Design dokumenter heisprosjekt, Gruppe 2: Andreas Moltumyr og Martin Mostad

System inndeling:

System inndelt i n noder (hver node er en process) knyttet sammen over nettverket gjennom TCP/UDP og IP.

Meldinger som sendes mellom noder:

- Knappetrykk
- Sette lys
- Heis status oppdateringer
- Ny ordre til heis
- Ordre gjennomført av heis
- Kø data fra aktiv node til backup på passive noder
- Meldinger om failover

Inne i hver node kommuniseres det over Go channels mellom ElevatorControl og NodeCommunication, og mellom OrderManager og NodeCommunication. NodeCommunication vil drive med forwarding av meldinger mellom de forskjellige modulene inne i en node, og de andre nodene (og deres moduler). Adressering internt vil derfor bli enkelt. F.eks. en modul ber om at en melding skal sendes til aktiv *orderManager* og NodeCommunication vet hvilken IP og port dette skal sendes til for å komme frem til orderManager.

Inne i hver modul vil kommunikasjon skje mellom de forskjellige submodulene ved bruk av funksjonskall. Vanligvis gjennom at tråd-submodulen kaller funksjoner fra de andre submodulene.

ElevatorControl modul:

- **ElevatorStatus:** Holder orden på heis status (etasje, bevegelsesretning, dør åpen/lukket, osv.) og innkapsling av dataen. Benytter Triple Modular Redundancy (TMR) for å beskytte minnet mot bit-flips.
- **Elev.h:** Grunnleggende heis I/O driver skrevet i C.
- **ElevatorDriver:** Oversetter Go funksjonskall til C funksjonskall i *Elev.h/Elev.c*.
- **ElevatorControlThread:** Heis kontroll tråden. Benytter funksjoner/metoder fra *ElevatorStatus* og *ElevatorDriver* til å styre en heis og snakker med resten av systemet gjennom *NodeMessageManager* modulen gjennom en Go channel.

NodeCommunication modul:

- **NodeConnections:** Holder orden på en tabell med node forbindelsesinformasjon (IP, port, Routing table) og innkapsling av denne dataen. Triple Modular Redundancy (TMR) for å beskytte data mot bit-flips.
- **NodeMessageManager:** Funksjoner for å ta imot, pakke, pakke-ut og sende meldinger.
- **TCP/UDPinterface:** TCP/IP, UDP/IP som tar imot meldinger som skal ut på nettet.
- **NodeCommunicationThread:** Nettverks kommunikasjonstråden. Benytter funksjoner/motoder fra *NodeConnections*, *NodeMessageManager*, *TCP/UDPinterface bibliotek*.

OrderManager modul:

- **OrderQueue:** Holder order på heisenes køer og heis status, og innkapsling av dataen. Benytter Triple Modular Redundancy (TMR) for å beskytte minnet mot bit-flips.
- **OrderEvaluator:** Modul som inneholder «costfunction» som blir brukt til å velge hvilken heis som skal utføre hvilke ordre.
- **ProcessPairControl:** Modul som inneholder funksjoner/metoder for fault detection (hearthbeat meldinger fra masterNode til backupNoder) og fault handling (Failover til annen node)
- **OrderManagerThread:** heis manager tråden. Benytter *OrderQueue, OrderEvaluator, ProcessPairControl modulen* til å reagere på ny tilstand og nye ordre, bestemme hvilken heis som er best skikket til å utføre ordre og drive backup lagring av kø i andre noders *OrderManager* modul i tilfelle nettverksproblemer, hangups, prosess krasj eller strømproblemer.

Fault detection, Fault handlig and Fault avoidance:

- Benytter **process pair** for å sørge for at systemet, i det tilfellet at en node streiker, fortsetter å fungere, men da med et redusert antall fungerende heiser.
    - En av nodene vil ha en aktiv(master) OrderManager (som tar valgene om hvilke heiser som skal utføre hva) modul mens de andre nodene vil ha passive(slave) OrderManagers hvor det blir lagret en backup av informasjonen om køene til de forskjellige heisene. Disse passive nodene lytter etter hearthbeat meldinger fra den aktive noden og er klare til å overta kontroll over systemet hvis de ikke mottar slike meldinger på en viss tid.
    - Køene skal på denne måten bli tatt vare på gitt at bare en av nodene streiker om gangen.
- **Triple Modular Redundancy** lagring av køer, status og nettverkstabeller.
    - Dataen lagres i tre helt like versjoner. Når data skal leses, sammenlignes de tre kopiene og hvis det er noen forskjell mellom de tre versjonene så vil majoriteten av de tre bestemme hvilken data som er riktig. Dette vil detektere, og håndtere enkle bit-flips feil.
- Oppdatering av køer, heis status og liknende kommuniseres mellom nodene ved hjelp av den **pålitelige TCP transport protokollen** for å sikre at data kommer frem og at dataen er i samme tilstand som den ble sendt i.