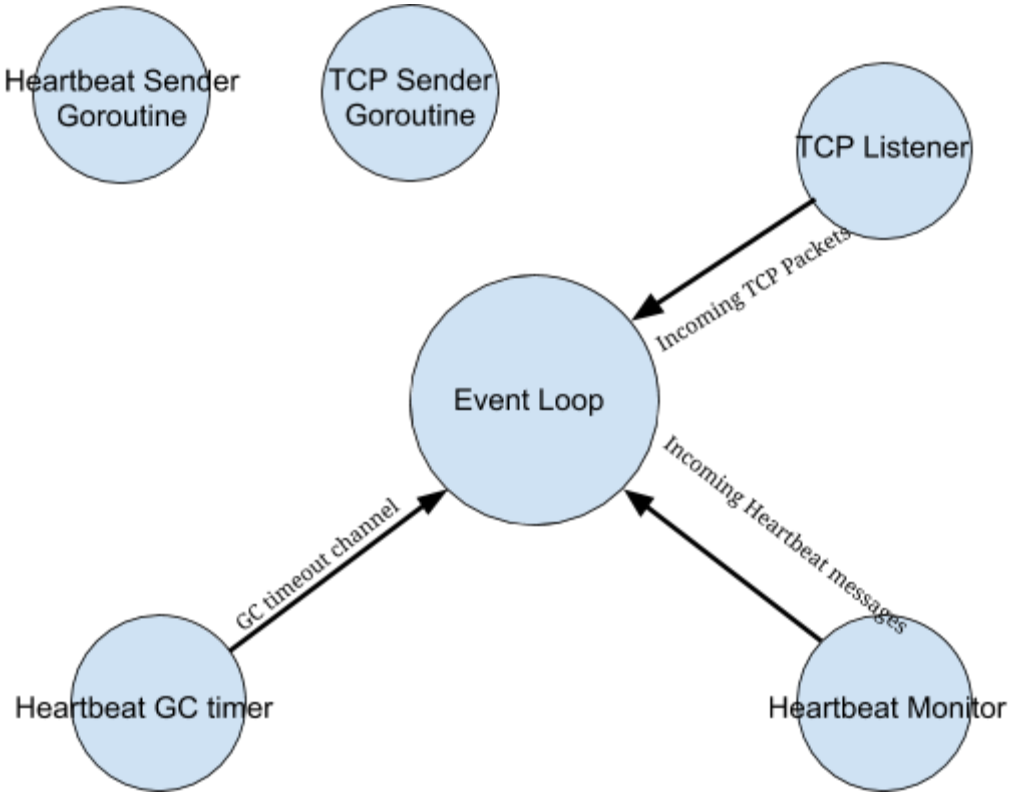
System Architecture:

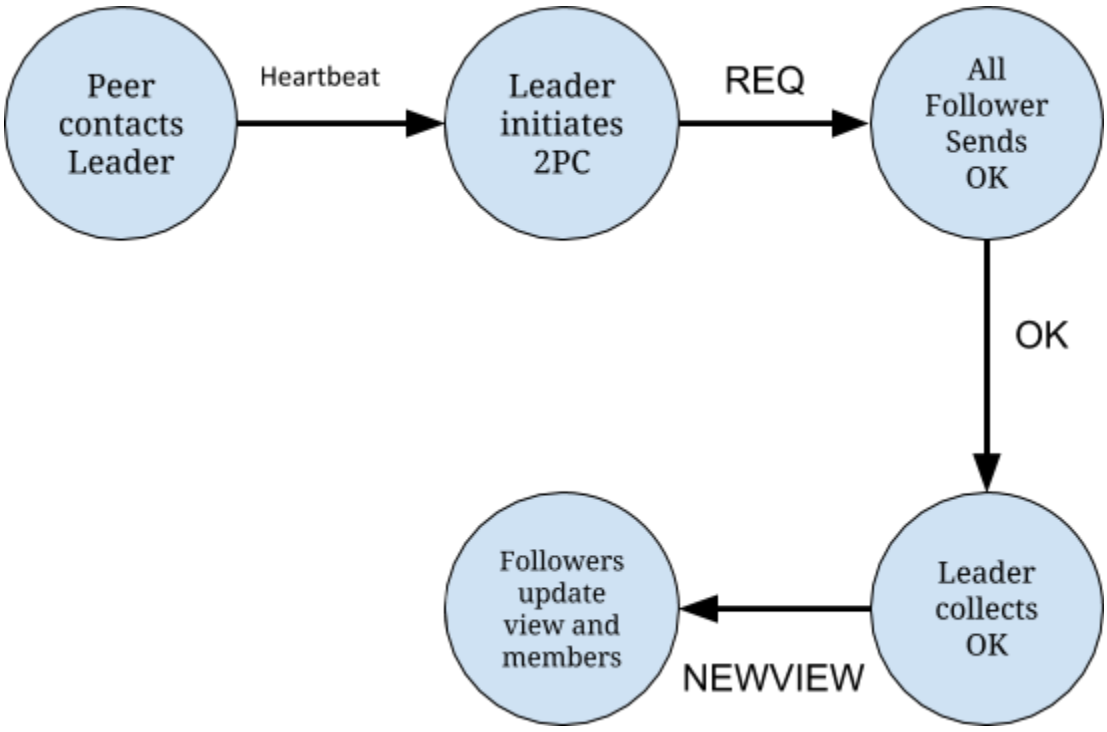
Since this project is written in Go, the design is centered around goroutines and channels. Each of the networking task is spawned in its own goroutine.

