Distributed Coordination — Last Time

- Happens before relationship
- Centralized and distributed semaphores for distributed systems

Distributed Coordination — Today

- Failure recovery

Recovery from Failure

Centralized system: reboot (if you are down, you are down)

Distributed system: when a node is down, the system is only partially down.

- failure detection
- reconfiguration
  - Election algorithms
  - Reaching agreement

Failure Detection:

- Site A sends a message to site B and receives no response. How do we tell if
  1. the message is lost, or
  2. site B is down, or
  3. the communication link to B is down?
- Try a handshake:
  - A $\rightarrow$ B: asking "are you up?"
  - B can reply "I am up."
  - If there is no response, wait a while and try again. Repeat some maximum number of times.
  - Does not differentiate between types of failures.

Reconfiguration for a Distributed System

If A believes B has failed:

- A must notify all other processors.
- If B performs a critical system function (deadlock detection, holds the token in a ring, or had centralized mutual exclusion control, ...), remaining processes must elect a replacement. B is called a *coordinator*.
- Assume each site is given a priority number, and the coordinator always has the highest number.
- If the coordinator fails, all processes must determine the process with the highest priority number. That process should start the replacement coordinator process.

Bully Election Algorithm

Suppose A is the highest numbered active process, but it doesn't know that for sure:

1. A believes that the coordinator is down.
2. A sends an election message to all higher priority (numbered) processes.
3. If A gets no message back within time T, it assumes they have all failed and elects itself the new coordinator.
   (a) It restarts the coordinator process, and
   (b) sends messages to all lower numbered processes to inform them it is now the coordinator.
4. When the lower-numbered processes get the message, they record this information so that they can direct future messages appropriately.

Bully Election Algorithm

Suppose A is not the highest numbered active process:

1. A believes that the coordinator is down.
2. A sends an election message to all higher priority (numbered) processes.
3. Some higher numbered process is alive and responds to the election message.
4. Upon receipt of the message, A starts a clock with timeout T', where T' > T.
5. The higher-numbered process repeats the election algorithm.
6. If A does not receive a message within time T', it repeats the election algorithm.
7. If A does receive a message within time T', the message identifies the new coordinator.

When a process recovers, it begins the election algorithm and wrestles control from the currently-active coordinator if the new process has the highest number.

Example of Bully Algorithm

Given P_1 thru P_5, and P_5 as coordinator. P_5 fails and P_2 is the first to detect the failure.

1.

2.

3.

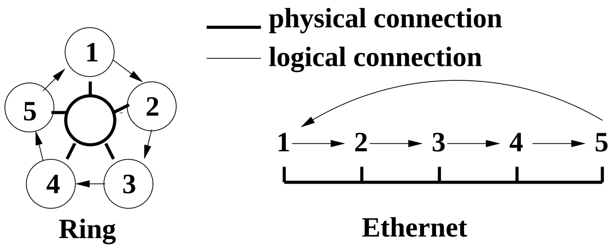
4.

5.

Electing a new leader in a virtual ring

A better algorithm than last time!

Assume a virtual network with a ring topology consisting of unidirectional links. When a node fails, the system can still send messages around the ring to the closest live neighbor in the virtual unidirectional ring.



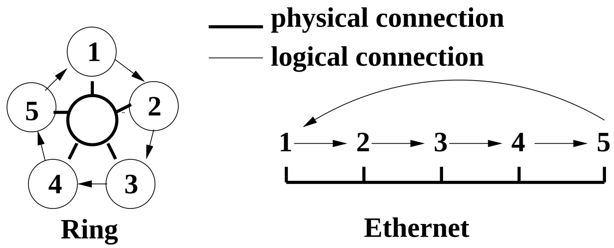
Again, assume that all nodes have a priority number and the largest becomes the new coordinator.

A believes C (the coordinator) has failed:

1. A creates an elect(A) message and sends it to its neighbor
2. When B receives an elect(A)
   (a) If B has not seen an elect message and Priority(B) ¡ Priority(A), B sends elect(A) to its neighbor.
   (b) Elseif B has not seen an elect message and Priority(B) ¿ Priority(A), B sends elect(B) to its neighbor (it substitutes itself in the elect message).
   (c) Elseif B has seen an election message, and if B = A, B is the new coordinator. B then sends King(B) to its neighbor.

$\Longrightarrow$ If X receives a King(B) message, it sets its coordinator to B and forwards King(B).

Example: Reconfiguration within a Ring



Given P_1 thru P_5, and P_5 as coordinator. P_5 fails and P_2 is the first to detect the failure.

Worst case?

Best case?

Summary of Distributed Coordination

- Synchronization is slower and even harder on a distributed system.
- Global knowledge vs. partial knowledge
- Failure detection and recovery are still very important research topics today.