paxos!

algorithm

consensus.

the consensus problem can be stated in a basic, generic manner: One or more systems may propose some value. How do we get a collection of computers to agree on exactly one of those proposed values?

consensus is a situation, if a collection of computers are asked to make a decision and all have to agree on single value.

consensus means agreement.

consensus.

paxos algorithm solves this problem for choosing a single value to attain consensus.

consensus deals with network that is unreliable and asynchronous - messages may get lost or arbitrarily delayed.

the formal properties for asynchronous consensus are:

- **validity.**
- **uniform agreement.**
- **integrity.**
- **termination.**

paxos.

paxos is an algorithm that is used to achieve consensus among a distributed set of computers that communicate via an asynchronous network

paxos simply selects a single value from one or more values that are proposed to it and lets all other nodes know what that value is.
a run of the Paxos protocol results in the selection of single proposed value.

paxos provides a feature called **abortable consensus.**
this means that some processes abort the consensus if there is contention while others decide on the value.

paxos.

assumptions for the algorithm are:

- concurrent proposals
- validity
- majority rule
- unicasts
- announcement

paxos.

paxos has three entities:

- proposers
- acceptors
- learners

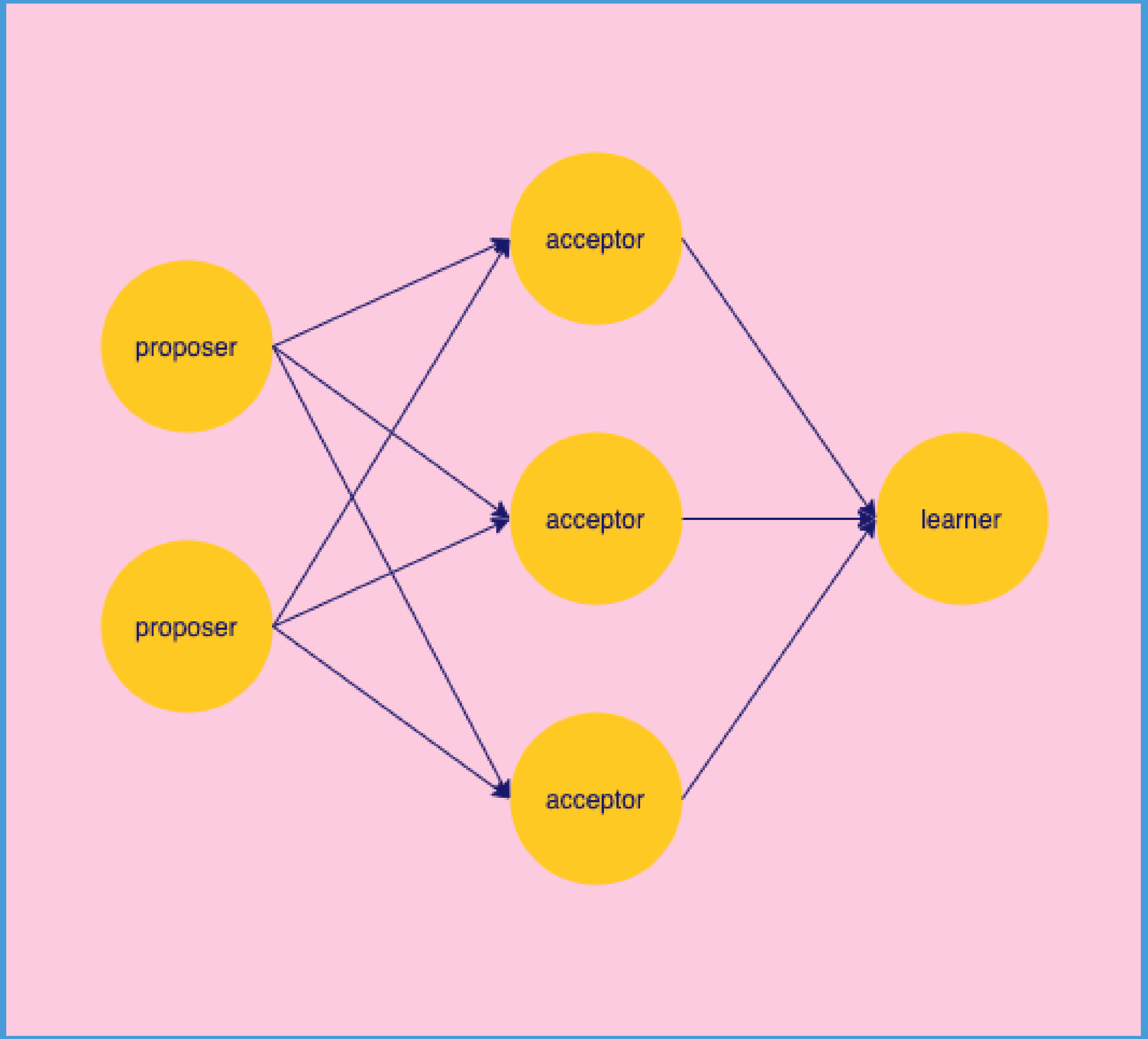
consider an env with multiple connected systems, where one or more of these nodes may concurrently propose a value.

