Q: Can failed nodes re-enter and participate in the network if they come back online?

A: *The Paxos protocol allows failed nodes to re-enter and participate in once they're back online. To see how this happens, we need to recall some of the basic facts about Paxos; specifically,*

a. *When a Proposer wants to propose a value, it sends a Prepare request message - the argument of this message is an id which is some unique number (i.e. it hasn't been used by another Proposer) to the majority (or all) Acceptors.*

b. *If there is a timeout or no acknowledgment of this from the Acceptors, the Proposer re-attempts with another Prepare request that has a higher ID number.*

c. *When an Acceptor receives a Prepare request message that is greater than what it has received before, (i) it will ignore all requests with a lower ID and (ii) respond with a Promise message that includes the highest-numbered accepted proposal that the Acceptor has seen so far.*

d. *If the Proposer obtains Promise messages from the majority, it sends an Accept request that includes the proposal with the highest number among the Promise responses.*

e. *If the Acceptors have not promised to ignore that ID, they then multi-cast an Accept response to both the Proposer and learners that includes the current ID and the value being accepted.*

The key thing to note here is that (c. ii) allows the Proposer to find out if the network has already achieved consensus on some value which it should accept.

Now let's go back to the case where a node has failed (and has missed consensus updates).

In this case, when the node comes back online and, when that node sends a request to the quorum of acceptors (Points a+c), having missed the previous updates, it sending a lower ID will result in its request being ignored. However, that node will re-attempt with a higher ID (and continue to do so) until it reaches an ID that is higher than what the quorum has previously seen (Point b). Eventually, it will reach an ID that allows its request to be accepted by the quorum. The key thing to note here is that the Proposer and Acceptors

include in their responses the previously committed value with the highest ID seen so far (Points c+d). Also, because the Acceptors inform the Proposer of a proposal value that is higher than what it had previously seen, in this case, the value the Proposer will end up proposing will correspond to the latest consensus update.

Q: What is going on in slide 69 of the Paxos lecture?

A: *In this slide, we're considering how to resolve a Consensus Problem when we have different proposals (c.f. my bank transfer example earlier today).*

First thing to note is that in Paxos we have 3 phases. We'll look at the first two for now: the Election Phase which is named Phase 1 in Paxos terminology. This splits into two parts, the Prepare Phase (Phase 1a) and the Promise Phase (Phase 1b). After Phase 1 is the Bill Phase or Phase 2. This phase also splits into two parts, an Accept Phase (Phase 2a) and an Accepted Phase (Phase 2b). We discuss this in slide 56.

Prepare Phase:

I. *A Proposer initiates the consensus process by sending a Prepare Message with a Proposal Number to the acceptors.*
II. *On receiving the Prepare Message, each Acceptor checks if the Proposal Number is greater than any previous Proposal Number it has seen.*

Promise Phase:
III. After receiving prepare messages, Acceptors respond with promise messages to the Proposer, in particular, if the Proposal Number is greater, the Acceptor replies with a Promise not to accept any proposals with lower numbers.
IV. The Promise Message includes the Acceptor's acceptance status and the highest-numbered proposal it has accepted (if any).
V.If an Acceptor has not accepted any proposals before, it responds with a Promise but with any accepted value.
VI. If an Acceptor has accepted a proposal before, it includes the Proposal Number and value in its Promise.

Accept Phase:

VII. *If a Proposer receives promises from a majority of acceptors (a quorum), it moves on to the Accept Phase.*
VIII. *In the Accept Phase, the Proposer sends an Accept Message to the acceptors with the same Proposal Number and the value it wants to be accepted.*
IX. *Acceptors check if the Proposal Number is not lower than any previous Proposal Number they have seen. If the Proposal Number is valid, the Acceptor accepts the Proposal and informs the Proposer.*

In slide 69, we're considering a setting where different acceptors want to promote different values, here Acceptor 1 (A1) wants to promote value B, Acceptor 2 (A2) wants to promote value C, and Acceptor 3 (A3) wants to promote value C. We're also assuming that the highest-numbered proposal accepted so far for A1 is 2, for A2 is 3 and for A3 is 4.

