Distributed Algorithms January 15th 2009

Final Exam

Rachid Guerraoui

Time Limit: 3 hours

Instructions:

- This exam is closed book: no notes or cheat sheets are allowed.
- Write your name on *each* page of the exam.
- If you need additional paper, please ask the TA.
- Read through each problem before beginning.
- Partial credit will be awarded, so explain your thinking carefully.
- State clearly any additional assumptions that you use which are not stated in the question.

Good Luck!

Problem	Points	Score
Non-Blocking Atomic Commit	10	
Early Deciding Consensus	10	
Total Order Bcast	10	
Total	30	

Problem 1. (Non-Blocking Atomic Commit)

Consider a system of N processes that communicate using a best-effort-broadcast service BEB. Any number of the processes may fail by crashing.

Part 1.a. (2p) Provide the specification for Non-Blocking Atomic Commit (NBAC).

Part 1.b. (2p) If processes have access to a failure detector P, can you implement NBAC? What if they have access to a failure detector $\diamond P$?

Note: A yes/no answer is enough–you do not need to prove your answer.

Part 1.c. (2p) Provide the specification for Consensus and for Uniform Consensus.

Part 1.d. (4p) Let $?P$ be the failure detector specified as follows:

- Its interface consists of a single **checkCrash** event which processes can trigger, which returns either 0 or 1.
- It guarantees *anonymous completion*: if any process fails, then eventually **checkCrash** returns 1 at every process.
- It guarantees *anonymous accuracy*: **checkCrash** returns 1 only if some process has failed.

Assume that, in this system, processes have at their disposal (1) a uniform consensus service UniCons; and (2) a $?P$ failure detector. Give an algorithm that implements NBAC, and show that it is correct.

Problem 2. (Early-Deciding Consensus)

Assume a synchronous system with N processes, at most $t < N$ of which may fail by crashing. Assume that time is divided into rounds, and in each round, each correct process p_i takes the following steps:

- p_i sends a message
- p_i receives a set of messages
- p_i updates its state

The communication service provides the following guarantees:

- Integrity: if p_i receives a message m in round r, then m was sent in round r by some process p_j.
- Synchrony: if p_i does not fail before the end of some round r, and if p_i sends message m in round r, then every other non-failed process receives message m in round r.

Part 2.a. (2p) Let $N = 3$ and $t = 1$. Provide an algorithm that solves consensus in this system in 2 rounds of communication. Explain why your algorithm is correct.

Part 2.b. (3p) Given an execution E, let $f(E)$ be the number of processes that fail in that particular execution. Give an algorithm that solves consensus in the given model within time $(f(E) + 2)$, for any execution E. Note that $f(E) \leq t$, and even though processes know t in all executions, you cannot assume that $f(E)$ is known by your algorithm.

If you cannot give an algorithm that solves consensus in $f(E) + 2$ synchronous rounds, you can write down the algorithm that solves consensus in $t + 1$ rounds, for partial credit.

Part 2.c. (3p) Prove that the algorithm in part **2.b** is correct.

Part 2.d. (2p) Is it possible to devise an algorithm (for all values of N) such that if there are no failures, uniform consensus terminates in 1 round? Note that this algorithm has to tolerate failures.

Problem 3. (Total Order Broadcast with Omission Failures)

Consider a system of N processes that communicate using a best-effort-broadcast service BEB. Assume that the processes have access to an eventually perfect failure detector $\diamond P$.

We assume that processes can be *omission-faulty*: a process p is said to be omission-faulty if either (i) p crashes, taking no further steps or (ii) p is subject to message transmission failures, in that messages sent or received by p may be lost (and hence not delivered to other processes), and messages sent to p may be lost (and hence p may not receive messages sent by other processes).

Let t be the maximum number of processes that might fail in this way. We assume that a majority of processes are correct in this model, therefore $t < N/2$.

Finally, processes have access to a best-effort-broadcast service, which only guarantees eventual message delivery between correct pairs of processes.

Part 3.a. (2p) Provide the specification of uniform Total Order Broadcast.

Part 3.b. (2p) Write down the specification for Eventual Leader Election. Then give an algorithm that implements Eventual Leader Election in this model and show that it is correct.

Part 3.c. (3p) Give an algorithm for Uniform Consensus in this model.

Part 3.d. (3p) Give an algorithm for Total Order Broadcast in this model.