

Κατανεμημένα Συστήματα I

Μάθημα Βασικής Επιλογής,

Χειμερινού Εξαμήνου

Τομέας Εφαρμογών και Θεμελιώσεων

Ιωάννης Χατζηγιαννάκης

Δευτέρα, 19 Οκτωβρίου, 2009
Αίθουσα Β3

Προηγούμενο Μάθημα

- ▶ Σύγχρονα Κατανεμημένα Συστήματα
- ▶ Μοντελοποίηση Συστήματος
- ▶ Πρόβλημα Εκλογής Αρχηγού
- ▶ Μελέτη ενός κατανεμημένου αλγόριθμου για Δίκτυα Δακτύλιου
- ▶ Μελέτη ενός κατανεμημένου αλγόριθμου για Γενικά Δίκτυα

Μοντέλο Σύγχρονου Δικτύου

- ▶ Μία συλλογή υπολογιστικών μονάδων ή 'επεξεργαστές'
 - ▶ κάθε επεξεργαστής εκτελεί μόνο μία διεργασία
- ▶ Οι μονάδες του συστήματος είναι συνδεδεμένες με ένα σύγχρονο δίκτυο
 - ▶ Ορίζουμε το σύγχρονο δίκτυο ως ένα **κατευθυνόμενο γράφημα** $G = (V, E)$
 - ▶ αποτελείται από $n = |V|$ **κορυφές** και $m = |E|$ **ακμές**
- ▶ Υποθέτουμε ότι κάθε κανάλι επικοινωνίας μπορεί να δεχτεί μόνο ένα μήνυμα τη φορά
 - ▶ τα κανάλια είναι οι ακμές του γραφήματος
- ▶ Θεωρούμε ότι υπάρχει ένα δεδομένο αλφάβητο M μηνυμάτων

Οι Καταστάσεις των Διεργασιών

- ▶ Κάθε διεργασία $u \in V$ χαρακτηρίζεται από ένα σύνολο καταστάσεων $states_u$
 - ▶ Ορισμένες τις ονομάζουμε **αρχικές καταστάσεις** $start_u$
 - ▶ Ορισμένες τις ονομάζουμε **καταστάσεις τερματισμού** $halt_u$
- ▶ Διαθέτει μια γεννήτρια εξερχόμενων μηνυμάτων $msgs_u : states_u \times nbrs_u^{out} \rightarrow M \cup \{null\}$
 - ▶ δεδομένης της τρέχουσας κατάστασης
 - ▶ δημιουργεί κάποια μηνύματα για τις γειτονικές διεργασίες
- ▶ Διαθέτει μία συνάρτηση αλλαγής κατάστασης $trans_u : states_u \times (M \cup \{null\})^{nbrs_u^i} \rightarrow states_u$
 - ▶ δεδομένης της τρέχουσας κατάστασης
 - ▶ τα μηνύματα που παραλήφθηκαν
 - ▶ υπολογίζει την επόμενη κατάσταση της διεργασίας

Έναρξη εκτέλεσης, Βήματα και Γύρος

- ▶ Αρχικά
 - ▶ όλες οι διεργασίες βρίσκονται σε κάποια αρχική κατάσταση
 - ▶ όλα τα κανάλια είναι άδεια
- ▶ Όλες οι διεργασίες, επαναλαμβάνουν 'συντονισμένα' τα ακόλουθα δύο βήματα:

1^ο Βήμα

1. Εφαρμογή της γεννήτριας μηνυμάτων
2. Παραγωγή μηνυμάτων για τους εξερχόμενους γείτονες
3. Αποστολή μηνυμάτων μέσω των αντίστοιχων καναλιών

2^ο Βήμα

1. Εφαρμογή της συνάρτησης αλλαγής κατάσταση
2. Διαγραφή όλων των μηνυμάτων από τα κανάλια.

- ▶ Ο συνδυασμός των δύο βημάτων ονομάζεται **γύρος**



Μέτρηση πολυπλοκότητας

- ▶ Σχεδιασμός Συστήματος
 - ▶ Ορισμός Ελάχιστων Απαιτήσεων
 - ▶ Επιλογή κατάλληλου κατανεμημένου αλγόριθμου
 - ▶ Πως μπορούμε να μετρήσουμε την απόδοση;

Χρονική πολυπλοκότητα

Το πλήθος των γύρων που απαιτούνται για να παραχθούν όλες οι ζητούμενες έξοδοι, ή μέχρι να τερματιστούν όλες οι διεργασίες (δηλ. να βρεθούν σε μια τερματική κατάσταση).

Πολυπλοκότητα επικοινωνίας

Ο συνολικός αριθμός μη μηδενικών μηνυμάτων (δηλ. δεν προσαμετρώνται τα null μηνύματα) που αποστέλλονται.



Πρόβλημα Εκλογής Αρχηγού

- ▶ Ορισμός Προβλήματος
- ▶ Αδυναμία αντιμετώπισης του προβλήματος όταν οι διεργασίες δεν έχουν μοναδικές ταυτότητες
- ▶ Δίκτυα Δακτυλίου

1. Αλγόριθμος των LeLann, Chang και Roberts
 - ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(n)$
 - ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(n^2)$
2. Αλγόριθμος των Hirschberg και Sinclair
 - ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(n)$
 - ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(n \log n)$

- ▶ Γενικά Δίκτυα

- ▶ Αλγόριθμος FloodMax
 - ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(\text{diam}(\mathcal{G}))$
 - ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(\text{diam}(\mathcal{G}) \cdot m)$



Ο Αλγόριθμος TimeSlice

Αλγόριθμος TimeSlice

Οι διεργασίες διατηρούν μια μεταβλητή **αρχηγός** η οποία αρχικά είναι **false**, και ένα **μετρητή γύρων** i , αρχικά 0. Κάθε διεργασία u λειτουργεί σε φάσεις $(0, 1, 2, \dots)$, όπου κάθε φάση διαρκεί n γύρους. Κάθε φάση v (που αποτελείται από τους γύρους $(v-1)n+1, \dots, vn$ αποσκοπεί στην περιφορά του μηνύματος που περιέχει την ταυτότητα της διεργασίας v .

Αν η διεργασία i υπάρχει, και μέχρι τον γύρο $(i-1)n+1$ δεν έχει λάβει κάποιο μήνυμα, θέτει την μεταβλητή **αρχηγός** σε **true** και στέλνει ένα μήνυμα με την ταυτότητα της στον δεξιόστροφο γείτονα της. Κάθε διεργασία που λαμβάνει το μήνυμα, στέλνει το μήνυμα στον δεξιόστροφο γείτονα της και τερματίζει.

- ▶ Βασίζεται στην γνώση του πλήθους των διεργασιών n



- ▶ Ο αλγόριθμος εκλέγει την διεργασία με την μικρότερη ταυτότητα l_{min}
- ▶ Καμία διεργασία εκτός της l_{min} δεν είναι σε κατάσταση 'εκλεγμένη'
- ▶ Η διεργασία l_{min} εκλέγεται αρχηγός στο γύρο $(l_{min} - 1)n + 1$
- ▶ Μέχρι τον γύρο $(l_{min} - 1)n + 1$ και μετά τον γύρο $l_{min}n$ κανένα μήνυμα δεν ανταλλάσσει
- ▶ Ο συνολικός αριθμός μηνυμάτων είναι n
- ▶ Η πολυπλοκότητα επικοινωνίας είναι $O(n)$
- ▶ Η χρονική πολυπλοκότητα είναι $n l_{min}$ - δεν φράσσεται ακόμα και σε δακτυλίους σταθερού μεγέθους.



