Κατανεμημένα Συστήματα

Μάθημα #7

Περιεχόμενα

- **Καθολική κατάσταση κατανεμημένου υπολογισμού**
- **Αλγόριθμοι Εκλογής Αρχηγού**
- **Εκλογή Αρχηγού σε Δακτύλιο – Αλγόριθμος Hirschberg-Sinclair**
- **Εκλογή Αρχηγού σε Δακτύλιο – Αλγόριθμος Variable Speeds**

Καθολική ιστορία κατανεμημένου υπολογισμού

- Η **τοπική ιστορία** μιας διεργασίας P_i συμβολίζεται με

$$h_i = e_i^1 e_i^2 \cdots e_i^{\mathbf{ci}}$$

  και αποτελεί την ακολουθία γεγονότων που έχουν εκτελεστεί στην P_i

- Η **καθολική ιστορία** H ενός κατανεμημένου υπολογισμού ορίζεται ως η ένωση των τοπικών ιστοριών όλων των διεργασιών που συμμετέχουν σε αυτόν, δηλ.,

$$H = h_1 \cup h_2 \cup \ldots \cup h_n$$

Καθολική κατάσταση (global state) κατανεμημένου υπολογισμού

Η **τοπική κατάσταση** μιας διεργασίας P_i αμέσως μετά την εκτέλεση του γεγονότος e_i^k συμβολίζεται με σ_i^k και περιέχει τις τιμές όλων των τοπικών μεταβλητών της.

Η **καθολική κατάσταση** Σ ενός κατανεμημένου υπολογισμού είναι η ένωση όλων των τοπικών καταστάσεων των επιμέρους διεργασιών $\Sigma = (\boldsymbol{\sigma_1^{k1}, \sigma_2^{k2}, \ldots , \sigma_n^{kn}})$

Η **καθολική κατάσταση μπορεί να αναπαρασταθεί γραφικά με την τομή.** Η **τομή** είναι ένα υποσύνολο της καθολικής ιστορίας, δηλαδή $C = (h_1^{c1}, h_2^{c2}, \ldots , h_n^{cn})$ και προσδιορίζεται μέσω του διανύσματος (c1,c2,...,cn)

Το σύνολο των τελευταίων γεγονότων $(e_1^{c1}, e_2^{c2}, \ldots, e_n^{cn})$ καλείται **σύνορο της τομής**

Καθολική κατάσταση (global state)

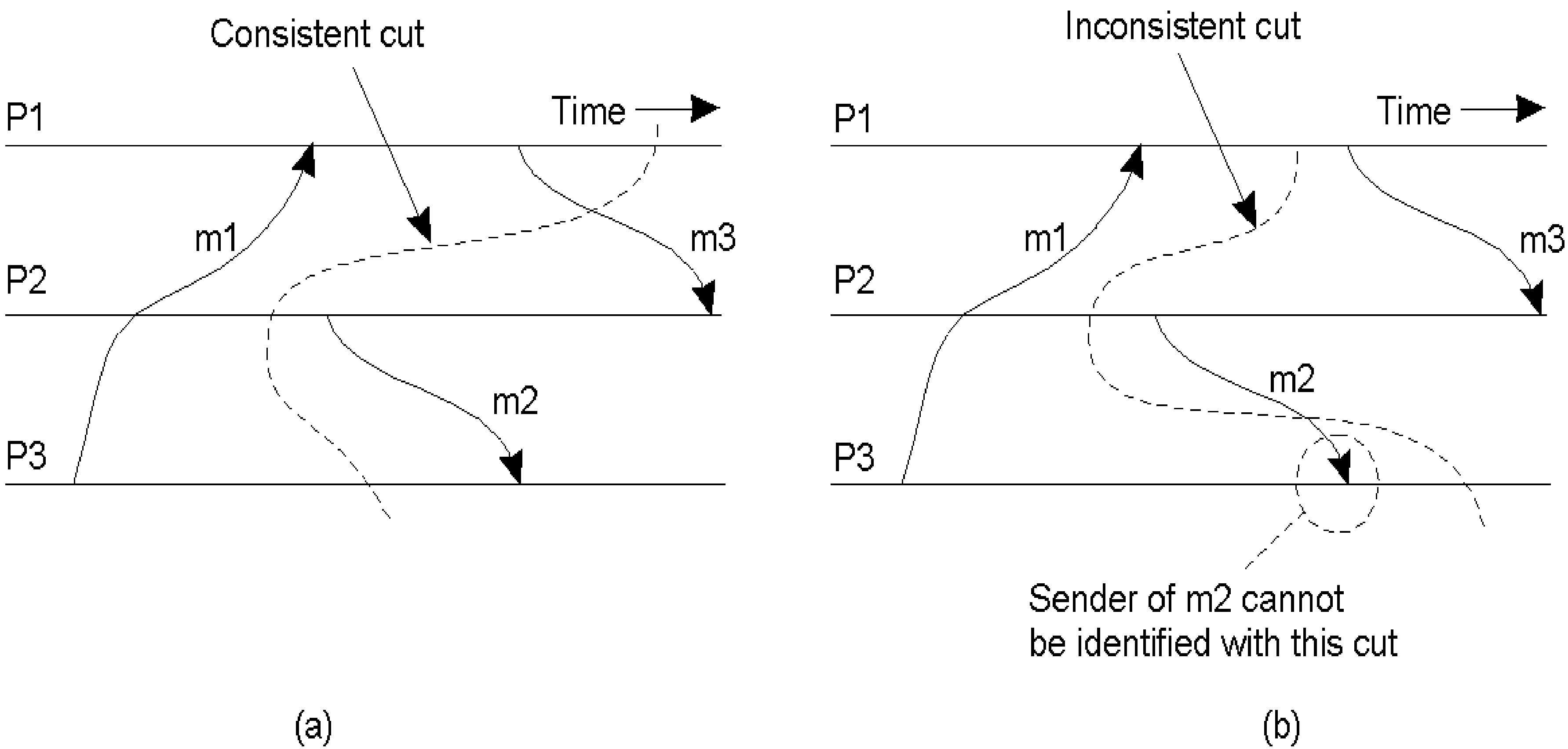
Μία **τομή C** είναι **συνεπής (consistent)** αν για όλα τα γεγονότα e και e’ ισχύει ότι