Now let's go through the steps:
I: *First, a Proposer selects a proposal number, in our case P1 is the Proposer and the Proposal Number is 5. It then sends this as a Prepare(5) request to the quorum (majority of acceptors).*

II: *The Quorum checks the proposal number - here the biggest previous number is 4 so this is OK.*

III,V: *The Acceptors then send back a Promise to the Proposer to never accept a future proposal less than the current one (5 in our case) (III). Each Promise note contains two terms appended to the current Proposal Number (5). The two terms are the highest number proposal that that Acceptor has seen so far and what that proposal is (V); so taking the value that each acceptor wants to promote and the Proposal Number we get for A1 the Promise Response is (2,B) for A2 it's (3,C) and A3 it's (4,D).*

Note that the current proposal number is still 5.

Next, Phase 2a:
VII: *The Proposer has received Promise responses from the quorum so moves to the Accept Phase.*

VIII: *The Proposer wants D to be accepted since it is the value with the highest Proposal Number along with the current Proposal Number (5) so it sends an Accept Message (5,D).*

IX: *The Acceptors check this Proposal Number is not lower than what they have seen, since here 5>4, 5>3 and 5>2 all Acceptors OK it and each informs the Proposer that it is accepted.*

The last part involves communicating the decision (which is that the Proposal for the value D has been accepted) to all nodes in the system, including learners and proposers - this is behind the arrow going down to the Learner L.

Q: Why does the proof of stake protocol help to encourage honest participation in a way that proof of work does not?

A: *In the proof of stake protocol, validators are selected in proportion to the quantity of their crytpo holdings. Participants with a larger share of the network are more likely to act honestly so as not to harm the network and cause a decline in the value of their holdings. Overall, influencing validation outcomes requires having a holding on the currency itself.*

Q: What is a majority in Paxos?

A: *In the vanilla variant of Paxos, given a set of N nodes, a count n is majority if n is greater than or equal to [N/2] +1 where [.] means the integer part e.g. [10/3]=3.*

Note that the concept of majority is different (more restrictive) that the largest subset. E.g. if a set of 20 nodes forms subsets of 9 nodes 6 nodes and 5 nodes, we do not have a majority.

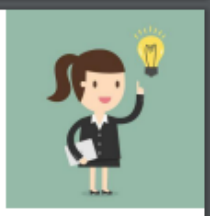
Q: In the image below, why do you add 1 to 45 for the total cloud time? This is slide 89 of week 6.

Example 2

- A biology lab generates 1TB of data per experiment
- One EC2 instance takes 2 hours per GB to process the data
- The are the equivalent of 25 instances locally
- The network transfer rate is 50Mbit/s
- Is it quicker in the cloud or locally?