we will have to communicate between proposers and acceptors for achieving consensus which needs protocols to figure it out

paxos.

distributed consensus protocol architectures:

- multiple proposers, single acceptor
- multiple proposers, multiple acceptors – FT



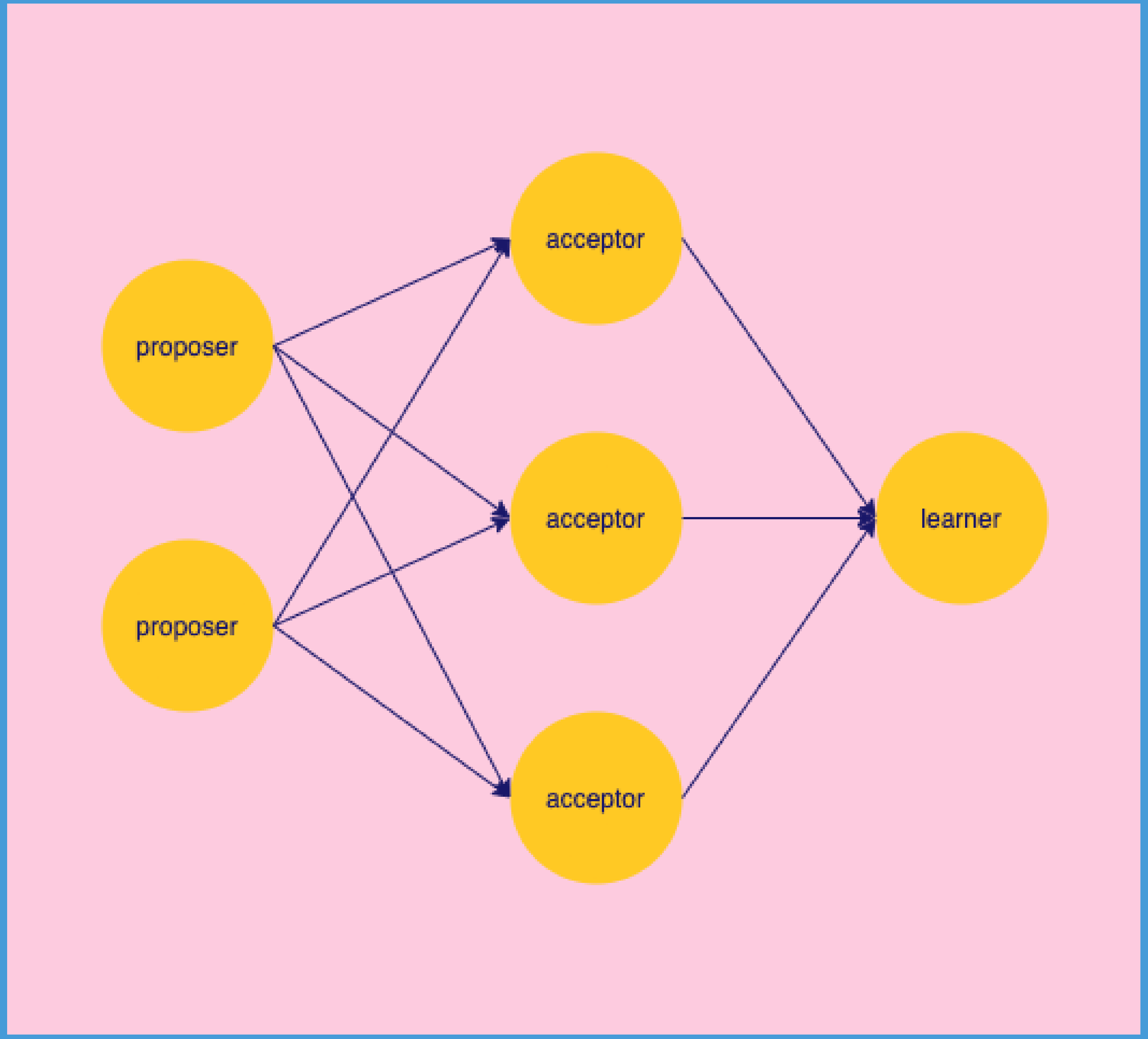
paxos.

