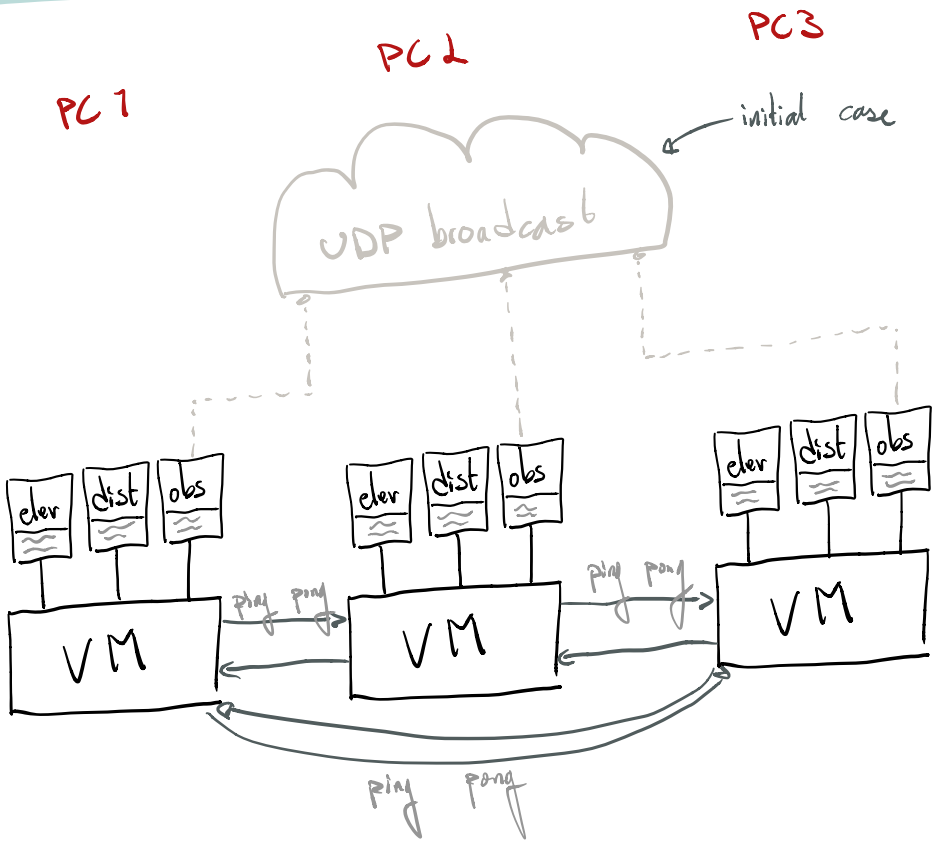
Elixir

- How Elixir works



- Why Elixir

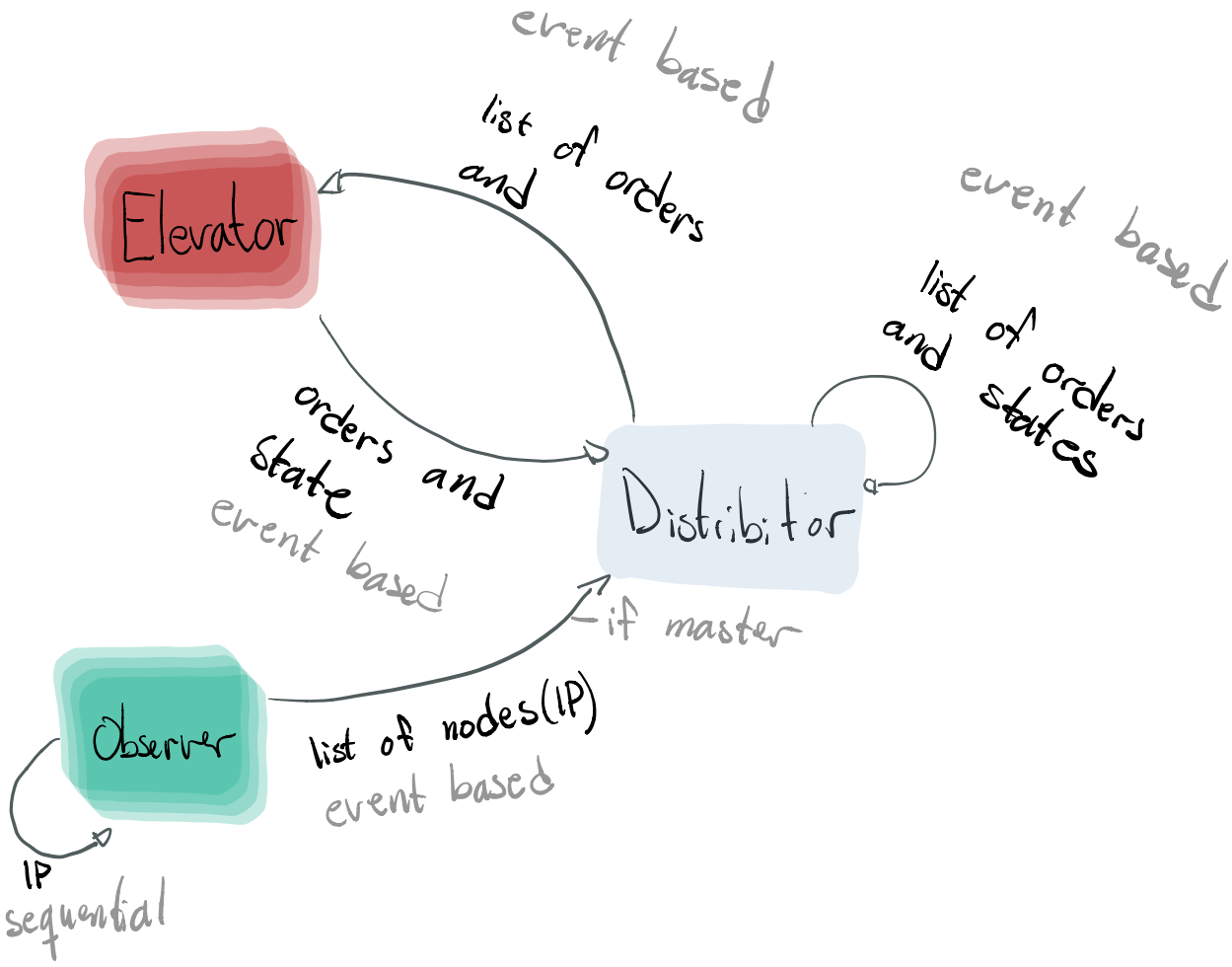
  . Predefined network module

  . More familiar syntax than Erlang

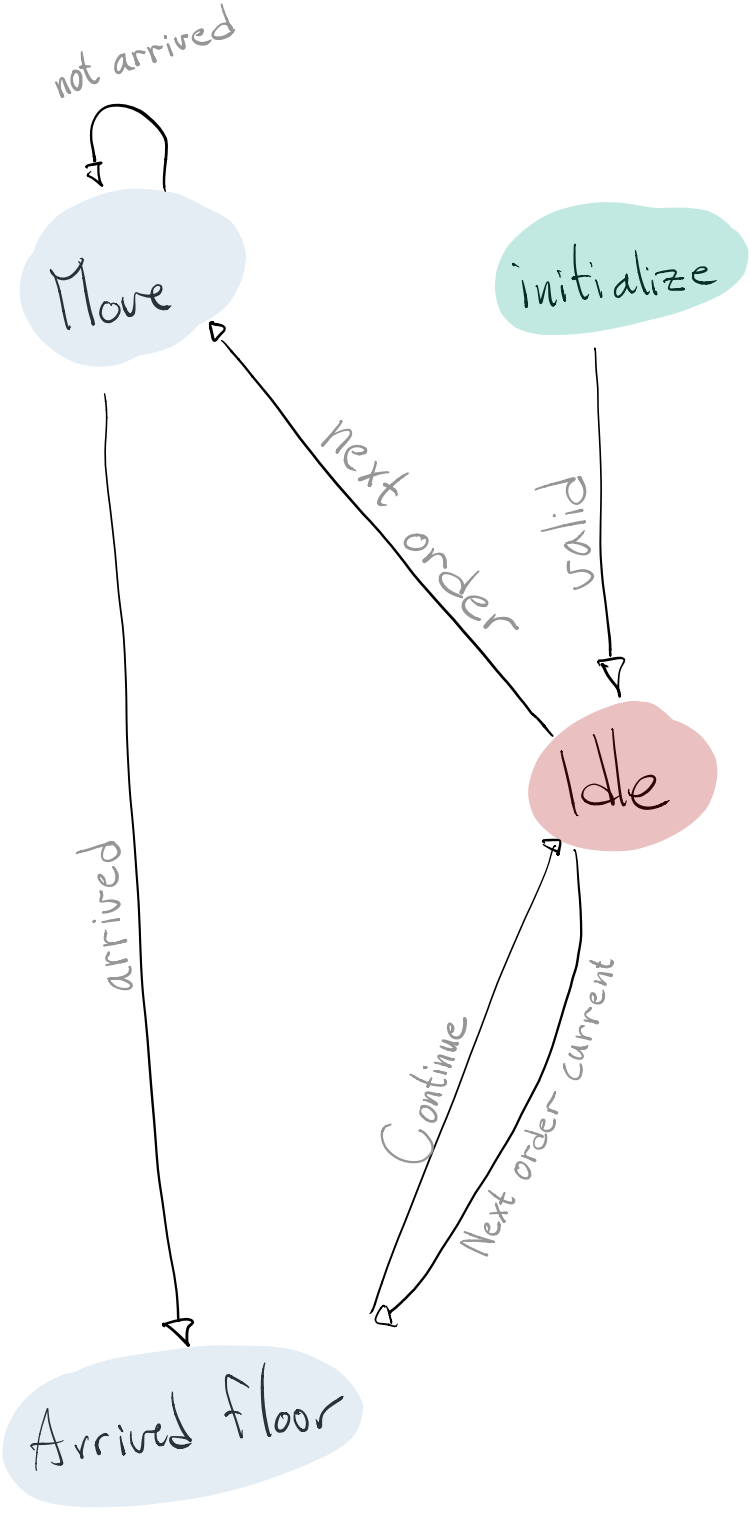
Design

What the system must handle

