Κινητός και Διάχυτος Υπολογισμός (Mobile & Pervasive Computing)

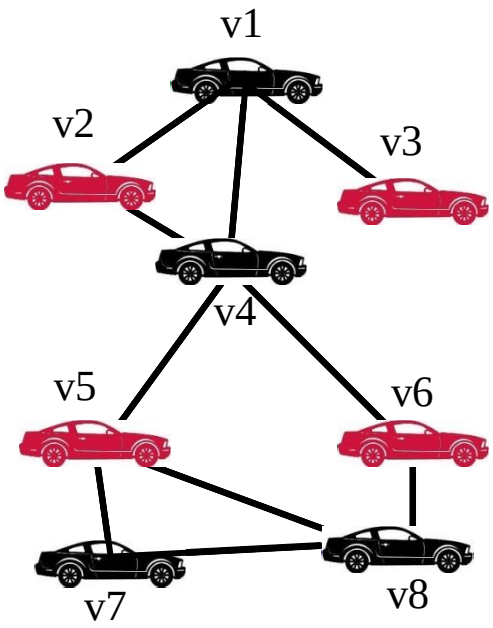
Δημήτριος Κατσαρός

Διάλεξη 16η

Περιεχόμενα

- **Clustering ad hoc δικτύων**
  - **Multi-Point Relays (MPRs)**
  - **Weakly Connected Dominating Set (WCDS)**
  - **Maximal Independent Sets (MIS)**
  - **Max-min d-cluster αλγόριθμος**

Η έννοια του MIS



- Έστω μη κατευθυνόμενο γράφημα G(V; E)
- Το υποσύνολο S ⊆ V είναι ένα independent set εάν κανένα ζεύγος κόμβων του S δεν είναι 1-hop γείτονες, π.χ., S={v1,v8}
  - Υπάρχουν πολλά
- Maximal independent set (MIS): είναι ένα independent set στο οποίο δεν μπορούμε να προσθέσουμε επιπλέον κόμβο, π.χ., S_{mal}={v3,v4,v8}
  - Υπάρχουν πολλά με διαφορετικά μεγέθη
- Maximum independent set: το MIS με το μέγιστο μέγεθος, π.χ., S_{mum}={v2,v3,v5,v6}
  - Πιθανόν να υπάρχουν πολλά , αλλά όλα με το ίδιο μέγεθος
  - NP-complete

Κάτω όριο επίδοσης

Θεώρημα:

Σε asynchronous ασύρματα ad hoc δίκτυα των οποίων ο unit-disk-graph είναι δακτύλιος, ο αριθμός των μηνυμάτων που θα στείλει οποιοσδήποτε κατανεμημένος αλγόριθμος για nontrivial CDS, είναι τουλάχιστον $\Omega(n \log n)$

Message-Optimal Αλγόριθμος CDS

Αποτελείται από δυο φάσεις:

1. Κατασκευή του Maximal Independent Set (MIS)
2. Dominating Tree

Βοηθητικοί ορισμοί (1/3)

- ***Independent set*** or stable set is a set of vertices in a graph no two of which are adjacent
- That is, it is a set V of vertices such that for every two vertices in V, there is no edge connecting the two
- Equivalently, each edge in the graph has at most one endpoint in V
- The size of an independent set is the number of vertices it contains

Βοηθητικοί ορισμοί (2/3)

- ***Maximal independent set*** or maximal stable set is an independent set that is not a subset of any other independent set
- That is, it is a set S such that every edge of the graph has at least one endpoint not in S and every vertex not in S has at least one neighbor in S

Βοηθητικοί ορισμοί (3/3)

- A maximal independent set is also a dominating set in the graph, and every dominating set that is independent must be maximal independent, so maximal independent sets are also called independent dominating sets
- A graph may have many maximal independent sets, of widely varying sizes; the largest maximal independent set is called the maximum independent set

Κατασκευή του MIS

- Δεδομένο ένα rooted spanning tree T
- Επίπεδο ενός κόμβου είναι ο αριθμός hops από τη ρίζα του spanning tree
- Rank ενός κόμβου είναι το ταξινομημένο ζεύγος του επιπέδου του και του ID του
- Λεξικογραφική διάταξη όλων των κόμβων
- Ranking διαδικασία
- Color Marking

Διαδικασία Ranking

Δομή:

