Distributed Systems

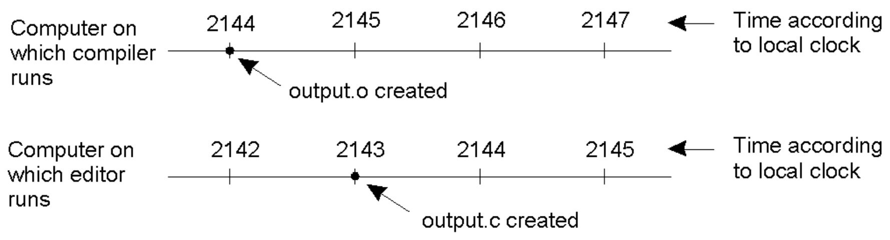
Delhi Technological University
Synchronization
Divyashikha Sethia
divyashikha@dce.edu

Objective:

•**Time Synchronization**

•**Mutual Exclusion**

Clock Synchronization


Computer on which compiler runs
2144
2145
2146
2147
Time according to local clock
output.o created
Computer on which editor runs
2142
2143
2144
2145
Time according to local clock
output.c created

When each machine has its own clock, an event that occurred after another event may nevertheless be assigned an earlier time.

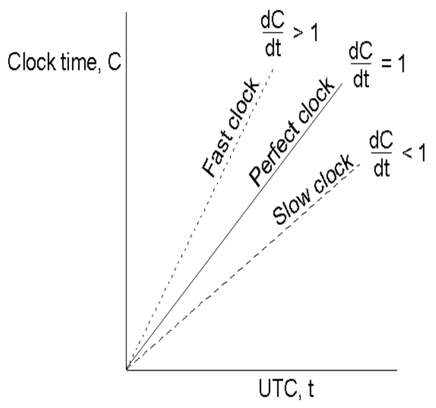
Example of Clock Difference

•Computer time with quartz crystal which oscillates and generates interrupts:
- Two registers: counter, holding register

•Multiple CPUs have crystals with slightly different frequencies causing the (software)

•Clocks gradually to get out of sync – Clock Skew

Clock Synchronization Algorithms



$$1 - \rho \le \frac{dC}{dt} \le 1 + \rho$$

•Constant p: maximum drift rate such that timer within specification
•Two clocks drifting from UTC in opposite direction after synchronization wil be 2p apart.
•Clock synch every 2p seconds

Relation between clock time &UTC (Universal Coordinated Time) tick at different rates.

GPS (Global Positioning System)

•GPS is a satellite-based distributed system
•used mainly for military applications
•many civilian applications, notably for traffic navigation
•GPS uses 29 satellites each circulating in an orbit at a height of approximately 20,000 km.
•Each satellite has up to four atomic clocks, which are regularly calibrated from special stations on Earth.
•Satellite continuously broadcasts its position, and time stamps each message with its local time.
•Broadcasting allows every receiver on Earth to accurately compute its own position using only three satellites.

GPS..

•Important facts:
1. It takes while before data on satellite's position reaches receiver,
2. Receiver's clock is generally not in synch with that of satellite.
•Calculating position and time with GPS:

-Message received from satellite i with timestamp T_i
delta(i) = Measured delay by receiver
delta(i) = $(T_{now} – T_i)$ + delta(r)
delta(r) = deviation of the receiver's clock from the actual time
-Since signal travels at speed of light, measured distance of satellite
di = c $(T_{now} – T_i)$
-Based on satellite coordinates $d_i = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$
-Equation available from different satellites
-On solving them get position of the receiver and also since it is correlated to time get the exact time on receiver.

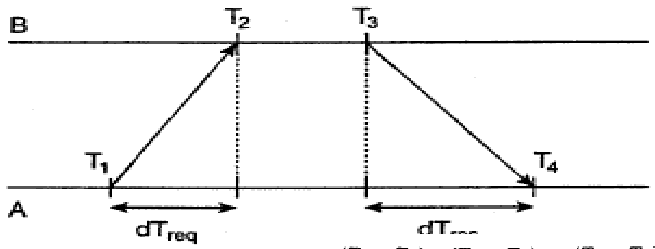
Clock Synchronization Algorithms

• Network Time Protocol (NTP)

•Berkeley Algorithm

•Reference Broadcast Synchronization (RBS)

Network Time Protocol
Cristian's Algorithm



A estimate offset relative to B: $\theta = T_3 - \frac{(T_2 - T_1) + (T_4 - T_3)}{2} = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$

- If theta < 0 A’s time is ahead of B’s and must set clock backwards

- Danger: source file is updated just before clock change and Object file compiled after clock update, then object might have time earlier than source file

NTP

The Network Time Protocol is an Internet standard whose goals are to:

- Enable clients across Internet to be accurately synchronized to UTC (universal coordinated time) despite message delays.

