Κινητός και Διάχυτος Υπολογισμός (Mobile & Pervasive Computing)

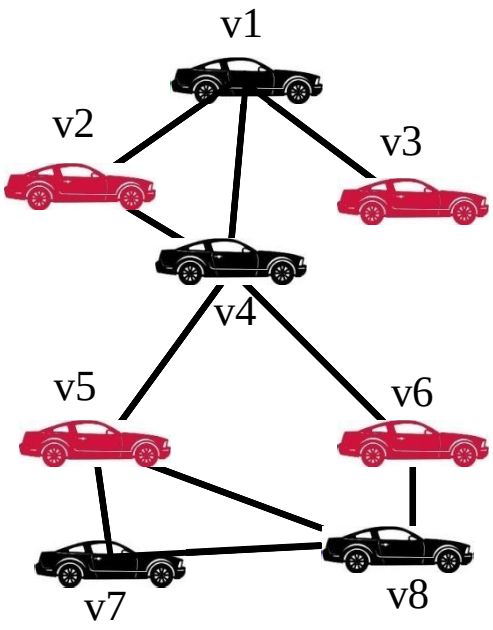
Δημήτριος Κατσαρός

Διάλεξη 14η

Περιεχόμενα

- **Clustering ad hoc δικτύων**
  - **Distributed Clustering Algorithm (DCS)**

Η έννοια του MIS



- Έστω μη κατευθυνόμενο γράφημα G(V; E)
- Το υποσύνολο S ⊆ V είναι ένα independent set εάν (είναι dominating set, και) 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

Clustering ad hoc δίκτυα

- Επιθυμητό να δημιουργήσουμε μια αφηρημένη δομή πάνω από το δίκτυο, έτσι ώστε τοπικές αλλαγές να μην χρειάζεται να γίνουν γνωστές σε όλο το δίκτυο
- Με χρήση υποδομών που λέγονται **clusters**
- **Clustering**: η διαδικασία ορισμού αυτών των υποδομών μέσα σε όλη την τοπολογία του δικτύου
- Οι κόμβοι διακρίνονται σε:
  - **Clusterheads**
  - **Gateways**
  - **Ordinal nodes**

Απλοί αλγόριθμοι clustering

- Οι παλιότεροι αλγόριθμοι επέλεξαν τα clusterhead με βάση:
  - ID
  - το node degree
- Γενίκευση αυτών με ανάθεση σε κάθε κόμβο ενός βάρους (weight)

Distributed Clustering Algorithm - DCA (1/8)

- Κάθε κόμβος έχει ένα ID και ένα weight≥0
- Δεν υπάρχουν δυο όμοια βάρη στο δίκτυο
- Στόχοι clustering
  - Κάθε ordinal κόμβος έχει ως γείτονα τουλάχιστον ένα clusterhead
  - Κάθε ordinal κόμβος συσχετίζεται με τον γειτονικό clusterhead που έχει το μεγαλύτερο weight
  - Δυο clusterhead δεν μπορεί να γειτνιάζουν
- Η τοπολογία δεν αλλάζει όσο εκτελείται ο αλγόριθμος

DCA (2/8)

- Ο αλγόριθμος εκτελείται γνωρίζοντας:
  - Το ID και το weight του κόμβου
  - Τα weights των γειτόνων
- (Μόνο) Δυο μηνύματα:
  - **CH(v)**: Αποστέλλεται από ένα clusterhead v
  - **JOIN(u,t)**: Αποστέλλεται από ένα ordinary κόμβο u όταν "μπαίνει" στο cluster του clusterhead t
- Τρεις (απλές) διαδικασίες:
  - **Init(start up)**
  - **OnReceivingCH(v)**
  - **OnReceivingJOIN(u,v)**

DCA (3/8)

- ***Κάθε κόμβος αποφασίζει ποιο ρόλο θα έχει τελικά, μόνο όταν όλοι οι κόμβοι στη γειτονιά του με μεγαλύτερο weight έχουν αποφασίσει ποιος θα είναι ο δικός τους ρόλος***

DCA (4/8)

- ***Init***
  - Κάθε κόμβος ξεκινά την εκτέλεση του αλγορίθμου την ίδια στιγμή
  - Μόνο οι κόμβοι με το μεγαλύτερο weight στη γειτονιά τους θα αποστείλουν ένα μήνυμα **CH**
  - Επειδή τα weights είναι μοναδικά, σίγουρα υπάρχει ένας τουλάχιστον κόμβος που θ' αποστείλει ένα μήνυμα **CH**
  - Όλοι οι υπόλοιποι κόμβοι αναμένουν να λάβουν ένα τέτοιο μήνυμα
  - Τότε έχουν δυο επιλογές:

