

# CS138 Homework Assignment 3

Due April 13, 2009

1. Slide XV-12 describes what happens both when a replica manager sends a gossip message and when it receives a gossip message. However, the last part of the pseudo code for receiving a gossip message (given below) is wrong (it might run forever). Please supply a corrected version.
  - while there exists request  $r$  in  $rm_b.log$  such that  $r.u.prev \leq rm_b.val.ts$ 
    - update  $val$  (by applying  $r.u.op$ )
    - $rm_b.val.ts = merge(rm_b.val.ts, r.TS)$
2. Lecture XIV discusses virtual synchrony. Slide XIV-14 discusses the use of “flush” messages as preparatory to installing a new view.
  - a. Why is it important that a process forward all unstable multicast messages in its cache *before* it sends a flush message?
  - b. In slide XIV-15, could P3 have sent flush messages before receiving  $cbcast1$ ?
  - c. In slide XIV-18, suppose P2 had started to do a  $cbcast$  and had sent it to P1 (only), but then both P1 and P2 crashed at the same time. Note that the definition of views allows there to be a difference of just one process in the memberships of successive views. How would this situation be modeled? What happens to the message that P2 attempted to multicast? (Note that there is a typo in the notes for this slide: the third sentence should state that P3 notices that it has an unstable multicast.)
3. The bully algorithm is an election algorithm that was discussed in class: each participant is assigned a unique identifier; the currently running process with the highest identifier is the coordinator — when a process resumes execution after crashing, it starts an election. We assume that all running processes can communicate with one another via multicast, and the total number of processes (running and failed combined) is known. When a process fails, it fails cleanly and comes back up (with the same identifier it had earlier) in a finite amount of time.
  - a. We’d like to design a variant of this algorithm, which we’ll call the noblesse oblige algorithm: the current coordinator continues to be the coordinator, despite recovery of other failed processes, until it fails. Please describe the details of the algorithm.
  - b. Suppose that it’s possible for the network itself to fail, so that the collection of processes might be partitioned into any number of non-intercommunicating pieces. It’s important that at most one of the pieces elects a coordinator. Furthermore, if at all possible, there should be exactly one piece with a coordinator. Explain how this could be done. (It may require some modifications to the election algorithm. What is required of a piece so that it may elect a coordinator?)
4. Explain why serial equivalence requires that once a transaction has released a lock, it may obtain no more locks, even if the transaction is guaranteed not to abort. Your answer should be fairly general — it should not depend on particular examples.