Implementation

Implementation Specifics:

- Python 3.0
- UDP sockets
- Git VCS
- Clickup task management board

What service are we implementing?

We want a client to be able to ask any of the members of the group, “what are the current members of the group?” ← This is the service.

- This means that each member must be able to respond to a client request
- Each member of the group must also have a list of the members of the group that it knows about, in order.

Assumptions Made

- No more than N/2 nodes will fail simultaneously, where N is the number of nodes in the group.
- Eventually, but not necessarily simultaneously in time, every member of the group will have the same view.

System Design

Group Membership

- There are three states for each member:
    - **Member**, if they belong to a group/when they start up;
    - **Candidate**, if the leader has failed and the node is asking for votes from other members of the group;
    - **Leader**, if the node receives a majority of votes from other members when it is already a candidate.

State Transitions

- Assuming we turn on, starting in state 'Member':
    - **If there is no heartbeat broadcast request from a leader:**
      Member -> Candidate,
        - **If there is no leader and we receive the majority of the votes: we become the leader;**
          Candidate -> Leader

      OR
        - **If there was no leader, but we lost the election, we become an ordinary member of the group.**
          Candidate -> Member

    - **If there is a heartbeat broadcast request from a leader:**
        - We are an ordinary member: we receive heartbeat messages, and update our group view if required.
        - We have a timeout on the length of time that we wait for a heartbeat request
            - If the timeout is reached, we assume the leader has died.
            - This means that we must start an election and form a new group.
            - This means we start again with the first state: Member->Candidate

- Assuming we are in state 'Leader':
  We can become a 'Member' if we see a 'Leader' with a higher term;
  Otherwise, we are the only leader.
    - We continuously **broadcast heartbeat requests**.
    - If we receive a response:
        - If the responder was part of the group, they remain part of the group.
        - If the responder was not part of the group, we update our group view and inform everyone else of the change.
    - If we don't receive a response
        - If there were previously responses from some nodes

            - We assume that these nodes have died/'left, and broadcast updates to any listening members informing them of the change.
        - If there were no responses previously
            - We are the only node in the system - we are the leader.
            - We continue to broadcast messages in the event that someone will join.

General Points of Discussion

- **Multiple Groups**
  The idea of group**s** was discussed, and deemed to be taken care of by the RAFT protocol:
    - Each node must have a leader: when they start, they will
        - Elect themselves leader, if there are no other nodes;
        - Vote for a candidate, if an election is taking place;
        - Join an existing group.
    - The RAFT protocol ensures that there can be no more than 1 leader at a time: this means that no node can be a member of more than 1 group.
    - The only case in which there may be multiple groups is when there is a network partition
        - In this case, the RAFT protocol ensures that every partition has its own leader
        - If this partition is healed, RAFT ensures that the leader with the highest term becomes the leader of all of the nodes

- **Group Membership Update Messages**
  The group membership update messages can be piggybacked on the heartbeat messages being sent from the leader to the other nodes in the group.
    - Each heartbeat message should have a sequence number. This ensures that we know when we have missed a heartbeat.
        - **To be decided**: if we know we have missed a heartbeat, do we discard any new messages and request this message?
            - I.e. if the last message we received was msg1, and then we received msg3, do we throw away msg3 and request msg2?
            - This would be **discarding** instead of **buffering**.
        - On the other hand, we could **buffer** the messages we have already received and not make an update to the group view
            - We could send a request for msg2 in our heartbeat-ACK for msg3, and once we receive msg2 we can then add msg2 and msg3 to the groupview (and log).
    - We need to have the sequencing of the messages because we are using UDP: an unreliable protocol.

- **Log File versus Group View**
  The idea of having the log file is that it will be easy to demonstrate that the changes over time are consistent across all of the nodes.
    - This will make demonstrating the reliability of our system good.

- **Member Identification**
  When a member joins the group, it should be given a number by the leader. This number uniquely identifies that member to the rest of the group.

- **Message Corruption**
    - We decided that it isn't an immediate concern to implement message corruption recovery: we can do this later.
    - The focus for the moment is on implementing the communication and group membership protocol.

- **Voting and Terms**
    - Votes are requested by candidates, from other members
    - Each member can either say
        - "Yes, I support you"; OR
        - "No, I don't support you"
    - Each node can only vote **once per leader term**.
        - This means that if two candidates ask for votes, the first one to get to member A will be the one to get its yes/no vote.
        - There is no possibility of member A voting 'yes' for both, because it only has one 'yes' vote per election term.

Timestamping as a sequencing approach

We decided that we can't use timestamps on messages, because of clock drift.

- The time on leader's clock could be very different to the time on a member's clock.
- Instead, we will use ordered sequence numbers: we can then identify when messages arrive out-of-order/don't arrive at all, and made the updates to the group view consistently.

Leaving versus Failing

- There should be no distinction between failing and leaving - if a node simply stops responding to heartbeat messages, it can be assumed to have left.
- This reduces the implementation overhead of having separate negotiation of leaving the group.
- If a node has failed and recovers, it will join as a new node the next time: its previous log and group view are irrelevant.