1. Κάθε κόμβος διατηρεί δυο τοπικές μεταβλητές x_1, x_2
   - x_1 = # των γειτόνων των οποίων τα επίπεδα δεν έχουν αναγνωριστεί ακόμα (αρχικά είναι ο # των γειτόνων)
   - x_2 = # των παιδιών που δεν έχουν αναφέρει ακόμα την περάτωση (αρχικά είναι ο # των παιδιών)
2. Κάθε κόμβος διατηρεί μια levelList
   - Αποθηκεύει τα επίπεδα των γειτόνων του (αρχικά είναι κενή)
3. Κάθε κόμβος διατηρεί την τοπική μεταβλητή y
   - Αποθηκεύει τον αριθμό των lower-ranked γειτόνων

Διαδικασία Ranking

- Ρίζα ανακοινώνει το επίπεδό της 0 με broadcasting ενός μηνύματος LEVEL
- Όταν ληφθεί ένα μήνυμα LEVEL:
  - levelList += (sender's level, sender's rank)
  - x_1- -
- Εάν sender = its parent στο T τότε
  - its level = sender's level + 1
  - Broadcast its LEVEL message
- Εάν $x_1 = 0$
  - y = # lower ranked γείτονες, υπολογισμένοι από τη levelList

Διαδικασία Ranking

- Εάν x_2=0 (είναι φύλλο) και το level του έχει τεθεί
  - Μετάδοση ενός μηνύματος LEVEL-COMPLETE στον πατέρα
  - Εάν είναι non-leaf:
    - x_2= # of children
  - Εάν είναι root:
    - x_2= # of children
    - Τερματισμός
- Όταν ληφθεί μήνυμα LEVEL-COMPLETE:
  - x_2- -

Διαδικασία Color Marking

- Αρχικά marked με white χρώμα
- Κάθε κόμβος διατηρεί blackList (ids of its black neighbors)
- Root marks black, broadcasts BLACK message
- Όταν ληφθεί μήνυμα ένα μήνυμα BLACK:
  - blackList += sender's ID
  - Εάν node = white
    - Mark itself gray
    - Broadcast GRAY message που περιέχει το level του
- Όταν ληφθεί μήνυμα ένα μήνυμα GRAY:
  - Εάν sender's rank < its rank
    - A white node decrements y by 1
    - If y = 1 after update, marks itself black, broadcasts BLACK message

Διαδικασία Color Marking

- Όταν ένα φύλλο γίνεται marked (με gray ή black), στέλνει μήνυμα MARK-COMPLETE στον πατέρα του
- Όταν ληφθεί ένα μήνυμα MARK-COMPLETE:
  - x_2- -
  - Εάν $x_2 = 0$ και not root:
    - Μεταδίδει μήνυμα MARK-COMPLETE στον πατέρα του
  - Εάν $x_2 = 0$ και root:
    - x_1= # of its neighbors
    - Τερματισμός

Κατασκευή του Dominating Tree T*

Δομή κάθε κόμβου:

- Διατηρεί μια τοπική boolean z
  - Αρχικά είναι 0, τίθεται στο 1 εάν κάνει join στο T*
- Local variable parent
  - Αποθηκεύει ID of its parent in T*, αρχικά κενή
- childrenList
  - ID of its children in T*, αρχικά κενή

Η ρίζα του T* είναι ένας gray γείτονας της ρίζας του T με το μεγαλύτερο αριθμό black γειτόνων

Ειδικά, η ρίζα στο T διατηρεί τη μεταβλητή root, και degree, αρχικοποιημένο στο 0

Κατασκευή του Dominating Tree T*

- Επιλογή μια ρίζας για το T* από το T
- Προσκάλεσε κόμβους για να ενωθούν στο T*
- Οι internal κόμβοι του T* σχηματίζουν ένα CDS

Επιλογή ρίζας για το T*

Βασική ιδέα: μια ρίζα του T* είναι γείτονας της ρίζας του T που έχει τον μεγαλύτερο αριθμό από black γείτονες

