14-736: DISTRIBTED SYSTEMS

LECTURE 8 * SPRING 2019 * KESDEN

COORDINATOR SELECTION: SELECTING A "SPECIAL HOST"

- Given *N* available hosts, where *N* isn't *1*, how do we pick one for a different role, e.g. coordinator, front-end server, etc?
  - Appoint one: A human simply picks
  - Elect one: Participating hosts pick

APPOINTING A COORDINATOR

- A human, e.g. system administrator, picks from available hosts
- Advantages:
  - Simple
  - Minimal development time
  - Agile
- Disadvantages
  - Slow, i.e. human speed
  - Requires human to detect failure, understand cause, determine participants, and react.

ELECTING A COORDINATOR

- The participants determine the need to pick a (new?) coordinator
- Participants "discuss" it among themselves
- Participants agree on coordinator
- New coordinator takes charge
- Advantages:
  - Automatic
- Disadvantages
  - Complexity (partitionings, etc)
  - Network traffic (storms)
  - Failure mode can be complex, e.g. many coordinators or none

BULLY APPROACH
(GARCIA-MOLINA '82)

- Probably the most common
- Simplest
- Can lead to storms (We'll see why)

BULLY ALGORITHM

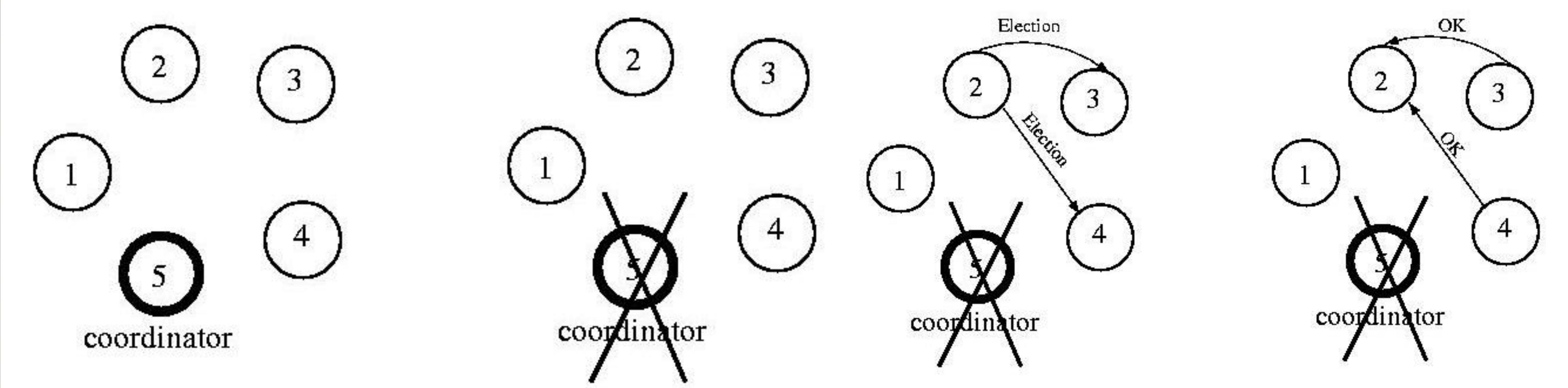
- Assumptions:
  - All messages are delivered within some T_m units of time, called the message propagation time.
  - Once a message is received, the reply will be dispatched within some T_p units of time, called the message handling time.
  - T_p and T_m are known.
- These are nice, because together they imply that if a response is is not received within $(2*T_m + T_p)$ units of time the process or connection has failed.
  - But, of course, in the real world congestion, load, and the indeterminate nature of most networks mean that a good amount of "slop" needs to be included.