$$\forall e', e: \ (e \in C) \wedge (e' \rightarrow e) \Rightarrow e' \in C$$

Διαφορετικά, η τομή καλείται **ασυνεπής (inconsistent).**

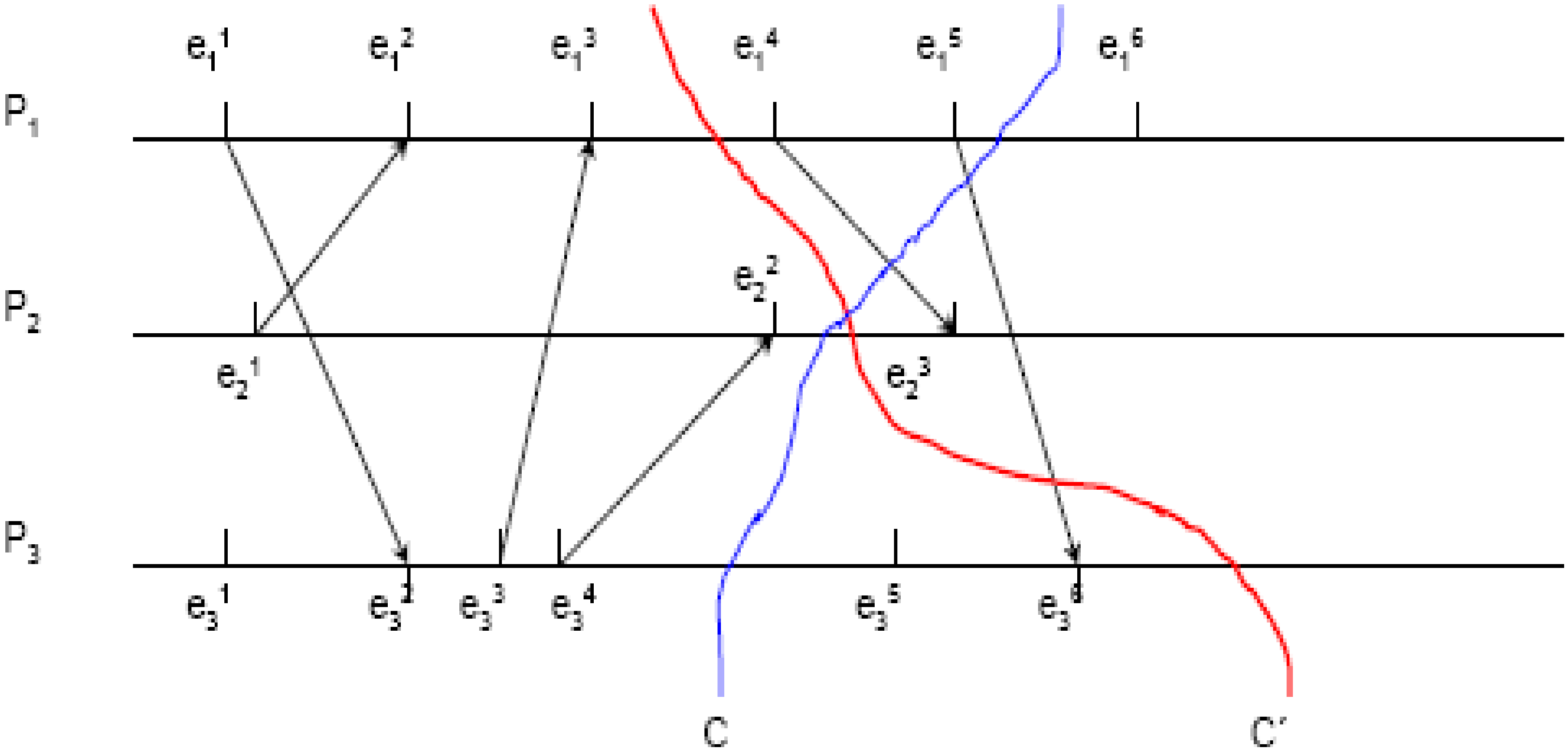
Αντίστοιχα, μια **καθολική κατάσταση είναι συνεπής** αν αντιστοιχεί σε μία συνεπή τομή.

Καθολική κατάσταση (global state)



(a) Συνεπής τομή **(b) Ασυνεπής τομή**

Καθολική κατάσταση (global state)



Consistent

$\Sigma=(\sigma_1^5, \sigma_2^2, \ldots, \sigma_3^4)$

Inconsistent

$\Sigma=(\sigma_1^3, \sigma_2^2, \ldots, \sigma_3^6)$

$(e_3^6 \in C') \wedge (e_1^5 \rightarrow e_3^6)$, but $e_1^5 \notin C'$

Κατασκευή καθολικών καταστάσεων

Αν μία διεργασία θέλει να γνωρίζει την καθολική κατάσταση ενός κατανεμημένου υπολογισμού πρέπει να την **«συλλέξει»** με τρόπο ώστε αυτή να **μην είναι**

- **ξεπερασμένη (obsolete)** – να μην αντικατοπτρίζει παρελθούσα κατάσταση του συστήματος
- **ατελής (incomplete)** –να περιέχει τις τοπικές καταστάσεις όλων των διεργασιών
- **ασυνεπής**

Η κατασκευή καθολικών καταστάσεων γίνεται από μια διεργασία – ενεργοποιητή. Αυτή ακολουθεί μια από τις εξής δύο στρατηγικές:

1. Παθητική: Όλες οι διεργασίες μόλις εκτελέσουν ένα γεγονός στέλνουν μήνυμα στην διεργασία – ενεργοποιητή, η οποία συλλέγει τις απαντήσεις και συνθέτει την καθολική κατάσταση

2. Ενεργητική: Η διεργασία – ενεργοποιητής ζητάει από τις υπόλοιπες διεργασίες τις τοπικές τους καταστάσεις για να συνθέσει την καθολική κατάσταση

Αλγόριθμοι Εκλογής Αρχηγού

Πρόβλημα: **Επιλογή μίας μόνο διεργασίας αρχηγού/συντονιστή προκειμένου να εκτελέσει ένα συγκεκριμένο καθήκον (εκτέλεση συγκεντρωτικών αλγορίθμων, επαναφορά από αδιέξοδο, να αποτελέσει τη διεργασία-ρίζα στη δημιουργία ενός δένδρου επικάλυψης)**

Δεν μπορεί αυθαίρετα μια διεργασία να αυτοανακηρυχθεί αρχηγός. Θα πρέπει να προηγηθεί η εκτέλεση ενός αλγορίθμου εκλογής.

- **Κάθε διεργασία πρέπει να αποφασίσει αν είναι ο αρχηγός ή όχι.**
- **Μία μόνο από τις διεργασίες θα πρέπει να αποφασίσει ότι αυτή είναι ο αρχηγός.**

Στόχος ενός αλγορίθμου εκλογής ***είναι να διασφαλίσει ότι όταν ξεκινήσει η διαδικασία εκλογής θα καταλήξει με τη συμφωνία όλων των διεργασιών ως προς το ποιος είναι ο νέος αρχηγός.***

Αλγόριθμοι Εκλογής Αρχηγού

Αρχικά

– ένα αυθαίρετο μη κενό σύνολο διεργασιών

– όλες οι διεργασίες ξεκινάνε από την ίδια κατάσταση (candidate)

Κάθε διεργασία εκτελεί τον ίδιο αλγόριθμο

Υπάρχουν δύο είδη τερματικών καταστάσεων για μια διεργασία:

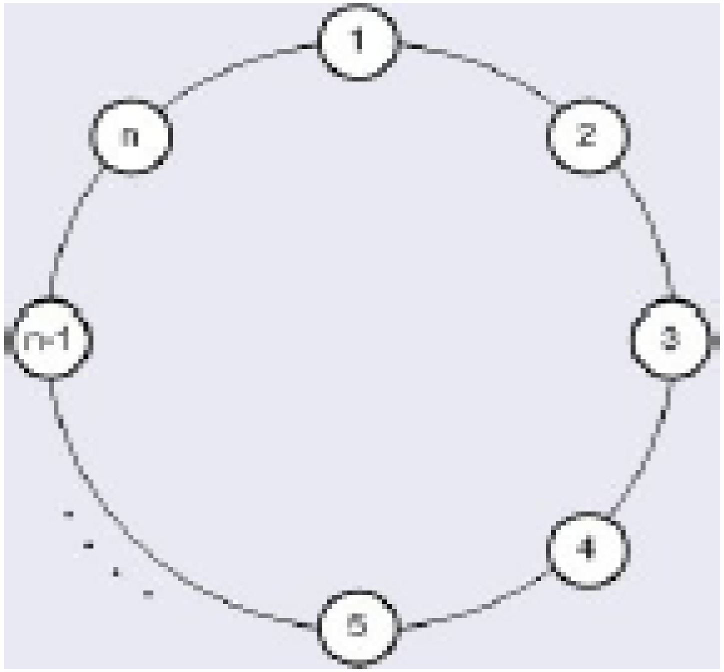
- **η τερματική κατάσταση στην οποία η διεργασία έχει εκλεγεί αρχηγός (*κατάσταση ισχύος ή αρχηγού*) και**
- **η τερματική κατάσταση στην οποία η διεργασία δεν είναι ο αρχηγός (*κατάσταση μη-ισχύος ή χαμένου*).**

Επιτρεπτές εκτελέσεις:

- **Κάθε διεργασία τελικά εισέρχεται σε μια τερματική κατάσταση, ισχύος ή μη.**
- **Μόνο μια διεργασία, ο αρχηγός, μπαίνει σε τερματική κατάσταση ισχύος.**

Εκλογή Αρχηγού σε Δακτύλιο

Έστω ένα κατανεμημένο σύστημα από *n* διεργασίες τοποθετημένες σε ένα δίκτυο δακτυλίου.



- **Οι διεργασίες έχουν μοναδικές ταυτότητες (IDs).**
- **Οι διεργασίες δεν γνωρίζουν τις ταυτότητες των υπόλοιπων διεργασιών**

Κάθε διεργασία γνωρίζει μόνο ποιος είναι ο αριστερός *(σύμφωνα με τους δείκτες του ρολογιού)* και ποιος ο δεξιός *(αντίθετα με τους δείκτες του ρολογιού)* της γείτονας.

