Agenda:

- CL algorithm wrap-up
- Safety & liveness
- reliable delivery
- Classifying faults, fault modes
- Two general problems

Channels

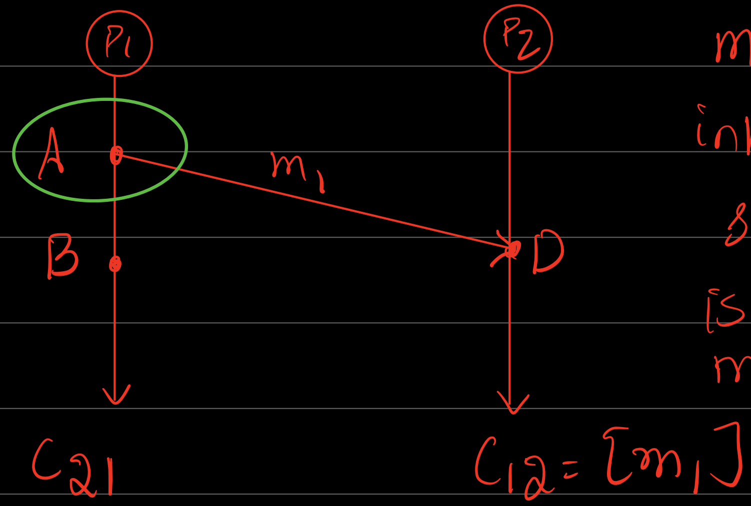
Assume every process has a channel to everyone else.

Graph should be strongly connected

If P_1 cant communicate directly with P_2, we can simulate communication i.e communicate via P_3 channel.

> Strongly connected is a requirement for the CL algorithm

why capture in-transit message



m_1 might contain info about the system. This info is required to make sense of the snapshot.

Safety & Liveness

- FIFO delivery
- Causal delivery
- TO delivery

} **safety properties**: properties that say that a "bad" thing won't happen.

$\boxed{\text{Liveness property: say a "good" thing will happen}}$

Eg: Eventual delivery

Eventual → keyword. If present, it is a liveness prop.

Safety properties

- say a "bad" thing won't happen
- Properties that can be violated in a finite

execution

Liveness properties

- say a "good" thing will happen
- cannot be violated in finite execution
- More difficult to assess.

Reliable (Eventual) delivery : Take 1

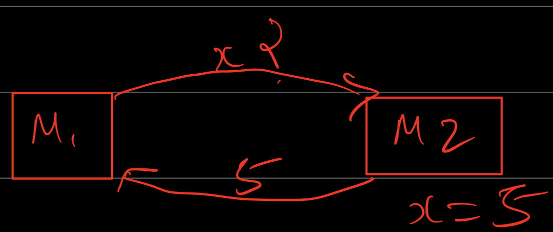
Let P_1 be a process that sends message 'm' to P_2. If neither P_1 & P_2 crashes, then P_2 eventually delivers 'm'.

and not all messages are lost.

All properties are either
- Safety
- Liveness
- Combo of safety & liveness.

Fault model

Tells you which kinds of faults can occur.



msg from M_1 gets lost - omission fault
msg from M_1 is slow - timing fault
M_2 crashed - crash fault
M_2 is slow - timing fault
msg from M_2 is slow - timing fault
msg from M_2 is lost - omission fault
M_2 lies - Byzantine fault

Crash fault: a process fails by halting. (stops sending / receiving msgs. May have internal messages)

Omission fault: a message is lost. (a process fails to send or receive one message)

timing fault: a process responds too late (or too-early)

Byzantine fault: a process behaves in an arbitary or even malicious way.

Eg: Assume Protocol X tolerates crash faults
Protocol Y tolerates omission faults.

