TTK4145, exercise 6:

Detailed design of the Network module

Hanne-Grete Alvheim, Maren Keini Haugen,
Iselin J. Nordstrøm-Hauge

February/March 2021

Contents

1 The Network Module - design

Our chosen module is the network module. As none of us have worked with sending messages between computers before, it will probably be the hardest module to implement. There will be one network module for each node, and these will send packages to each other. Since we want to implement a peer-to-peer network topology, all the network modules will be designed in the same way. In addition to being connecred to each other, the elevator's network module will be connected to the elevator's corresponding order distributor module and fsm module (see figure 1).

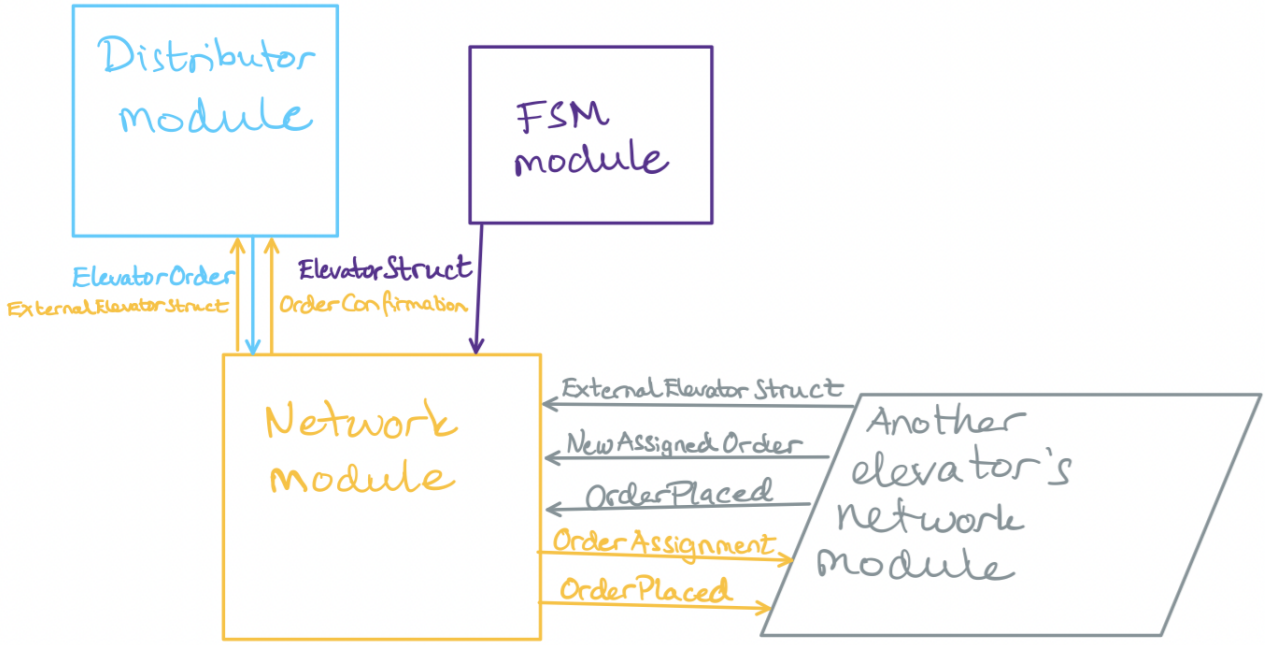

Figure 1: Figure showing how the network module interacts with the other modules.

Inputs of the network module:

- New assigned order [from another elevator's network module].
  Name: NewAssignedOrder
  Data type: JSON object (slice converted to uint8[])
- New order to send + ID of receiving elevator [from Order Distributor module].
  Name: ElevatorOrder
  Data type: slice

- Other elevators' elevator information structs [from other elevators network modules].
  Name: ExternalElevatorStruct
  Data type: JSON object (struct converted to uint8[])
- Own elevator struct [from fsm module]
  Name: ElevatorStruct
  Data type: struct
- Order placed confirmation message [from other elevators network modules].
  Name: OrderPlaced
  Data type: JSON object (slice converted to uint8[])

Outputs of the network module:

- Order placed confirmation message [to Order distributor module]
  Name: OrderConfirmation
  Data type: slice
- Other elevators' information structs [to Order Distributor module].
  Name: ExternalElevatorStruct
  Data type: struct
- Order placed confirmation message [to other elevator's network module]
  Name: OrderPlaced
  Data type: serialized JSON object (slice converted to uint8[])
- New assigned order [to another elevator's network module].
  Name: OrderAssignment
  Data type: JSON object (slice converted to uint8[])

The network module is stateless bacause we don't need to save any of the information it processes in the module. We will only use threads and functions to handle packages. These will be called from the Order Distributor module, and from the fsm module.

In order to connect the inputs and outputs, we need several functions. If one for instance considers the input NewAssignedOrder, this will be sent into the network as a serialized object - hence, our network module needs a function that will deserialize data sent to it from other nodes. In order to send messages we will use JSON, and we also need a way to serialize the messages sent from the network module. Hence we need the funtions

- function Serialize(struct/slice)
  This function will use the json.Marshal function from the package encoding/json to serialize a struct or a slice, given as input to the network module.

- function Deserialize(JSON object)
  This function will use the json.Unmarshal function from the package encoding/json to deserialize a struct or a slice, given as input to the network module.

The elevator will also need functions to send messages between the different modules. As of now, we plan to send elevator structs and slices both between network modules, and from the network module of the single elevator to its own order distributor module and fsm module. Therefore, we need

- function BroadcastElevatorState(OwnElevatorStruct): broadcast its own elevatorstruct, given as input from the fsm module, to a shared port where all of the elevators are listening.

- function SendOrder(ElevatorOrder): receives ElevatorOrder from the order distributer module, and sends it to the elevator the order is assigned to. The ID of the elevator for which the order is assigned is given in Elevator order, and based on the ID the node sending the message can find its designated listening port.

- function PlacedOrderConfirmation(NewAssignedOrder): When the elevator has been given an assigned order (NewAssignedOrder) from another elevator, PlacedOrderConfirmation is a function that will be used to send an "acknowledgement" (OrderPlaced) back with the same order and elevatorID, so the elevator knows that it takes it.

- function takeAssignedOrder(NewAssignedOrder): This function will use the assigned order from another node (OrderAssigned), deserialize it and send it to the fsm module, so that it can be added to its own struct and also send to the order distributor module so it can send that it is taken to the order supplier.

- function readElevatorBroadast():
  Here a listeningport will be used where the elevator gets sent the serialized structs of the other elevators. The function will then read the serialized structs and use the deserialize function to turn them into normal structs. These will then be sent to the order distributor module, so the output is ExternalElevatorStruct.

- function ConfirmedOrder(OrderPlaced): will use the input OrderPlaced, a confirmation message from another elevator, and send that the order is placed (OrderPlaced) to the order distributor module.