Κατανεμημένα Συστήματα
Εκλογή Αρχηγού
Leader Election

Χάρης Μανιφάβας
Τμήμα Εφ. Πληροφορικής & Πολυμέσων
ΤΕΙ Κρήτης

Leader election

- Definition
  - In distributed computing, **leader election** is the process of designating a single process as the organizer of some task distributed among several computers (nodes)
  - Many algorithms used in distributed systems require a coordinator, initiator, or otherwise someone to perform some special role
- Before the task is begun
  - All network nodes are unaware which node will serve as the "leader," or coordinator, of the task
  - In general, all processes in the distributed system are equally suitable for the role
  - Election algorithms are designed to choose a coordinator
- After a leader election algorithm has been run
  - Each node throughout the network recognizes a particular, unique node as the task leader

Leader election

- Ορισμός
  - Εκλογή αρχηγού (leader election) ονομάζουμε την επιλογή μίας από τις διεργασίες ενός συστήματος προκειμένου αυτή να εκτελέσει ένα συγκεκριμένο καθήκον
  - Παράδειγμα
    - Απαιτείται εκλογή αρχηγού στην εκτέλεση ενός συγκεντρωτικού αλγορίθμου στον οποίο μία συγκεκριμένη διεργασία πρέπει να αναλάβει το ρόλο του συντονιστή

Leader election

- Κάθε αλγόριθμος εκλογής αρχηγού πρέπει να ικανοποιεί τις παρακάτω βασικές απαιτήσεις:
  - Κάθε κόμβος-διεργασία πρέπει να εκτελεί τον ίδιο τοπικό αλγόριθμο
  - Ο αλγόριθμος πρέπει είναι κατανεμημένος, δηλαδή ο υπολογισμός πρέπει να μπορεί να ξεκινήσει από ένα αυθαίρετο μη κενό σύνολο διεργασιών
  - Σε κάθε δυνατή τελική κατάσταση του αλγορίθμου πρέπει να υπάρχει μόνο μία διεργασία η οποία θα είναι στην κατάσταση του αρχηγού (leader), ενώ όλες οι υπόλοιπες πρέπει να είναι στην κατάσταση του ηττημένου (lost)

Leader election

- Procedure
  - Any process can serve as coordinator
  - Any process can “call an election”
    - Initiate the algorithm to choose a new coordinator
    - There is no harm (other than extra message traffic) in having multiple concurrent elections
- Elections may be needed
  - When the system is initialized
  - When a new process enters the system
  - If the coordinator crashes or retires

Leader election

- Procedure
  - The network nodes communicate among themselves in order to decide which of them will get into the "leader" state
  - If all processes are exactly the same, with no distinguishing characteristics, there is no way to select one of them to be special
    - For that, they need some method in order to break the symmetry among them
  - If each node has **unique identities**, then the nodes can compare their identities, and decide that the node with the highest identity is the leader
  - Leader election algorithms are designed to be economical in terms of total bytes transmitted, and time

Leader election

- Assumptions
  - Every process/site has a unique ID
    - The network address
    - A process number (as in MPI)
  - Every process in the system should know the values in the set of ID numbers
    - Although not which processors are up or down
    - What the processes do not know is which ones are currently up and which ones are currently down
  - The process with the highest ID number will be the new coordinator
  - When the election algorithm terminates a single process has been selected and every process agrees and knows its identity

Leader election

- Υποθέσεις
  - Στους αλγορίθμους που θα περιγραφούν παρακάτω, θεωρείται ότι κάθε διεργασία έχει ένα μοναδικό αναγνωριστικό
  - Με τα μοναδικά αναγνωριστικά είναι δυνατή η αποστολή μηνυμάτων σε συγκεκριμένες διεργασίες, αλλά και η εκλογή ως αρχηγού της διεργασίας με το μικρότερο (ή το μεγαλύτερο) αναγνωριστικό
  - Σε αυτή την περίπτωση, το πρόβλημα της εκλογής αρχηγού ονομάζεται και πρόβλημα εύρεσης ακρότατου (extrema-finding problem)

