Design debrief

Part 1 : Common mistakes and misunderstandings

1: Bad reasons for TCP

"TCP already has acknowledge" or "TCP is reliable"

For an open connection there is only `send()` and `recv()` functions

There is no `bool hasAcked()` function

If you need application-level ack, it must be done at the app-level

Use TCP if you need:

- All messages to arrive in order, and getting no messages at all is ok
- A "session" of requests and responses (not just single request/response)
- A **lot** of simultaneous unique connections

Let's imagine that two endpoints Alice and Bob want to communicate over an unreliable network

And for those of you who are about to pull TCP out of your ass, just imagine that Alice is partitioned, and they're all sending to Bob, and Bob is replicated - where's your precious TCP now?

- Peter Alvaro : "I see what you mean" : https://www.youtube.com/watch?v=R2Aa4PivG0g @ 12:55

2: Magic boxes that do "sync stuff"

"This module synchronizes the orders"

...ok, but how?

Do you really need consistency?

- Does it matter if two Elevators disagree on some data?
- Or the result of some calculation?

Or do you just need uniqueness of some calculation?

Or just a single backup of some data?

3: Too much... stuff

"First we have acknowledgements on data X, then we calculate result A, then acknowledge that everyone knows result A

Then Elevator 1 does work Y, and the rest have a timeout on the response from Elevator 1 where if it does not respond with Y after T seconds, they calculate B

Then there is an acknowledgement procedure on the data B such that some other elevator does work Y, but if that also fails, there is a timeout U which definitely triggers behavior Y on all elevators

4: Misunderstanding Specification

- Hall button lights should show the same on all workspaces
  - Under **normal** conditions this should always work
  - Under **exceptional** conditions (packet loss) this does not have to always work
  - Under **faulty** conditions (broken elevator) this does not work (because it ded, son)
- All button panels on all workspaces must work
- You have three computers, and they all have an elevator attached
  - You cannot put some immortal master program "in the cloud"

5: Assumptions about time on a network

Time can not always be the same everywhere

- You will not be able to "synchronize the watches" on the computers
  - (Especially when we add network delays during testing)

Different times in different places can happen at the same time

- Two buttons of the same kind (eg up at floor 1) can be pressed simultaneously
  - Both real-world simultaneously and idealized-timeline simultaneously

Time doesn't always move forward

- You can get a reply to a response you haven't sent yet, if you just restarted

---

Commutative (any order) and **Idempotent** (any repetition) are good properties

Design debrief

Part 2 : Not-mistakes (takes?)

Things we already have

https:// github.com/TTK4145/Project-resources

- State machine / Elevator algorithm
- Cost function

https://github.com/TTK4145/?q=driver

- Elevator hardware/simulator driver code

https://github.com/TTK4145/?q=Network

- Network modules

It's not stealing if it's copying

- (Also, it's not murder if it's robots)

Things a solution must have

- There must be some three-way communication over the network
  - Some kind of acknowledgement thing going on
  - We need a guarantee that some backup exists
  - 4- or 5-way is ok, 6 or 7 is pushing it, 8+ is wrong
- There must be some timeout somewhere
  - Can't send a "hey btw we are dead" message
  - 2 timers is ok, 3 is pushing it, 4+ is wrong