BULLY ALGORITHM

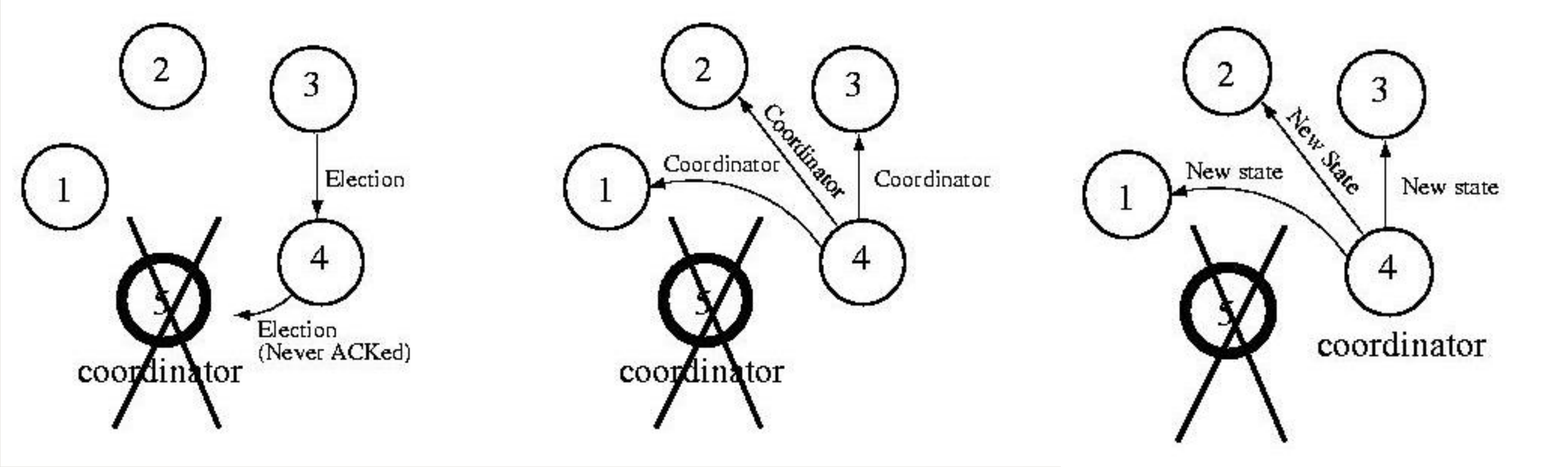
The idea behind the Bully Algorithm is to elect the highest-numbered processor as the coordinator.

- If any host thinks that the coordinator has failed, it tries to elect itself by sending a message to the higher-numbered processors.
- If any of them answer it loses the election.
  - At this point each of these processors will call an election and try to win themselves.
- If none of the higher-ups answer, the processor is the highest numbered processor, so it should be the coordinator.
  - So it sends the lower level processors a message declaring itself the coordinator
  - After they answer (or the ACK of a reliable protocol), it starts doing its job as coordinator
    - E.g. It starts to query participants to find out what they know, then begins providing coordination, etc.
- If a new processor arrives, or recovers from a failure, it gets the state from the current coordinator and then calls an election
  - Or, for efficiency, just remains a participant and lets the new coordinator lead, until it fails, etc.

BULLY ALGORITHM



BULLY ALGORITHM



INVITATION ALGORITHM: ASSUMPTIONS AND GOAL

- Goal: The *Invitation Algorithm* provides a protocol for forming groups of available participants within partitions, and then merging them into larger groups as partitions heal or failed coordinators are returned to service.
  - In many ways, it is like a self-healing, partitionable *Bully Algorithm*
- Assumption: In practice communication failure, high latency, and/or congestion can partition a network.
- Assumption: A collection of participants under the direction of a coordinator, can perform useful work, even if other such groups exists.
  - In other words, partitioned participants can still organize and make progress

INVITATION ALGORITHM: ASSUMPTIONS AND GOAL

- Groups are named using a *group number*.
  - The group number is unique among all groups, is changed every time a new group is formed, and is never reused.
  - To accomplish this, the group number might be a simple sequence number attached to the processor ID.
  - The sequence number component can be incremented each time the processor becomes the coordinator of a new group.
- Basic idea:
  - Partitioned participants (or whole group) elect their own coordinator
  - Coordinators "yell out" periodically to participants outside their group asking each if it is a coordinator.
  - When a coordinator answers, the coordinators pick one to coordinate, and merge their groups, electing a new coordinator
    - Choosing the coordinator is Bully-like, with higher nodes winning.