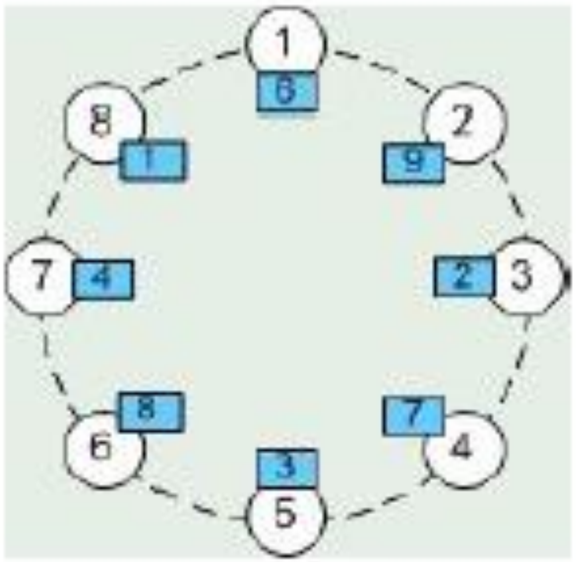
Θεωρούμε ότι οι διεργασίες είναι αριθμημένες από 1 έως n χωρίς οι ίδιες να γνωρίζουν αυτή την αρίθμηση

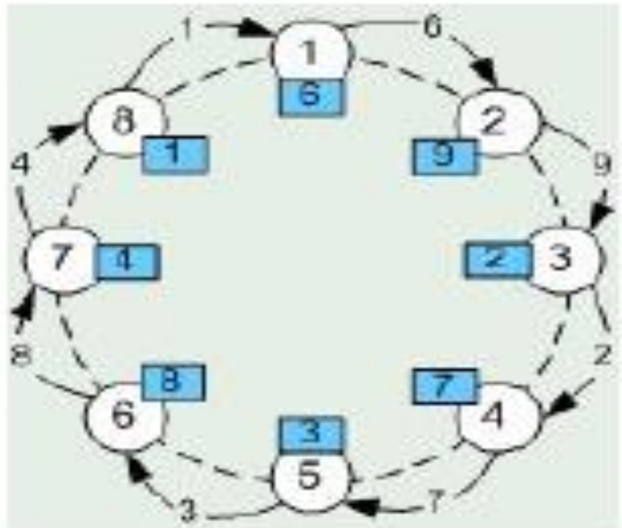
Αλγόριθμος Εκλογής Αρχηγού σε Δακτύλιο

Ενέργειες κάθε διεργασίας p:

- **Αποστολή του ID της στα αριστερά.**
- **Όταν η p λάβει ένα ID (από δεξιά) κάνει τα εξής:**
  - **αν είναι μεγαλύτερο από το δικό της, το προωθεί προς τα αριστερά**
  - **αν είναι μικρότερο από το δικό της, το αγνοεί (και δεν το προωθεί)**
  - **αν είναι ίσο με το δικό της, αποφασίζει πως αυτή είναι ο αρχηγός στο σύστημα και στέλνει ένα μήνυμα τερματισμού προς τα αριστερά**
- **Όταν η p λάβει μήνυμα τερματισμού, το προωθεί προς τα αριστερά και εισέρχεται σε τερματική κατάσταση μη-ισχύος.**

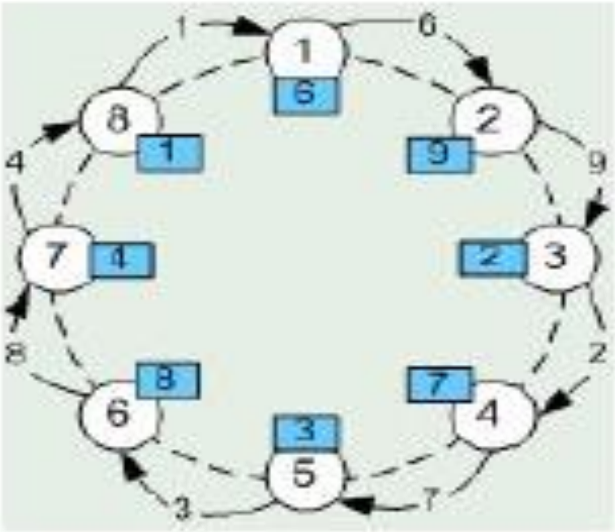
Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

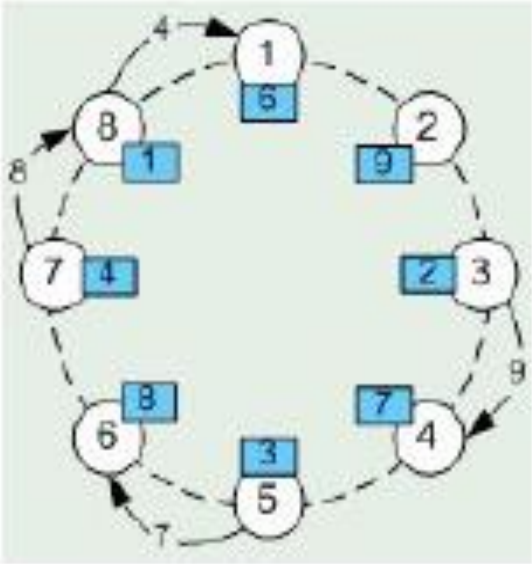




Βήμα 1

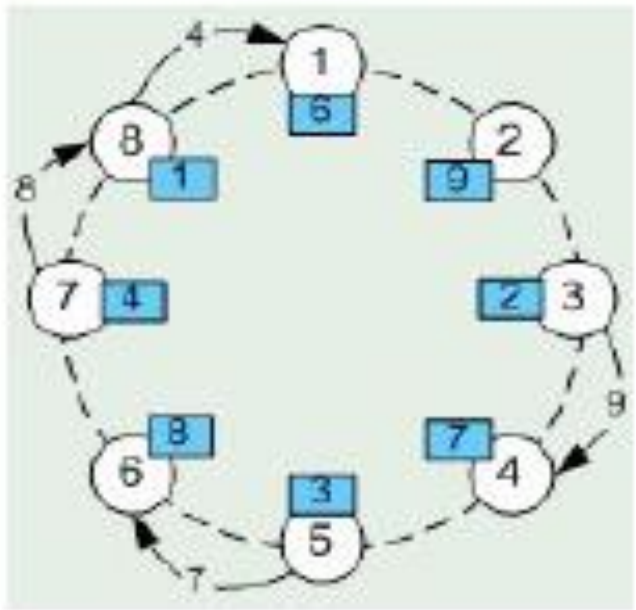
Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

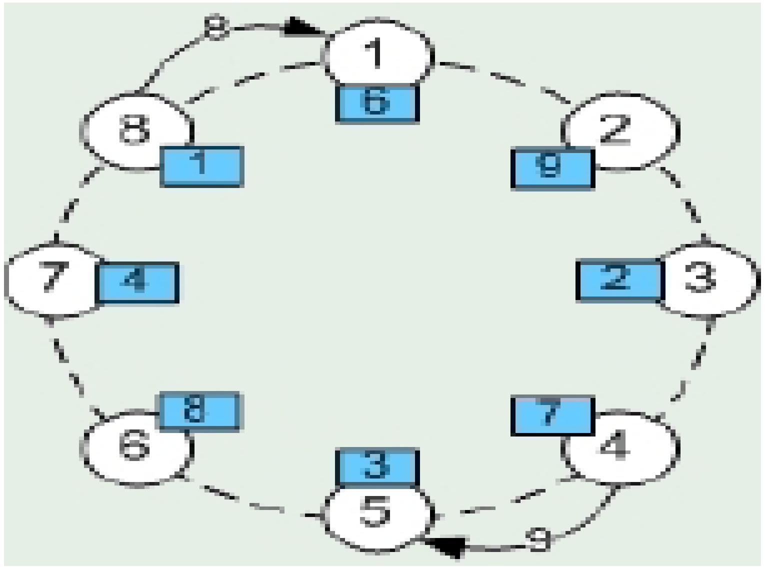




Βήμα 1 **Βήμα 2**

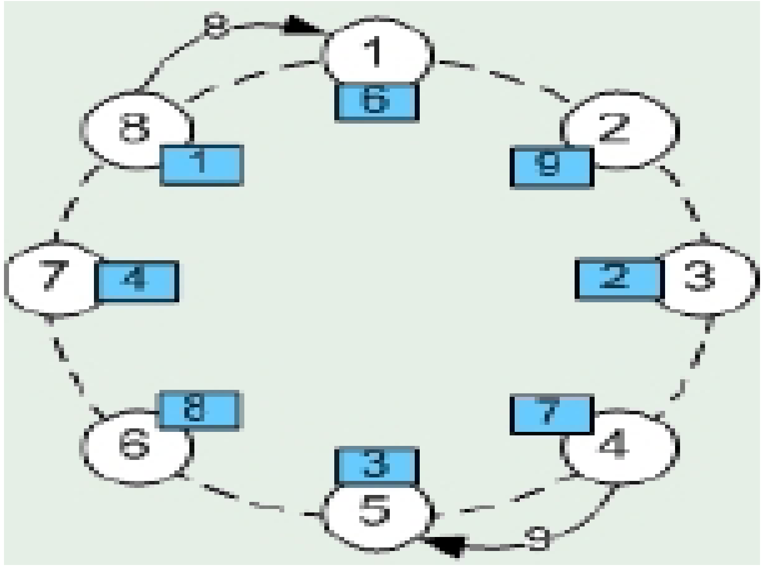
Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

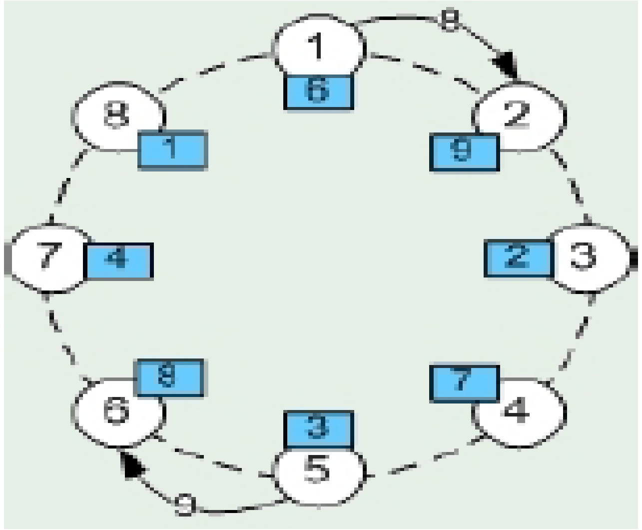




Βήμα 2 **Βήμα 3**

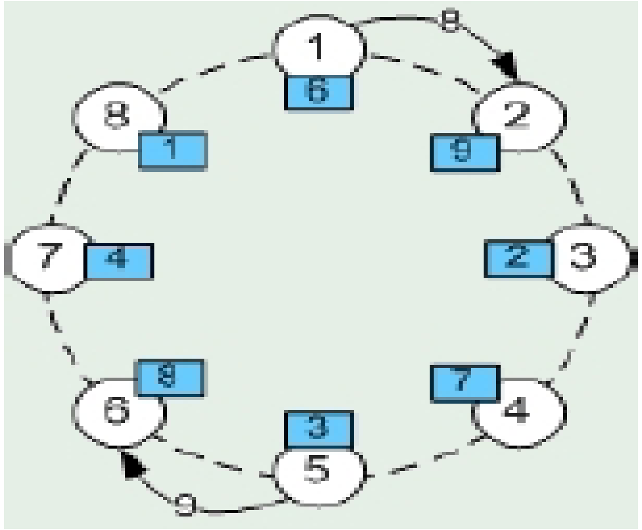
Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο



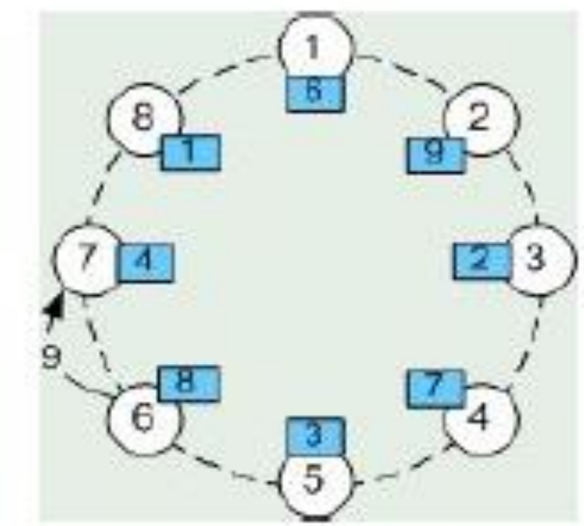


Βήμα 3 **Βήμα 4**

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

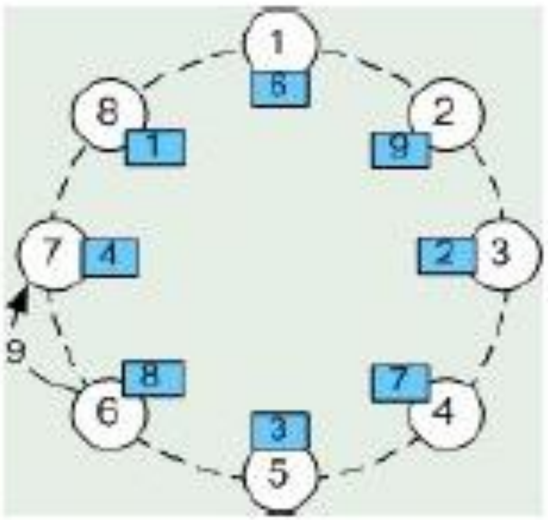


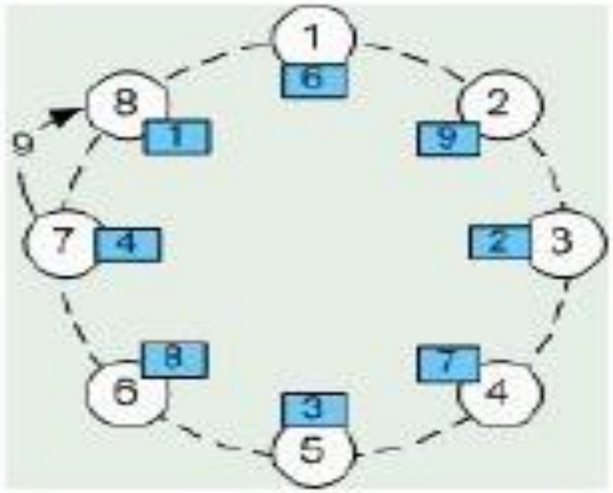
Βήμα 4



Βήμα 5

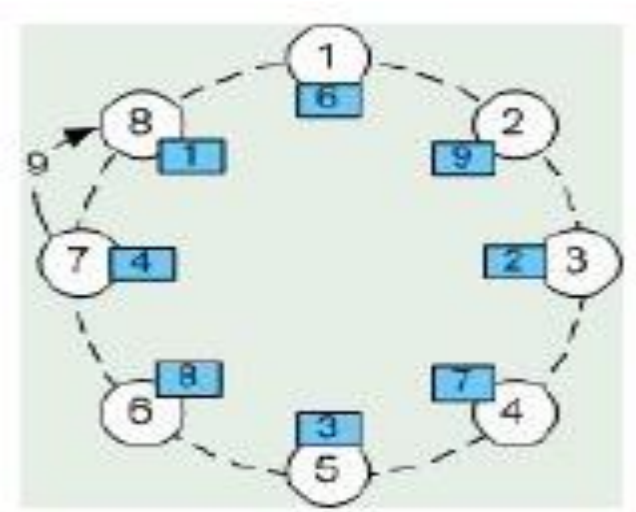
Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο



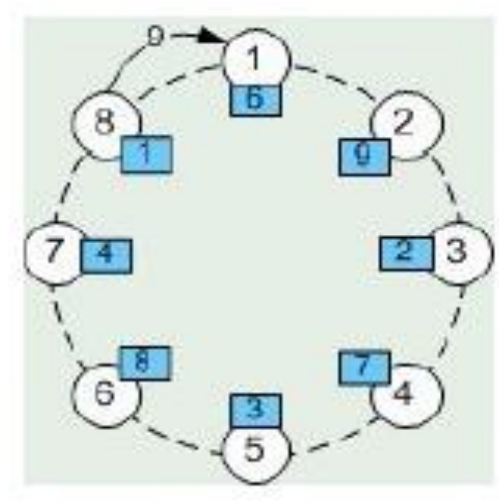


Βήμα 5 **Βήμα 6**

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

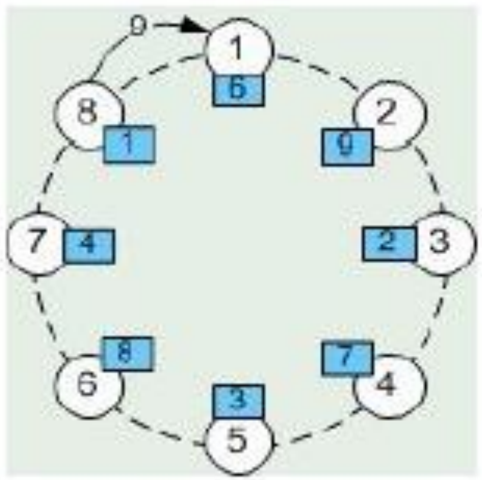


Βήμα 6

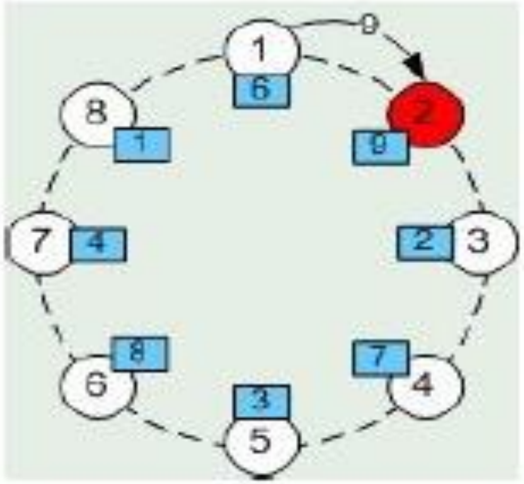


Βήμα 7

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο



Βήμα 7



Βήμα 8

Ορθότητα - Πολυπλοκότητα

Ορθότητα αλγορίθμου: Θα **εκλεγεί ως αρχηγός** η διεργασία με **το μεγαλύτερο ID**. Το μήνυμα με αυτό το ID θα περάσει από όλες τις διεργασίες.

Χρονική πολυπλοκότητα: 2n = O(n)

Η Πολυπλοκότητα Επικοινωνίας **εξαρτάται από τη διάταξη των διεργασιών**

- Το μέγιστο ID θα περάσει από όλες τις διεργασίες (*n* μηνύματα)
- Το δεύτερο μέγιστο ID θα ταξιδέψει έως ότου συναντήσει το μέγιστο ID
- Το τρίτο μέγιστο ID θα ταξιδέψει έως ότου συναντήσει το μέγιστο ή το δεύτερο μέγιστο ID.
- κ.ο.κ

Πολυπλοκότητα Επικοινωνίας: απαιτούνται *O*(n^2) μηνύματα

Η **χειρότερη διάταξη των IDs** είναι σε φθίνουσα σειρά κατά την ωρολογιακή φορά

Τότε:

- το δεύτερο μέγιστο ID συνεισφέρει *n-1* μηνύματα
- το τρίτο μέγιστο ID συνεισφέρει *n-2* μηνύματα
- το τέταρτο μέγιστο ID συνεισφέρει *n-3* μηνύματα
- κ.οκ.

Άρα συνολικά απαιτούνται

$$\sum_{i=1}^{n}(n-i+1) = \sum_{i=1}^{n} i = \Theta(n^2)$$

μηνύματα

Εκλογή Αρχηγού σε Δακτύλιο που απαιτεί *O(nlogn)* μηνύματα (διατηρώντας τη χρονική πολυπλοκότητα σε O(n)) (Αλγόριθμος Hirschberg-Sinclair)

m-γειτονιά μιας διεργασίας ***p***: **το σύνολο των διεργασιών που βρίσκονται σε απόσταση το πολύ *m* από την *p* στο δακτύλιο** (είτε προς τα αριστερά ή προς τα δεξιά).

Περιγραφή Αλγορίθμου

Λειτουργεί σε $k=0,\ldots,\log n -1$ φάσεις.

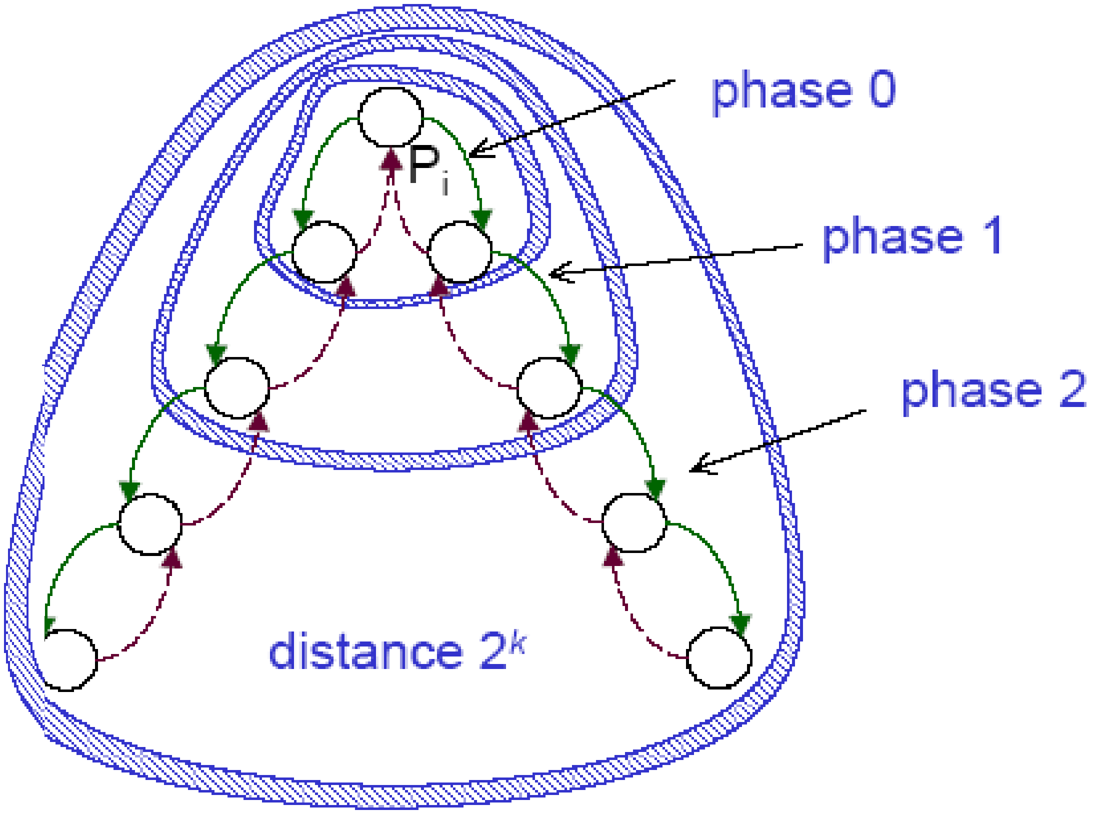
- Στην ***k-οστή*** φάση, ένας επεξεργαστής προσπαθεί να γίνει ο προσωρινός αρχηγός της ***2^k-γειτονιάς*** του.
- Μόνο οι επεξεργαστές που εκλέγονται αρχηγοί στην ***k-οστή*** φάση θα συνεχίσουν στην ***(k+1)-οστή*** φάση.

Αλγόριθμος Hirschberg-Sinclair

Περιγραφή *k-οστής Φάσης*

- Κάθε διεργασία *p*, που εκλέχθηκε προσωρινός αρχηγός στην *(k-1)-οστή φάση*, στέλνει μηνύματα <*probe*> με το ID της σε όλους τους κόμβους στην ***2^k-γειτονιά*** της. Κάθε μήνυμα <*probe*> περιέχει τον αριθμό της τρέχουσας φάσης ***k***, και έναν μετρητή ***d*** (του μήκους του μονοπατιού που έχει διανυθεί).
- Μια διεργασία αγνοεί ένα μήνυμα τύπου <*probe*>, αν αυτό περιέχει ID που είναι μικρότερο από το δικό της.
- Όταν ένα μήνυμα τύπου <*probe*> φθάσει στην τελευταία διεργασία στη τρέχουσα γειτονιά, τότε αυτή η διεργασία στέλνει στην *p* ένα μήνυμα τύπου <*reply*>.
- Αν η *p* λάβει το <*reply*> και από τις δύο κατευθύνσεις, αποφασίζει πως είναι ο αρχηγός της ***2^k-γειτονιάς*** της στη φάση *k*.
- Η διεργασία που θα λάβει το δικό της μήνυμα τύπου <*probe*>, τερματίζει σε κατάσταση ισχύος (στέλνοντας μήνυμα τερματισμού στις υπόλοιπες).

Αλγόριθμος Hirschberg-Sinclair



Algorithm 5 Asynchronous leader election: code for processor p_i, $0 \le i < n$.

Initially, $asleep$ = true

upon receiving no message:
    if $asleep$ then
        $asleep$:= false
        send $\langle$probe, $id, 0, 1\rangle$ to left and right

upon receiving $\langle$probe, $j, k, d\rangle$ from left (resp., right):
    if $j = id$ then terminate as the leader
    if $j > id$ and $d < 2^k$ then // forward the message
        send $\langle$probe, $j, k, d+1\rangle$ to right (resp., left) // increment hop counter
    if $j > id$ and $d \ge 2^k$ then // reply to the message
        send $\langle$reply, $j, k\rangle$ to left (resp., right)
// if $j < id$, message is swallowed

upon receiving $\langle$reply, $j, k\rangle$ from left (resp., right):
    if $j \ne id$ then send $\langle$reply, $j, k\rangle$ to right (resp., left) // forward the reply
    else // reply is for own probe
        if already received $\langle$reply, $j, k\rangle$ from right (resp., left) then
            send $\langle$probe, $id, k+1, 1\rangle$ to left and right ! // phase k winner

Αλγόριθμος Hirschberg-Sinclair
Πολυπλοκότητα μηνυμάτων

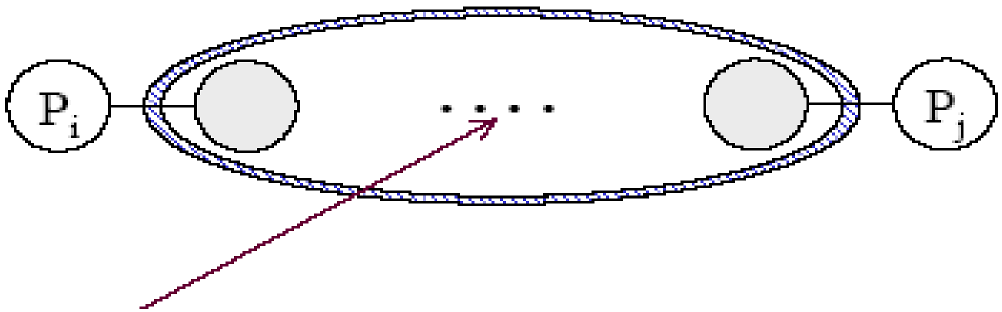
- **Κάθε μήνυμα ανήκει σε μία φάση και στέλνεται από μία συγκεκριμένη διεργασία**
- **Η απόσταση εξερεύνησης στη φάση *k* είναι 2^k , $k>=0$**
- **Ο αριθμός των μηνυμάτων που αποστέλλονται προερχόμενα από μία συγκεκριμένη διεργασία στη φάση *k* είναι το πολύ**

  $4 \cdot 2^k$ (σήματα <*probe*> και <*reply*>)
- **Ο αριθμός των διεργασιών που εκλέγονται αρχηγοί στη φάση *k* είναι το πολύ $n/(2^k+1)$.**

  Δύο αρχηγοί της *k-οστής* φάσης θα πρέπει να έχουν ανάμεσά τους τουλάχιστον 2^k διεργασίες αφού για να εκλεγούν αρχηγοί θα πρέπει να έχουν το μεγαλύτερο ID σε μια ακτίνα 2^k γύρω τους.
- **Υπάρχουν το πολύ log n φάσεις αφού σε κάθε φάση διπλασιάζεται η απόσταση εξερεύνησης**
- **Συνολικά μηνύματα:** $n + \sum_{i=1}^{\log n} 4 \cdot 2^i \cdot n/(2^{i-1}+1) = O(n \log n)$
- **Χρονική πολυπλοκότητα:** $2 \cdot (2^0+2^1+2^2+\ldots+2^{\log n}) = O(n)$

Αλγόριθμος Hirschberg-Sinclair Πολυπλοκότητα μηνυμάτων

The closest together than two *k* temporal leader, P_i and P_j, can be is if the left side of P_i's *k*-neighbourhood is exactly the right side of P_j's *k*-neighbourhood.

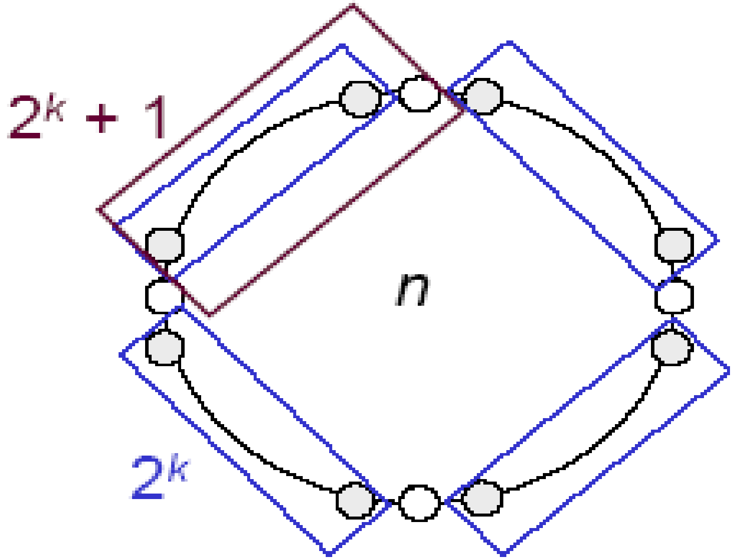


That is there are 2^k processes in between P_i and P_j.

The maximum number of phase *k* temporary leaders is achieved when this pattern continues around the ring.

Thus, in a group of $2^k + 1$, at most one can initiate messages along paths of length 2^{k+1}.

Αλγόριθμος Hirschberg-Sinclair Πολυπλοκότητα μηνυμάτων



The number of leader in this case is:

$$\frac{n}{2^k + 1}$$

Εκλογή Αρχηγού σε Δακτύλιο – Αλγόριθμος Μεταβλητών Ταχυτήτων (Variable Speeds Algorithm)

- **Αρχηγός εκλέγεται** η διεργασία **με το μικρότερο ID**
- Η βασική ιδέα του αλγορίθμου βρίσκεται στη **διαφοροποίηση της "ταχύτητας"** με την οποία τα μηνύματα **"ταξιδεύουν"** στο δίκτυο.
- Συγκεκριμένα, τα μηνύματα μιας διεργασίας με μικρό ID ταξιδεύουν πιο γρήγορα από τα μηνύματα μια άλλης με μεγαλύτερο ID.
- Μία διεργασία εκλέγεται αρχηγός όταν το μήνυμα με την ταυτότητά της κάνει το γύρο του δικτύου και επιστρέψει στην ίδια διεργασία.
- Το μήνυμα μιας διεργασίας με **ταυτότητα *i*** ταξιδεύει στο δίκτυο με **καθυστέρηση 2^i-1 γύρους**. Οι διεργασίες, δηλαδή, που λαμβάνουν μήνυμα με τη ταυτότητα *i*, προωθούν το μήνυμα αυτό στον αριστερό γείτονά τους μετά από 2^i-1 γύρους.

Αλγόριθμος Variable Speeds

Οι διεργασίες διατηρούν τις εξής μεταβλητές:

- την **μεταβλητή my_ID** με τιμή το ***ID*** της διεργασίας
- την **μεταβλητή min_seen_ID** με αρχική τιμή το άπειρο
- την **μεταβλητή *leader =\{true,false\}*** με αρχική τιμή ***false***
- **την μεταβλητή candidate =*\{true,false\}***

Κάθε διεργασία *k* γνωρίζει την ταυτότητα της (ID_k) και εάν

1. είναι υποψήφια για αρχηγός *(candidate=true)* οπότε συμμετέχει στην εκλογή αρχηγού

2. δεν είναι υποψήφια για αρχηγός *(candidate=false)* *και* απλά χρησιμοποιείται για την προώθηση των IDs των διαδικασιών που συμμετέχουν στην εκλογή αρχηγού.