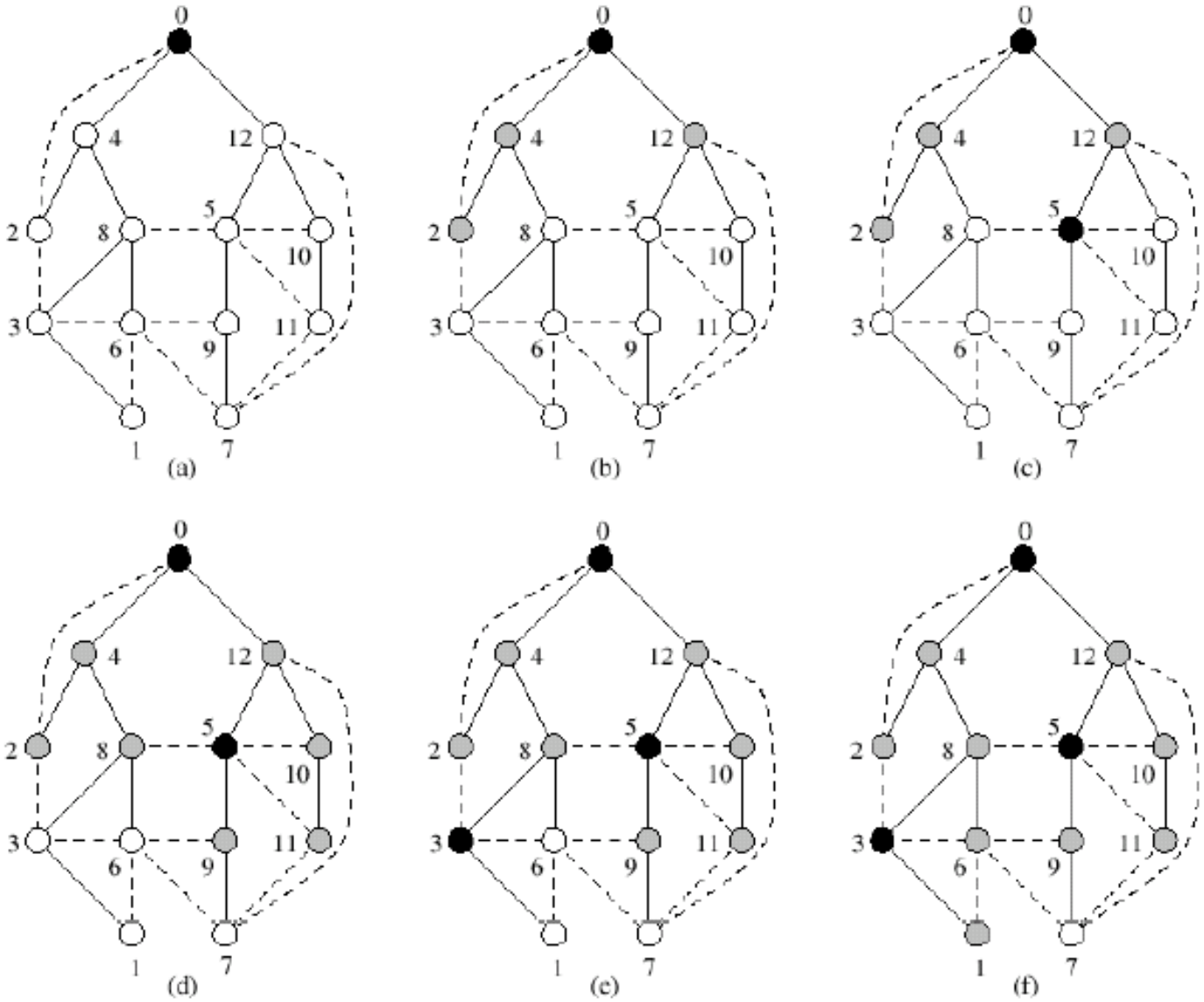
- Η ρίζα του T θέτει τη μεταβλητή x_1 ίση με τον αριθμό των γειτόνων, και κάνει broadcast ένα μήνυμα QUERY
- Όταν ληφθεί το μήνυμα QUERY από έναν gray:
  - Ο κόμβος μεταδίδει στον αποστολέα ένα μήνυμα REPORT που περιέχει τον # των black γειτόνων του
  - Ρίζα ελαττώνει x_1 κατά 1; Θέτει degree ίσο με #black γειτόνων εάν είναι μεγαλύτερη; Θέτει root variable ίσο με αυτό το id.
- Εάν $x_1 = 0$, στέλνει μήνυμα ROOT στον κόμβο που το id του είναι στην root variable
- Αυτός ο κόμβος γίνεται η ρίζα του T*

