L-10 :- recap of safety and liveness; reliable delivery

Safety property

=> "Something bad never happen"

→ can be violated in a finite execution.

→ ex :- FIFO/causal/TO delivery

Liveness property

→ something good eventually happen.

→ cannot be violated in a finite execution.

eg: reliable delivery.

Reliable delivery :-

→ let P_1 be a process that sends a message m to process P_2.

→ If neither P_1 nor P_2 crashes, then P_2 eventually delivers m_2.

Reliable delivery is a liveness property but it is not a safety property.

Fault models

9/11

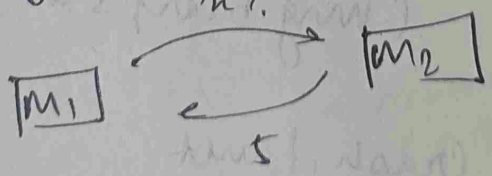
So when you are designing a system, you need to start by making some certain assumptions about the environment in which that system operates.

So in particular if you are trying to build a fault tolerant system, then you need to have some sort of idea about what kind of faults may occur, so you know what kind of faults you need to tolerate.

So we need some way of defining and classifying faults and putting them in categories.

And a "fault model" is something that tells you which categories of faults can occur so that you know which ones you need to worry about

Lets take a eg from Lec1



Things that can go wrong:-

(i) message from M_1 to M_2 gets lost

(ii) " " M_1 " " is slow.

(iii) M_2 crashes

(iv) M_2 is slow

(v) message from M_2 to M_1 is slow

(vi) " " M_2 to M_2 gets lost

(vii) M_2 lies!

Let's classify all these faults into categories.

(i) & (v) → Omission fault
(Msg getting lost is Omission fault)

(ii) & (vii) → Timing fault
(iv) (msg being slow)

(iii) & (vi) → Crash fault

D.

Types of faults:

- Omission faults - A message is lost
- Timing fault - a message or a process is slow
- Crash fault - a process crashes.
- Byzantine fault :- malicious or arbitrary behaviour.

Hierarchy of faults

crash fault :- a process fails by halting. (stops sending or receiving any messages)
(Well the process can have ongoing internal event, but it won't matter as you can't communicate with it)

omission fault :- a message is lost.
(a process fails to send or receive a message)

Timing fault → a process responds too late

Byzantine Fault :- a process behaves in an arbitrary or even malicious way.
eg:- mimicking other faults.

Note: we cutted out timing fault, cause we will talking about asynchronous system. (messaging Queue type ki, producer consumer alag alag kar rahe apna kaam, bas the unhe ni h ek req bhej ke)
to is course ke liye timing fault hamara concern ni h.

But why do we have this hierarchy?

