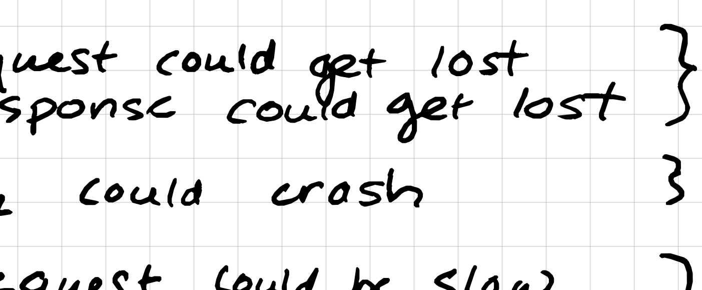
CSE138 Lecture 8

this time:

- ✓ fault classification and fault models
- ✓ the two generals problem
- ✓ "common knowledge"
- ✓ idempotence and "exactly-once" message delivery
- forms of fault tolerance
- reliable broadcast

(if time)

Faults



- request could get lost
- response could get lost } omission fault
- M_2 could crash } crash fault
- request could be slow
- response could be slow
- M_2 could be slow } timing fault
- M_2 could lie,
- messages could be corrupted (say, by an attacker!) } Byzantine fault

Fault classification

in a crash fault – a process fails by halting. (stops sending/receiving messages)

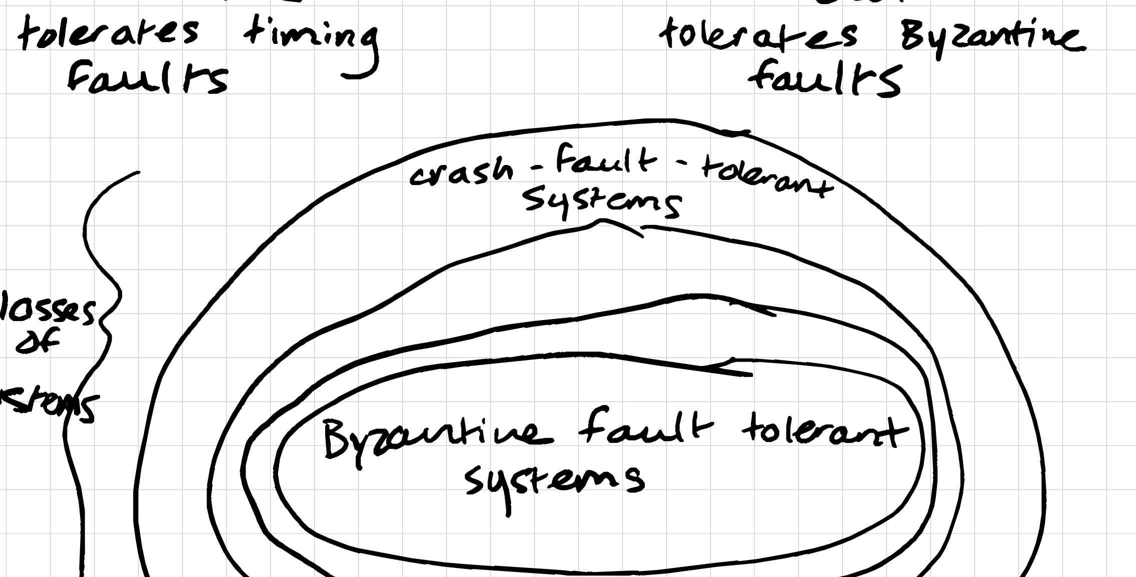
in an omission fault – a message is lost. (a process fails to send or receive a message)

in a timing fault – something happens "too late" ~~or "too early"~~

in a Byzantine fault – a process behaves in an arbitrary or malicious way.

