Homework 4

CS425/ECE428 Spring 2020

Due: 11:59 p.m. on April 4. **No late submissions!**

1. Timeout values in RAFT. Please show your work in deriving these values. If you are not comfortable working with probability, you may write a simple simulation. Please include (concise, commented) code of your simulation with your solution, and make sure to run the simulation enough times that the standard deviation is below 1%.
   (a) (2 points) Consider a RAFT cluster with a heartbeat interval of 30 ms, and a **maximum** one-way communication delay of 20 ms. (No minimum is known.) Assuming zero processing delay, what is should the *minimum* election timeout be set to?
   (b) (2 points) Suppose now that the election timeout is chosen uniformly at random from the range [150,300] ms. Assume also that one-way delay for all messages is *exactly* 10 ms, and no processing delay exists.

   Suppose that the leader for term 1 fails and its 4 followers receive its last heartbeat at exactly the same time. The first follower sets its election timeout to 150 ms. What are the odds that another follower also calls an election for term 2?
   (c) (3 points) Now compute the general probability that an election is called by two or more followers for term 2, assuming now that each follower sets its election timeout uniformly at random in the range [150,300] ms
   (d) (3 points) Now compute the probability of a *split vote* in term 2; i.e., the probability that no leader is elected in term 2.

2. We will adapt the model from the FLP proof to the RAFT system. In particular, a *configuration* will include the state of each RAFT node (candidate / follower / leader) and the current term. An *event* consists of the receipt of a message by a node, changes in that node's state, and the sending of one or more new messages. An event can also be precipitated by a timeout expiring. Here are two example events:

   - Node 1's election timeout expires. It updates the current term to term 1, it changes its state from follower to candidate, and it sends RequestVote messages to nodes 2 and 3.
   - Node 1 receives AppendEntries from Node 2, with current term being 2. It updates the current term to 2, and changes its state to follower. It sends an acknowledge RPC and resets its election timeout.

   To simplify the discussion, do not worry about timeout values and assume instead that an election timeout event is *always* valid for any node.
   (a) (3 points) Considering a 3 node cluster, and starting with an initial state of all nodes in follower state, with current term = 0, list a sequence of events that results in Node 1 transitioning to the leader state with the current term = 1.
   (b) (3 points) Starting from the same initial state, list a sequence of events that results in Node 2 transitioning to the leader state with the current term = 2.
   (c) (4 points) For the above sequences, at each event, list which nodes *could* be elected leader for term 1 in *some* future execution starting from that event. (This is similar to the valence concept of the FLP proof.)

3. Consider a Bitcoin network with $N = 64$ nodes. Let us model the propagation of a newly mined block. Note that this will be a simplified model that disregards several complexities of the protocol.

   At time t, there are N_t nodes that have a copy of the block. $N_0 = 1$, which is the block that mines it. Each of the nodes picks a random other node to send the block to.[1] This node already has the block with probability $(N_t - 1)/(N - 1)$ and it does not have the block with probability $(N - N_t)/(N - 1)$. Therefore, the expected number of nodes that receive the block are $N_t(N - N_t)/(N - 1)$. We will therefore use the recurrence:

$$N_{t+1} = \lceil N_t + N_t(N - N_t)/(N - 1) \rceil$$

   (a) (3 points) Starting with $N_0 = 1$, how many rounds until all nodes receive the block?

   (b) (5 points) Calculate the chance that a chain split occurs. In each round, each node that has not yet received the block will mine a conflicting block with probability $1/(600 \times N)$. Again, you may use a simulation to calculate this if you wish, but make sure to include your simulation code and use enough trials to get an accurate estimate of the true value.

   (c) (2 points) (Unrelated to above) Find some number such that `echo` *`netid n`* `| sha256sum` results in a number with at least 5 leading zeros, with your own netid. E.g.:

```
$ echo nikita 90242 | sha256sum
00000b8556ab757a1a7a6a3ab4b43ff0045975e439593b98a8281f244ab4a772  -
```

[1] In Bitcoin nodes only send blocks to random *neighbors* but we will simplify the analysis here.