Αλγόριθμος Variable Speeds

Αρχικά κάθε υποφήφια διεργασία k με my_ID=ID_k στέλνει το ID_k στην αριστερή της διεργασία στον γύρο $2^{IDk} – 1$.

Όταν η υποψήφια διεργασία i λάβει μήνυμα με την ταυτότητα ID_j της διεργασίας j κάνει τα εξής:

if $ID_j > ID_i$ ή $ID_j >$ της τιμής της μεταβλητής min_seen_ID της i **then**

αγνοεί το μήνυμα και δεν το προωθεί

else

θέτει min_seen_ID = ID_j

προωθεί το μήνυμα μετά από $2^{IDj} – 1$ γύρους

Όταν η μη υποψήφια διεργασία i λάβει μήνυμα με την ταυτότητα ID_j της διεργασίας j κάνει τα εξής:

if $ID_j >$ της τιμής της μεταβλητής min_seen_ID της i then

αγνοεί το μήνυμα (και δεν το προωθεί)

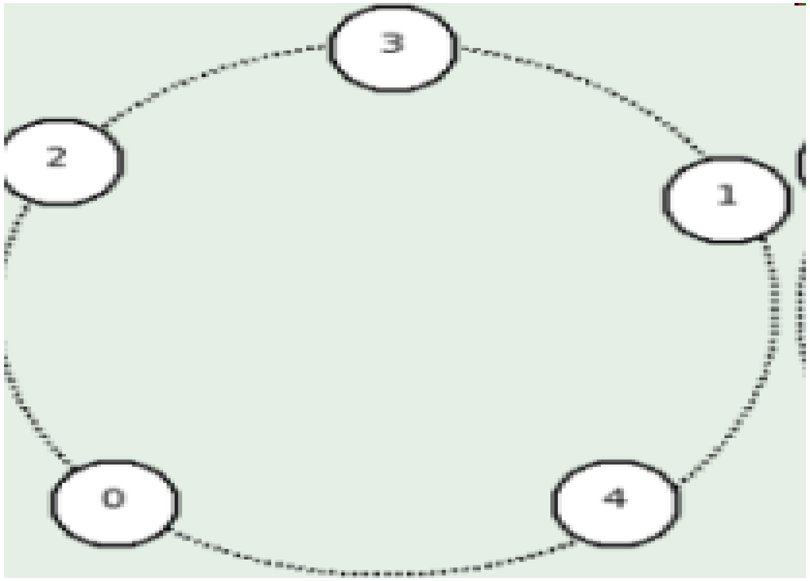
else

θέτει min_seen_ID = ID_j

προωθεί το μήνυμα μετά από $2^{IDj} – 1$ γύρους

Όταν η υποψήφια διεργασία i λάβει πίσω μήνυμα με την ταυτότητά της (ID_i) αποφασίζει πως αυτή είναι ο αρχηγός θέτοντας leader = true

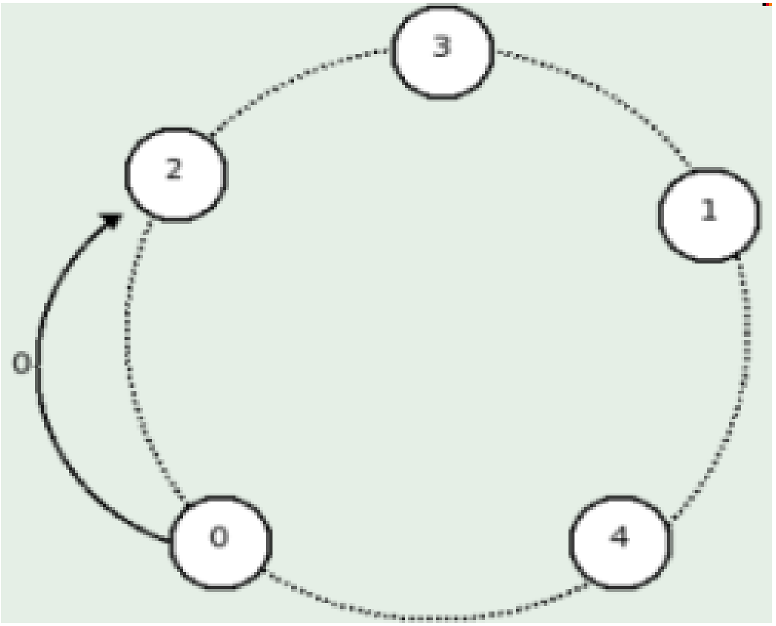
Παράδειγμα Εκτέλεσης του Αλγορίθμου Variable Speeds



Οι διεργασίες 1 και 0 είναι υποψήφιες

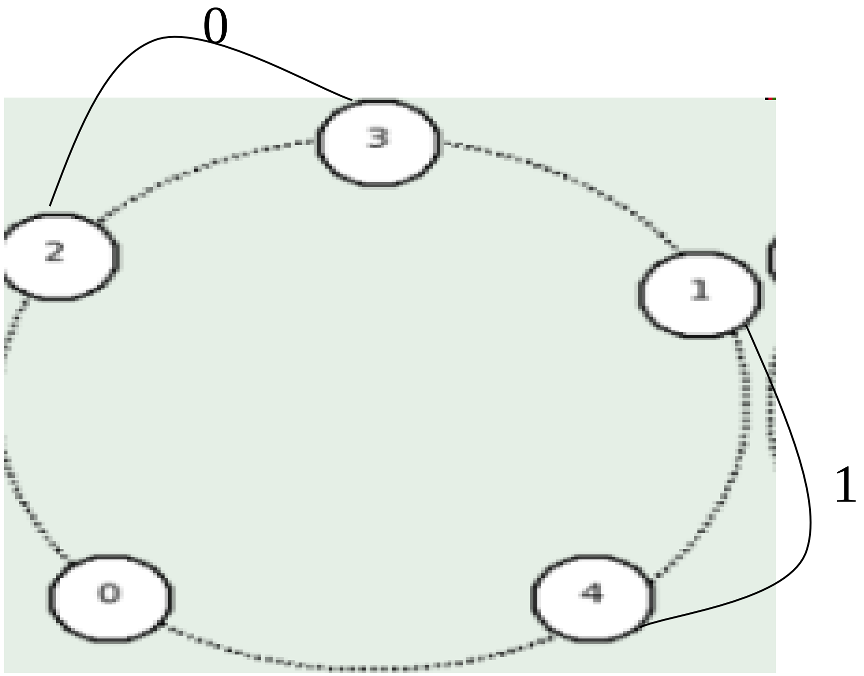
Παράδειγμα Εκτέλεσης του Αλγορίθμου Variable Speeds

- **Στον πρώτο γύρο η διεργασία 0 στέλνει την ταυτότητά της στην αριστερή γειτονική της διεργασία. Η διεργασία 1 δεν έχει δικαίωμα να προωθήσει το μήνυμά της στον πρώτο γύρο.**



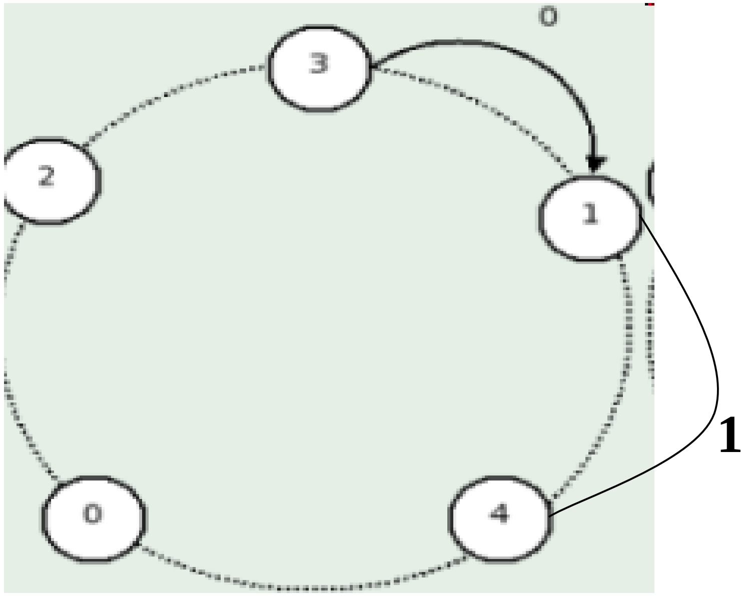
Παράδειγμα Εκτέλεσης του Αλγορίθμου Variable Speeds

- **Στον δεύτερο γύρο το μήνυμα της 1 προωθείται στην διεργασία 4 ενώ η διεργασία 2 προωθεί την ταυτότητα της 0 στην 3.**



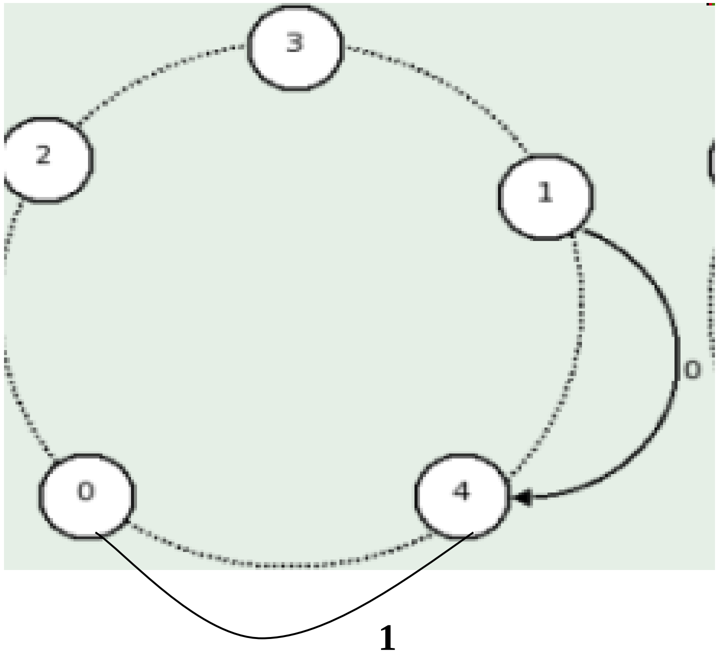
Παράδειγμα Εκτέλεσης του Αλγορίθμου Variable Speeds

- **Στον τρίτο γύρο η διεργασία 4 δεν έχει δικαίωμα να προωθήσει την ταυτότητα της 1. Η διεργασία 3 προωθεί την ταυτότητα της 0 στην 1.**



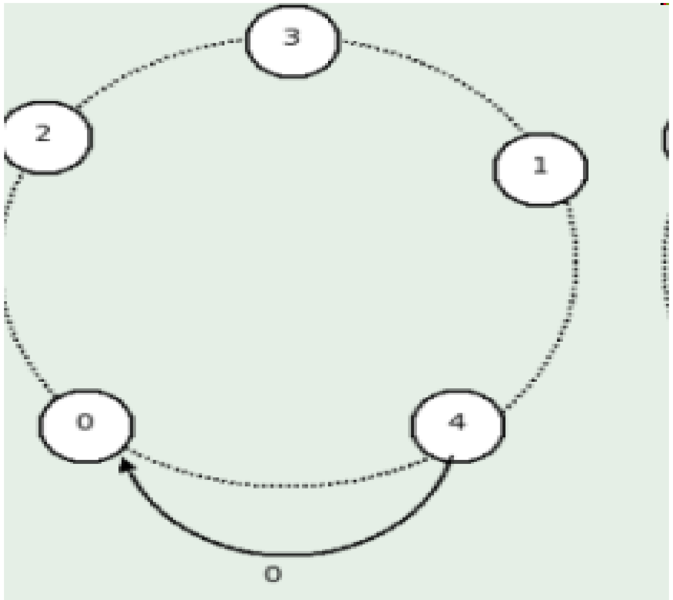
Παράδειγμα Εκτέλεσης του Αλγορίθμου Variable Speeds

- **Στον τέταρτο γύρο το μήνυμα της 1 προωθείται στην 0. Επειδή η 0 είναι υποψήφια συγκρίνει τη δικιά της ταυτότητα με την ταυτότητα που φέρει το μήνυμα. Επειδή 0<1 η διεργασία 0 δεν θα μεταδώσει περαιτέρω το μήνυμα της 1. Η διεργασία 1 προωθεί την ταυτότητα της 0 στην 4.**



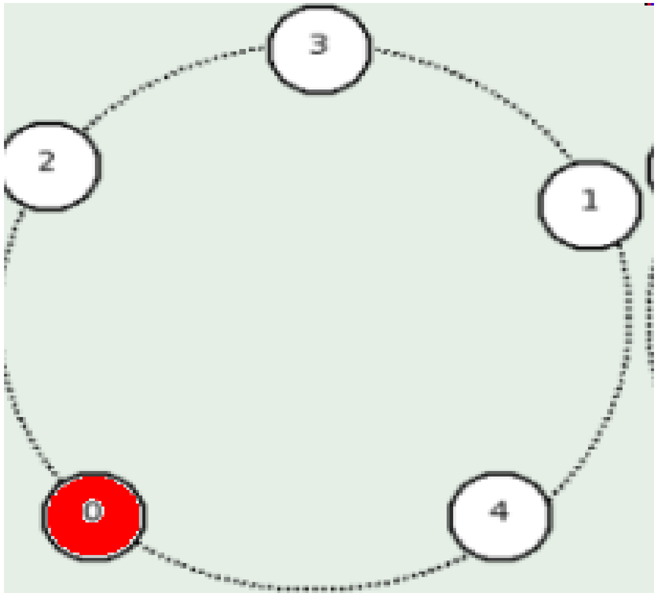
Παράδειγμα Εκτέλεσης του Αλγορίθμου Variable Speeds

- **Στον πέμπτο και τελευταίο γύρο η ταυτότητα της 0 έχει επιστρέψει σ’ αυτήν. Η διαδικασία 0 θέτει *leader* = *true*.**



Παράδειγμα Εκτέλεσης του Αλγορίθμου Variable Speeds

- **Τερματισμός**



Χρονική πολυπλοκότητα του Αλγορίθμου Variable Speeds

Θεώρημα. Η χρονική πολυπλοκότητα του αλγορίθμου Variable Speeds είναι $n2^m$, όπου m είναι το ID της διεργασίας με τη μικρότερη ταυτότητα σε ένα δακτύλιο n διεργασιών.

Απόδειξη.

Έστω i η διεργασία με το μικρότερο ID, δηλαδή $ID_i = m$. Τότε το m διακινείται n φορές (δημιουργεί n μηνύματα) έως ότου φθάσει ξανά στη διεργασία i. Οι διεργασίες, που λαμβάνουν μήνυμα με τη ταυτότητα m, προωθούν το μήνυμα αυτό στον αριστερό γείτονά τους μετά από 2^m-1 γύρους.

Επομένως, ο συνολικός χρόνος είναι $n(2^m\text{-}1 + 1) = n2^m$

Πολυπλοκότητα Επικοινωνίας του Αλγορίθμου Variable Speeds

Θεώρημα. Ο αριθμός των μηνυμάτων που διακινούνται από τον αλγόριθμο Variable Speeds σε ένα δακτύλιο *n* διεργασιών είναι O(*n*).

Απόδειξη.

Έστω *i* η διεργασία με το μικρότερο ID, δηλαδή $ID_i = m$. Τότε το *m* δημιουργεί *n* μηνύματα.

Έστω *j* η διεργασία με το δεύτερο μικρότερο ID, δηλαδή $ID_j = q$.

Οι διεργασίες, που λαμβάνουν μήνυμα με τη ταυτότητα *q*, προωθούν το μήνυμα αυτό στον αριστερό γείτονά τους μετά από 2^q-1 γύρους.

Πολυπλοκότητα Επικοινωνίας του Αλγορίθμου Variable Speeds

Επειδή η χρονική πολυπλοκότητα είναι $n2^m$ η διεργασία j δημιουργεί

$$\begin{aligned} & \frac{n2^m}{2^q - 1 + 1} \\ = \; & \frac{n2^m}{2^q} \\ = \; & n2^{m-q} \\ \leq \; & n2^{-1} \\ = \; & \frac{n}{2} \text{ messages.} \end{aligned}$$

Η τρίτη μικρότερη διεργασία δημιουργεί $n/4$ μηνύματα, κ.ο.κ. Επομένως, συνολικά διακινούνται $n+n/2+n/4+n/8+\ldots = n(1+1/2+1/4+\ldots) < 2n = O(n)$ μηνύματα