multiple proposers, multiple acceptors - FT

if a **quorum**, or **majority**, of the acceptors chooses a particular proposed value then that value will be considered chosen.

if an acceptor **crashes** after it accepted a proposal, other acceptors know that a value was chosen.

instead of just having a proposer propose a value, we will use a **two-phase protocol**



paxos.

two-phase protocol:

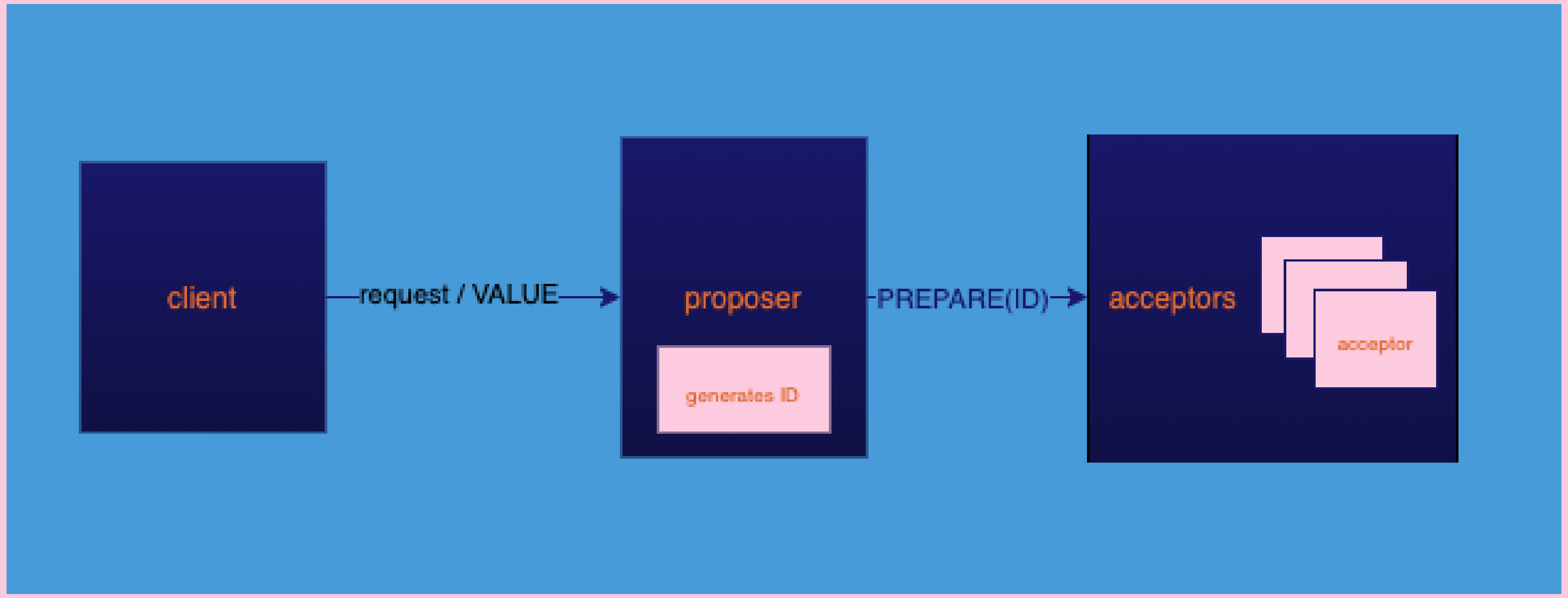
- phase 1: **PREPARE - PROMISE**
- phase 2: **PROPOSE - ACCEPT**

paxos.

phase 1_
PREPARE - PROMISE :

a proposer receives a consensus request for a VALUE from a client

proposer creates a unique proposal number, ID, and sends a PREPARE(ID) message to at least a majority of acceptors.