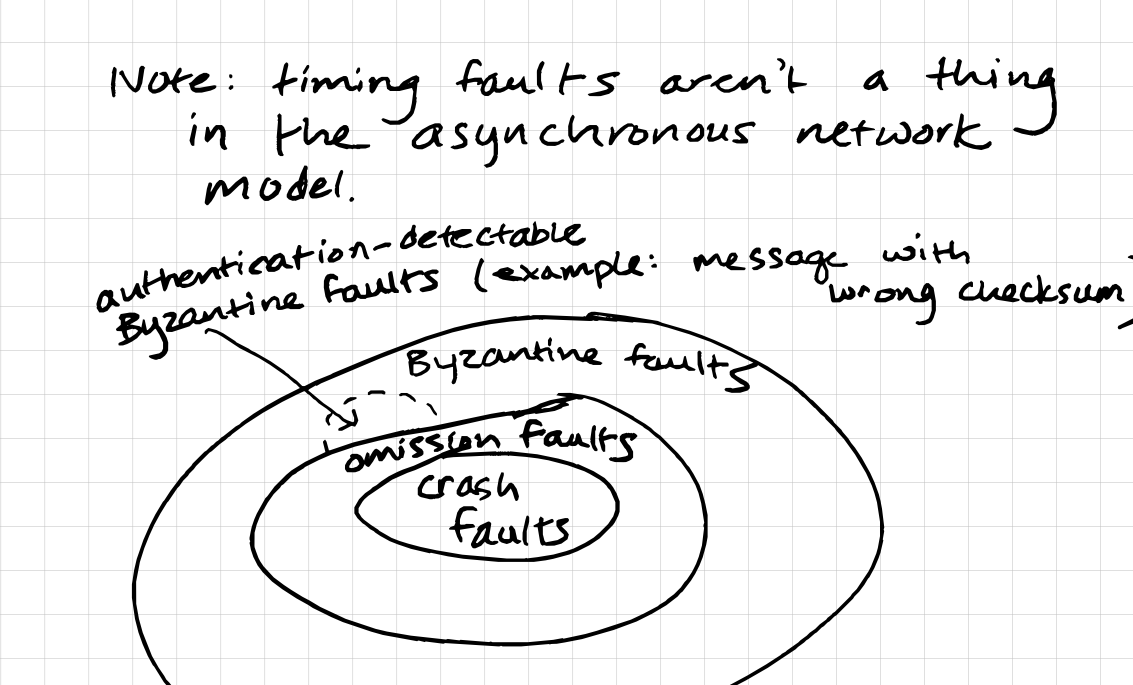
Protocol X tolerates crash faults

Protocol Y tolerates omission faults

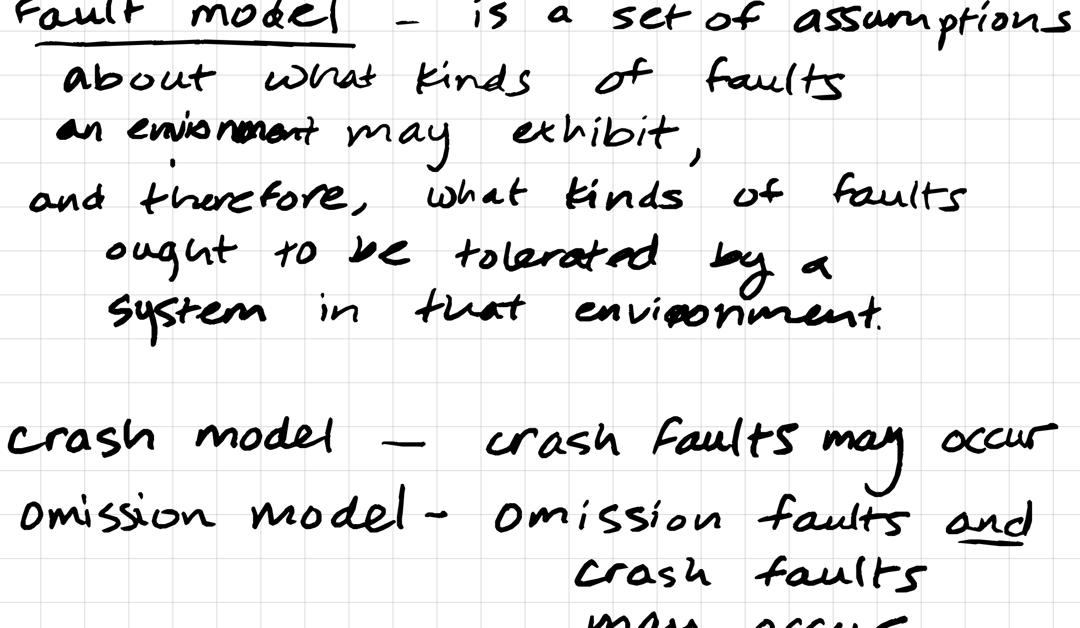