DCA (5/8)

- ***On receiving CH***
  - Λαμβάνοντας ένα μήνυμα **CH** από ένα γείτονα u, ο κόμβος v ελέγχει εάν έχει λάβει μηνύματα από όλους τους γείτονες z (τέτοιους ώστε $w_z > w_u$) κάποιο μήνυμα **JOIN(z,x)**
  - Στην περίπτωση αυτή, ο v δεν θα λάβει μήνυμα **CH** από αυτούς τους z, και συνεπώς ο u είναι ο κόμβος με το μεγαλύτερο weight στη γειτονιά του κόμβου v, ο οποίος έχει στείλει μήνυμα CH
  - Επομένως, ο v προσκολλάται στον κόμβο u, δηλαδή εκπέμπει **JOIN(v,u)** και τερματίζει την εκτέλεση του αλγορίθμου
  - Εάν υπάρχει κάποιος κόμβος z ($w_z > w_u$) που δεν έχει στείλει ακόμα κάποιο μήνυμα, ο κόμβος v καταγράφει τον u (ως υποψήφιο δικό του clusterhead), και περιμένει το μήνυμα του z

DCA (6/8)

- ***On receiving JOIN* (1/3)**
  - Λαμβάνοντας ένα μήνυμα **JOIN(u,t)**, ο κόμβος v "ελέγχει" εάν ο ίδιος έχει στείλει προηγουμένως ένα μήνυμα CH, δηλ., **CH(v)**
    - Εάν αυτό ισχύει, τότε ελέγχει εάν ο κόμβος u θέλει να ενσωματωθεί στο δικό του cluster, δηλ., t = v
    - Κατόπιν, εάν όλοι οι γείτονες z του v με $w_z < w_v$ έχουν εκπέμψει την επιθυμία τους σε ποιο cluster θέλουν να ενταχθούν, ο κόμβος v εγκαταλείπει την εκτέλεση του αλγορίθμου (έχει μάθει τα cluster-members του)
      - Στην περίπτωση αυτή ο κόμβος v δεν ενδιαφέρεται για τους γειτονικούς του κόμβυς y (με $w_y > w_v$), εάν υπάρχουν κάποιοι, διότι αυτοί οι κόμβοι σίγουρα έχουν ενσωματωθεί στο cluster κάποιου κόμβου x με $w_x > w_v$

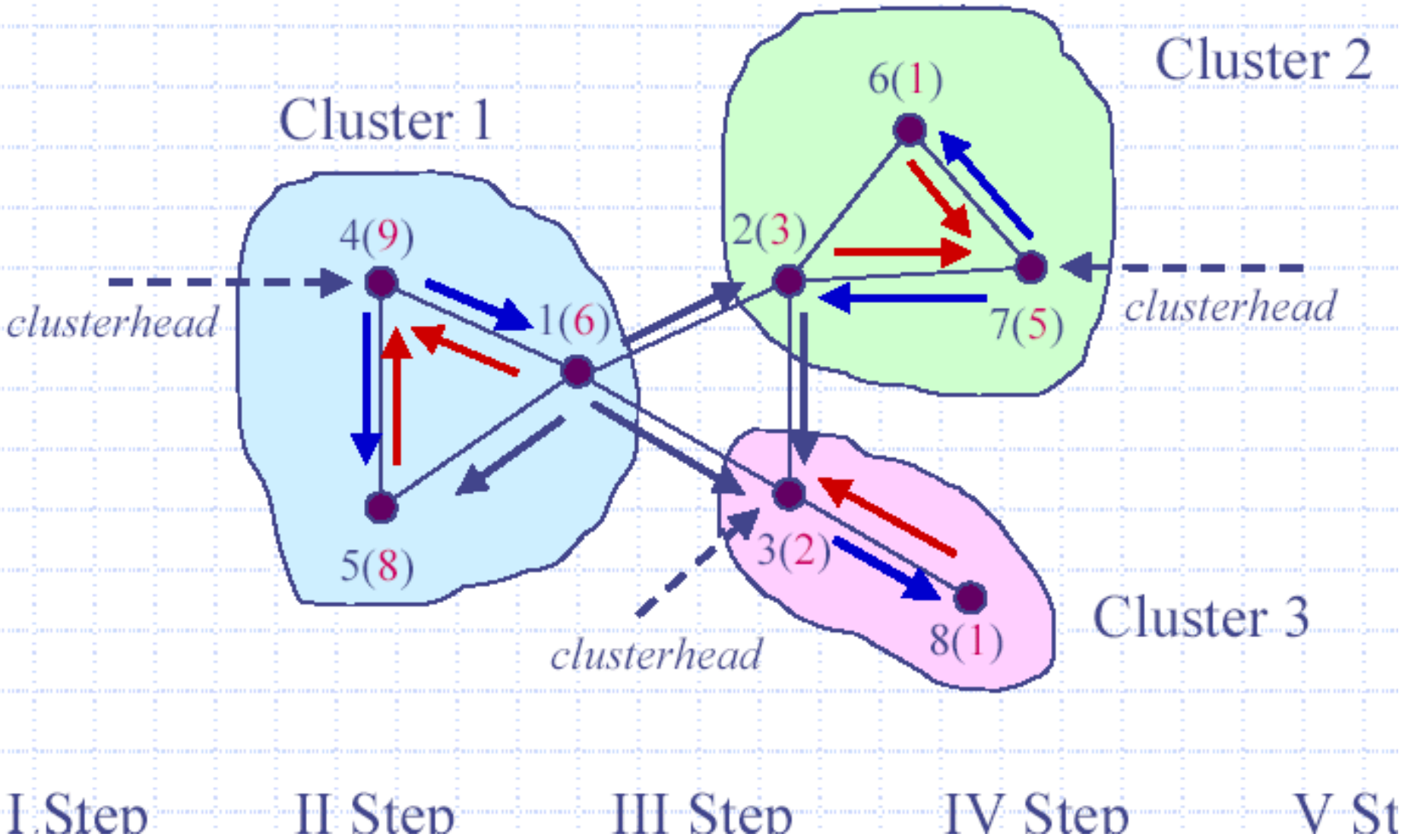
DCA (7/8)