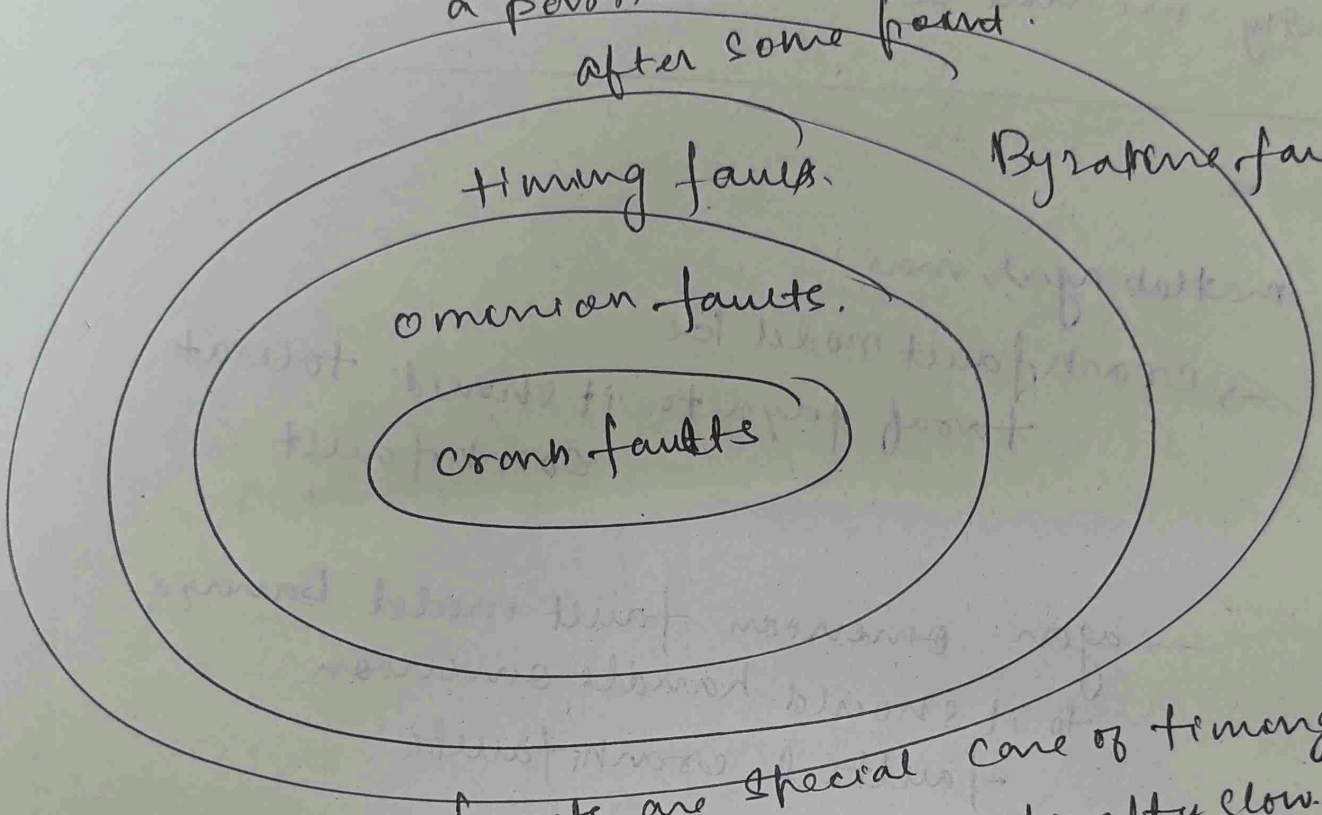
Let say we have two protocol:

Protocol X tolerates crash fault.

Protocol Y tolerates omission faults.

Q) Does protocol Paxo tolerate crash faults?

→ Yes,
crash faults can be thought as a special case of omission faults, where all the messages to and from a particular process get lost after some point.



Omission faults are special case of timing faults where messages are infinitely slow.

Paper
"Atomic Broadcast: From Simple message Diffusion to Byzantine Agreement"
by Cristian et al. (1994)

A fault model is a specification that specifies what kinds of faults a system may exhibit (and thus defines what kinds of faults are to be tolerated by the system)

mostly, we will be looking at omission faults.

---

matlab agar mai
→ crash fault model le
taraf gaya to it should tolerate
crash fault.

→ agar omission fault model banaya
to it should handle omission
-faults & crash faults.

→ Similarly arbitrary jitne upar jayege
class mai, utna hi complex
hote jayega.