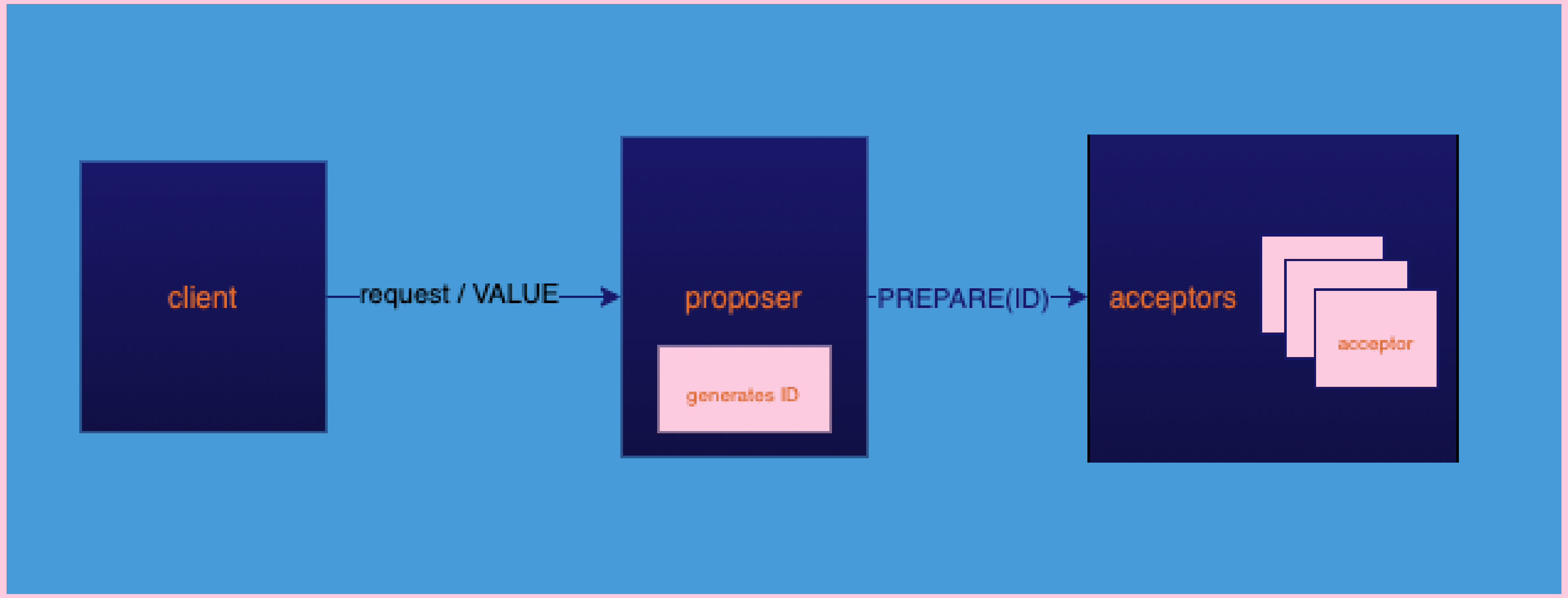


ID should be:

- unique for each request, for that it uses a global process identifier for the least significant bits of the number
- must be bigger than any previously used identifier used in the cluster.

paxos.

phase 1_
PREPARE - PROMISE :



each acceptor that receives the **PREPARE** message looks at the **ID** in the message and decides:

```
Is this ID bigger than any round I have previously received?
    If yes
        store the ID number, max_id = ID
        respond with a PROMISE message
    If no
        do not respond (or respond with a "fail" message)
```

paxos.

phase 2_
PROMISE - ACCEPT :

if a proposer received a **PROMISE** message from the majority of acceptors, it now has to tell the acceptors to accept that proposal.

if not, it has to start over with another round of Paxos.

the proposer tells all the acceptors what value to accept.

it sends
PROPOSE(ID, VALUE)
to a majority or all of the acceptors.