- Provide a reliable service that can survive lengthy losses of connec - tivity. This means having redundant paths and redundant servers.

- Enable clients to synchronize frequently and offset the effects of clock drift.

- Provide protection against interference; authenticate that the data is from a trusted source

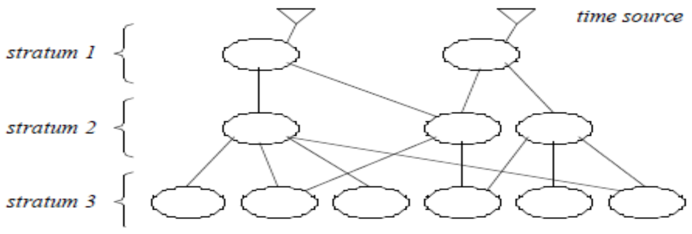
NTP..

•protocol is set up pair-wise between servers

$$\delta = \frac{(T_2 - T_1)+(T_4 - T_3)}{2}$$

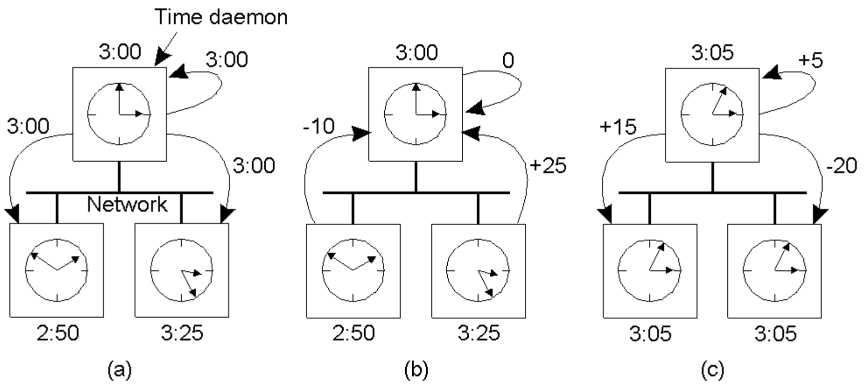
•Minimal value for delay taken into consideration
•NTP applied symmetrically allows both B and A adjust their time
•What if B’s time is more accurate?
•NTP divides server in terms of strata for accuracy
•Accurate server will be of starta 1
•A contacts B and finds a clock difference , it will update time only if it has a higher strata
•B’s strata: k
•If A does adjust it’s time wrt to B then A’s starta becomes k+1
•Due to symmetry if A’s strata is higher then B will adjust time

NTP



•The NTP servers are arranged into strata.
•The first stratum contains the primary servers, which are machines that are connected directly to an accurate time source.
•Second stratum contains the secondary servers. These machines are synchronized from the primary stratum machines.
•The third stratum contains terti -ary servers that are synchronized from the secondaries, and so on.
•Together, all these servers form the synchronization subnet

The Berkeley Algorithm



a) The time daemon asks all the other machines for their clock values
b) The machines answer
c) The time daemon tells everyone how to adjust their clock

Time Synchronization in WSN

•A **Wireless Sensor Network** (WSN)
- spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion, pollutants
- Cooperatively pass their data through the network to a main location.
- Use: control, machine health monitoring, environment and habitat monitoring, healthcare applications, surveillance, and traffic control.

•Time Synchronization :
- basic communication, movement, location, proximity.

•Parts of time synchronization: send time, access time, propagation time, and receive time.

•Synchronization protocol Reference Broadcast Synchronization (RBS)

Sensor Networks..

