ELEVATOR PROJECT
TTK4145 Real-Time Programming

Group 20
Michael Soukup, Petter Rossvoll
21/4-2012

Introduction
We have chosen to implement our elevator program in C with a module-based approach in mind. There are four modules in addition to the drivers provided, which will be described shortly in the modules section. Our model is build on the master-slave principle: One master elevator receives information by network communication and then make requests to other elevators (slaves) present at the network. The slaves are actually servers - they may accept or decline requests from the master. Backups of orders and elevator statuses are sent to a chosen master candidate, which will take role as a new master if the current master goes down or lose connection. The idea of introducing a master candidate role into the master-slave model also solves the problem with slaves voting to become new masters. Orders from inside the elevator is always handled locally, while elevator reservations are sent to the master which finds an optimal elevator for the job. A minimum requirement is that an elevator can always run locally and serve all local orders.

Modules
-Network

This module provides methods for finding other elevators at the network and for communication between master and slave. Initially, an UDP broadcast is made to make itself visible at the network, and if no master is found, the elevator takes role as master and starts listening for UDP broadcasts. If the broadcast gets picked up, a TCP connection is made for direct communication. When the master goes down the master candidate will start listening for broadcasts right away, and the slaves will repeat the process.

-Local queue

A linked list for storing both local and requested orders with internal help functions to manage the queue.

-System

Provides the methods needed for running the elevator locally. This includes a state machine, callback functions for interaction with the panel and elevator initialization.

-Master

Another linked list for storing elevator statuses and methods for determine where to request jobs and sending backup.

Use case

-Driving elevator

Action: A person is inside an elevator and pushes a command button.

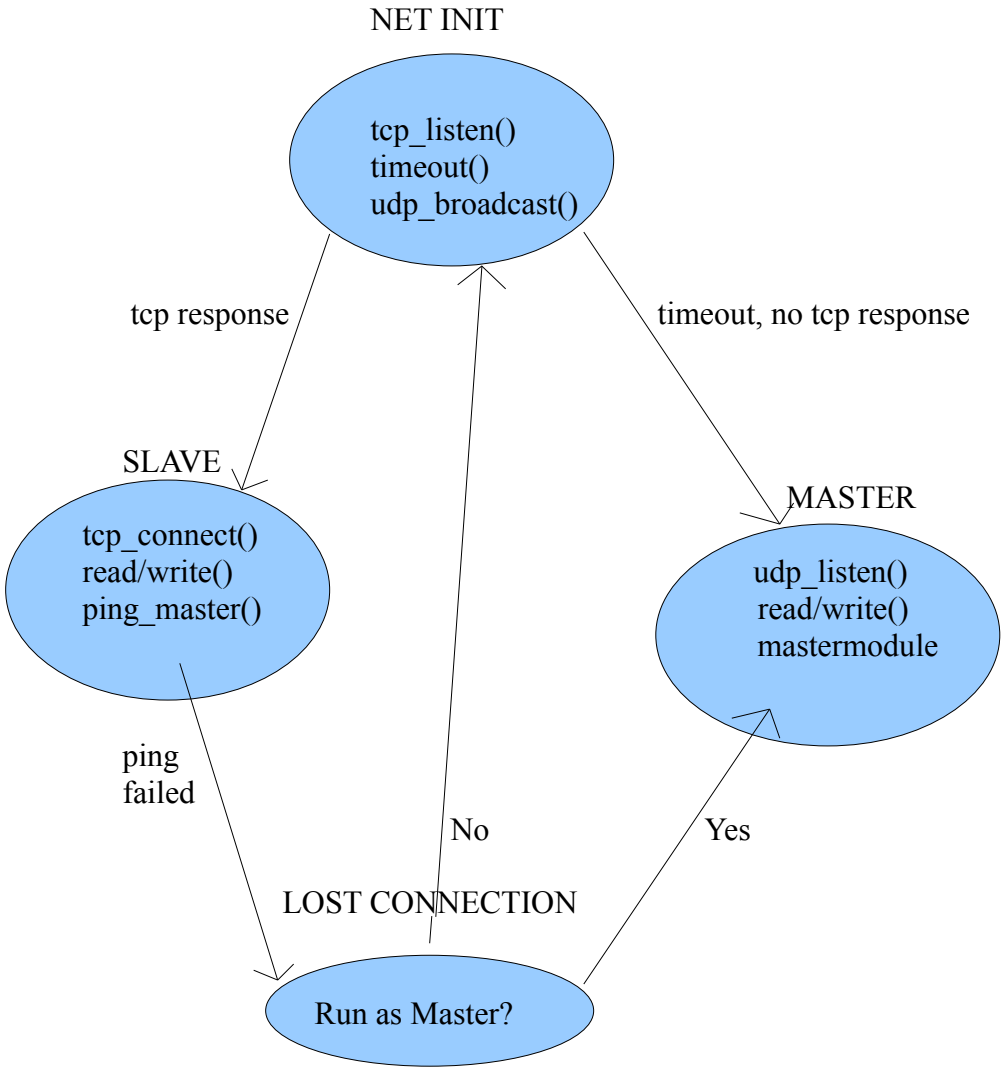
1. Queue order and update state, send status to master.
2. Start driving towards destination and pick up orders along the way which are in the same direction.
3. Stop elevator at destination and open door.