Leader election

- Υποθέσεις
  - Οι αλγόριθμοι που εκλέγουν ως αρχηγό τη διεργασία με το μικρότερο (ή το μεγαλύτερο) αναγνωριστικό μειονεκτούν σε ένα βασικό σημείο:
    - Αυτή η διεργασία δεν αποτελεί τη βέλτιστη επιλογή από άποψη, για παράδειγμα, επίδοσης ή αξιοπιστίας
    - Γι' αυτό έχουν παρουσιαστεί αλγόριθμοι βασισμένοι σε προτιμήσεις (preference-based algorithms), όπου η εκλογή βασίζεται στις προτιμήσεις των διεργασιών, οι οποίες στηρίζονται σε εκτιμήσεις περί αξιοπιστίας, τοποθεσίας, κ.λπ.
    - Τέλος, αξίζει να σημειώσουμε ότι υπάρχουν αλγόριθμοι εκλογής στους οποίους δεν θεωρούνται γνωστά τα αναγνωριστικά των διεργασιών
    - Οι αλγόριθμοι αυτοί είναι κυρίως πιθανοτικοί αλγόριθμοι (probabilistic algorithms)

Leader election

- Στους αλγορίθμους εκλογής, οι διεργασίες του συστήματος διακρίνονται σε εκκινητές και μη εκκινητές
  - Ως εκκινητής (initiator) χαρακτηρίζεται μια διεργασία που ξεκινά την εκτέλεση του τοπικού αλγορίθμου της αυτόματα, π.χ. με την ικανοποίηση μιας συνθήκης
  - Χαρακτηριστικό ενός εκκινητή είναι ότι η πρώτη ενέργεια που εκτελεί είναι η αποστολή ενός μηνύματος
  - Στον αλγόριθμο υπάρχουν και μη εκκινητές (non-initiators) οι οποίοι απλώς συμμετέχουν στον αλγόριθμο και ξεκινούν την εκτέλεση του δικού τους τοπικού αλγορίθμου μόνο όταν λάβουν κάποιο μήνυμα

Bully algorithm

- The **bully algorithm** is a method in distributed computing for dynamically selecting a coordinator by process ID number
  - The algorithm gets its name out of the fact that a process with a higher ID number will bully a lower ID process out of the coordinator position as soon as it comes online
  - The algorithm was devised by Garcia-Molina in 1982
    - Elections in a Distributed Computing System, IEEE Transactions on Computers, Vol. C-31, No. 1, January (1982) 48-59

Bully algorithm

- The **bully algorithm** is a method in distributed computing for dynamically selecting a coordinator by process ID number
  - When a process P determines that the current coordinator is down because of message timeouts or failure of the coordinator to initiate a handshake, it performs the following sequence of actions:
    - P broadcasts an election message (inquiry) to all other processes with higher process IDs
    - If P hears from no process with a higher process ID than it, it wins the election and broadcasts victory
    - If P hears from a process with a higher ID, P waits a certain amount of time for that process to broadcast itself as the leader. If it does not receive this message in time, it re-broadcasts the election message
  - Note that if P receives a victory message from a process with a lower ID number, it immediately initiates a new election

Bully algorithm

- At any moment, a process can get an ELECTION message from one of its lower-numbered colleagues
  - When such a message arrives, the receiver sends an OK message back to the sender to indicate that it is alive and will take over
  - The receiver then holds an election, unless it is already holding one
  - Eventually, all processes give up but one, and that one is the new coordinator
  - It announces its victory by sending all processes a message telling them that starting immediately it is the new coordinator

Bully algorithm

- If a process that was previously down comes back up, it holds an election
  - If it happens to be the highest-numbered process currently running, it will win the election and will take over the coordinator's job
  - High-numbered processes “bully” low-numbered processes out of the election, until only one process remains
  - Thus the biggest guy in town always wins, hence the name "Bully Algorithm"

Bully algorithm

- Λειτουργία
  - Υποθέτουμε ότι κάθε διεργασία έχει ένα αναγνωριστικό το οποίο εκφράζει την προτεραιότητά της μεταξύ των άλλων
  - Κάθε διεργασία γνωρίζει τις προτεραιότητες όλων των υπολοίπων διεργασιών και ως αρχηγός εκλέγεται η διεργασία με τη μεγαλύτερη προτεραιότητα
  - Επιπλέον θεωρείται ότι κάθε διεργασία μπορεί να αποτυγχάνει και να επανέρχεται οποιαδήποτε στιγμή
  - Έτσι το ερώτημα είναι ποια είναι η ζωντανή διεργασία με τη μεγαλύτερη προτεραιότητα κατά τη χρονική στιγμή που εκτελείται ο αλγόριθμος αυτός

Bully algorithm

- Λειτουργία
  - Μια διεργασία, που μόλις έχει επανέλθει και έχει το μεγαλύτερο αναγνωριστικό, αναγκάζει οποιαδήποτε διεργασία με μικρότερη προτεραιότητα κατέχει εκείνη τη στιγμή την αρχηγία να της την παραχωρήσει
  - Γι' αυτό, ο αλγόριθμος είναι γνωστός και ως αλγόριθμος εξαναγκασμού (bully algorithm)
  - Αν μια διεργασία που έχει επανέλθει δεν έχει τη μεγαλύτερη προτεραιότητα, τότε ξεκινά αμέσως τη διαδικασία επαναπροσδιορισμού του αρχηγού

Bully algorithm

- Λειτουργία
  - Όταν μια διεργασία ανακαλύψει ότι ο αρχηγός δεν είναι ενεργός (π.χ. αν έχει πολλή ώρα να λάβει μήνυμα από αυτόν), τότε μεταδίδει σε όλες τις διεργασίες με μεγαλύτερη προτεραιότητα από αυτήν ένα μήνυμα <election> για να τις ενημερώσει ουσιαστικά ότι πρέπει να γίνει εκλογή αρχηγού
  - Στη συνέχεια, περιμένει απάντηση <0K> από κάποια από τις διεργασίες αυτές
  - Το μήνυμα <OK> δείχνει ότι η διεργασία που το έστειλε δεν έχει αποτύχει και συνεπώς αυτή μπορεί να «αναλάβει δράση» για τη διαδικασία εκλογής

Bully algorithm

- Λειτουργία
  - Αν η διεργασία που έστειλε το μήνυμα <election> δεν λάβει απάντηση <0K> μέσα σε ένα συγκεκριμένο χρονικό διάστημα, τότε θεωρεί ότι όλες οι διεργασίες με μεγαλύτερο αναγνωριστικό από αυτήν έχουν αποτύχει και αυτοανακηρύσσεται αρχηγός, μεταδίδοντας την προτεραιότητά της, δηλαδή την προτεραιότητα του νέου αρχηγού, σε όλες τις διεργασίες που έχουν μικρότερη προτεραιότητα από αυτήν.
  - Σε περίπτωση που λάβει μήνυμα <0K> από κάποια διεργασία με μεγαλύτερη προτεραιότητα μέσα στο χρονικό όριο, τότε θέτει ένα νέο χρονικό όριο μέσα στο οποίο θα πρέπει να της γνωστοποιηθεί το αναγνωριστικό του νέου αρχηγού
  - Σε περίπτωση που δεν λάβει το αναγνωριστικό του αρχηγού, ξαναστέλνει το μήνυμα <election> που έστειλε και αρχικά

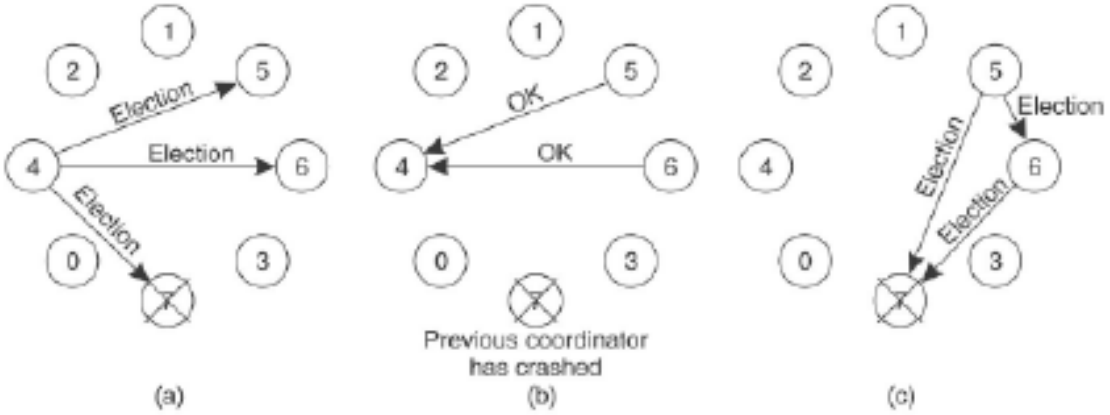
Bully algorithm

- Λειτουργία
  - Όταν μια διεργασία με μεγαλύτερη προτεραιότητα από αυτήν που ξεκίνησε τη διαδικασία εκλογής (έστω ρ) λάβει το μήνυμα <election>, στέλνει μια απάντηση <OK> στην ρ (που προφανώς έχει μικρότερη προτεραιότητα από αυτήν) και ξεκινά το δικό της αλγόριθμο εκλογής στέλνοντας με τη σειρά της ένα μήνυμα <election> σε όλες τις διεργασίες με μεγαλύτερη προτεραιότητα από αυτήν
  - Αν αυτή η διεργασία έχει την υψηλότερη προτεραιότητα, μπορεί να αυτοανακηρυχθεί αρχηγός αμέσως
  - Έτσι η ευθύνη της εκτέλεσης της διαδικασίας εκλογής μεταβιβάζεται σταδιακά στη διεργασία που έχει τη μεγαλύτερη προτεραιότητα (και είναι σε λειτουργία)
  - Αυτή η διεργασία εκλέγεται τελικά ως νέος συντονιστής

Bully algorithm

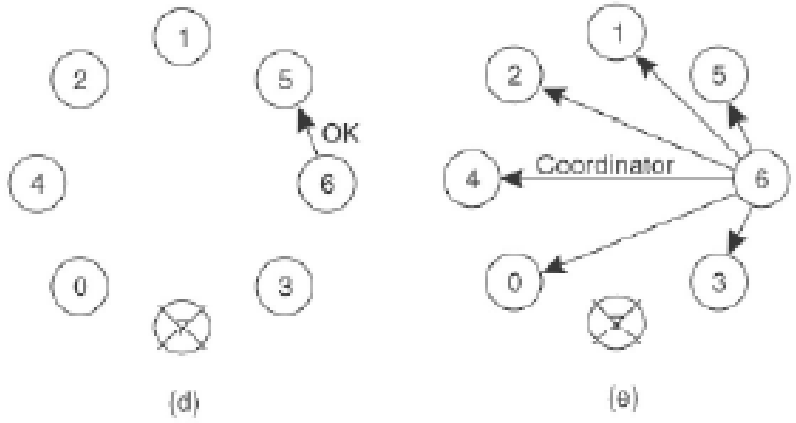
Αν μια διεργασία ρ είναι εκκινητής
Αν η ρ έχει το μεγαλύτερο δυνατό αναγνωριστικό Αυτοανακηρύσσεται αρχηγός
Γνωστοποιεί το αναγνωριστικό της σε όλες τις διεργασίες που έχουν μικρότερο αναγνωριστικό Διαφορετικά
Στέλνει σε όλες τις διεργασίες με μεγαλύτερο αναγνωριστικό από αυτήν ένα μήνυμα <election> (τις ενημερώνει ότι πρέπει να γίνει εκλογή αρχηγού)
Περιμένει για κάποιο συγκεκριμένο χρονικό διάστημα ένα μήνυμα <OK>.από κάποια από αυτές Αν η ρ λάβει κάποιο μήνυμα <OK>
Περιμένει (για κάποιο συγκεκριμένο χρονικό διάστημα) να λάβει το αναγνωριστικό του αρχηγού Αν το χρονικό διάστημα εξαντληθεί KOL δεν λάβει το αναγνωριστικό
Ξαναστέλνει μήνυμα <election>
Αν το χρονικό διάστημα εξαντληθεί και η ρ δεν λάβει κάποιο μήνυμα <OK>
Ανακηρύσσεται αρχηγός (όλες ol διεργασίες με μεγαλύτερο αναγνωριστικό έχουν αποτύχει) Γνωστοποιεί στις διεργασίες που έχουν μικρότερο αναγνωριστικό ότι είναι ο αρχηγός Αν η ρ λάβει ένα μήνυμα <election> από μια διεργασία ρ'
Αν έχει μεγαλύτερο αναγνωριστικό από την ρ' Στέλνει ένα μήνυμα <OK> στην ρ'
Αν μια διεργασία ρ δεν είναι εκκινητής
Αν λάβει ένα μήνυμα <election> από μια διεργασία ρ'
Αν έχει μεγαλύτερο αναγνωριστικό από την ρ'
Στέλνει απάντηση <OK> στην ρ'
Αναλαμβάνει το ρόλο του εκκινητή

Bully algorithm



- In Fig. 6-20 we see an example of how the bully algorithm works
  - The group consists of eight processes, numbered from 0 to 7
  - Previously process 7 was the coordinator, but it has just crashed
  - Process 4 is the first one to notice this, so it sends *ELECTION messages to all the processes higher than it, namely 5, 6, and 7* as shown in Fig. 6-20(a)
  - Processes 5 and 6 both respond with *OK, as shown in* Fig. 6-20(b)
  - Upon getting the first of these responses, 4 knows that its job is over
  - It knows that one of these bigwigs will take over and become coordinator
  - It just sits back and waits to see who the winner will be (although at this point it can make a pretty good guess)
  - In Fig. 6-20(c), both 5 and 6 hold elections, each one only sending messages to those processes higher than itself

Bully algorithm - Example



- In Fig. 6-20 we see an example of how the bully algorithm works
  - In Fig. 6-20(d) process 6 tells 5 that it will take over
  - At this point 6 knows that 7 is dead and that it (6) is the winner
  - If there is state information to be collected from disk or elsewhere to pick up where the old coordinator left off, 6 must now do what is needed
  - When it is ready to take over, 6 announces this by sending a *COORDINATOR message to all running processes*
  - When 4 gets this message, it can now continue with the operation it was trying to do when it discovered that 7 was dead, but using 6 as the coordinator this time
  - In this way the failure of 7 is handled and the work can continue
  - If process 7 is ever restarted, it will just send the others a *COORDINATOR* message and bully them into submission

Bully algorithm - Example

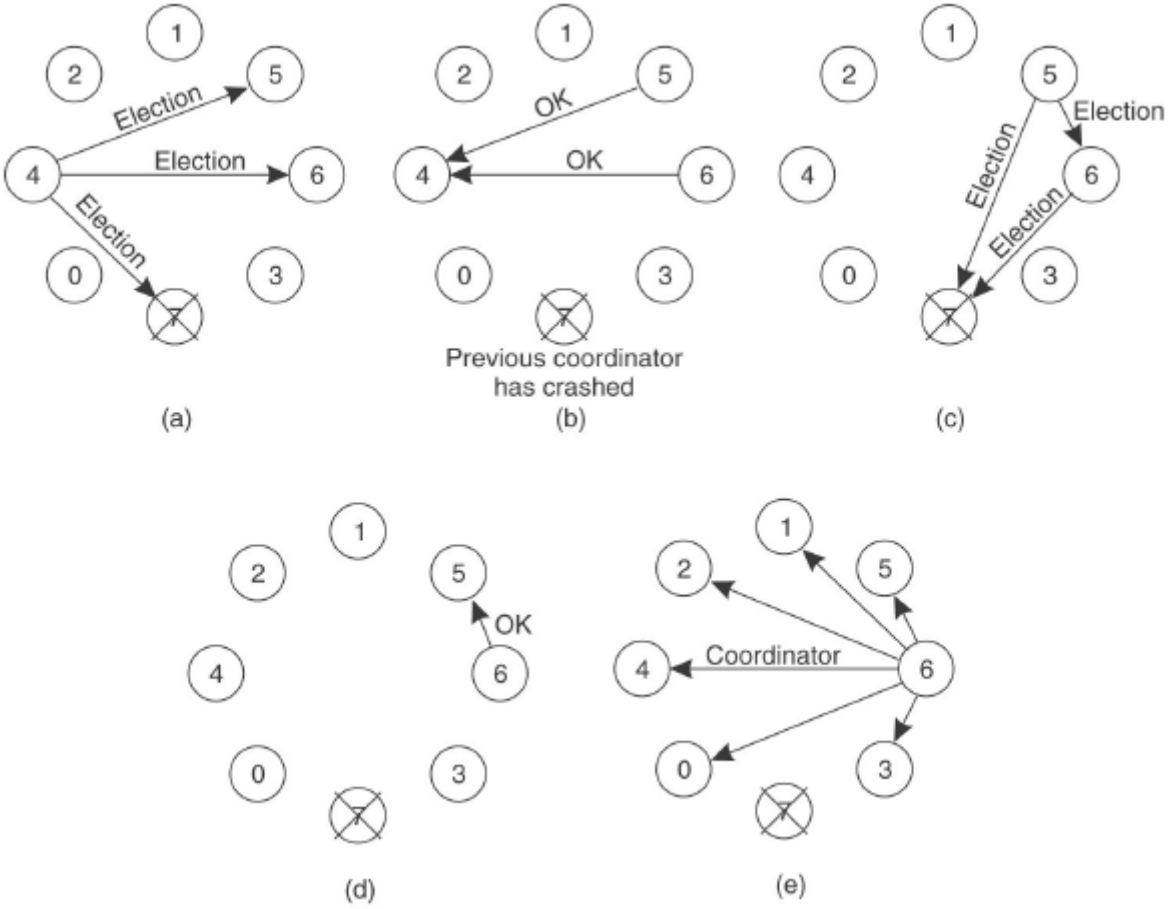


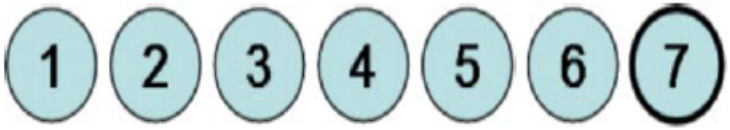
Figure 6-20. The bully election algorithm. (a) Process 4 holds an election. (b) Processes 5 and 6 respond, telling 4 to stop. (c) Now 5 and 6 each hold an election. (d) Process 6 tells 5 to stop. (e) Process 6 wins and tells everyone.

Bully algorithm - Example

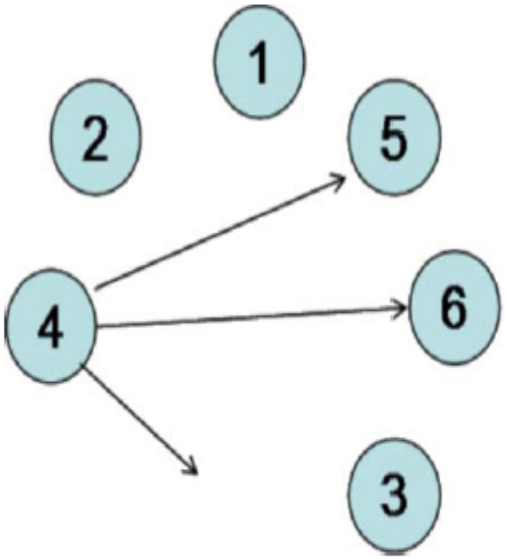
- Στο Σχήμα φαίνεται ένα παράδειγμα εκτέλεσης του παραπάνω αλγορίθμου
  - Οι κύκλοι με έντονο περίγραμμα παριστάνουν διεργασίες με το ρόλο του εκκινητή, ενώ ο κύκλος με έντονα διακεκομμένο περίγραμμα παριστάνει τον κόμβο που εκλέγεται αρχηγός του συστήματος
  - Αρχικά η διεργασία 7 που ήταν μέχρι εκείνη τη στιγμή αρχηγός τίθεται εκτός λειτουργίας
  - Η διεργασία 4 είναι αυτή που πρώτη ανακαλύπτει την απουσία αρχηγού στο δίκτυο και ξεκινά τον αλγόριθμο εκλογής (αναλαμβάνει το ρόλο του εκκινητή)
  - Επειδή γνωρίζει ότι υπάρχουν διεργασίες με μεγαλύτερο αναγνωριστικό από αυτήν, στέλνει μηνύματα <election> στις διεργασίες αυτές, δηλαδή στις 5, 6 και 7
  - Από αυτές, μόνο οι δύο πρώτες απαντούν με μήνυμα <OK>, εφόσον μόνο αυτές είναι σε λειτουργία
  - Έτσι η διεργασία 4 σταματά και απλώς περιμένει να ενημερωθεί για την ταυτότητα του νέου αρχηγού
  - Στη συνέχεια, οι διεργασίες 5 και 6 (που έχουν αναλάβει πια το ρόλο του εκκινητή) στέλνουν μηνύματα <election> στις 6 και 7 και 7, αντίστοιχα
  - Από αυτές, μόνο η 6 στέλνει <OK> στην 5 και την αναγκάζει να περιμένει να ενημερωθεί για το νέο συντονιστή
  - Αφού, λοιπόν, η 6 βλέπει ότι η 7 (η μόνη διεργασία με υψηλότερη προτεραιότητα από αυτήν) δεν της έχει απαντήσει, συμπεραίνει ότι η 7 έχει αποτύχει και κατά συνέπεια ανακηρύσσεται αρχηγός
  - Έτσι, στο επόμενο βήμα ενημερώνει όλες τις διεργασίες με μικρότερο αναγνωριστικό από αυτήν ότι αυτή είναι πλέον ο νέος αρχηγός

Bully algorithm - Example

Let's assume that we have 7 nodes:

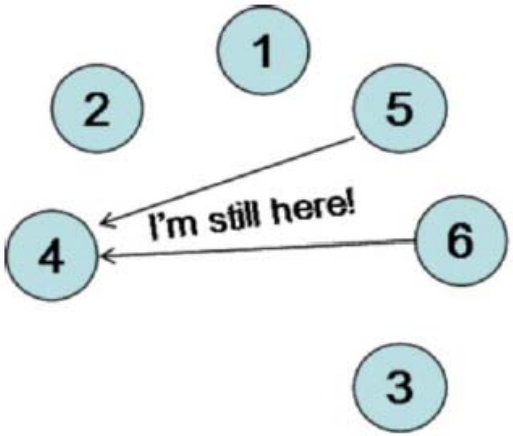


At this current moment in time, node '7' is the coordinator, because it is the highest numbered node. But then node '7' leaves, so who is now the coordinator? Lets say node '4' decides it wants to be the coordinator, then we have:

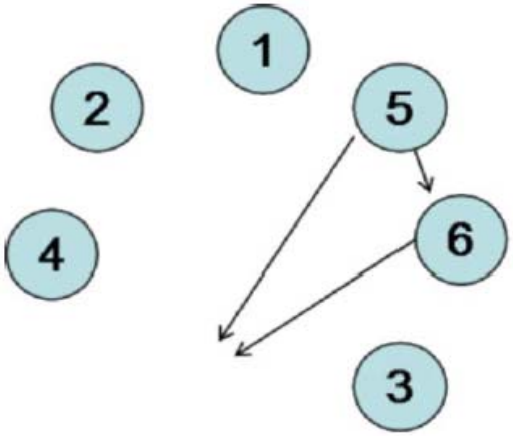


Here, node '4' has sent a ELECTION to nodes '5', '6' and '7', this is because none of the nodes know that node '7' has left yet. In this case, node '7' cannot reply, yet, nodes '5', and '6' are still here, so they reply to '4' that they are still here:

Bully algorithm

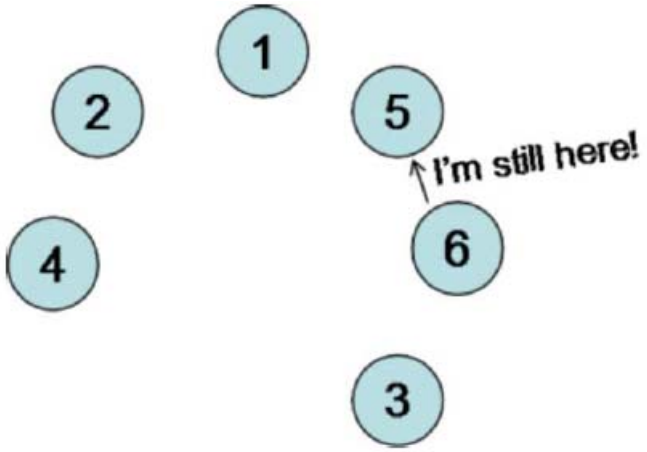


Now we have nodes '5' and '6' sending out elections. In this case, node '5' sends an election to node '6' and '7'. Whereas node '6' just sends an election to node '7':

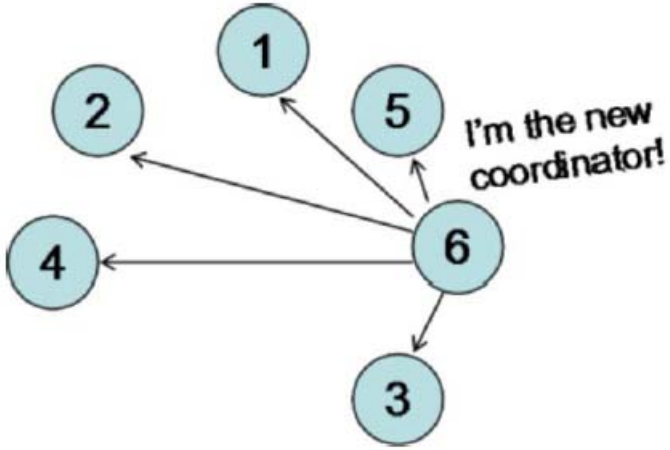


Now, since node '6' is still here, then node '6' tells node '5' that it is still here:

Bully algorithm



Finally, since node '7' has not replied to node '6', this implies that node '7' no longer exists, meaning that node '6' is the new coordinator. So node '6' tells all the other nodes:



Bully algorithm

- Analysis
  - Works best if communication in the system has bounded latency so processes can determine that a process has failed by knowing the upper bound (UB) on message transmission time (T) and message processing time (M)
    - UB = 2 * T + M
  - However, if a process calls an election when the coordinator is still active, the coordinator will win the election

Bully algorithm

- Suppose two processes detect the demise of the coordinator simultaneously and both decide to hold an election using the bully algorithm. What happens?
  - If two processes simultaneously detect that the coordinator has failed, they will both begin the bully algorithm by sending out an election message
  - The algorithm assumes that all processes know the IDs of other processes, so a message will be sent to all of the processes in the system that has a higher ID then themselves
  - The process that sent the election message will then set a timeout period and wait to see if a message is returned in that period
  - Processes with a higher ID will send a response to those with a lower ID and will begin its own election algorithm
  - In the case of two processes simultaneously starting the election, the process with the lowest ID of the two processes that started the algorithm will receive a response indicating they are out of the running of becoming the new coordinator
  - So each node that receives a election message and has a higher identifier will run its own bully algorithm
  - Eventually the process with the highest id will become the coordinator and send out a message to the other processes indicating it is the new coordinator
  - If eventually the process that fails come back up it will then hold a new election
  - So in the case of two processes starting the bully algorithm simultaneously, if one of the nodes has the highest identifier of all nodes in the system then it will be chosen as the new coordinator, if not then other nodes with higher identifiers will run the algorithm

Bully algorithm

- A recovering process starts an election
  - It will become the new coordinator if it has a higher identifier than the current leader
  - Is this a feature that is necessary for the algorithm to work correctly?
  - Under which circumstances can it be dropped?
  - To start election process with every recovered process is not completely necessary
  - In my opinion starting an election during process recovery can be dropped if recovered process has lower ID as currently running leader

Bully algorithm

- Suggest how to adapt the Bully algorithm to deal with temporary network partition (slow communication) and slow processes

Bully algorithm - Demonstration

- What happens if two processes notice at the same time that the leader has crashed?
  - As it was mentioned in the report, start and end of election process are implemented as an atomic functions (class ReentrantLock is used)
  - This prevents another process from being executed during the time start/end election process is running, until is completely finished
  - As nowadays computing systems are discrete systems, using atomic functions prevents multiple processes to start multiple elections at a same time

- According to the algorithm many parallel elections run when a coordinator crashes. Do you implement parallel elections?
  - Parallel elections are not implemented
  - There is only used "lock“ method to deal with the concurrency of several logically active processes

Bully algorithm - Demonstration

- How do you count the election steps? Is it the number of messages sent before the leader is found?
  - Election steps are counted as number of messages send before leader is found

- How many are the steps when you press start for the first time?
  - When the processes are created for the first time, the process with highest ID number is implicitly set as an coordinator, therefore there is no election process performed

Bully algorithm - Demonstration

- Is there a logic on who and when polls the leader?
  - Processes are set to poll the leader every 3 seconds
  - The processes poll the leader serially
  - Of course there is limited number of processor units in every computer, therefore every process is periodically given a short amount of processor time, the order and timing depends on operating system and JVM management of processor time
  - Therefore this "logic" is quite hard to predict
  - I think the order doesn't change as it depends on information which is held in the deactivated processes queue, which is the same after activation

Bully algorithm - Demonstration

- Is there a limit for the number of processes - threads?
  - There is no theoretical limit of processes, but of course in reality, the processes are limited by hardware of the computer on which the application is running
  - The main bottleneck is physical memory of current machine

Bully algorithm - Demonstration

- Is there a limit for the number of processes - threads?
  - quote from http://bytes.com/topic/java/answers/16952-maximum-number-threads:
    - "JVM, There are green_thread and native_thread JVMs There is also np-threads implementation (I think from IBM) where in n native threads map to p number of JVM threads (I think the JDK1.4 mixed-mode JVM is of this type).
    - Then comes the OS, I believe linux has a per process restriction (built into the kernel) of around 128 native threads (I don't recall the exact number)
    - If a JVM is spawning native thread for every thread in the JVM then the performance is based on the OS, behavior, number and everything else is dependent on the OS. Native threads are better in performance than green_threads implementation where the JVM manages its own threads. On the other hand greenthreads behave exactly the same on multiple platforms, since the JVM is managing them."
  - Furthermore ... there is some info about thread limit in windows OS
    - http://msdn.microsoft.com/en-us/library/ms686774(v=vs.85).aspx
  - In unix like systems there is no thread limitations
    - You can limit total number of running processes using "ulimit" (man ulimit) command