* Guarantees about elevators:
  + What should happen if one of the nodes loses its network connection?
    - Dersom en node mister nettverkstilkobling så den prøve å koble seg til nettet på nytt. Gitt watchdog timer som følger med på når den sist var koblet til network så restart.
  + What should happen if one of the nodes loses power for a brief moment?
    - Tilstand kan lagres hos nabo og man kan dermed dobbeltsjekke hva som er oppdraget den holder på med. Hvis den har msitet power så må den minnes på hva den gjorde.
  + What should happen if some unforeseen event causes the elevator to never reach its destination, but communication remains intact?
    - Man har en viss tid til å utføre oppdraget. Dersom det tar for lang tid så vil en annen heis kunne overta oppdraget og avlaste den andre heisen.
* Guarantees about orders:
  + Do all your nodes need to "agree" on a call for it to be accepted? In that case, how is a faulty node handled?
    - Hvis det blir registrert et knappetrykk så legges den bare til. Antar at knappetrykk betyr knappetrykk og at det ikke skyldes noe defekte signaler. Heis må si ifra at den har lagt til et knappetrykk kontra spørre om å legge den til.
  + How can you be sure that a remote node "agrees" on an call?
    - Confirmation meldinger (message passing acknowledgement)
  + How do you handle losing packets between the nodes?
    - Analogi bussprotokoll
    - Sender på nytt dersom acknowlegdement ikke kommer
  + Do you share the entire state of the current calls, or just the changes as they occur?
    - 1. Alle må ha en felles forståelse av køen(ordrene) for at man skal kunne vite hvilke oppdrag en heis har og hvilke oppdrag som skal gjøres. Alle blir backup til hverandre. Alle har hver sin ordreliste derimot, men ordre de håndterer på et gitt tidspunkt.
    - For either one: What should happen when an elevator re-joins after having been offline?
      1. Som nevnt så burde den minnes på sitt siste oppdrag og få tilbake forståelse for den totale køen.

*Pencil and paper is encouraged! Drawing a diagram/graph of the message pathways between nodes (elevators) will aid in visualizing complexity. Drawing the order of messages through time will let you more easily see what happens when communication fails.*

* Topology:
  + What kind of network topology do you want to implement? Peer to peer? Master slave? Circle? Something else?
    - Peer to peer:
      * Mye kobling og mye kommunikasjon, men desto mer sikkert
      * Flere potensielle feil
    - Circle:
      * Kommunikasjon må gå gjennom hverandre som kan bli veldig upraktisk med tanke på måten vi tenker å implementere køen og ordrehåndtering for heisene.
    - Per nå: Peer to Peer.
  + In the case of a master-slave configuration: Do you have only one program, or two (a "master" executable and a "slave")?
    - How do you handle a master node disconnecting?
    - Is a slave becoming a master a part of the network module?
  + In the case of a peer-to-peer configuration:
    - Who decides the order assignment?
      * Alle, heisen selv må sjekke om det finnes relevante ordre i køen som den kan utføre. (eks den er 4->1 og den finner 2->1). Blir litt førstemann til mølla prinsipp, men med acknowledge og threading slik at bare en og en kan akksessere køen av gangen. Dermed kan ikke 2 heiser ende opp med samme ordre.
      * En = master slave?
    - What happens if someone presses the same button on two panels at once? Is this even a problem?
      * Skal ikke bli et problem gitt at begge bare legges til i ordreliste og heisene kun kan finne ut hva som er relevant en av gangen.
* Technical implementation and module boundary:
  + Protocols: TCP, UDP, or something else?
    - If you are using TCP: How do you know who connects to who?
      * Do you need an initialization phase to set up all the connections?
    - If you are using UDP broadcast: How do you differentiate between messages from different nodes?
      * Ja, meldinger fra heisene må skilles. Dette fordi vi må bruke meldingene til å avgjøre om en heis f.eks. skal restartes og da må vi vite hvilken det er. Så må man også vite hvilken heis som gjør hva for å kunne minne dem på hva de skal gjøre hvis de mister strøm eller liknende.
    - If you are using a library or language feature to do the heavy lifting - what is it, and does it satisfy your needs?
      * Go og bruker pakken net, funker fjell
  + Do you want to build the necessary reliability into the module, or handle that at a higher level?
    - Ja/Nei vet ikke helt hva som blir spurt om her
  + Is detection (and handling) of things like lost messages or lost nodes a part of the network module?
    - Nei, network module er bare selve kommunikasjonen. Dette med feilhåndtering kan være en annen modul.
  + How will you pack and unpack (serialize) data?
    - Do you use structs, classes, tuples, lists, ...?
    - JSON, XML, plain strings, or just plain memcpy?
    - Is serialization a part of the network module?
      * Dette kan jeg ikke svare på nå, hehe. Dont know
      * Structs er nice i c da da, men vi får si i GO (lang)

Alle har felles forståelse om køen og dens innhold, men de minner hverandre på hvilket oppdrag de holder på med.

Når oppdrag utført for en heis så må den si ifra med oppdrag utført. Hvis ikke oppdrag utført har kommet så kan vi anta at heisen holder på med oppdraget. På denne måten kan vi håndtere om en heis mister power og da «glemmer» hvilket oppdrag den holdt på med.

Ettersom de sender oppdateringer med hvilket oppdrag de holder på med fast tidspunkt så kan man detektere om netverk tilkobling blir mistet. Watchdog og restart hvis tilkobling borte for lenge. Wattchdog lokalt på heisen ofc.

Hvis heis har utført oppdrag på tiden den hadde mistet tilkoblingen så kan man sjekke hvor heisen er i forhold til oppdraget de andre heisene husker. Hvis den er i etasjen så er oppdraget utført og heisen kan rapportere oppdrag utført. Hvis ikke så må den dra til etasjen. Heisene skal da kun ha et aktivt oppdrag om gangen.

Oppdatere oppdraget underveis?

* Eks: skal ned fra 4 -> 1, også skal noen ned fra 2->1