•Determination of sensor location

•Determination of relative proximity

•Save energy

•Speed of moving object

Drawbacks of using wired synchronization protocols for wireless

•**NTP**
- Not possible for wireless server node powered with battery power to have atomic clock

•**GPS**
- GPS receiver on each wireless node due to cost and power requirement is unfeasible.
- time accuracy dependent on number of satellites receiver communicates and may hence vary
- Line of site problem for sensor within buildings, in wildlife, under sea

Wireless Packet delay

•Send time

•Access time

•Propagation time

•Receiver time

Clock Synchronization in Wireless Networks

•Sensor networks: Nodes resource constrained

•Reference broadcast synchronization (RBS) clock synchronization:
-does not assume that there is a single node with an accurate account of time
-Aims at merely internally synchronizing the clocks
-Lets only receivers synchronize
-sender broadcasts a reference message that will allow its receivers to adjust their clocks
-Propagation time measured from the time it leaves NIC of sender

RBS

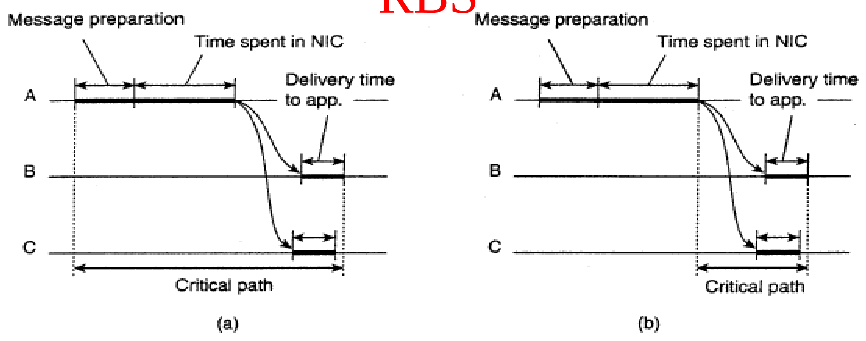
• Traditional time synchronization protocols sender will transmit the timestamp information and the receiver will synchronize.

•**RBS**
 - receiver to receiver synchronization.
 - Third party will broadcast a beacon to all the receivers which have no timing information
 - Receivers compare their clocks to one another to calculate their relative phase offsets.

•Example RBS:
 - one broadcast beacon sent to two receivers.
 - receivers record when packet was received wrt local clocks.
 - two receivers exchange timing information and be able to calculate the offset.

• RBS with n receivers with multiple broadcast message exchange

RBS



a. Usual critical time in calculating network delays
b. Critical Path for RBS

• Propagation time in this case is measured from the moment that a message leaves the network interface of the sender

• Message preparation time and time spent in NIC to access network are not critical to the propagation time in RBS

• Eliminates uncertainty of sender by removing the sender from the critical path. only uncertainty is the propagation and receive time.

RBS..

• When node broadcasts reference message m, each node p records the time Tp,m (p's local clock) that it received m.

• Two nodes p and q can exchange each other's delivery times in order to estimate their mutual, relative offset . M is the total number of reference messages sent

$$Offset[p,q] = \frac{\sum_{k=1}^{M}(T_{p,k} - T_{q,k})}{M}$$

Logical Clocks

•Cooperating Processes must agree on the order of events

•Time of day at which the event occurred does not matter

•Processes can agree on the order in which related events occur.

•Logical Clocks help get event sequence numbers that make sense system-wide.

Lamport's Logical Clocks

•Clock synchronization requires happens-before relation:
a ~ b is "a happens before b" i.e all processes agree that a happened before b. Observed in the following:

-a and b events in same process and a happens before b=> a ~ b
- a is an event of message sent from a process, b is the event of message being received by another process, then a ~ b because message cannot be received before being sent
- Transitive relation: if a ~ band b ~ c, then a ~ c.

Lamport's Logical Clocks

•Every event, a, we can assign it a time value C(a) on which all processes agree.

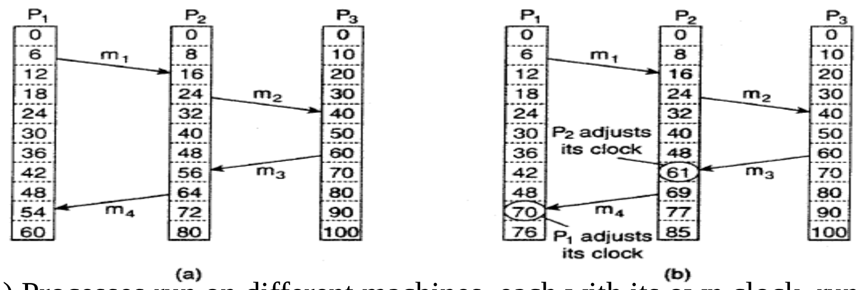
• if a ~ b, then C(a) < C(b).