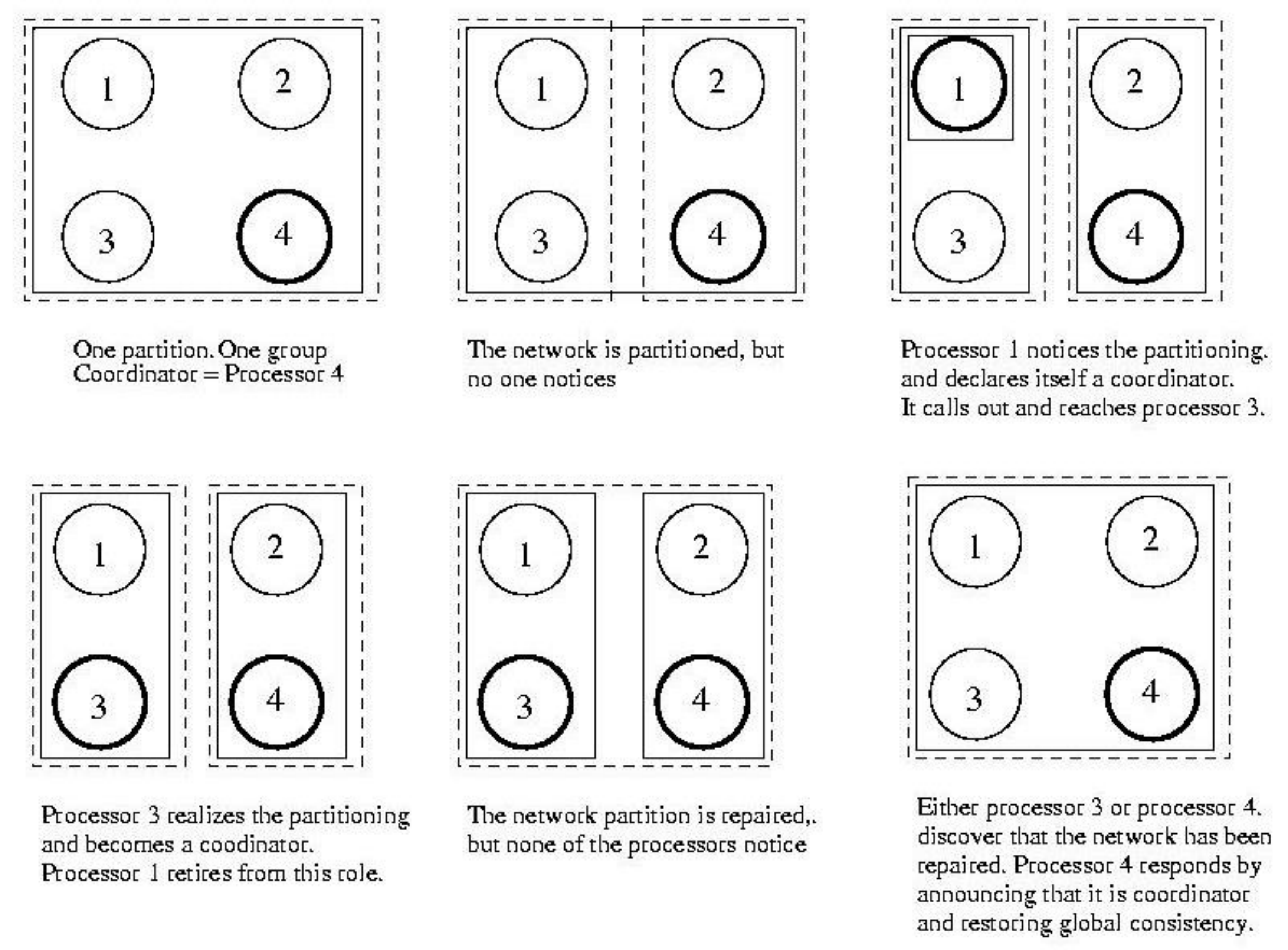
INVITATION ALGORITHM: MERGING GROUPS

- One might think that it is acceptable for the coordinator that initiated the merge to be the coordinator of the new group.
- But it might be the case that two or more coordinators were concurrently looking for other coordinators and that their messages may arrive in different orders.
- To handle this situation, there should be some priority among the coordinators -- some method to determine which of the perhaps many coordinators should take over.
  - One way of doing this might be to use the processor ID to act as a priority.
  - Perhaps higher-numbered processors ignore queries from lower-level processors.
  - This would allow lower-level processors to merge the groups with lower priority coordinators during this operation.
  - At some later time the higher-level coordinators will each act to discover other coordinators and merge these lower-priority groups.
  - Perhaps receiving the query will prompt the higher-level coordinator to try to merge its group with others sooner than it otherwise might.
  - An alternative would be for a coordinator only to try to merge lower-level coordinators.