- ▶ Επανεξετάζουμε την περίπτωση όπου οι διεργασίες δεν έχουν ταυτότητες
- ▶ Διακρίνουμε την ύπαρξη μιας συμμετρίας στο δίκτυο
- ▶ Αναφέρεται επίσης ως το **πρόβλημα διάσπασης της συμμετρίας**
- ▶ Ο αλγόριθμος των Itai & Rodeh λύνει το πρόβλημα
 - ▶ Βασίζεται στον αλγόριθμο LCR
 - ▶ **Βασική διαφορά:** οι διεργασίες δεν έχουν μοναδική ταυτότητα
- ▶ Κάθε διεργασία διαλέγει τυχαία μια ταυτότητα από το σύνολο $\{1, \dots, n\}$
- ▶ Ψάχνουμε να βρούμε την διεργασία με την μεγαλύτερη ταυτότητα
 - ▶ Αν διαπιστώσουμε ότι η μεγαλύτερη ταυτότητα δεν είναι μοναδική επαναλαμβάνουμε την διαδικασία



- ▶ Καμία διεργασία δεν μπορεί να αποφασίσει μοναμερώς να σταματήσει
- ▶ Πρέπει πρώτα να σιγουρευτεί ότι υπάρχει τουλάχιστον ένας ακόμα που προσπαθεί να εκλεγεί
- ▶ Σε αντίθετη περίπτωση
 - ▶ Έστω κατάσταση όπου καμία διεργασία δεν ενδιαφέρεται να εκλεγεί αρχηγός
 - ▶ Ο αλγόριθμος πέφτει σε αδιέξοδο
- ▶ Αντιμετωπίζουμε το πρόβλημα ως εξής:
 - ▶ Σε κάθε φάση απενεργοποιούνται όσες διεργασίες **καταλαβαίνουν** ότι δεν μπορούν να εκλεγούν
 - ▶ Έχουν δηλαδή μάθει ότι κάποια άλλη διεργασία είναι ισχυρότερη υποψήφια
 - ▶ Δεν είναι δυνατόν να παραιτηθούν όλες οι διεργασίες μαζί σε μία φάση



Αλγόριθμος IR

Οι διεργασίες διατηρούν μια μεταβλητή **αρχηγός**=false και μια μεταβλητή **φάση**=0. Σε κάθε φάση οι διεργασίες διαλέγουν μια ταυτότητα από το σύνολο $\{1, \dots, n\}$ και την στέλνουν στον δεξιόστροφο γείτονα τους ως εξής: (**φάση,ταυτότητα,μετρίτης,μοναδική**). Ο **μετρίτης**=0 και **μοναδική**=true. Μόλις λάβουν μία ταυτότητα από τον αριστερόστροφο γείτονα, την συγκρίνουν με την δικιά τους. Αν είναι μεγαλύτερη, την προωθούν στον δεξιόστροφο γείτονα και αυξάνουν τον μετρίτη. Αν είναι μικρότερη, δεν κάνουν τίποτα. Αν είναι ίδια, ελέγχουν τον **μετρίτη**: αν είναι μικρότερος από n τότε θέτει την λογική μεταβλητή **μοναδική**=false και προωθούν το μήνυμα. Αν **μετρίτης** $\geq n$ και **μοναδική**=false τότε διαλέγουν μια νέα ταυτότητα, θέτουν **φάση++** και επαναλαμβάνουν τον αλγόριθμο. Αλλιώς μεταβαίνουν στην κατάσταση **εκλεγμένη** θέτοντας την μεταβλητή **αρχηγός** στην τιμή true.



Χαρακτηριστικά του Αλγόριθμου IR

- ▶ Ο αλγόριθμος εξελίσσεται σε φάσεις
 - ▶ Κάθε φάση διαρκεί $\mathcal{O}(n)$ γύρους
 - ▶ Οι διεργασίες ανταλλάσσουν $\mathcal{O}(n \log n)$ μηνύματα
 - ▶ Μπορούμε να δείξουμε ότι ο αριθμός των απαιτούμενων φάσεων για την εκλογή αρχηγού είναι $\frac{en}{n-1}$
 - ▶ Δηλαδή σταθερός αριθμός φάσεων $\mathcal{O}(1)$
- ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(n)$
- ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(n \log n)$