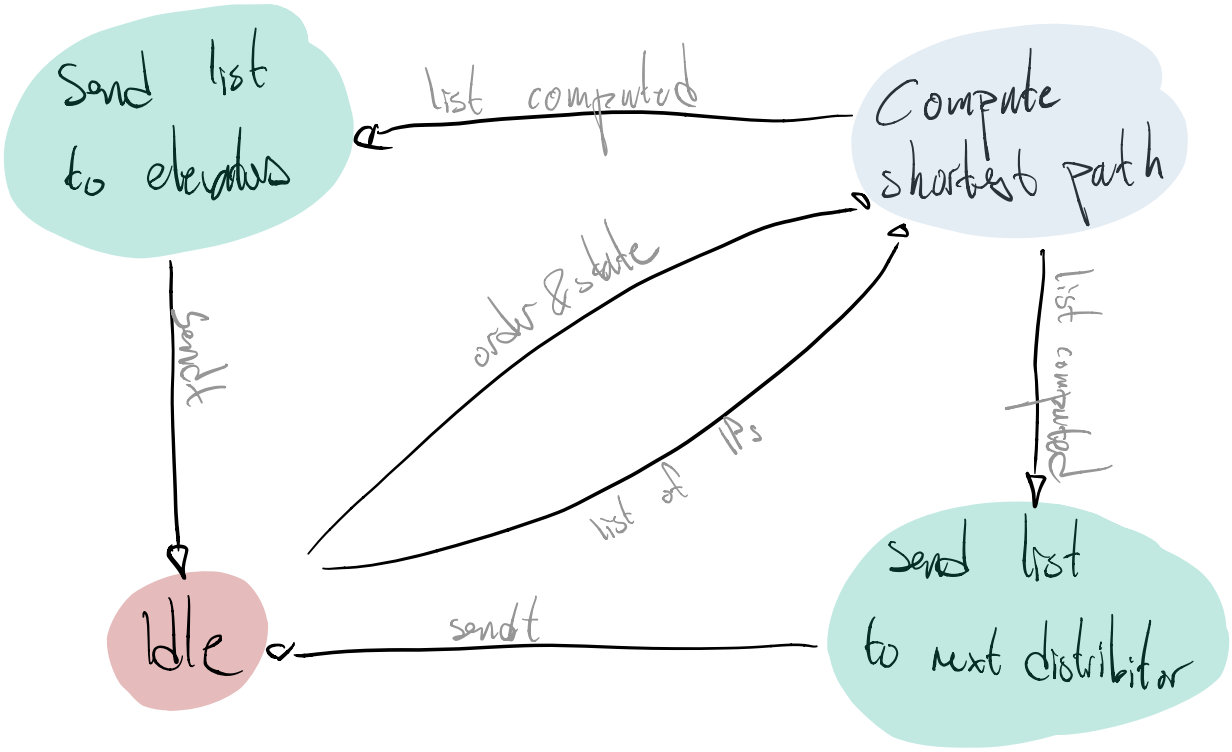
- Order recieved
- A node is killed (stops/crashes/ disappear / lose connection)
- ~~A node is loosing connection~~
- ~~A node is reconnected~~
- Conflict between masters (same situation)
- No master
- A node starts (init)
  - loose UDP package
- Floor sensor date recieved/ detection


event based
list of orders and
Elevator
event based
list of orders and states
orders and state
event based
Distributor
if master
Observer
list of nodes(IP)
event based
IP
sequential

