Project Design Description Group 71

Kristoffer Onstad Schie kristoffer.o.schie@ntnu.no
Dina-Madelen Sandlie Hegna dina.s.hegna@ntnu.no
Svein-Thore Wighus stwighus@stud.ntnu.no

January 2025

1 Design

1.1 Strategy for fault toleance

- The button light contract The button lights up when the order has been added to the queue of the cab tasked with that floor.
- Network unreliability All nodes wait for acknowldgement from the master of new orders and tasks.
- Spontaneous crashes and unscheduled restarts if a program crashes and restarts it sends a message that it is back online. Sets it world view by matching with the master broadcast. It also checks if the current master has a higher ID than it self. if so it sends a message to the master that it assumes its role.
- Normal operation of hall or cab calls from the button press to the opening of the door Button press. Node sends pickup request to master and waits for acknologement. if no acknoledgement is given in 1s the node adds it to its own queue, master calculated best suited node and sends an order to the best suited node and waits for an acknoledgement. Once the best suited node recvies the order it adds it to its queue and acknoledges the master, if the master does not recive an acknoledgement within 200ms it sends the request to the 2nd best alternative. The master then acknoledges the node that first recived the button press which in turn turns on the light. once the selected node has serviced all orders infront of this specfic order it goes to the specific floor and opens the door and waits 5s for the door to close. any button press inside the cab is added to that nodes queue and informed to the master.
- The network disconnecting a node with active hall requests (detection -¿ takeover) If the broadcast from the master stops, each node assumes that the master has gone offline. as every node knows the masters worldview they wait for the new master to be appointed based on ID, if the network

is completely down. every node will think that it is the master and act independently.

if a node does not respond to the master when given an order or takes a certain amount of time(200ms) to process an order the master redistributes the orders of that specific node.

- A node with an active cab order crashing All computers runs the same program but with diffrent IDs The master role is decided by the lowest ID. every 100ms? The master broadcasts a message with its ID and its world view whitch is then updated in each node. If 500ms passes without a broadcast message is recived the node waits untill 200ms*ID has passed an then checks again, if no master broadcast has been recived then that slave is promoted to master, this ensures that the node with the lowest ID enherits the master role.

- all of the above in the presence of network packet loss Packetloss for the broadcast will need a minimum of 12 losses in a row to affect the system. if a

1.2 Network topology and choice of protocols

1.3 Why Rust?

We chose Rust as we see it as more relevant for use in other projects than GO or Elixir. GO seems to focus more on simplicity and Rust seems to focus more on control and speed. Eventhough GO and Elixir are well suited for concurrency, we do see them as optimal for hardwareinterfacing as Rust. Software development in Rust is slower and more complex compared to GO. Unlike GO and Elixir, Rust has no need for a garbage collector and instead uses ownership and borrowing to be memory safe. https://bitfieldconsulting.com/posts/rust-vs-go