• If a is sending of message by one process and b is reception of message by another process, then C (a) and C(b) must be assigned values such that all agree with C(a) < C(b).

•C, must always go forward (increasing), never backward (decreasing).

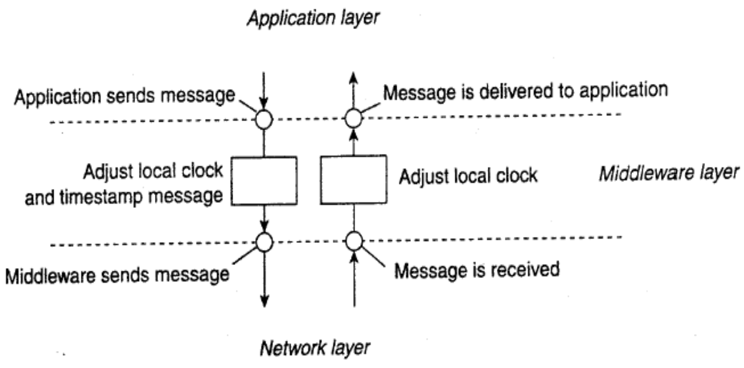
• Corrections to time can be made by adding a positive value, never by subtracting one

Lamport algorithm for assigning time to events



a) Processes run on different machines, each with its own clock, running at its own speed.
When the clock has ticked 6 times in process PI, it has ticked
8 times in process P2and 10 times in process P3' Each clock runs at a constant rate, but the rates are different due to differences in the crystals.
b) Lamport's algorithm corrects the clocks.

Positioning of Lamport's logical clocks in distributed systems.



Lamport's logical clock

each process *Pi maintains a local counter Ci* which is updated as follows:

1.Before executing an event (i.e., sending a message over the network, delivering a message to an application, or some other internal event), Pi executes Ci = Ci + 1
2. When process Pi sends a message m to Pj, it sets m's timestamp ts (m) equal to Ci after having executed the previous step.
3. Upon receipt of message m, process Pj adjusts its own local counter as Cj = max (Cj, ts(m)), after which it then executes the first step and delivers the message to the application.

•Some case may require no two events ever occur at the exact same time
•Attach number of process in which the event occurs to the low-order end of the time eg: event of time 40 at process Pi will be 40.i

Example: Totally-**Ordered** Multicasting

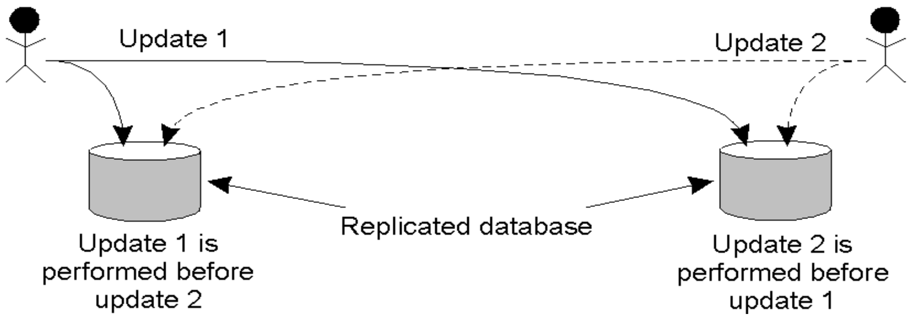
San Francisco customer adds $100, NY **bank** adds 1% interest
San Fran will have $1,111 and NY will have $1,110
Updating a replicated database and leaving it in an inconsistent state.
Can use Lamport's to totally order
(San Francisco)

(New York)

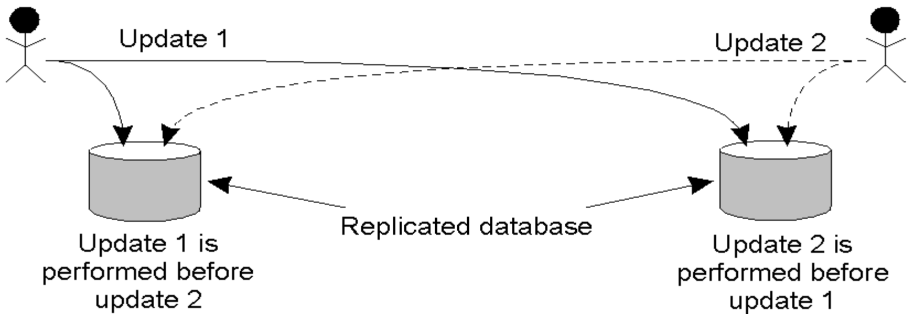
(+$100)