Πολυπλοκότητα Επικοινωνίας του Αλγόριθμου IR (1)

- ▶ Ας εξετάσουμε μια φάση του αλγόριθμου
- ▶ Έστω $p(i, d)$ η πιθανότητα το μήνυμα της διεργασίας που διάλεξε ταυτότητα i να ταξιδέψει απόσταση d

$$p(i, d) = \begin{cases} p_i^{d-1} & \text{if } d = n \\ p_i^{d-1}(1 - p_i) & \text{otherwise} \end{cases}$$

- ▶ Μας ενδιαφέρει να υπολογίσουμε την μέση απόσταση D που ταξιδεύει κάθε μήνυμα
 - ▶ Αυτό εξαρτάτε από την ταυτότητα i
 - ▶ Αν $i = n$ τότε $\bar{D} = n$
 - ▶ διαφορετικά αν $i < n$ τότε υπολογίζεται ως εξής



Πολυπλοκότητα Επικοινωνίας του Αλγόριθμου IR (2)

$$\begin{aligned} \bar{D}(i) &= \sum_{d=1}^n d \cdot p(i, d) \\ &= n \cdot p_i^{n-1} + (1 - p_i) \sum_{d=1}^{n-1} d \cdot p_i^{d-1} \\ &= \frac{1 - np_i^{n-1} + (n-1)p_i^n}{1 - p_i} + np_i^{n-1} \\ &= \frac{1 - p_i^n}{1 - p_i}, \quad 1 \leq i \leq n \end{aligned}$$

- ▶ Κάθε διεργασία έχει πιθανότητα $\frac{1}{n}$ να διαλέξει σαν ταυτότητα τον αριθμό i
- ▶ Η διεργασία με ταυτότητα i έχει πιθανότητα $\frac{1}{n}$ να εκλεχθεί αρχηγός



Πολυπλοκότητα Επικοινωνίας του Αλγόριθμου IR (3)

- ▶ Οπότε ο μέσος αριθμός μηνυμάτων N σε μία φάση φράσσεται από:

$$\begin{aligned} N &\leq \sum_{i=1}^n \bar{D}(i) = n + \sum_{i=1}^{n-1} \frac{1 - (\frac{1}{n})^n}{1 - \frac{1}{n}} \\ &\leq n + \sum_{i=1}^{n-1} \frac{1}{1 - \frac{1}{n}} = n + n \sum_{i=1}^{n-1} \frac{1}{n-1} \\ &= n + n \cdot \mathcal{O}(\log n) \end{aligned}$$

- ▶ Άρα ο αριθμός μηνυμάτων σε μία φάση είναι $\mathcal{O}(n \log n)$



Προηγούμενο Μάθημα

Προηγούμενο Μάθημα
Σύγχρονα Καταμεμημένα Συστήματα
Εκλογή Αρχηγού

Καταμεμημένες Δομές

Αναζήτηση κατά Εύρος
Συντομότερα Μονοπάτια

Εύνοψη Μαθήματος

Εύνοψη Μαθήματος
Βιβλιογραφία
Επόμενο Μάθημα



Αναζήτηση κατά Εύρος

Σε ένα σύγχρονο δίκτυο G , η αναζήτηση κατά εύρος απαιτεί την κατασκευή ενός επικαλυπτικού δέντρου $T(G)$, με ρίζα την διεργασία u_0 όπου οι κορυφές που είναι σε απόσταση d από την u_0 στο G , βρίσκονται στο επίπεδο d στο δέντρο $T(G)$.