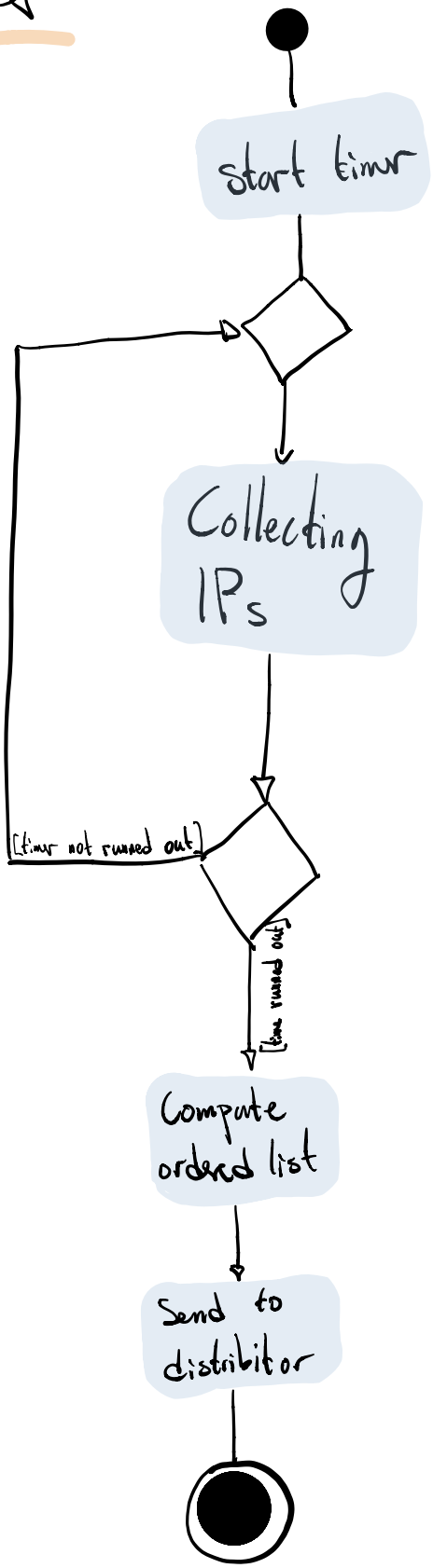
- Submodules
  - Elevator

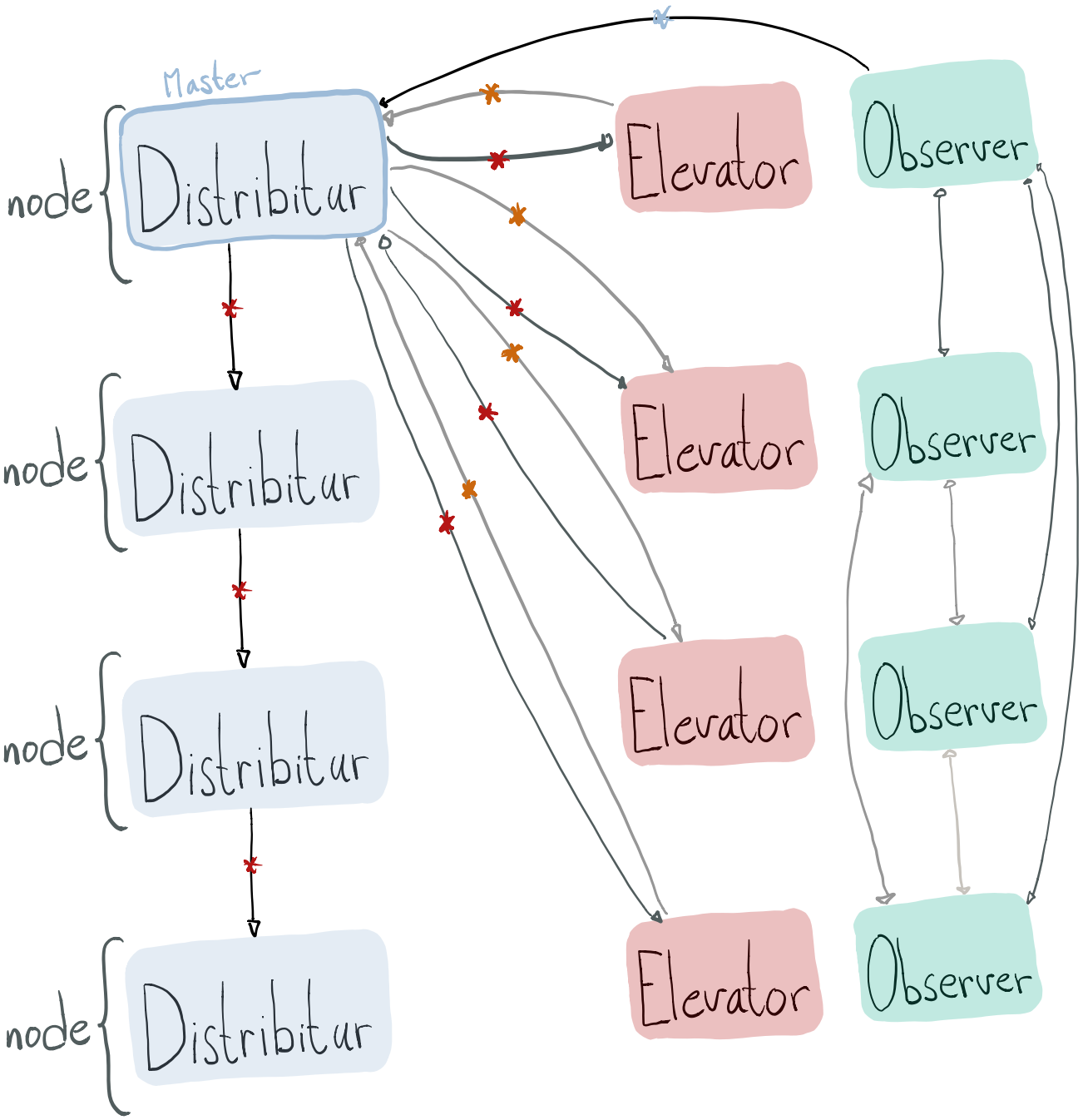


- Distributor



- Observer


start timer
Collecting IPs
[timer not runned out]
[time runned out]
Compute ordered list
Send to distributor