(+1%)

Lamport Timestamps
Totally Ordered Multicasting


Update 1
Update 2
Replicated database
Update 1 is performed before update 2
Update 2 is performed before update 1

Lamport Timestamps
Totally Ordered Multicasting


Update 1
Update 2
Replicated database
Update 1 is performed before update 2
Update 2 is performed before update 1

Totally-**Ordered Multicast**

•A **multicast** operation by which all messages are delivered in the same order to each receiver.

•**Lamport** Details:

- •Each message is timestamped with the current logical time of its sender.
- •**Multicast** messages are conceptually sent to the sender.
- •Assume all messages sent by one sender are received in the order they were sent and that no messages are lost.

Totally-**Ordered Multicast**

Lamport Details (cont):

- •Receiving process puts a message into a local queue **ordered** according to timestamp.
- •The receiver multicasts an ACK to all other processes.
- •Key Point from **Lamport**: the timestamp of the received message is lower than the timestamp of the ACK.
- •All processes will eventually have the same copy of the local queue consistent global ordering.

Totally-**Ordered Multicast**

•Process can deliver queued message to application it is running only when that message is at the head of the queue and has been acknowledged by each other process.

•Message is removed from queue and handed over to application; associated acknowledgments can simply be removed.

• Since each process has same copy of queue, all messages are delivered in same order everywhere.

Mutual Exclusion

Mutual exclusion : makes sure that concurrent process access shared resources or data in a serialized way.
If a process , say P_i , is executing in its critical section, then no other processes can be executing in their critical sections

Example: updating a DB or sending control signals to an IO device

Distributed mutual exclusion algorithms

Two different categories:

1.Token-based solutions:

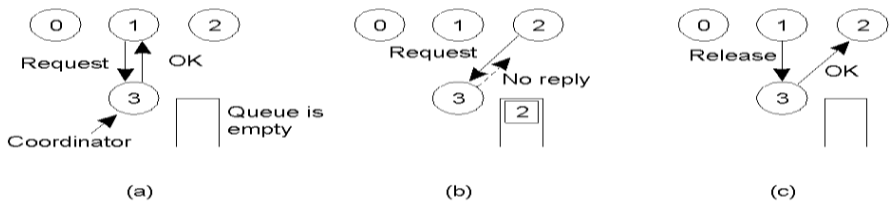
- Pass special message (token) between processes for mutual exclusion
- Who ever has that token is allowed to access the shared resource.
- When finished, token is passed on to a next process.
- If process having token is not interested it simply passes it on.
- Advantage:
    - Fair chance to processes to access resource (no starvation)
    - Easily Avoids deadlocks - several processes are waiting for each other to proceed
- Disadvantage:
    - when token is lost (e.g., because process holding it crashed), a distributed procedure required to ensure that new token is created

Distributed mutual exclusion algorithms

2. **Permission-based approach:**

- Process wanting to access the resource first requires permission of other processes
- Type:
    - Centralized Algorithm
    - Distributed Algorithm

Mutual Exclusion: A Centralized Algorithm



a) Process 1 asks the coordinator for permission to enter a critical region. Permission is granted
b) Process 2 then asks permission to enter the same critical region. The coordinator does not reply.
c) When process 1 exits the critical region, it tells the coordinator, when then replies to 2

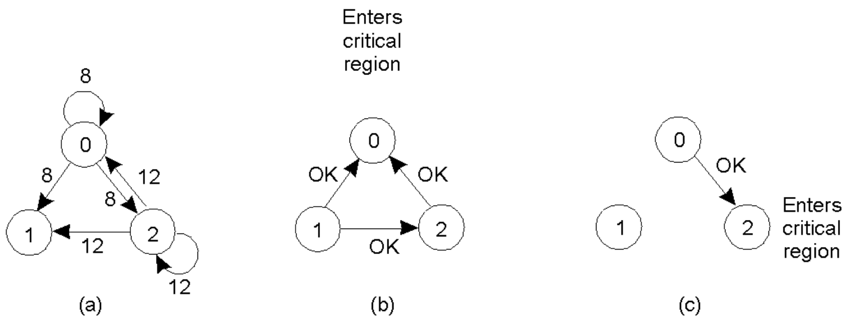
Centralized algorithms contd..