Πρόσκληση κόμβων

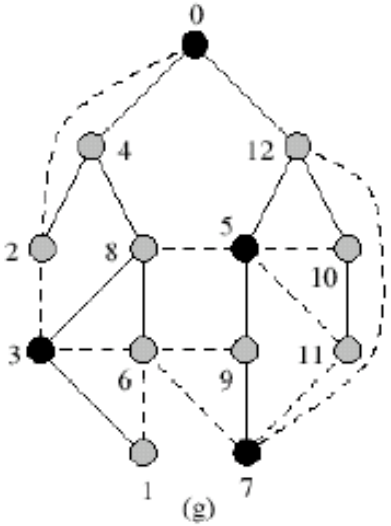
- Η ρίζα του T* θέτει z=1 και broadcasts μήνυμα INVITE2
- Όταν ληφθεί το μήνυμα INVITE2
  - Εάν color=black, z=0 τότε
    - Θέτει z =1
    - Parent = sender's ID
    - Μετάδοση JOIN μηνύματος στον αποστολέα (sender)
    - Broadcast μήνυμα INVITE1
- Όταν ληφθεί μήνυμα INVITE1
  - Εάν color=gray, z=0 τότε
    - Θέτει z = 1
    - Parent = sender's ID
    - Μετάδοση JOIN μηνύματος στον αποστολέα
    - Broadcast μήνυμα INVITE2
- Όταν ληφθεί μήνυμα JOIN από έναν κόμβο
  - Προσθήκη του sender's ID στην chidrenList

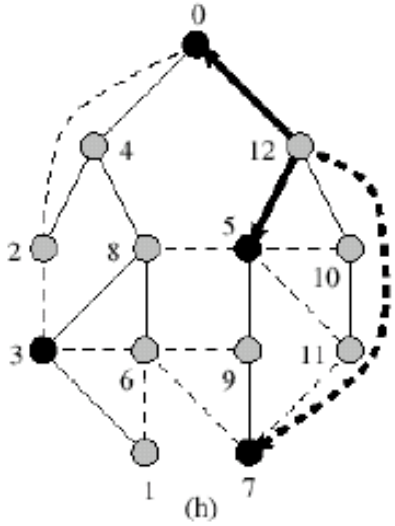
Κατασκευή του MIS και Dom. Tree T*

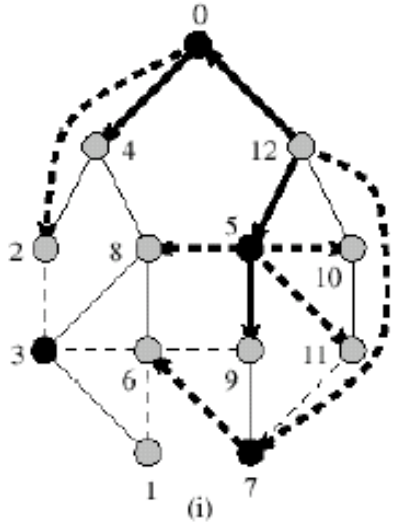
Κατασκευή MIS
(a) - (g)



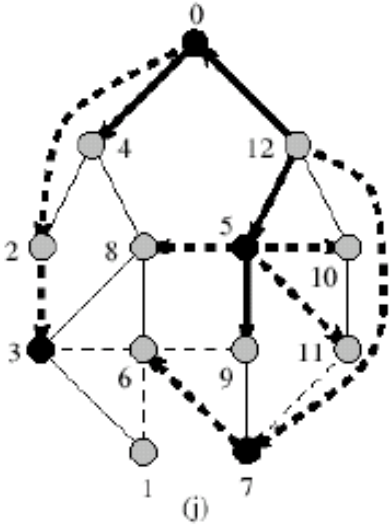
Κατασκευή του MIS και Dom. Tree T*

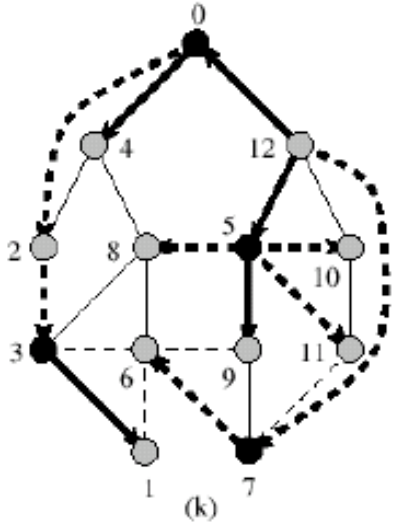






Κατασκευή DomTree (h) - (k)





CDS = {12, 0, 5, 7, 2, 3}

Ανάλυση επίδοσης

- Approximation factor <= 8
- Πολυπλοκότητα χρόνου: O(n)
- Πολυπλοκότητα μηνυμάτων: O(n*logn)
- Nontrivial CDS

Περιεχόμενα

- **Clustering ad hoc δικτύων**
  - **Weakly Connected Dominating Set (WCDS)**
  - **Maximal Independent Sets (MIS)**
  - **Max-min d-cluster αλγόριθμος**

