Thursday 16-20, station 14, ID 87

Christopher Yuxiang Chen - christyc@stud.ntnu.no

Edvard Skei Tønset - edvarsto@stud.ntnu.no

We have decided to use a Peer to Peer network topology, where there will be a master slave structure for deciding which elevator executes which orders.
Every elevator will periodically send out “I’m alive” messages as a way to detect crashes or disconnections, and keep track of other elevators’ orders to obtain the required fault tolerance.

Normal operation of order from button pressed til doors open:

1. One elevator receives an order and starts broadcasting that it has seen an order.
2. The elevators receiving this message will broadcast the same message and add that it has also seen this order.
3. When the master elevator receives an order that has been seen by all elevators alive, it will delegate the order to the most efficient elevator and when every elevator has confirmed the delegation they turn on the button lights, as well as storing who executes which order.
4. If the delegated elevator dies before finishing the order, it will be noticed by the master as a lack of “I’m alive” messages, and the order will be delegated to another elevator.
5. When the delegated elevator reaches its floor it will open its doors and send that the order has been executed When every node has confirmed that they have seen it they will turn off the button lights, and remove the stored order.

We plan on using UDP as our protocols since our information will be broadcasted periodically for everyone to see. Given the right implementation there will be no issues that the messages are received in the wrong order or that it won’t be received.

We have decided to use Golang for implementing the system since we felt it was more intuitive to understand threads and sockets in Go, and because we found more resources regarding the project for this language.

We have decided to divide the system into modules. We plan on having network modules, one for Peer nodes and one for the Master node. These will handle communication for the respective elevator. We will also implement a middleman module for conveying and interpreting messages between network and elevator. We will have modules handling elevator functionality - this will closely correspond with the code handed out for a single elevator.

If an elevator loses power or goes offline, its alive signal will no longer reach the others. The other elevators will have an overview of which existing hall orders the dead elevator had, and the master will distribute these orders among the remaining elevators. When the elevator comes back online, the others will get it up to speed on existing orders and also make it aware of its own cab orders. In our implementation, the master doesn’t have any more information than the others, it is simply pointed out to make the decisions. Therefore, if the master dies, another node will be pointed out to become the new master. The procedure for handling the orders of the dead elevator remains the same.