Advantages
- •Fair algorithm, grants in the order of requests
- •The scheme is easy to implement
- •Scheme can be used for general resource allocation

Critical Question: When there is no reply, does this mean that the coordinator is "dead" or just busy?

Shortcomings
- •Single point of failure. No fault tolerance
- •Confusion between No-reply and permission denied
- •Performance bottleneck of single coordinator in a large system

A Distributed Algorithm Ricart and Agrawala



a) Two processes want to enter the same critical region at the same moment.
b) Process 0 has the lowest timestamp, so it wins.
c) When process 0 is done, it sends an OK also, so 2 can now enter critical region.

Distributed Algorithm

•mutual exclusion is guaranteed without deadlock or starvation.
•The number of messages required per entry is now 2(n - 1), where the total number of processes in the system is n.
•No single point of failure exists

•Instead n points of failure:
 - If any process crashes, it will fail to respond to requests. This silence will be interpreted (incorrectly) as denial of permission, thus blocking all subsequent attempts by all processes to enter all critical regions
•Solution:
When a request comes in, the receiver always sends a reply, either granting or denying permission.
Whenever either a request or a reply is lost, the sender times out and keeps trying until either a reply comes back or the sender concludes that the destination is dead.
After a request is denied, the sender should block waiting for a subsequent *OK message.*

Distributed Algorithm…

•Group multicast:
- each process must maintain the group membership list itself, including processes entering the group, leaving the group, and crashing.
- The method works best with small groups of processes that never change their group memberships

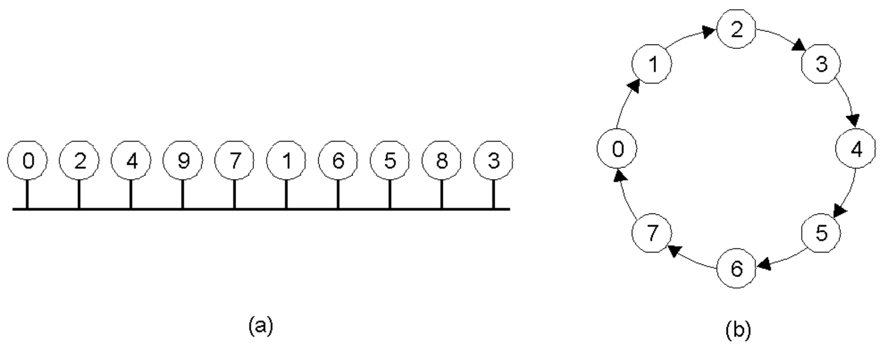
•Variation getting permission from majority
- getting permission from everyone is really overkill
- grant permission when it has collected permission from a simple majority of the other processes, rather than from all of them

Distributed Algorithm…

Disadvantage: algorithm is slower, more complicated, more expensive, and less robust than the original centralized one

Advantage: Distributed nature of the algorithm and not dependent on a centralized mechanism which may become a bottleneck for the algorithm

A Toke Ring Algorithm



a) An unordered group of processes on a network.
b) A logical ring constructed in software.

A Toke Ring Algorithm

Advantage:
•Only one process has the token at any instant, so only one process can actually get to the resource
•Since token circulates in well defined order no starvation

Disadvantage:
•Detecting Loss of token:
Absence of token for a long period not an indication of loss of token since token could ne used by a node in the ring.
•Process Crashes:
 - Requires process receiving token to acknowledge receipt.
 - If not receipt indicates process has crashed
 - Remove dead process and pass token to the next one in the logical ring
 - Need prior information of the ring structure

Comparison

Algorithm	Messages per entry/exit	Delay before entry (in message times)	Problems
Centralized	3	2	Coordinator crash
Distributed	2 (n – 1)	2 (n – 1)	Crash of any process
Token ring	1 to ∞	0 to n – 1	Lost token, process crash

A comparison of three mutual exclusion algorithms.

Election Algorithm

• Distributed Algorithm require one process to act as coordinator, initiator, or otherwise perform some special role.
• Assume one process per machine for simplicity with a unique number (network address)
• election algorithms attempt to locate the process with the highest process number and designate it as coordinator
•All processes know process number of every other process but not the state of the process (alive /dead)
•Election Algorithm must ensure in the end of the election all processes agree on the new coordinator.