Εισαγωγικά

- Είδαμε ότι επιβάλλουμε δόμηση σε ένα MANET διαμέσου της δημιουργίας clusters
- Τα clusterheads σχηματίζουν ένα **d-hop dominating set**
- Τα clusterheads αποτελούν ένα virtual backbone και χρησιμοποιούνται για δρομολόγηση πακέτων στους κόμβους εντός cluster
- Αλγόριθμος σχηματισμού d-clusters σε ένα MANET
  - Εκτός των clusterheads, ο αλγόριθμος περιγράφει και τη διαδικασία αναγνώρισης των gateway κόμβων
  - Ο αλγόριθμος έχει time complexity της τάξης των O(d) γύρων

Χαρακτηριστικά νέου αλγορίθμου

- Δεν απαιτούνται synchronized clocks
- Περιορίζει τον #μηνυμάτων που ανταλλάσσονται μεταξύ των κόμβων σε O(d)
- Ελαχιστοποιεί το μέγεθος των βοηθητικών δομών δεδομένων
- Ελαχιστοποιεί τον αριθμό των clusterheads ως συνάρτηση του d
- Σχηματισμός του backbone με χρήση των gateways
- Επαν-εκλογή των ίδιων clusterheads όταν είναι δυνατόν: *stability*
- Έλεγχος του αριθμού των clusterheads και της πυκνότητας των clusters density διαμέσου της παραμέτρουd
- Κατανομή της ευθύνης διαχείρισης των clusters ισότιμα μεταξύ όλων των κόμβων: *fairness*

Max-min d-clustering (1/7)

- Ο αλγόριθμος εκτελείται για 2d γύρους ανταλλαγής μηνυμάτων
- Κάθε κόμβος διατηρεί δυο πίνακες, WINNER και SENDER, κάθε ένας μήκους 2d node ids: ένα id ανά γύρο ανταλλαγής μηνυμάτων
- **Βήμα1**: Αρχικά, κάθε κόμβος θέτει το WINNER ίσο με το δικό του node id

- **Βήμα2**: (Floodmax) – Κάθε κόμβος εκπέμπει τοπικά τη δική του τιμή WINNER προς όλους τους 1-hop γείτονές του. Σε ένα γύρο, ο κόμβος επιλέγει τη μεγαλύτερη τιμή, μεταξύ της δικής του τιμής WINNER και των τιμών που λαμβάνει στο γύρο αυτό, ως νέα τιμή WINNER για τον εαυτό του. Αυτή η διαδικασία συνεχίζεται για d γύρους

Max-min d-clustering (2/7)

- **Βήμα 3**: Έπεται του βήματος Floodmax και διαρκεί επίσης d γύρους. Είναι ίδιο με το βήμα Floodmax εκτός του ότι ένας κόμβος επιλέγει τη μικρότερη, αντί για την μεγαλύτερη τιμή, ως νέα WINNER τιμή

- **Βήμα 4** (overtake): Στο τέλος κάθε γύρου πλημμυρίδας, ένας κόμβος αποφαίνεται εάν θα διατηρήσει την τρέχουσα τιμή WINNER ή θα την αλλάξει σε κάτι που έλαβε στον προηγούμενο γύρο πλημμυρίδας

- **Βήμα 5** (node pair): Ένα node pair είναι οποιοδήποτε node id, το οποίο προκύπτει τουλάχιστον μια φορά ως WINNER ταυτόχρονα στον 1ο (Floodmax) και στον 2ο (Floodmin) από τους d γύρους πλημμυρίδας για έναν κόμβο

