Dr. Annette Bieniusa
M.Sc. Peter Zeller

TU Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Exercise 5: Programming Distributed Systems (SS 2019)

Prepare this sheet for the exercise on Wednesday, June 5th.

1 Atomic broadcast and consensus

In the lecture you have seen, that atomic broadcast can be implemented using consensus (Atomic Broadcast Algorithm, Lecture 6 “Replication, FLP”, Slide 17).

a) Show that it is also possible to do the reverse: Describe an algorithm that implements consensus by using atomic broadcast. Your algorithm should provide a *propose* method to clients and trigger a *decide*-event when a proposed value is decided.

b) Assume we change the atomic broadcast algorithm from the lecture and only use best-effort broadcast instead of reliable broadcast. Show that this would make the algorithm incorrect by giving an execution that violates one of the properties of consensus.

c) Consider the following modification to the atomic broadcast algorithm from the lecture. Instead of two sets `delivered` and `received`, we only use a single set `pending`, which should contain the messages that have been received but not yet delivered.

Why does this “optimization” not work correctly?

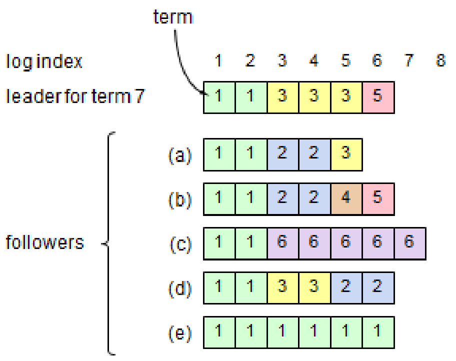
```
State:
  $k_p$            // consensus number
  pending    // messages received but not a-delivered by process

Upon Init do:
  $k_p$ ← 0;
  pending ← ∅;
Upon a-Broadcast(m) do
  trigger rb-Broadcast(m);
Upon rb-Deliver(m) do
  pending ← pending ∪ {m};

Upon pending ≠ ∅ do
  $k_p$ ← $k_p$ + 1;
  propose($k_p$, pending);
  wait until decide($k_p$, msg$^{k_p}$)
  ∀ m in msg$^{k_p}$ in deterministic order do trigger a-Deliver(m)
  pending ← pending \ msg$^{k_p}$
```

2 Raft

These questions are taken from the Raft quiz at `https://ramcloud.stanford.edu/~ongaro/userstudy/quizzes.html`.

To learn about the details of the Raft algorithm, read the Raft paper "In Search of an Understandable Consensus Algorithm (Extended Version)" by Diego Ongaro and John Ousterhout (`https://raft.github.io/raft.pdf`*). You can also watch the video at* `https://youtu.be/YbZ3zDzDnrw`.

a) Consider the figure below, which displays the logs in a cluster of 6 servers just after a new leader has just been elected for term 7 (the contents of log entries are not shown; just their indexes and terms). For each of the followers in the figure, could the given log configuration occur in a properly functioning Raft system? If yes, describe how this could happen; if no, explain why it could not happen.



b) Suppose that a hardware or software error corrupts the nextIndex value stored by the leader for a particular follower. Could this compromise the safety of the system? Explain your answer briefly.

c) Suppose that you implemented Raft and deployed it with all servers in the same datacenter. Now suppose that you were going to deploy the system with each server in a different datacenter, spread over the world. What changes would you need to make, if any, in the wide-area version of Raft compared to the single-datacenter version, and why?

d) Each follower stores 3 pieces of information on its disk: its current term, its most recent vote, and all of the log entries it has accepted.

- Suppose that the follower crashes, and when it restarts, its most recent vote has been lost. Is it safe for the follower to rejoin the cluster (assuming no modifications to the algorithm)? Explain your answer.
- Now suppose that the follower's log is truncated during a crash, losing some of the entries at the end. Is it safe for the follower to rejoin the cluster (assuming no modifications to the algorithm)? Explain your answer.