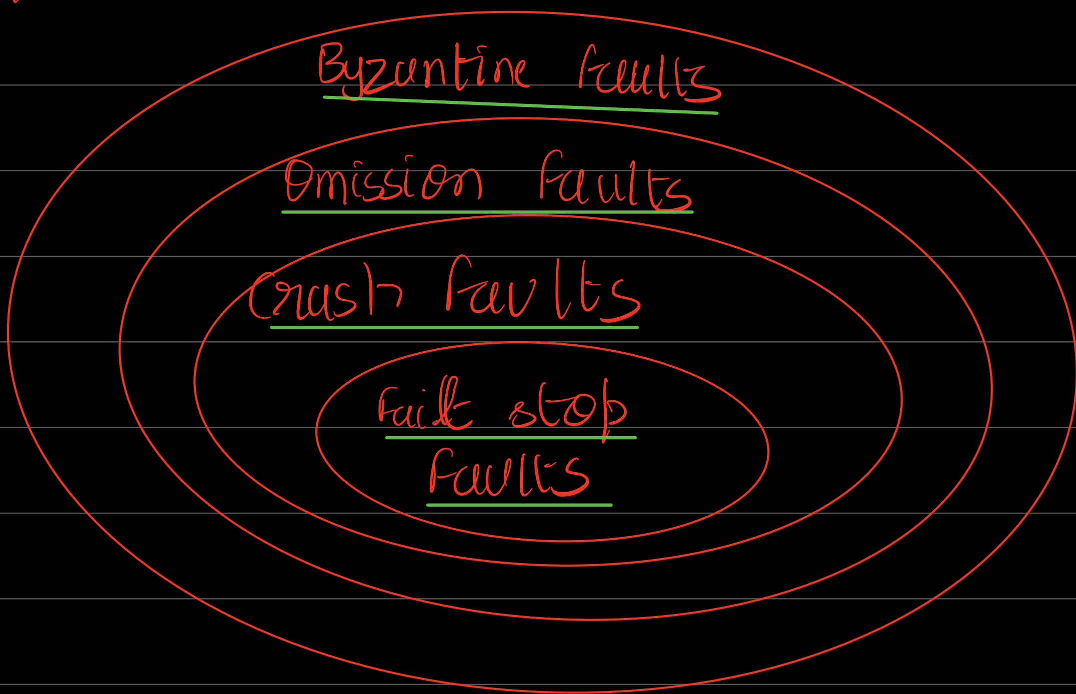
Q: Does Y also tolerate crash faults?

YES!

Crash faults are a special case of omission faults.

Q: Protocol Z handles byzantine faults. Does Z also tolerate omission faults?

YES.



Crash fault: process fails by halting

Fail-stop fault: process fails by halting, and everyone knows it crashed.

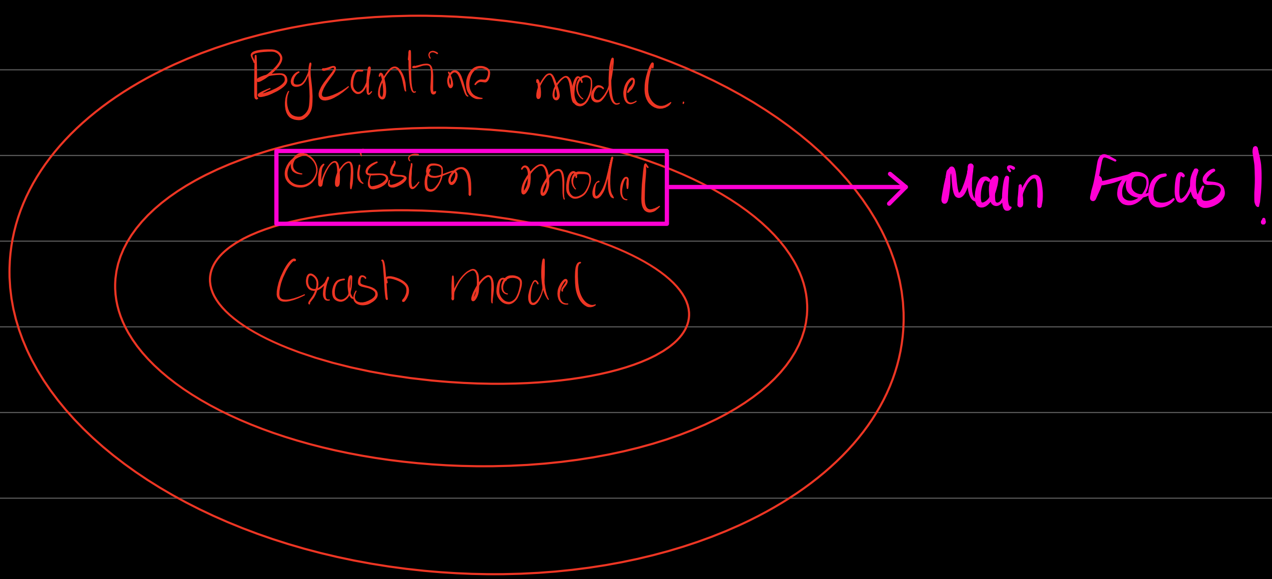
Mostly faults.