1. **TCP Listener goroutine:** This goroutine accepts TCP connections from the peers, decode the sent data and forwards it to the event loop over a channel. Event loop processes the data and sends appropriate response.
2. **Heartbeat Monitor goroutine:** This goroutine accepts UDP heartbeat packets from the peers and forwards it to the event loop over a channel. Event loop adds the timestamp to a hashmap
3. **Heartbeat GC timer goroutine:** This goroutine sends a signal at a regular interval to the event loop. Event loop garbage collects dead processes and in case of master, initiates a DELETE operation.
4. **Heartbeat Sender goroutine:** This goroutine keeps sending heartbeat messages to all the peers in the membership list at regular intervals.
5. **TCP Sender goroutine:** This goroutine is spawned from event loop every time a TCP message is to be sent to a peer.

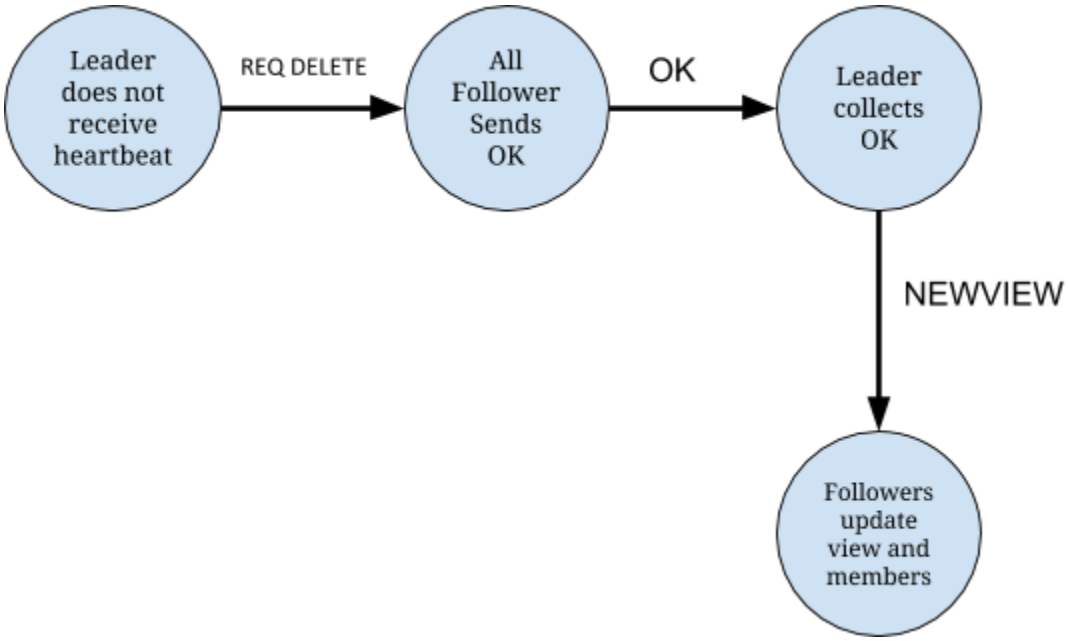


Event Loop: Event loop waits for a message to be available on one of the three channels from the goroutines. As data becomes available on one of the channels, the data is processed and messages are sent. The channels in the event loop are accessed using as multi-way blocking select.

State Diagram for ADD operation:



State Diagram for DELETE operation:



State Diagram for LEADER FAIL: