Chapter 18 Distributed Coordination

- Event Ordering
- Mutual Exclusion
- Atomicity
- Concurrency Control
- Deadlock Handling
- Election Algorithms
- Reaching Agreement

Election Algorithms

- Determine where a new copy of the coordinator should be restarted
- Assume that a unique priority number is associated with each active process in the system, and assume that the priority number of process P_i is i
- Assume a one-to-one correspondence between processes and sites
- The coordinator is always the process with the largest priority number. When a coordinator fails, the algorithm must elect that active process with the largest priority number
- Two algorithms, the bully algorithm and a ring algorithm, can be used to elect a new coordinator in case of failures

Bully Algorithm

- Applicable to systems where every process can send a message to every other process in the system
- If process P_i sends a request that is not answered by the coordinator within a time interval T, assume that the coordinator has failed; P_i tries to elect itself as the new coordinator
- P_i sends an election message to every process with a higher priority number, P_i then waits for any of these processes to answer within T

Bully Algorithm (Cont.)

- If no response within T, assume that all processes with numbers greater than i have failed; P_i elects itself the new coordinator
- If answer is received, P_i begins time interval T', waiting to receive a message that a process with a higher priority number has been elected
- If no message is sent within T', assume the process with a higher number has failed; P_i should restart the algorithm

Bully Algorithm (Cont.)

- If P_i is not the coordinator, then, at any time during execution, P_i may receive one of the following two messages from process P_j
  - P_j is the new coordinator ($j > i$). P_i, in turn, records this information
  - P_j started an election ($j > i$). P_i, sends a response to P_j and begins its own election algorithm, provided that *Pi* has not already initiated such an election
- After a failed process recovers, it immediately begins execution of the same algorithm
- If there are no active processes with higher numbers, the recovered process forces all processes with lower number to let it become the coordinator process, even if there is a currently active coordinator with a lower number

Ring Algorithm

- Applicable to systems organized as a ring (logically or physically)
- Assumes that the links are unidirectional, and that processes send their messages to their right neighbors
- Each process maintains an active list, consisting of all the priority numbers of all active processes in the system when the algorithm ends
- If process Pi detects a coordinator failure, I creates a new active list that is initially empty. It then sends a message elect(i) to its right neighbor, and adds the number i to its active list

Ring Algorithm (Cont.)

- If P_i receives a message elect(*j*) from the process on the left, it must respond in one of three ways:
  1. If this is the first *elect* message it has seen or sent, P_i creates a new active list with the numbers *i* and *j*
     - It then sends the message *elect(i),* followed by the message *elect(j)*
  2. If $i \neq j$, then the active list for P_i now contains the numbers of all the active processes in the system
     - P_i can now determine the largest number in the active list to identify the new coordinator process
  3. If $i = j$, then P_i receives the message *elect(i)*
     - The active list for P_i contains all the active processes in the system
       - P_i can now determine the new coordinator process.