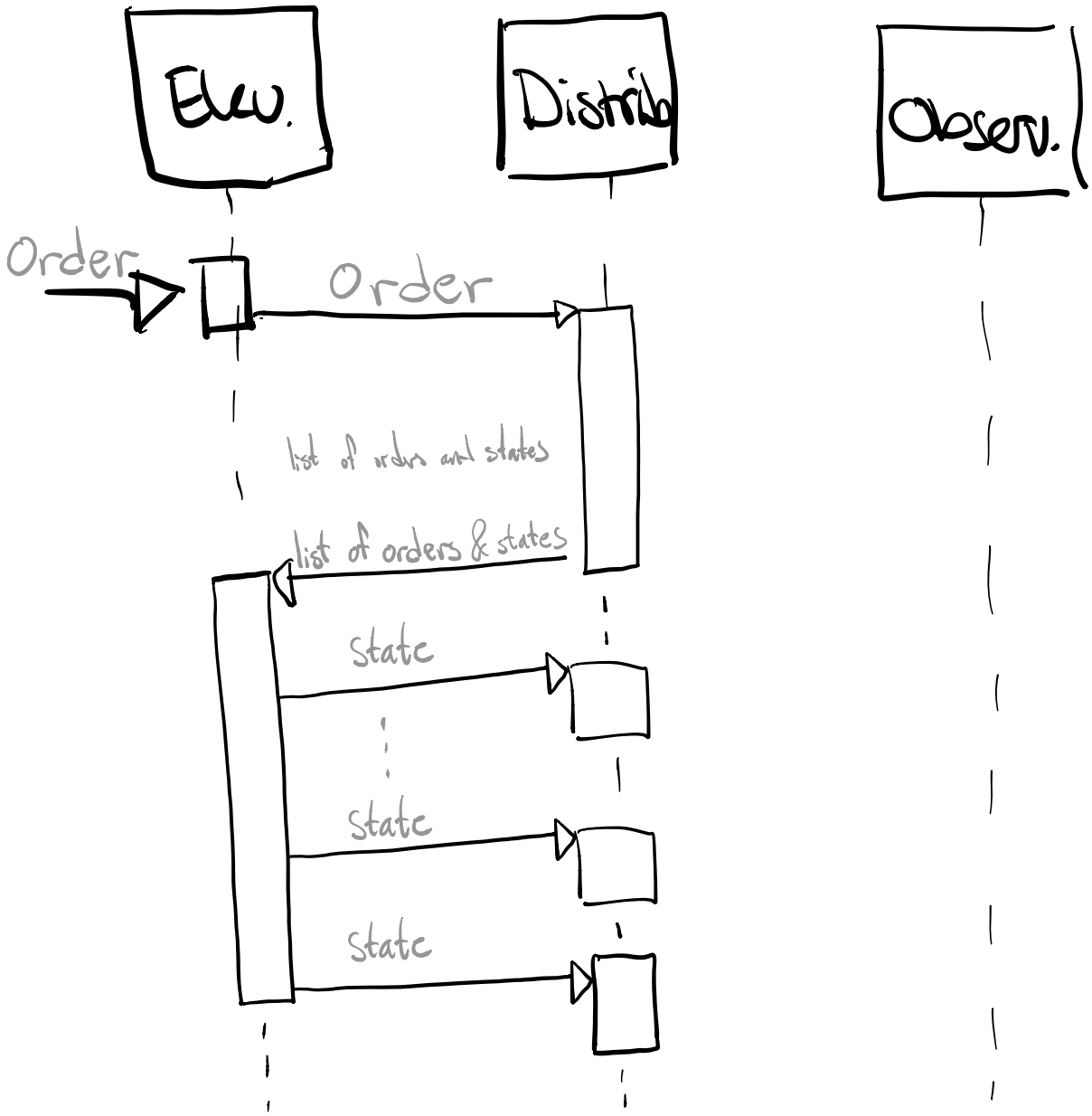

Master
node
Distributur
node
Distributur
node
Distributur
node
Distributur
Elevator
Elevator
Elevator
Elevator
Observer
Observer
Observer
Observer

- List of orders and states
  - Complete list of all orders and states
  - For backup
  - For confirmation of orders (lights)
- Orders and state
  - Current state of explicit elevator