- ***On receiving JOIN* (2/3)**
  - Εάν ο κόμβος v ΔΕΝ έχει στείλει μήνυμα CH, πριν αποφασίσει ποιος θα είναι ο ρόλος του, πρέπει να γνωρίζει τι αποφάσισαν για τον εαυτό τους οι κόμβοι z με $w_z > w_v$
  - Εάν ο v έχει λάβει μήνυμα από όλους αυτούς τους κόμβους, τότε ελέγχει τη φύση αυτών των μηνυμάτων
  - Εάν όλα αυτά τα μηνύματα είναι μηνύματα **JOIN**, αυτό σημαίνει ότι οι κόμβοι z ενσωματωθήκαν σε κάποιο cluster ως ordinal κόμβοι
    - Άρα, ο v είναι ο κόμβος με το μεγαλύτερο weight στη γειτονιά του μεταξύ εκείνων που δεν έχουν αποφασίσει ακόμα ποιος θα είναι ο ρόλος τους

DCA (8/8)

- ***On receiving JOIN* (3/3)**
  - Στην περίπτωση αυτή, ο κόμβος v θα είναι clusterhead
  - Στο σημείο αυτό, ο κόμβος v ελέγχει επίσης εάν κάθε γείτονας y (με $w_y < w_v$) έχει ήδη ενσωματωθεί σε κάποιο άλλο cluster
  - Εάν αυτό συμβαίνει, ο κόμβος v τερματίζει την εκτέλεση του αλγορίθμου
    - Θα είναι CH με ένα μόνο κόμβο, δηλ., τον εαυτό του
  - Εναλλακτικά, εάν ο v έχει λάβει τουλάχιστον ένα μήνυμα CH από κάποιον z, τότε ενσωματώνεται στο cluster του κόμβου-γείτονα με το μεγαλύτερο weight και τερμάτίζει την εκτέλεση του αλγορίθμου

Παράδειγμα DCA



Πολυπλοκότητα του DCA

Είδαμε ότι κάθε κόμβος περιμένει πρώτα την απόφαση των γειτόνων του με το μεγαλύτερο βάρος

Αυτός ο "χρόνος αναμονής" μπορεί να οριστεί ως συνάρτηση της *απόστασης* του κόμβου από έναν από τους init κόμβους

Αυτή η "blocking distance" D_b εξαρτάται από την τοπολογία του δικτύου και όχι από τον αριθμό των κόμβων

- **Πρόταση**: Κάθε κόμβος v στο V στέλνει ακριβώς ένα μήνυμα μέσα σε D_b+1 βήματα
- **Πόρισμα 1**: Ο DCA τερματίζεται ορθά σε D_b+1 βήματα το πολύ
- **Πόρισμα 2**: Η πολυπλοκότητα μηνυμάτων του DCA είναι **n =|V|**