Discussion Session 08
Raft | Homework 8 | Timer
Raft

- A consensus algorithm
- Built with the goal to be understandable
- Well defined RPCs

**Figure 1**: Replicated state machine architecture. The consensus algorithm manages a replicated log containing state machine commands from clients. The state machines process identical sequences of commands from the logs, so they produce the same outputs.
Leader Election in Raft:

- Safety - Only one Leader at a time
- Liveness - A new leader is elected when the current leader crashes

Server States -

- Follower - Every Node starts in this state
- Candidate - After Election timeout, a node transitions to this state
- Leader - After winning the election, a node transitions to this state
Timer

Perform an action after certain time has elapsed

```python
from threading import Timer
from time import sleep, time

current_milli_time = lambda: int(round(time() * 1000))

def f(arg):
    print("[{}]“{} Timer Action Triggered! {}\u201d.format(current_milli_time(), arg))

t = Timer(3, f, ["Argument Passed In."])  
print("[{}]“{} Starting the Timer\u201d.format(current_milli_time()))
t.start()
sleep(5)
print("[{}]“{} Ending the program.\u201d.format(current_milli_time()))
```
Timer

Canceling a Timer

```python
from threading import Timer
from time import sleep, time

current_milli_time = lambda: int(round(time() * 1000))

def f(arg):
    print("{}] Timer Action Triggered! {}").format(current_milli_time(), arg)

t = Timer(3, f, ["Argument Passed In."])
print("{}] Starting the Timer".format(current_milli_time()))
t.start()
sleep(1)
print("{}] Canceling the Timer.".format(current_milli_time()))
t.cancel()
sleep(4)
print("{}] Ending the program.".format(current_milli_time()))
```
Other Ideas

1. Use messaging Queues such as ZeroMQ
2. The Event class in the threading module
Follower -

1. Keep listening for heartbeats from the Leader and reset election timeout
2. If heartbeat was not received before election timeout expired, transition to Candidate State
3. If RequestVote call received, vote for only one node in a term
Candidate -

1. Increment the term, vote for yourself and request for votes from other nodes
2. Set a timeout on RequestVote RPC call, and send out requests only to nodes that haven’t responded yet
3. Become a leader if majority votes received and stop election timeout.
4. Make sure that the election timeout is larger than the heartbeat frequency
Homework 8 - Notes

Leader -

1. Sends heartbeats (appendEntries) and stay a Leader as long as other nodes respond with success=True
2. If a response to heartbeat (appendEntries) is False, revert back to a follower, update term and set votedFor = None
3. A Leader cannot transition directly to the candidate state. Stop the election timeout once in Leader state
Homework 8 - Notes

General -

1. Use Logging and print the time with each log message
2. Write simple tests locally and run them multiple times. Your program should not “mostly” work. Use logs to debug the case when there is a failure
3. Redirect logs to a file so that you can automate testing and access logs
Resources

When unsure about a certain scenario -

https://raft.github.io/raftscope/index.html

Questions?