such a broadcast comes in, all remote computations are killed. Of course, if the network is partitioned, some orphans may survive. However, when they report back, their replies will contain an obsolete epoch number, making them easy to detect.
Solution 3 is a variant on this idea, but less Draconian. It is called gentle reincarnation. When an epoch broadcast comes in, each machine checks to see if it has any remote computations, and if so, tries to locate their owner. Only if the owner cannot be found is the computation killed.
Solution 4, expiration, in which each RPC is given a standard amount of time, T, to do the job. If it cannot finish, it must explicitly ask for another quantum, which is a nuisance. On the other hand, if after a crash the server waits a time T before rebooting, all orphans are sure to be gone. The problem to be solved here is choosing a reasonable value of T in the face of RPCs with wildly differing requirements.