Real Time Programming TTK4145

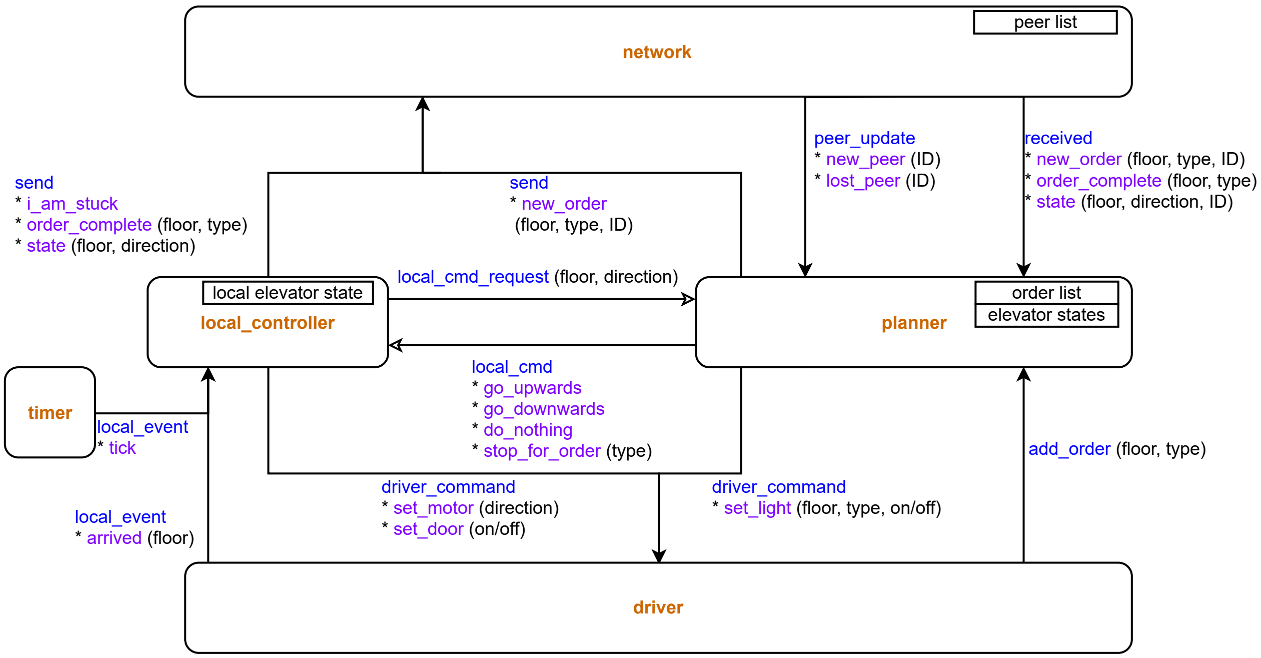
Design

Didrik Rokhaug
Bern Johan Damslora

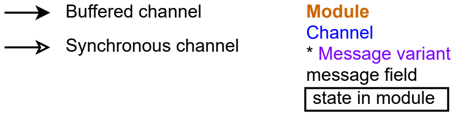
January 30, 2017

NTNU – Trondheim
Norwegian University of
Science and Technology

Module diagram



Legend



The modules each have their own thread, passing messages over multiple-in-single-out thread-safe channels. These channels are in most cases buffered queues to allow nearly simultaneous events to be processed in sequence without blocking a thread. The exception is the local_cmd channel and its corresponding request channel, because the content in one is a specific reply to the request in the other.

Bernt Johan Damslora
Didrik Rokhaug

Interfaces and draft specification:

network handles network interaction. It should maintain a list of peers by broadcasting a UDP beacon, and listening for the same.

Inputs:
send messages should be handled by serializing the message and transmit it to all peers in a reliable way. Failing that, it should announce to all peers that the peer(s) that could not be reached are down. The i_am_stuck variant should also trigger this announcement.

Outputs:
peer_update should be sent as follows:
* new_peer (ID) should be sent when a new peer appears (detected new beacon).
*lost_peer (ID) should be sent when a peer is lost, either by the beacon timing out or an announcement.
received should be sent when a message has been received over the network.

driver is the hardware interface, detecting rising edges on buttons and floor sensors, and executing low-level commands to the motor and lights

Inputs:
driver_command should be handled as follows:
* set_motor (direction) should set the motor direction accordingly, that being either UP, DOWN or STOP
* set_light (floor, type, on/off) should turn on/of the appropriate lamp (types being UP, DOWN or CAB)
* set_door (on/off) should open/close the door

Outputs:
add_order (floor, type) should be sent when a rising edge on a button is pressed
local_event of the variant arrived (floor) should be sent when a rising edge is detected on a floor sensor

timer is a module producing a message at a regular short interval (20 ms, for example).
Outputs:
local_event should be sent as follows:
*tick should be sent at a regular interval.

local_controller is a state machine, responding to an event (time passage or the floor sensor) by sending a local_cmd_request for the next step in handling the orders. The "moving" state has a countdown variable, used to detect mechanical faults preventing the elevator from reaching its destination.

Inputs:
local_event should be handled as follows:
* tick should decrement a countdown for the current state, if it is "moving". If it reaches zero, i_am_stuck should be passed to send.
should cause a local_cmd_request (floor, direction) to be sent, with the current floor and direction of movement in the fields.
* arrived (floor) should update the current floor and direction variables, and pass these in a state to send.
should cause a local_cmd_request (floor, direction) to be sent, with the current floor and direction of movement in the fields.
local_cmd should trigger the state transitions in the state machine, and send any appropriate driver_command messages. If an order was handled, an order_complete (floor, type) should be passed to send.

Outputs:
send
driver_command
local_cmd_request (floor, direction)

planner maintains a list of all elevator orders in the network and delegates new orders.

Inputs:
received
* new_order (floor, type, ID) should add the order to the order list
* order_complete(floor, type) should remove all orders with that floor and type from the order list.
* state(floor, direction, ID) should update the elevator state data
peer_update
* new_peer(ID) should cause all orders to be passed to send as new_order messages, ensuring the new peer has the same order list.
* lost_peer(ID) should cause all orders with that ID to be re-delegated, as if they had just been called locally.
add_order (floor, type) should cause the module to choose the best elevator for the order, add it to the order list, pass it to send as a new_order, and pass an appropriate set_light to driver_command.
local_cmd_request (floor, direction) should be answered with an appropriate local_cmd based on the order list and the supplied state.
Outputs:
add_order (floor, type)
driver_command
local_cmd

Bernt Johan Damslora
Didrik Rokhaug