  - Recieved order of explicit elevator
- List of nodes
  - Sorted by IP
  - Top is master
  - If an IP is not recieved in x time, schratch it

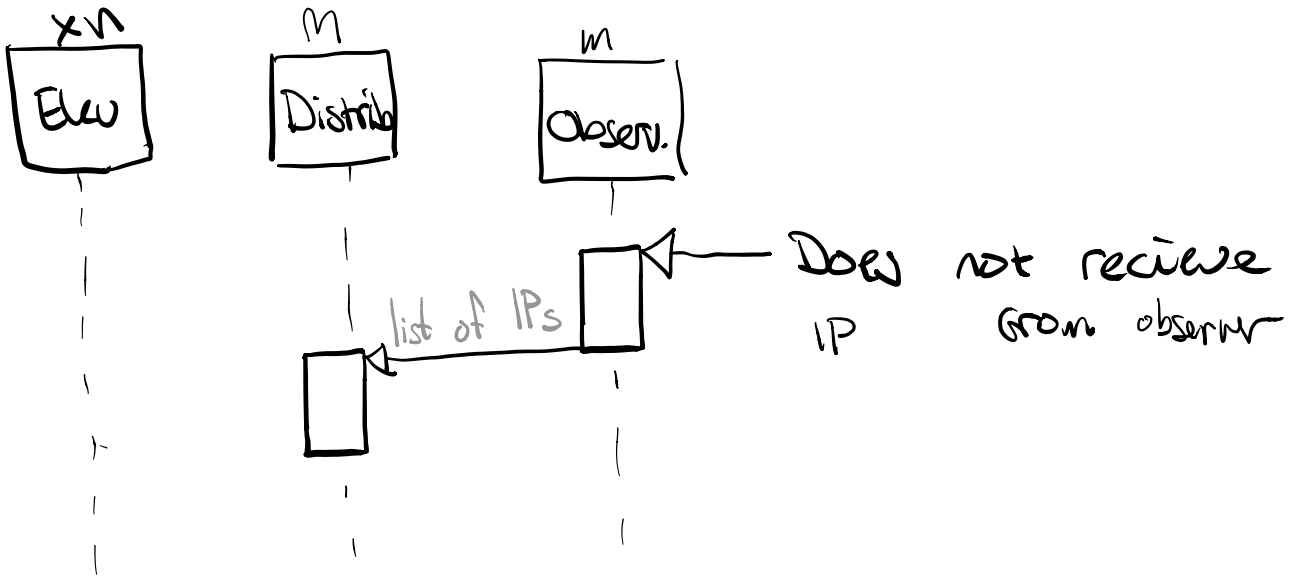
Communication

- Distributar → elevator
  - ↳ Entire list of all orders and states
- Elevator → distributar
  - ↳ Orders and states
- Observer → observer
  - ↳ is alive = IP