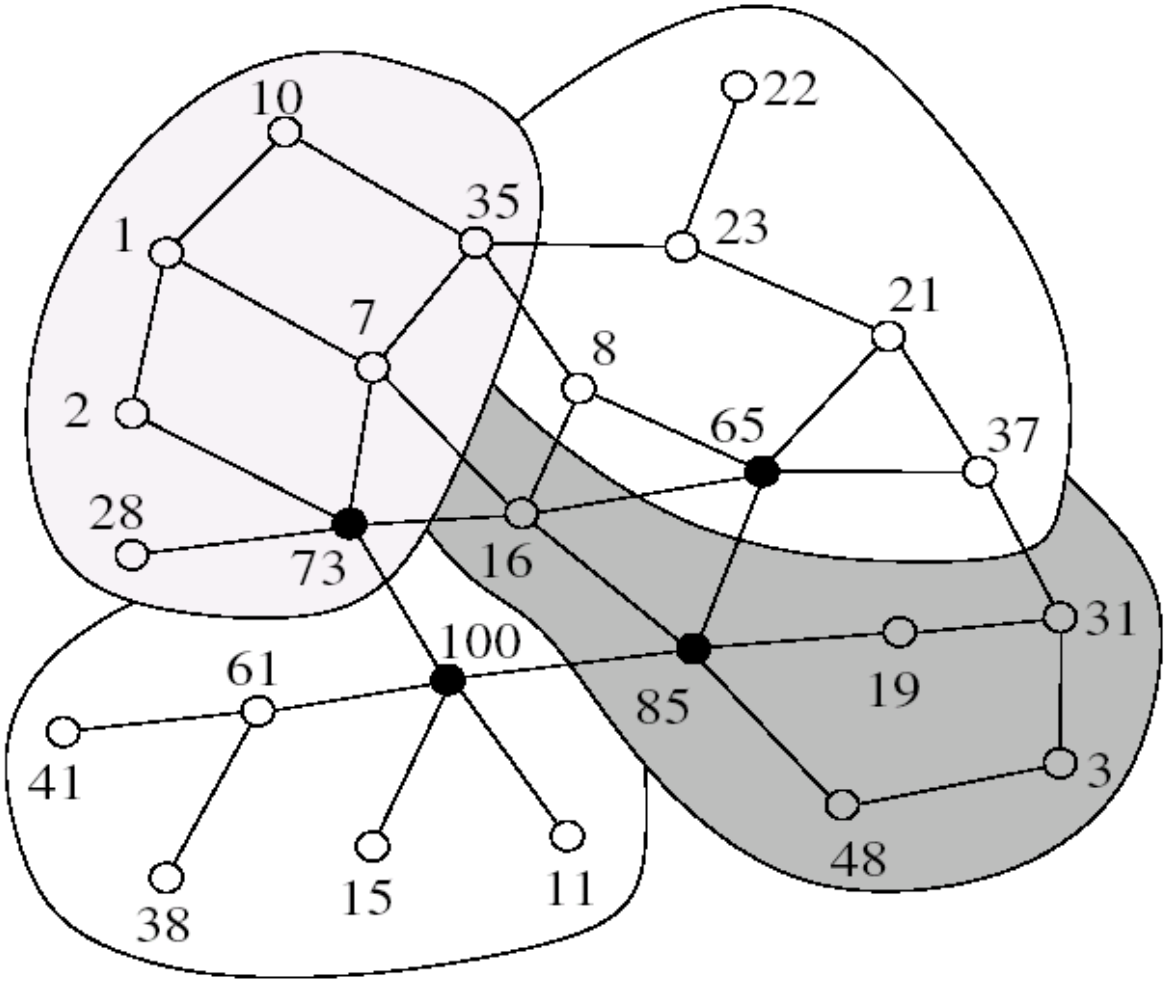
Max-min d-clustering (3/7): Κριτήρια επιλογής clusterheads

- Μετά τη συμπλήρωση του 2ου από τους d γύρους, κάθε κόμβος κοιτάζει τις "εγγραφές" του για τους 2d γύρους rounds πλημμυρίδας. Οι επόμενοι κανόνες εξηγούν τα λογικά βήματα του αλγορίθμου που εκτελεί κάθε κόμβος πάνω στις εγγραφές του
- ***Rule 1***: Πρώτα, κάθε κόμβος ελέγχει εάν έχει λάβει το δικό του αρχικό node id στο floodmin. Εάν αυτό συμβαίνει, τότε δηλώνει τον εαυτό ως clusterhead και δεν εκτελεί τα υπόλοιπα αυτής της φάσης. Αλλιώς, εκτελεί τον Rule 2
- ***Rule 2***: Κάθε κόμβος ψάχνει για node pairs. Αφού ένας κόμβος αναγνωρίσει όλα τα node pairs, επιλέγει το ελάχιστο minimum node pair ως clusterhead. Εάν ο κόμβος δεν βρει κάποιο node pair, τότε εκτελεί τον Rule 3
- ***Rule 3***: Εκλέγεται το maximum node id στους πρώτους d γύρους της πλημμυρίδας ως clusterhead για τον κόμβο

Max-min d-clustering (4/7): Επιλογή gateways και convergecast

- Για ελάττωση της πολυπλοκότητας, η επικοινωνία ξεκινά από τις παρυφές ενός cluster, gateway nodes, εσωτερικά προς το clusterhead
- Εάν κάποιοι κόμβοι γείτονες ενός κόμβου έχουν επιλέξει διαφορετικό clusterhead, τότε ο κόμβος είναι gateway node (1-hop local broadcast)
- Η δομή δεδομένων SENDER χρησιμοποιείται για να προσδιοριστεί σε ποιον επόμενο κόμβο να σταλεί το convergecast μήνυμα
- Ο αλγόριθμος μεγιστοποιεί τον αριθμό των gateways δημιουργώντας ένα backbone με πολλαπλά μονοπάτια μεταξύ γειτονικών clusterheads (για λόγους αξιοπιστίας)

Max-min d-clustering (5/7): Επιλογή gateways και convergecast



3-cluster formation in a network of 25 nodes.