- ▶ Η κατασκευή αυτής της δομής είναι χρήσιμη σε πολλές περιπτώσεις
 - ▶ Γρήγορη μετάδοση πληροφορίας
 - ▶ Πρόβλημα Εκλογής Αρχηγού
 - ▶ Πρόβλημα Καταμέτρησης



Παράδειγμα (1)

Έλεγχος Χώρου Στάθμευσης Αυτοκινήτων

Για λόγους αυτόματου ελέγχου, η εταιρεία τοποθετεί ένα πλήθος από μονάδες εφοδιασμένες με αισθητήρες έτσι ώστε να καλύπτουν όλο τον χώρο στάθμευσης. Οι μονάδες συνδέονται μέσω ενός ασύρματου δικτύου (IEEE 802.11b). Το σύστημα υπολογίζει αυτόματα το σύνολο των σταθμευμένων αυτοκινήτων.

- ▶ Οι μονάδες έχουν ξεχωριστή διεύθυνση στο δίκτυο
- ▶ Οι μονάδες χρησιμοποιούν φωτογραφικές μηχανές
- ▶ Ο χειριστής του συστήματος χρησιμοποιεί έναν τερματικό σταθμό που είναι συνδεδεμένος στο ασύρματο δίκτυο



Παράδειγμα (2)

- ▶ Ο τερματικός σταθμός δημιουργεί ένα δέντρο αναζήτησης κατά εύρος
- ▶ Κάθε μονάδα μετράει το πλήθος των αυτοκινήτων που 'βλέπει' και το στέλνει στον γονέα της στο δέντρο
- ▶ Ο γονέας συγκεντρώνει τις τιμές που έλαβε από τα παιδιά της στο δέντρο, τις προσθέτει με την δικιά της και στέλνει το σύνολο στον δικό της γονέα
- ▶ Το άθροισμα που υπολογίζει η ρίζα του δέντρου είναι το πλήθος των σταθμευμένων αυτοκινήτων



Άλλες Εφαρμογές

Πρόβλημα Εκλογής Αρχηγού:

- ▶ Ο αλγόριθμος κατασκευάζει ένα δέντρο αναζήτησης κατά εύρος για κάθε διεργασία
- ▶ Η διεργασία με την 'μέγιστη ταυτότητα' εκλεγεται αρχηγός και όλες οι υπόλοιπες μεταβαίνουν στην κατάσταση 'μη-εκλεγμένη'
- ▶ Δεν χρειάζεται να γνωρίζουν το πλήθος n ή την διάμετρο του δικτύου $diam(G)$

Υπολογισμός Διαμέτρου Δικτύου:

- ▶ Ο αλγόριθμος κατασκευάζει ένα δέντρο αναζήτησης κατά εύρος για κάθε διεργασία
- ▶ Κάθε διεργασία υπολογίζει το 'ύψος' του δέντρου της
- ▶ Κάθε διεργασία χρησιμοποιεί το δέντρο της για να ανακαλύψει το μέγιστο ύψος, δηλ. την διάμετρο του δικτύου



Ο Αλγόριθμος SynchBFS

Αλγόριθμος SynchBFS

Οι διεργασίες διατηρούν μια μεταβλητή **μαρκαρισμένη** η οποία αρχικά είναι `false` και μια μεταβλητή **γονέας** με αρχική τιμή 0. Αρχικά, η διεργασία u θέτει την μεταβλητή **μαρκαρισμένη** ως `true`, την μεταβλητή **γονέας** με την ταυτότητα της, και στέλνει ένα μήνυμα 'αναζήτησης' σε όλους τους γείτονες της. Σε κάθε γύρο, εάν μια διεργασία λάβει ένα μήνυμα 'αναζήτησης' και η τιμή της μεταβλητής **μαρκαρισμένη** είναι `false`, τότε θέτει την μεταβλητή σε `true`, θέτει την μεταβλητή **γονέας** με την ταυτότητα της διεργασίας από όπου έλαβε το μήνυμα, και στον επόμενο γύρο στέλνει ένα μήνυμα 'αναζήτησης' σε όλους τους γείτονες της.