Two generals problems

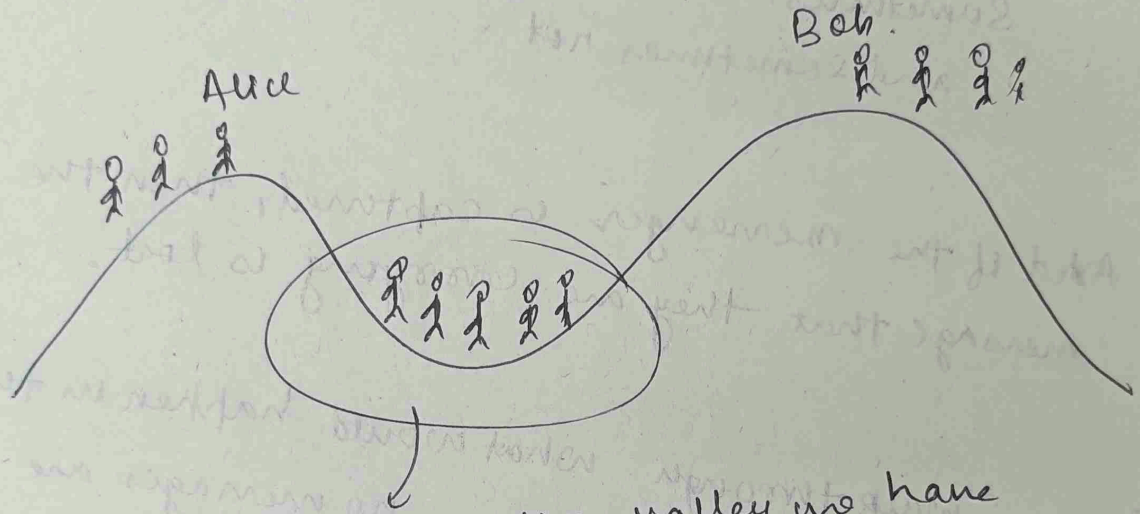
Suppose we have two generals with their armies.

→ General Alice

and

→ General Bob.

and each of them are encamped on a hill.



So, Alice and Bob knows that neither one of them on their own is strong enough to attack the enemy.

But if they worked together, then they would outnumber the enemy.

So, they'd like to both attack the enemy at the same time.

The only problem is that they are too far away to communicate. ~~via any signal or any~~

The only way to send a message is to send a messenger through the valley, who sometimes can be captured by the enemies and sometimes not.

And if the messenger is captured, then the message that they are carrying is lost.

So, lets walk through what would happen in the best case scenario where no messages are lost and everything goes fine.

↳ So let's say Alice wants to attack tomorrow at dawn. ~~early morning~~

So she sends a message over to Bob. Saying "I want to attack tomorrow at dawn".

And so let's say Bob gets two message.

At this point, Bob knows that Alice has prepared to attack at dawn.

So, at this point should Alice go and attack?