89

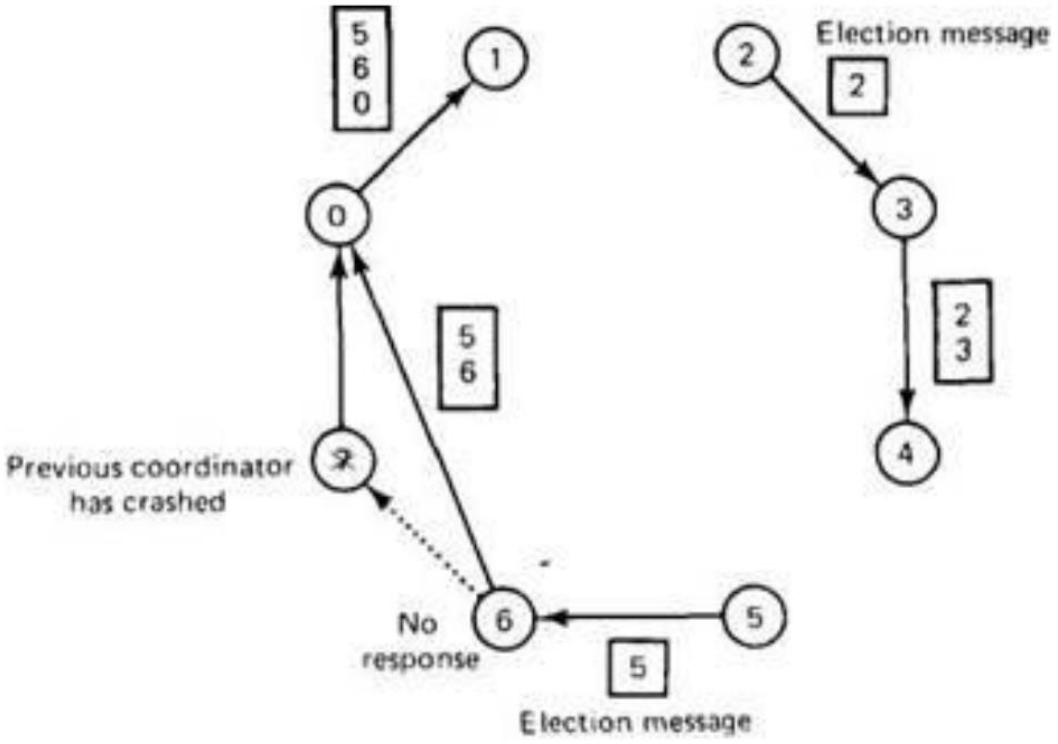
Example 2



- Local Computation time is 1000x2/25 = 80 hrs
- Transfer time is (1000GB x 1000MB/GB x 8bits/Byte)/50Mbits/sec = 160,000 second or 45hrs
- Total Cloud time is 45+1= 46hrs
- May be advisable to move to cloud (Need to consider cost, age of local hardware and other factors)

A: *The 45 hours we calculated in the slide is just the transfer time. To make the comparison with doing the computation locally, we need to calculate the total time which we can get by adding +1 hour to the transfer time (we did the same for Example 1 on slide 87).*

Ring Algorithm



Q: Can you please explain the steps that occur in the above diagram? In this diagram case, will the new coordinator be 6?

A: *Here is the explanation for your question about the Ring algorithm.*

To understand the example on the slide, we need to first of all state the key properties of the method:

I) Each process has its own unique ID number.
II) All processes in the ring are connected to a successor and predecessor process.
III) The processes are arranged in a logical ring and each process knows the ring order.
IV) Within an election cycle, the process with the highest ID number will be elected as the coordinator.
V) Processors skip faulty nodes: upon detecting that a node is down, process n will first send its message to process n+1, if after some fixed period process n+1 is unresponsive, process n will forward its message directly to process n+2.

And the key steps of the algorithm:

1. In the Ring method, the algorithm starts a process of selecting a new coordinator once any process connected in the ring realises that the coordinator is down.
2. The initiating process (i.e. the one that notices the coordinator is down first) prepares an election message that contains its own ID number and then sends this message to its successor in the ring.
3. On receiving an election message, each process appends their own ID to the message before forwarding the message to its successor.
4. The election cycle is terminated once a process receives an election message that contains its own ID.
5. When the election cycle is terminated, the initiator of the election will forward an election message to its successor. Each node will forward the election message to their successor until the election message has been circulated among all active members.

So let's go through the example on the slide.

In this example, Process 7, which was the coordinator has crashed. Processes 5 and 7 have noticed this and according to Step 1, both have initiated elections.

Process 2 has sent its election message with its ID to Process 3, and Process 3 has sent the election message to Process 4 (both Process 2 and Process 3 have appended their IDs as acc. to Step 3). Points I - III make this possible (Steps 2,3).

Process 5 has prepared an election message which it has sent to Process 6. Process 6 has appended its own ID to the election message to Process 7) (Steps 2,3) but Process 6 has not received a response in a fixed time period. Consequently, Process 6 sent the election message to Process 7 (c.f. Point V).

The election messages will continue to progress through the ring, eventually, the election message initiated by Process 5 will reach Process 5 with the ID numbers [5,6,0,1,2,3,4] and the election message initiated by Process 2 will reach 2 with the ID numbers [2,3,4,5,6,0,1] (Point 4).

In both cases, the highest ID contained in the returned election message is 6 so both Process 2 and Process 5 ***will elect Process 6*** *(Point IV) by sending election messages to their successor nodes (Step 5).*