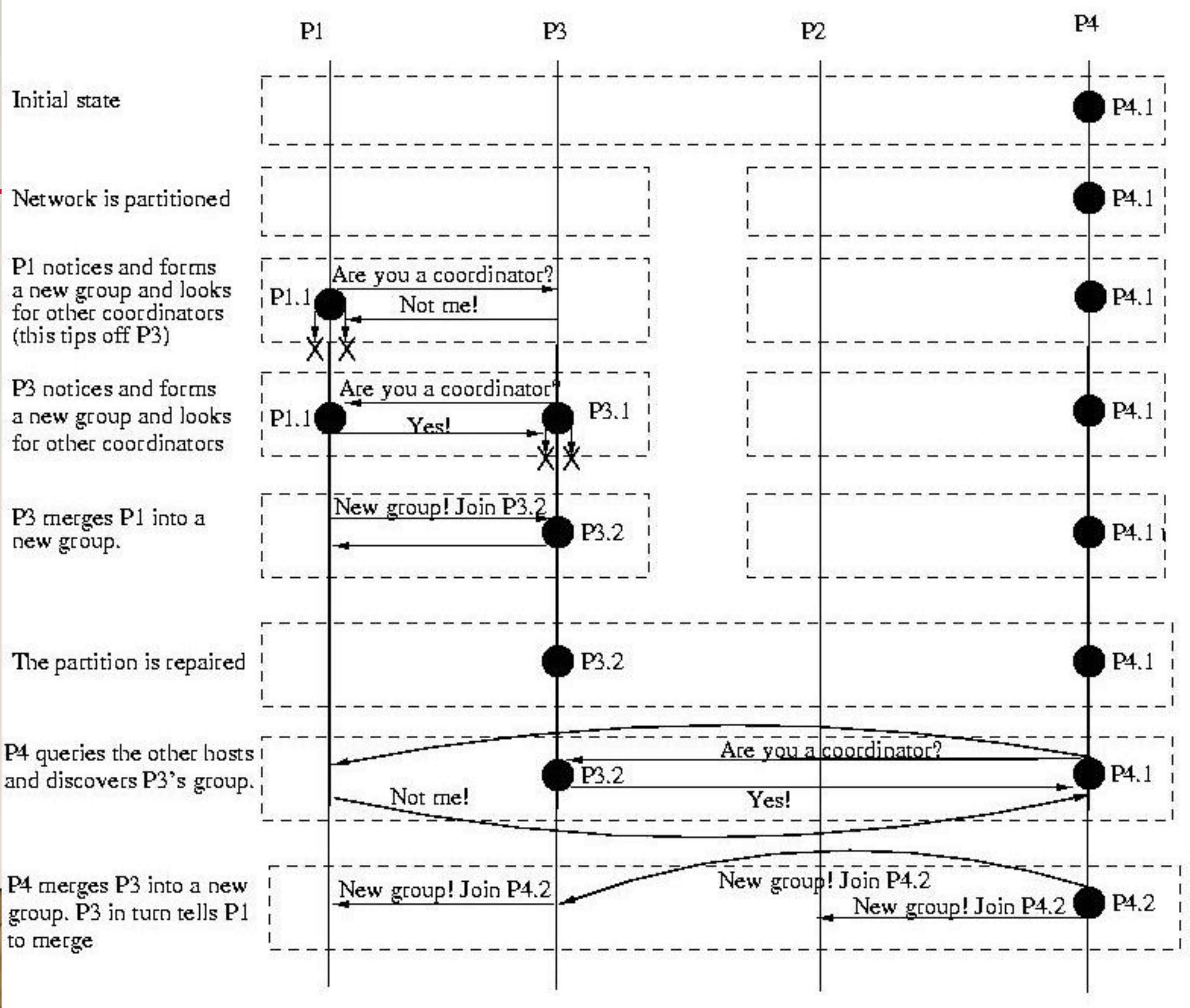
INVITATION ALGORITHM: OPTIONS

- Perhaps higher-numbered processors ignore queries from lower-level processors.
  - This would allow lower-level processors to merge the groups with lower priority coordinators during this operation. At some later time the higher-level coordinators will each act to discover other coordinators and merge these lower-priority groups.
  - Perhaps receiving the query will prompt the higher-level coordinator to try to merge its group with others sooner than it otherwise might.
- An alternative would be for a coordinator only to try to merge lower-level coordinators.
- Or perhaps processors delay some amount of time between the time that they look for other coordinators and the time that they start to merge these groups.
  - This would allow time for a higher-priority coordinator to search for other coordinators (it knows that there is at least one) and ask them to merge into its group.
  - If after such a delay, the old coordinator finds itself in a new group, it stops and accepts its new role as a participant.
  - In this case, it might be useful to make the delay inversely proportional to one's priority. For example, there is no reason for the highest-priority processor to delay. But the lowest priority processor might want to delay for a long time.

INVITATION ALGORITHM



INVITATION ALGORITHM



RING ELECTION

- Another approach, Ring election, is very similar to token ring synchronization, except no token is used.
- Assumptions:
  - We assume that each processor is logically ordered, perhaps by IP address, so that each processor knows its successor, and its successor's successor, and so on.
  - Each processor must know the entire logical structure.

RING ELECTION

- When a processor discovers that the coordinator has died, it starts circulating an ELECTION message around the ring.
- Each node advances it in logical order, skipping failed nodes as necessary.
- Each node adds their node number to the list.
- Once this message has made its way all the way around the ring, the message which started it will see its own number in the list.
- It then considers the node with the highest number to be the coordinator, and this messages is circulated.
- Each receiving node does the same thing. Once this message has made its way around the ring, it is removed.
- If multiple nodes concurrently discover a failed coordinator, each will start an ELECTION.
- This isn't a problem, because each election will select the same coordinator. The extra messages are wasted overhead, while this isn't optimal, it isn't deadly, either.

RING ELECTION