Protocol Z tolerates timing faults

Protocol U tolerates Byzantine faults



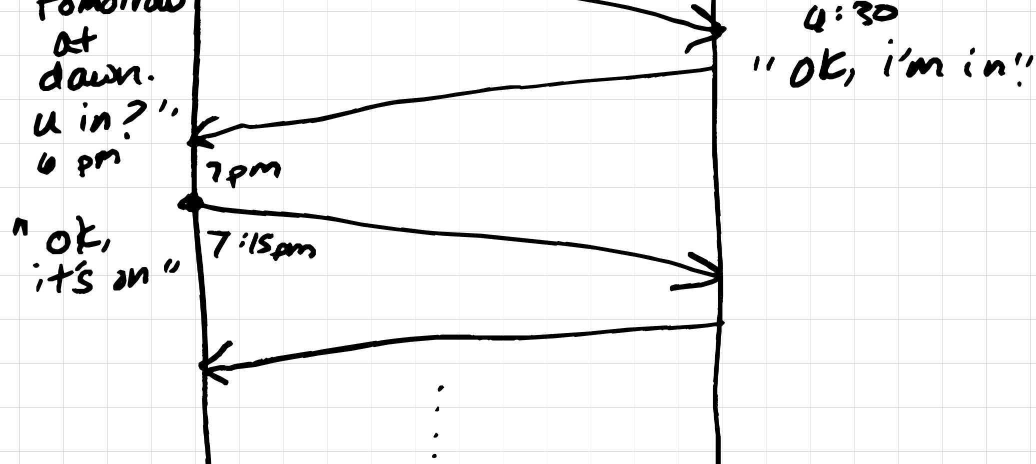
Note: timing faults aren't a thing in the asynchronous network model.