→ No, because Alice don't know whether Bob got the message

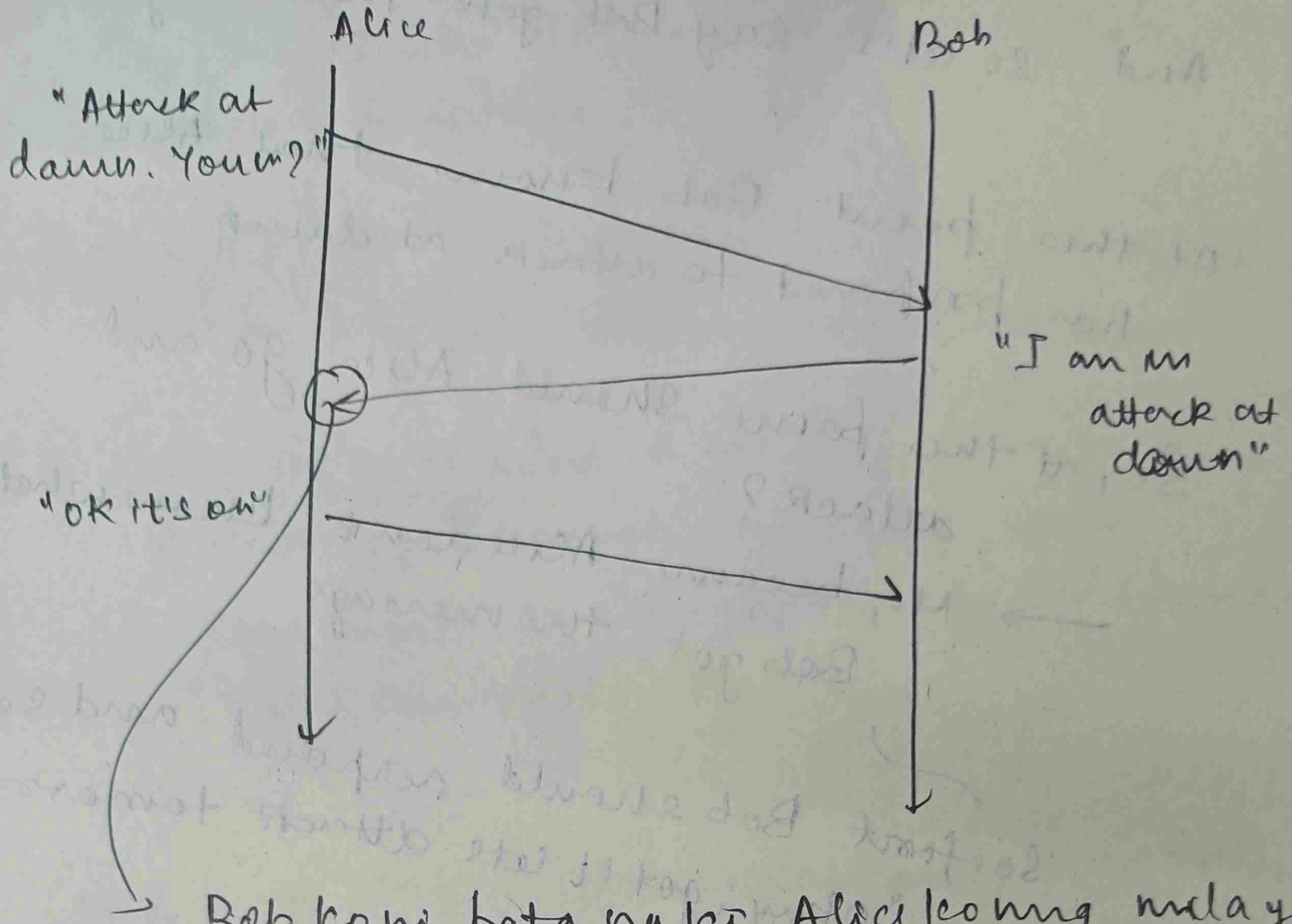
So first Bob should respond and say "okay, got it lets attack tomorrow at dawn"

So, let's say this msg from Bob to Alice also gets delivered.

Now, Alice knows that Bob got the msg.

Now is it safe to go ahed and attack at dawn?

→ No, because Bob don't know ki uska acknowledgement Alice ko mila hya ni



Bob ko ni pata na ki Alice ko mng mila yani,
to tabtak ~~Bob~~ Bob Attack ni kar sakta,
to Alice bhyta h, "ki mng mil gya, bal pakka",
ab Alice ko pata ni h ki Bob ko mng mila yani

to ye process chalte rahega,
kb tareeka h ise roke ka

"So, it turns out that in the common model, it is impossible for Alice and Bob to ever attack and know whether other one will Attack, because it will require sending an infinite number of acknowledgments and so on."

→ And this is a very common problem in any two device communication where there are chances of failures.

So, are there any workarounds :-

→ make a plan in advance (technical term: of common knowledge)

(eg:- matlab Alice aur Bob alag alag hall jaane se pehle hi decide karle saara plan.

we say that there's common knowledge of some piece of information P when :-

- everyone knows P
- everyone knows that everyone knows P
- everyone knows that everyone knows that everyone knows P.
- everyone know . . . that everyone knows P.

Are there any work arounds?

→ Although it's impossible to solve this problem but in practice, it is possible for the generals to increase their confidence that the other general will attack.

So one way to do it, would be for Alice to send a whole bunch of messages at regular intervals, all saying "attack at dawn".

Sooner or later one gets through and Bob sends an ack and at that point Alice can stop sending messages.

So, if Alice receives the ack and then Alices msgs stop coming through, then overtime Bob will gradually grow more certain that Alice receives the ack, right?

But there can still be the case that she is still sending them and all of them are getting lost

So, but prob of answering that would be low.