8 Concurrent and Distributed Systems (mk428)

You are developing a distributed system in which you want some task to be assigned to exactly one node at any given time. If that node crashes, the task must be automatically reassigned to a different node, but you also want to ensure that there are never two or more nodes executing the task at the same time.

This is known as a *lease*. A lease is a concurrency primitive similar to a lock (only one node may hold a lease at any time); the difference is that a lease times out if it is not renewed for some time. After timing out, another node can acquire the lease.

(*a*) Briefly summarise how leader election works in the Raft consensus algorithm, and discuss the commonalities and differences between leader election and a lease. (Focus only on leader election, and ignore the rest of the Raft algorithm. Include the role of the term number in your explanation.) [5 marks]

(*b*) In a partially-synchronous system with crash-recovery failures, is it possible to guarantee that a lease is always held by exactly one node? Justify your answer. [5 marks]

(*c*) You are asked to design a lease algorithm for a system in which the set of nodes is not known in advance, and may change over time. Can the Raft leader election algorithm be used here? Why/why not? [2 marks]

(*d*) A colleague proposes the following lease algorithm:

- There are three servers, each storing a value that is initially null.
- Assume every client has a unique ID *clientId* $\neq$ null. Every 10 seconds, each client that wants to acquire the lease, or currently holds the lease, sends a request (acquire, *clientId*) to all three servers.
- When a server with current stored value v receives (acquire, n):
  If $v = \mathsf{null} \lor v = n$, or if its value was last set more than 30 seconds ago, then it sets its value to n and replies true. This counts as "setting the value", even if the value does not change.
  If its value was last set to $v \neq n$ less than 30 seconds ago, it leaves its value unchanged and replies false.
- If a client receives two or more true responses from the servers, it now holds the lease, otherwise it does not hold the lease.

Discuss the strengths and weaknesses of this algorithm. What faults does it tolerate? What assumptions does it make for its correctness? How might the algorithm be improved to avoid some assumptions or weaknesses? [8 marks]