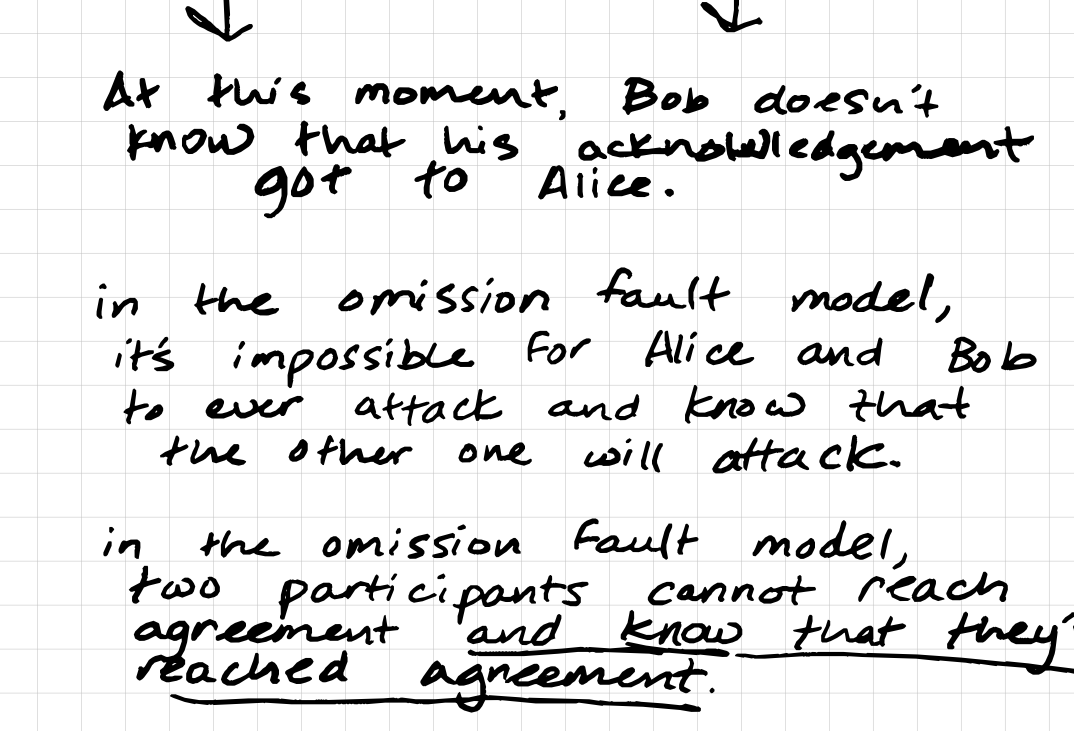


a fault model – is a set of assumptions about what kinds of faults an environment may exhibit, and therefore, what kinds of faults ought to be tolerated by a system in that environment.

- ☆ crash model – crash faults may occur
- ☆ omission model – omission faults and crash faults may occur
- Byzantine model – all the faults ¡¡

Classic thought experiment: the two generals problem





At this moment, Bob doesn't know that his acknowledgement got to Alice.

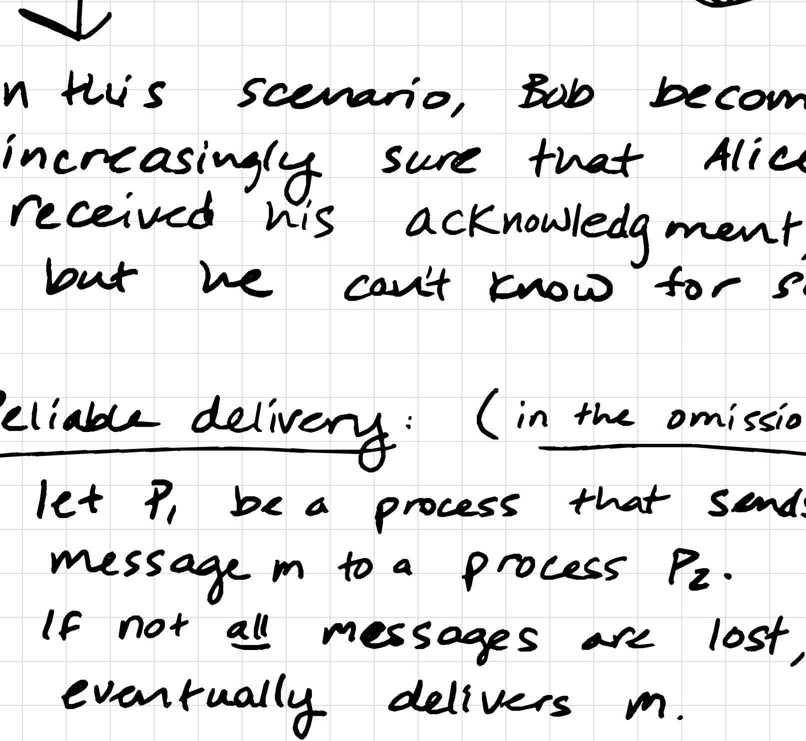
in the omission fault model, it's impossible for Alice and Bob to ever attack and know that the other one will attack.

in the omission fault model, two participants cannot reach agreement and know that they've reached agreement.

we say that there is "common knowledge" of some piece of information p if:

- everyone knows p,
- everyone knows that everyone knows p,
- everyone knows … everyone knows p

↑ infinite # of "everyone knows that"



in this scenario, Bob becomes increasingly sure that Alice received his acknowledgment, but he can't know for sure.