Q: Why are timing faults excluded?

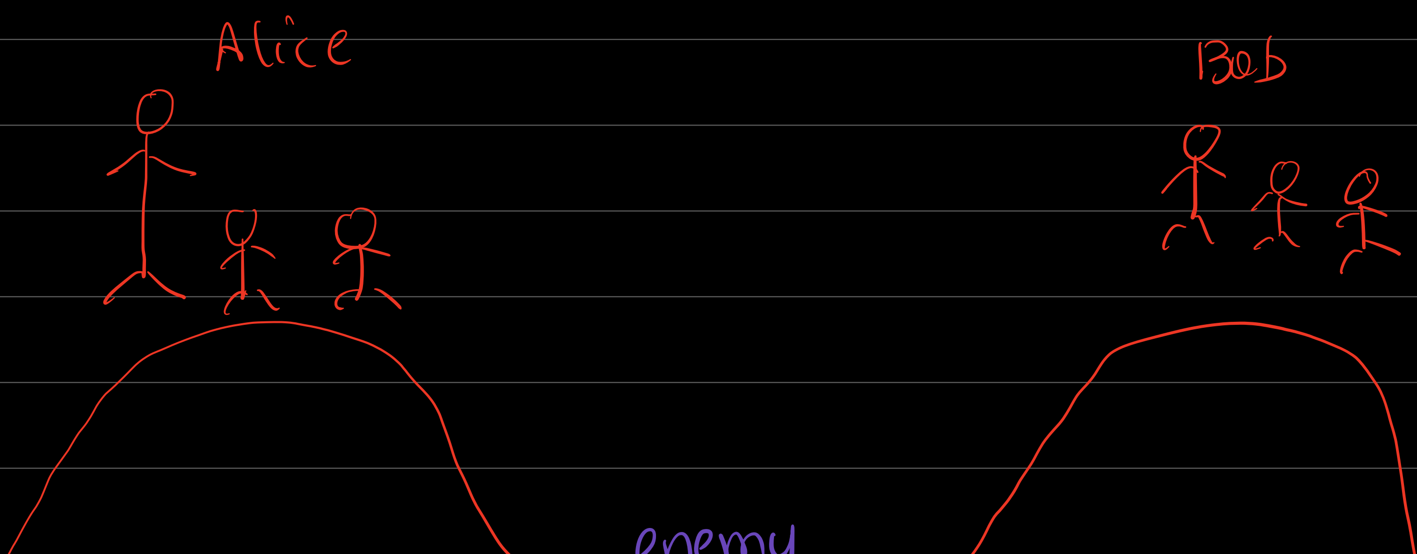
In asychronous world, there are no timing faults!

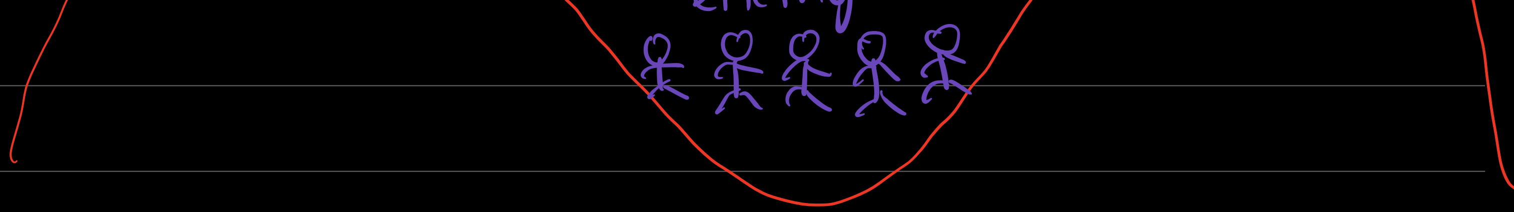
Fault model: Formal definition

Specification that says what kinds of faults a system can exhibit, & this tells you what kinds of faults need to be tolerated.