•Algorithm:
- Bully Algorithm
- Ring Algorithm

Election Algorithms

- Bully Algorithm
- Ring Algorithm

Bully Algorithm

•Process with highest number will be coordinator. It will Bully and take over the other to become the leader.

•Any process Pi notices that coordinator is no longer responding to requests, it initiates an election:

1. Pi sends ELECTION message to all processes with higher numbers.
2. If no one responds, Pi wins election and becomes coordinator.
3. If one of the higher-ups answers, it takes over. Pi's job is done.

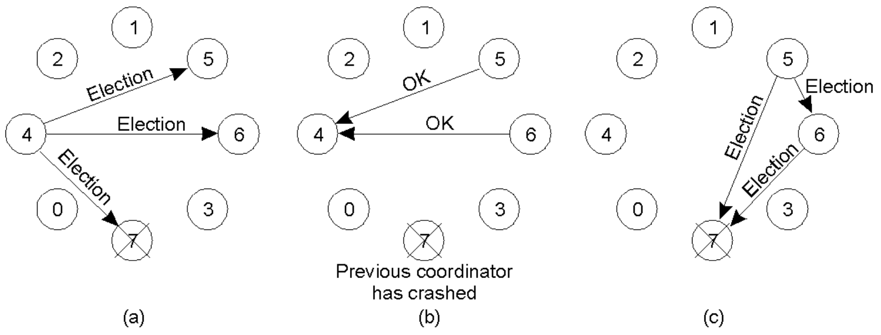
• If process Pi receives ELECTION message from lower numbered process , it sends OK message indicating that it is alive and will take over.

•In the end all process give up except one that is the new coordinator

•If a process that was previously down comes up it holds election.

- If it has the highest number among all running processes it will win over and become coordinator

The Bully Algorithm (1)



The bully election algorithm

- Process 4 holds an election
- Process 5 and 6 respond, telling 4 to stop
- Now 5 and 6 each hold an election

Bully Algorithm

•If 7 is ever restarted, it will just send an the others a *COORDINATOR* message and bully them into submission.

Ring Algorithm

•No token.

•Assumes processes are physically or logically ordered

• If a process notices that coordinator is not functioning, builds *ELECTION* message containing its own process number and sends message to successor.

• If successor is down, sender skips over successor and goes to next member along the ring, until a running process is located

• Each step along way, sender adds its own process number to list in message effectively making itself a candidate to be elected as coordinator

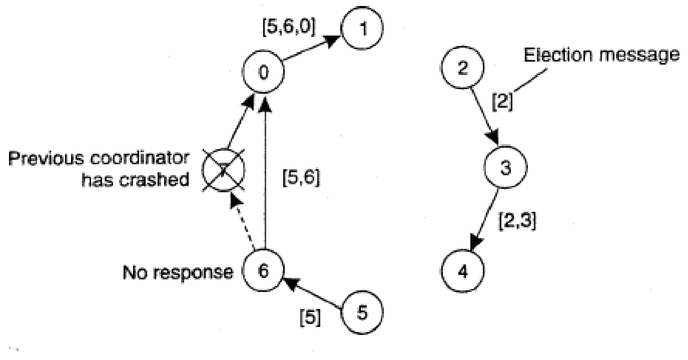
• Eventually, the message gets back to the process that started it all

Ring Algorithm..

•Message type is changed to *COORDINATOR* and circulated once again, to inform everyone the coordinator is member with the highest number and who the members of the new ring are.

•Once message has circulated once, it is removed and processes resume work.

Election in Ring



Elections in Wireless Environments

•Wireless Sensor Networks are wireless networks that usually consist of a great number of far distributed devices that are equipped with sensors (instruments that measure quantities in our environment) to monitor physical or environmental phenomenon. These devices work autonomous and are logically linked by self-organizing means.

•Wireless Sensor Networks is a class of special wireless ad hoc networks. A wireless ad hoc network is a collection of wireless nodes, that communicate directly over a common wireless channel. There is no additional infrastructure needed for ad hoc networks. Therefore, every node is equipped with a wireless transceiver and has to be able to act as an router, to process packets to their destinations.

•Characteristic: ability to self-organize after they were deployed.

Elections in Wireless Environments

•Difference:
- WSNs used for monitoring and collecting data
- Ad hoc networks focus more on the communication aspects.