-Ordering elevator

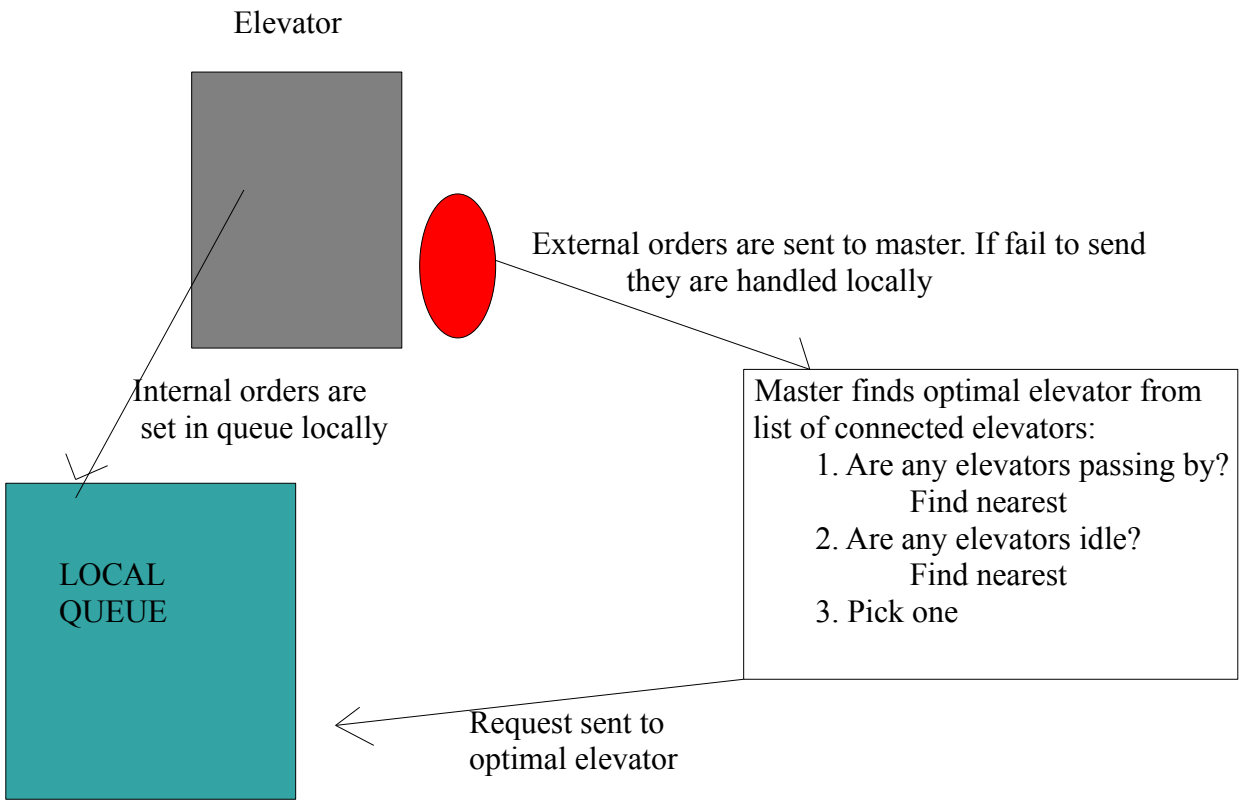
Action: Someone makes a request for an elevator.

1. Send order to master, master sends backup to master candidate.
2. Server finds an optimal elevator and makes a request for this order.
3. Elevator receives order, sets light and queues the order, and sends status to master.
4. Order is served and removed locally, master is updated and removes the order.

Network state diagram



Order handling



Backup

Master chooses first slave as master candidate. Sends backup of list of slaves and list of orders to master candidate. If master goes down, master candidate initialize as master and starts setting up connection with slaves in list and sends requests.