Two generals problem - 1975



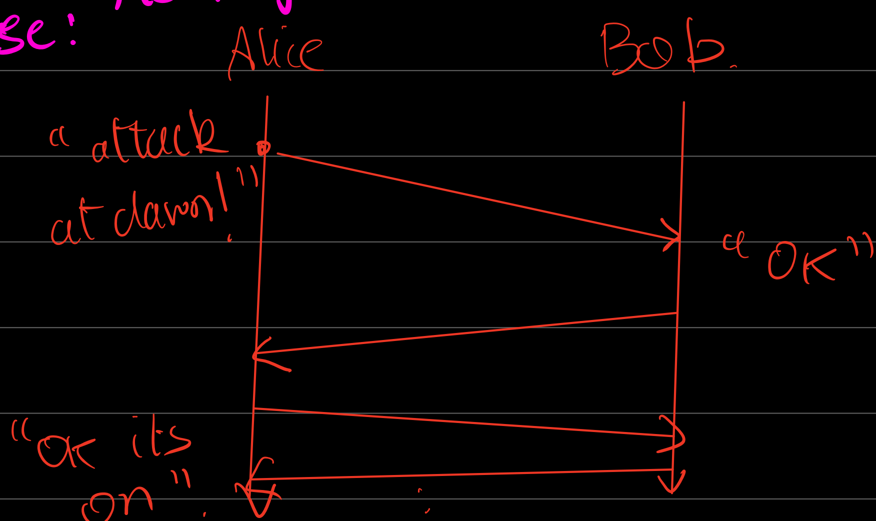


Alice & Bob can't defeat enemy on their own. They need to work together!

Problem: They are too far from each other to communicate directly. Instead, they will send a messenger through the valley with a message. The messenger:

- sometimes will be captured by enemy
- sometimes won't be captured.

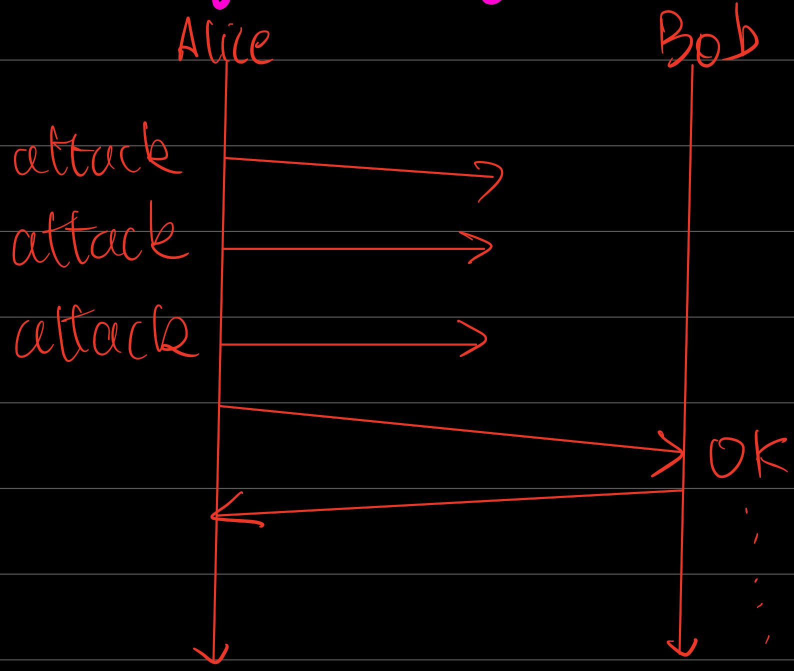
Best case: **No msgs lost**



Continues forever.

In the omission model, it is impossible for Alice or Bob to attack and know for sure that the other would attack

2nd case: Msgs at regular intervals



Once Bob sends "OK", alice stops sending messages. As time passes, Bob can grow more confident that alice has received the message

Good enough solution in practice.

Workaround #1: probabilistic certainity

Workaround #2: $\boxed{\text{common knowledge}}$

Case where Alice & Bob comm before reaching the hill.

Everyone knows p,
everyone knows that everyone
knows p

⋮

Continues forever.

TCP does not solve it, since it faces
the same common knowledge problem.