- ▶ Δεν γνωρίζουν το πλήθος των διεργασιών (n)
- ▶ Έχουν μοναδικές ταυτότητες



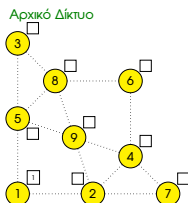
Παράδειγμα Εκτέλεσης Αλγόριθμου SynchBFS

Αρχικό Δίκτυο

Το δίκτυο έχει 9 μονάδες, 14 κανάλια
Η διεργασία 1 ξεκινά την εκτέλεση του αλγορίθμου.

Η διεργασία 1 θεωρείται μαρκαρισμένη.

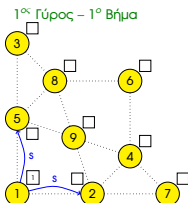
Όλες οι άλλες διεργασίες δεν είναι μαρκαρισμένες.



Παράδειγμα Εκτέλεσης Αλγόριθμου SynchBFS

1^{ος} Γύρος - 1^ο Βήμα

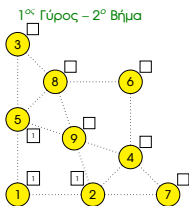
Η διεργασία 1 στέλνει μήνυμα 'αναζήτησης' σε όλους του γείτονες της.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS

1^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα "αναζήτησης" σε όλους του γείτονες της.



1^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες 2, 5 μαρκάρονται.

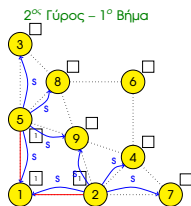
Οι διεργασίες 2, 5 θέτουν την 1 ως γονέα στο δέντρο.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS

2^{ος} Γύρος – 1^ο Βήμα

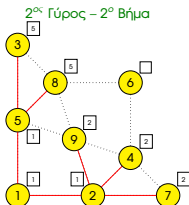
Οι διεργασίες 2, 5 στέλνουν μήνυμα "αναζήτησης" σε όλους τους γείτονες τους.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS

2^{ος} Γύρος – 1^ο Βήμα

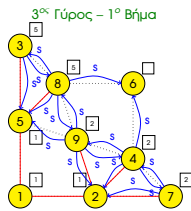
Οι διεργασίες 2, 5 στέλνουν μήνυμα "αναζήτησης" σε όλους τους γείτονες τους.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS

3^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες 3, 4, 7, 8, 9 στέλνουν μήνυμα "αναζήτησης" σε όλους τους γείτονες τους.



Χαρακτηριστικά του Αλγόριθμου SynchBFS

- ▶ Ο αλγόριθμος SynchBFS κατασκευάζει ένα δέντρο αναζήτησης κατά εύρος.
- ▶ Η δομή του δέντρου δεν είναι αποθηκευμένη σε κάποια 'κεντρική' διεργασία
- ▶ Η χρονική πολυπλοκότητα είναι $O(\text{diam}(G))$
 - ▶ Στην πραγματικότητα είναι η μέγιστη απόσταση από την u_0
 - ▶ Στο παράδειγμα, η διάμετρος είναι 4 - η μέγιστη απόσταση από την u_0 είναι 3
- ▶ Η πολυπλοκότητα επικοινωνίας είναι $O(m)$



Βελτίωση Πολυπλοκότητας Μηνυμάτων

Μπορούμε να μειώσουμε τον αριθμό των μηνυμάτων που χρησιμοποιεί ο αλγόριθμος ως εξής

- ▶ Οι διεργασίες μπορούν να αναγνωρίσουν το κανάλι από το οποίο έλαβαν ένα μήνυμα
- ▶ Οι διεργασίες δεν στέλνουν μηνύματα 'αναζήτησης' στα κανάλια από τα οποία έλαβαν ένα μήνυμα 'αναζήτησης'

Στο παράδειγμα εκτέλεσης του SynchBFS τα συνολικά μηνύματα μειώνονται κατά 10 - δηλ. 18



Μετάδοση Μηνύματος

Μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο για την μετάδοση ενός μηνύματος σε όλο το δίκτυο