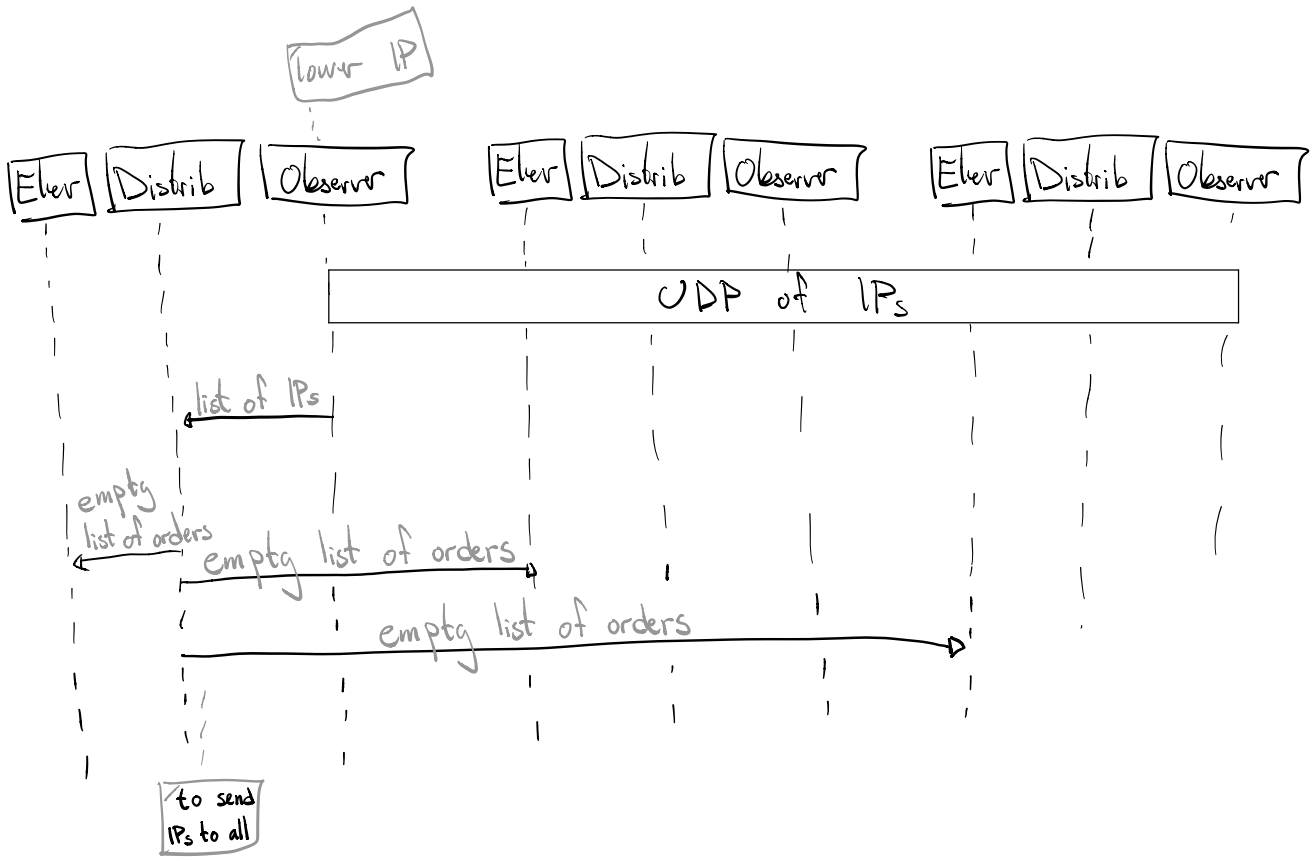
Order is recieved



- A node is killed (eighter master or not)



- Initial setup



• Conflicts between masters (scalable)