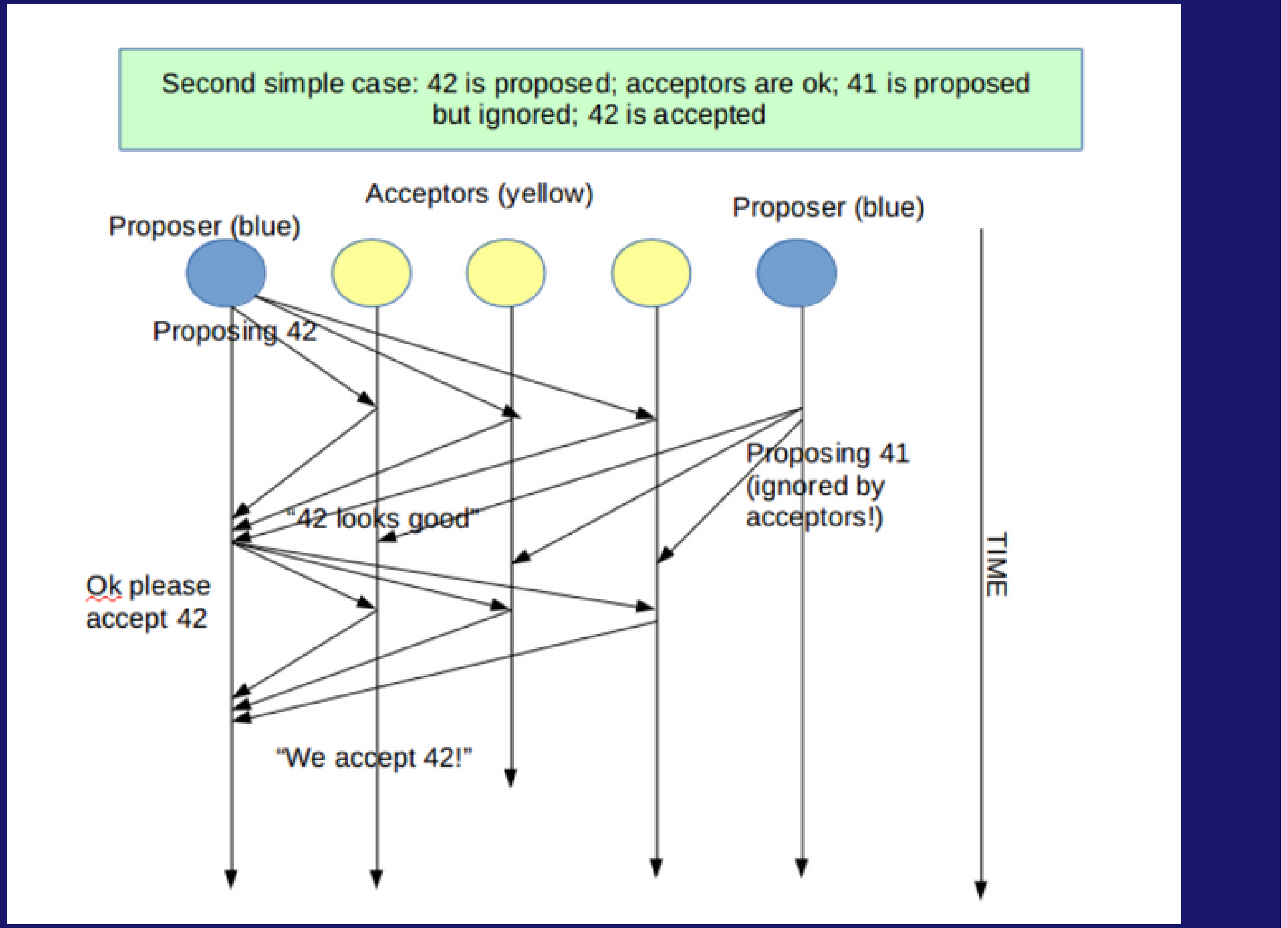
each acceptor now decides whether to accept the proposal
it accepts the proposal if the ID number of the proposal is still the largest one that it has seen

paxos.

phase 2_
PROMISE - ACCEPT :

an acceptor promised not to accept proposals from PREPARE messages with **smaller numbers** but can and will **accept proposals with higher numbers.**
the logic the acceptor uses is:

```
is the ID the largest I have seen so far, max_id == N?
if yes
    reply with an ACCEPTED message & send ACCEPTED(ID, VALUE)
to all learners
if no
    do not respond (or respond with a "fail" message)
```


Second simple case: 42 is proposed; acceptors are ok; 41 is proposed but ignored; 42 is accepted
Proposer (blue)
Acceptors (yellow)
Proposer (blue)
Proposing 42
Proposing 41 (ignored by acceptors!)
"42 looks good"
TIME
Ok please accept 42
"We accept 42!"

paxos for the real world.

group management.

cluster of systems that are running Paxos needs to be administered

Each proposer needs to know the **set of acceptors** so it can communicate with them and needs to know the learners

byzantine failures.

usually systems running paxos, doesn't suffer byzantine failures, either they run and communicate correctly or they stay silent

however these do exist, we can guard against network problems with mechanisms such as checksums or, in case of malicious interference, use digital signatures

in the distributed system env, misbehaving proposer that may inadvertantly set its proposal ID to **infinity**

got
questions?