Reliable delivery: (in the omission model)

let P_1 be a process that sends a message m to a process P_2.
If not all messages are lost, P_2 eventually delivers m.

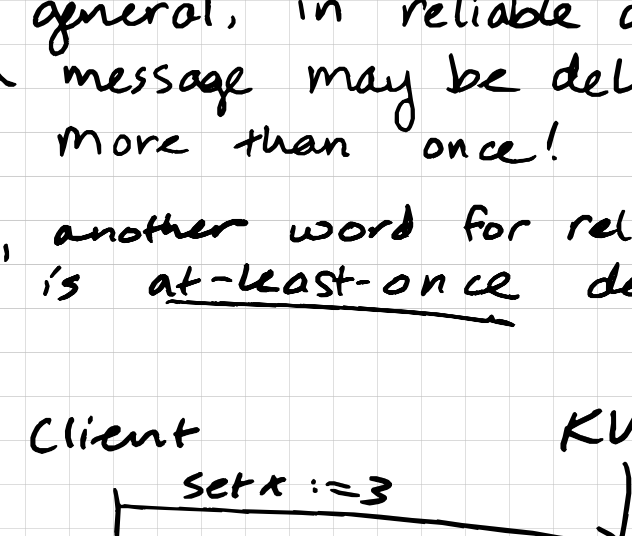
Implementing reliable delivery:

To send a message:

- put it in a send buffer.
- whenever a given timeout expires, send what's in the buffer.
- whenever an ack is received for a sent message, delete it from the send buffer.

To receive & deliver a message:

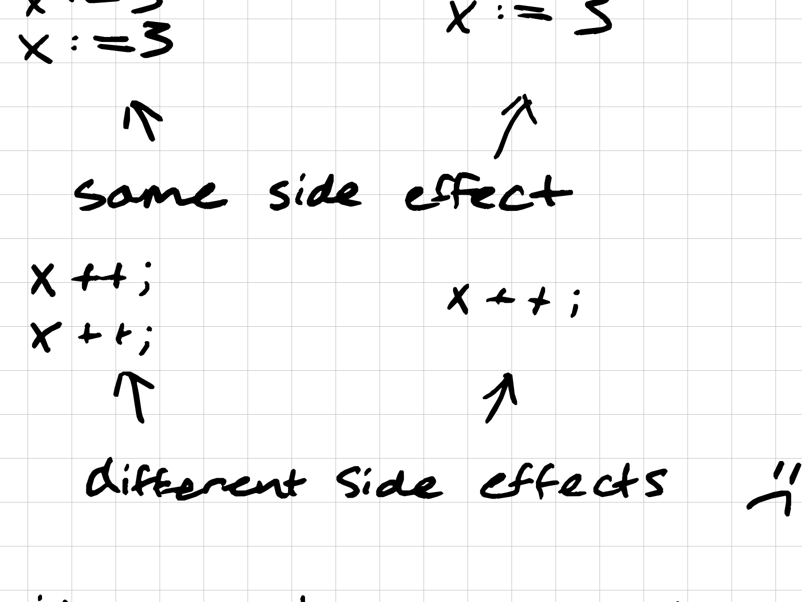
- Ack the sender.



Bob could get Alice's message twice.

In general, in reliable delivery, a message may be delivered more than once!

So, another word for reliable delivery is at-least-once delivery.



x := 3
x := 3

x := 3

↑ same side effect ↑

x++;
x++;

x++;

↑ different side effects ↑ ¡¡

An idempotent message is one that can be delivered multiple times and have the same effect on the delivering process as if it had been delivered just once.

Forms of fault tolerance

A system has safety and liveness properties that should hold.

e.g. C-L snapshot algorithm
- safety: any snapshots you take are consistent snapshots
- liveness: actually terminates

say that a marker message gets lost.

liveness is compromised.

but safety is not!

the best form of fault tolerance preserves both safety and liveness

but sometimes we have to make a choice.