Max-min d-clustering (6/7): Επιλογή gateways και convergecast

Node	10	1	2	7	35	8	23	22	21	65	37	31
Max 1	35	10	73	73	35	65	35	23	65	85	65	37
Max 2	35	73	100	100	73	85	65	35	85	100	85	85
Max 3	73	100	100	100	100	100	85	65	100	100	100	100
Min 1	73	73	100	100	73	100	65	65	85	100	100	100
Min 2	73	73	73	73	65	73	65	65	65	85	85	100
Min 3	65	73	73	65	65	65	65	65	65	65	65	85
Result	73	73	73	73	73	65	65	65	65	65	65	85

19	85	16	100	73	28	41	61	11	48	3	15	38
85	100	85	100	100	73	61	100	100	85	48	100	61
100	100	100	100	100	100	100	100	100	100	85	100	100
100	100	100	100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100	100	100	100
100	85	73	100	73	100	100	100	100	100	100	100	100
100	85	100	100	73	100	100	100	100	100	100	100	100

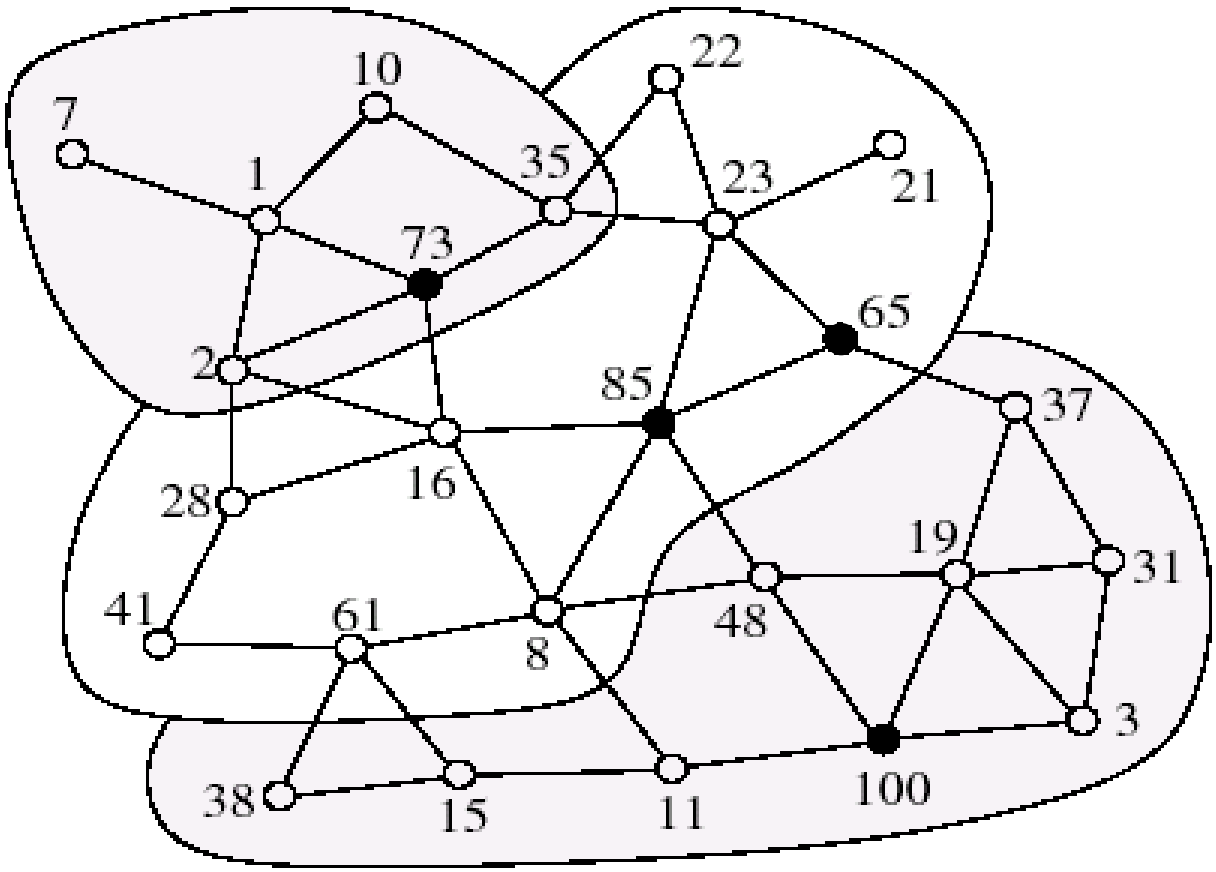
WINNER

3-cluster formation in a network of 25 nodes.

Max-min d-clustering (7/7): Ορθότητα

- **Υπόθεση 1:** Κατά τη διαδιακασίες floodmin και floodmax, κανένα node id δεν θα μεταδοθεί σε απόσταση μεγαλύτερη από d-hops από τον αρχικό κόμβο από όπου προήλθε
- **Υπόθεση 2:** Όλοι οι κόμβοι που επιβιώνουν από τη διαδιακασία floodmax εκλέγουν τους εαυτούς τους ως clusterheads
- ***Λήμμα 1:*** Εάν ο κόμβος A εκλέξει τον κόμβο B ως δικό του clusterhead, τότε ο B γίνεται clusterhead

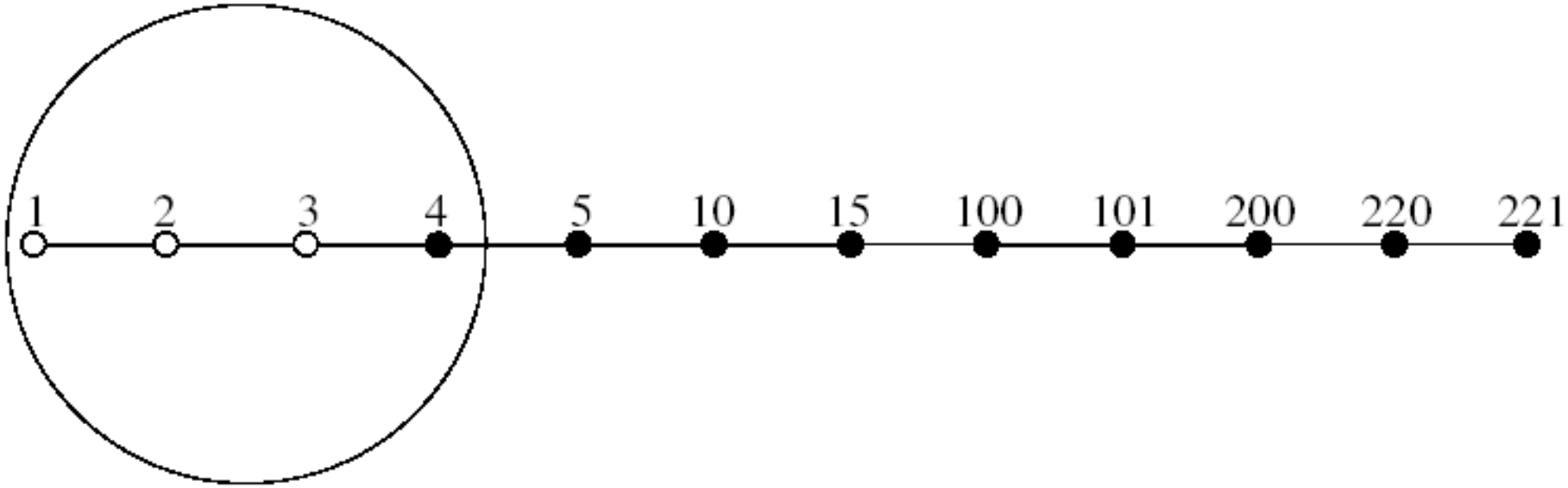
Παράδειγμα



3-cluster formation after topology change.

Παράδειγμα τοπολογίας δικτύου που παράγεται με εκτέλεση του αλγορίθμου πάνω σε 25 κόμβους. Βλέπουμε τέσσερα clusterheads σε μικρή απόσταση μεταξύ τους, τα οποία είναι οι κόμβοι 65, 73, 85, και 100

Παράδειγμα (χειρότερη επίδοση)



Σενάριο χειρότερης περίπτωσης για τον αλγόριθμο, d=3

Πολυπλοκότητα

- Αφού κανείς κόμβος δεν βρίσκεται σε απόσταση μεγαλύτερη από d hops από το clusterhead του, η διαδικασία convergecast θα απαιτήσει O(d) γύρους μηνυμάτων
  - Επομένως, η time complexity του αλγορίθμου είναι O(2d + d) γύροι = O(d) γύροι
- Κάθε κόμβος πρέπει να διατηρεί 2d node ids στη δομή WINNER, και τον ίδιο αριθμό από node ids στη δομή SENDER
  - Επομένως, η storage complexity είναι O(d)