•Traditional election algorithms not valid since:
- message passing is not reliable
- topology of the network changes

Elections in Wireless Environments

•To elect leader, any node in network, called source, can initiate an election by sending an ELECTION message to its immediate neighbours (i.e., the nodes in its range).

•When a node receives an ELECTION for first time, it designates sender as its parent, and sends out an ELECTION message to all its immediate neighbours, except for parent.

•When node receives an ELECTION message from a node other than its parent (i.e. later) , it only acknowledges it.

• R has designated Q as parent, forwards ELECTION message to immediate neighbors and waits for acks before it can ack to Q.

•Neighbors which have already selected parent will ACK to R.

Elections in Wireless Environments

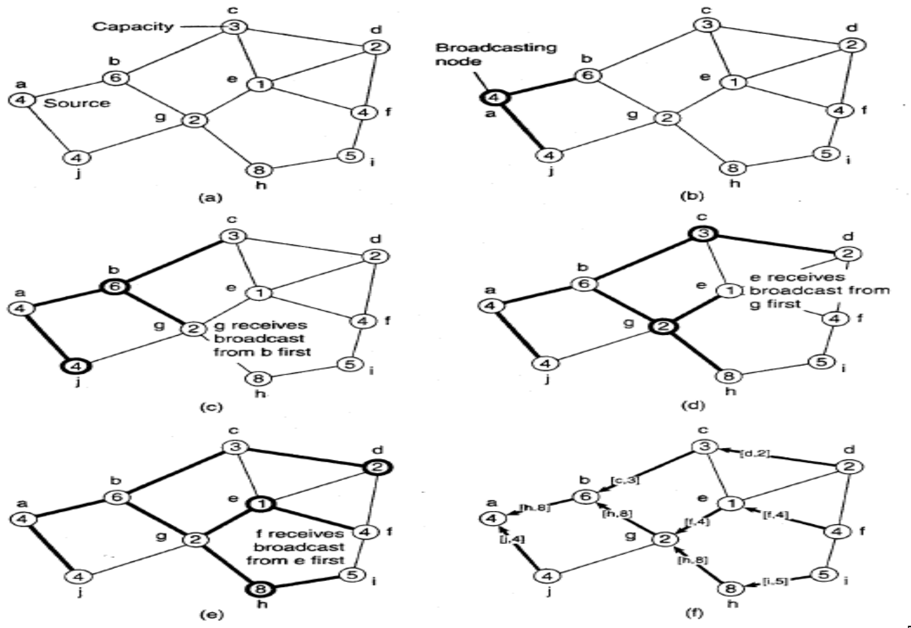
•If all neighbours already have a parent, R is a leaf node and will be able to report back to Q quickly and it will also report information such as its battery lifetime and other resource capacities

•Q will compare R's capacities to other downstream nodes, and select best eligible node for leadership.

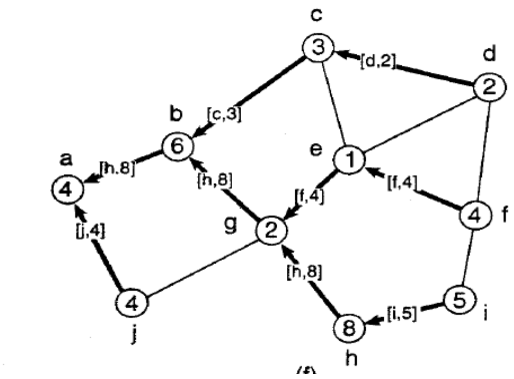
•Q had sent an ELECTION message because its own parent P had done so as. Hence when Q eventually acknowledges the ELECTION message previously sent by P, it will pass the most eligible node to P as well.

•In this way, source will eventually get to know which node is best to be selected as leader, after which it will broadcast this information to all other nodes.

Election algorithm in a wireless network


Capacity
Source
Broadcasting node
g receives broadcast from b first
e receives broadcast from g first
f receives broadcast from e first
[d,2]
[c,3]
[h,8]
[f,4]
[j,4]
[i,5]
(a)
(b)
(c)
(d)
(e)
(f)

Election algorithm in a wireless network



References

•http://www.cs.rutgers.edu/~pxk/rutgers/notes/content/08-clocks.pdf

•http://www1.cse.wustl.edu/~jain/cse574-06/ftp/time_sync/index.html#Section1.0

•Distributed Systems Principles and Paradigms, Andrew Tanenbaum