Extra

CS60002: Distributed Systems
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur
End Semester Examination, Spring 2016

Total Marks: 100 **Time: 3 Hours**

Answer ALL Questions

1. **(a)** Define masking and non-masking fault tolerant systems, with an example of each. (4)

   (b) Define the Interactive Consistency problem. Show how you can solve the interactive consistency problem if you are given an algorithm to solve the consensus problem. You cannot change the given algorithm for the consensus problem, you can only use it. (4 + 6)

   (c) Describe how the quorum consensus based concurrency control techniques work. (5)

   (d) Consider the 2-phase commit protocol. Clearly explain how the coordinator recovers if it has failed (i) before writing the commit record in the log but after receiving AGREED messages from all other processes, and (ii) after writing the commit record in the log and getting back some (but not all) acknowledgements from the other processes. (3 + 3)

2. **(a)** Define with examples linearizability and sequential consistency. Briefly discuss why sequential consistency is considered to be a good consistency model to adopt in many practical distributed systems. (6 + 4)

   (b) Consider a replicated system using active replicas that stores a large set of records with each record having a large number of fields. Read/write requests to a record can be for any subset of the fields of the record. The replicas are to be hosted in high end servers spread over a large geographical area, completely connected with low bandwidth links. Reads and writes to the system are expected to be occurring at a similar rate and quite frequently, though it has been seen that (i) when a record is written, usually only a small number of fields in it are modified at a time and (ii) each record is not updated very frequently (but since there is a large number of records, writes to some record keep on occurring in the system).

   It is needed that a read of any record should always get the result of the latest write to the record. It is also required that if no further writes occur to a record, all copies of the record should have the same values within a finite time. Propose a design for the system. Your answer should have the following in order (you will be penalized if you do not organize your design this way): (i) list your design goals (what would you attempt to minimize/reduce or maximize/increase etc.), (ii) list the design issues you have to address, (ii) for each issue listed, state what design choice you make, clearly justifying your reason for making the choice. You must not use a central node, and you should try to make your system efficient for a multiuser environment. (20)

3. **(a)** List the steps clearly when some number of bytes of a file is read in GFS (List the steps in sequence as Step 1, Step 2, Step 3…, do not just write long paragraphs). Give two reasons why the central master node of a cluster does not become a performance bottleneck in GFS. (7 + 3)

(b) In the context of DFS, describe the advantages and disadvantages of stateful vs. stateless service. (5)

(c) What kind of file access pattern and network environment is NFSv3 best suited for? Justify your answer briefly and clearly pointing out design decisions made in NFSv3 based on the above. (5)

4. Consider a synchronous, bidirectional ring in which a token circulates in one direction in a fault-free state. A node may use the token when it gets it if it needs to, else it sends it to the next node in the ring. Nodes may exhibit crash fault (so the ring may be broken) and recover (and the ring may be formed again), but links are reliable and never go down. It is also given that at most one node can be in the crashed state at any time, the failure of a node can be instantaneously detected by its two neighbors in the ring (do not worry how), message delays are zero, and a node does not crash when it has the token.

Design an algorithm so that the token will continue to circulate in some manner to all non-failed nodes even when a node has crashed. Your algorithm must ensure that after a crashed node recovers, eventually the token must circulate in one direction only (though the direction may be the same or opposite of the direction followed before the crash) if no further failures happen.

Assume that the nodes are numbered from 1 to n in the clockwise direction. Write the code for a node k clearly, stating the messages first and then the receive/send rules for the messages. You can assume the existence of a predicate *failed(i)* which returns (do not worry how) *true* if node i has crashed, and *false* if i is alive (so any neighbor of a node i can just check the value of *failed(i)* to see if it has crashed or not, no messages need to be sent for this) (15)

5. **(a)** Explain clearly (showing the content of relevant messages and their encryption) how an authenticator sent in a Kerberos message prevents a malicious attacker from replaying a stolen ticket for a service. (5)

(b) How is a server authenticated to a client in mutual authentication implemented in Kerberos (show relevant messages)? Clearly justify why the messages shown ensure server authentication. (5)