- ▶ Η διεργασία u_0 θέλει να στείλει το μήνυμα M σε όλες τις διεργασίες του δικτύου
- ▶ Η διεργασία u_0 ξεκινάει την κατασκευή του δέντρου στέλνοντας το μήνυμα 'αναζήτησης' το οποίο περιέχει και το μήνυμα M
- ▶ Οι άλλες διεργασίες επισυνάπτουν και αυτές με την σειρά τους το μήνυμα M με τα μηνύματα 'αναζήτησης' που στέλνουν
- ▶ Εφόσον το δέντρο περιέχει όλες τις διεργασίες, το μήνυμα M τελικά παραλαμβάνετε από όλες τις διεργασίες του δικτύου



Πλήρη Γνώση (1)

Είναι απαραίτητο κάθε διεργασία να γνωρίζει και τα 'παιδιά' της
Τροποποιούμε τον αλγόριθμο SynchBFS ως εξής:

- ▶ Κάθε διεργασία που λαμβάνει ένα μήνυμα 'αναζήτησης' επιστρέφει στον αποστολέα ένα μήνυμα 'γονέας' ή 'μη-γονέας' ανάλογα του αν ο αποστολέας είναι ο γονέας της διεργασίας

Κατά την ολοκλήρωση της εκτέλεσης του SynchBFS', όλες οι διεργασίες γνωρίζουν τα 'παιδιά' τους
Ο αλγόριθμος SynchBFS' απαιτεί το πολύ $\text{diam}(G) + 2$ γύρους και χρησιμοποιεί $2 \cdot m$ μηνύματα

Η χρονική πολυπλοκότητα και η πολυπλοκότητα μηνυμάτων παραμένουν σταθερές



Πλήρη Γνώση (2)

Μπορούμε να μειώσουμε τον αριθμό των μηνυμάτων που χρησιμοποιεί ο αλγόριθμος

Τροποποιούμε τον αλγόριθμο SynchronBFS' ως εξής:

- ▶ Κάθε διεργασία που λαμβάνει μήνυμα 'αναζήτησης' απαντά στον αποστολέα με ένα μήνυμα 'γονέας' εφόσον ο αποστολέας είναι ο γονέας της διεργασίας
- ▶ Εάν η διεργασία που έστειλε το μήνυμα 'αναζήτησης' σε μία γειτονική διεργασία, δεν λάβει μήνυμα 'γονέας' στον επόμενο γύρο, θεωρεί ότι δεν επιλέχθηκε ως γονέας της γειτονικής διεργασίας

Ο αλγόριθμος SynchronBFS' χρησιμοποιεί $m + n - 1$ μηνύματα



Τερματισμός (1)

Πως μπορεί η διεργασία u_i να μάθει πότε ολοκληρώθηκε η κατασκευή του δέντρου;

- ▶ Δεν είναι γνωστή η διάμετρος του δικτύου και το πλήθος των διεργασιών n

Βασίζομαστε την παραλλαγή SynchronBFS' όπου οι διεργασίες απαντάνε στον αποστολέα ενός μηνύματος 'αναζήτησης' με ένα μήνυμα 'γονέας' ή 'μη-γονέας' ανάλογα του αν ο αποστολέας είναι ο γονέας της διεργασίας.

Εφόσον οι διεργασίες μπορούν να αναγνωρίσουν το κανάλι από το οποίο έλαβαν ένα μήνυμα, δεν στέλνουν μηνύματα 'αναζήτησης' στα κανάλια από τα οποία έλαβαν ένα μήνυμα 'αναζήτησης'.



Τερματισμός (2)

Επεκτείνουμε τον SynchronBFS' ως εξής:

- ▶ Κάθε διεργασία που λαμβάνει ένα μήνυμα 'αναζήτησης' επιστρέφει στον αποστολέα **στον επόμενο γύρο** ένα μήνυμα 'μη-γονέας' αν ο αποστολέας δεν είναι ο γονέας της διεργασίας
- ▶ Η διεργασία καθυστερεί την αποστολή του μηνύματος 'γονέας' έως ότου λάβει κάποιο μήνυμα 'γονέας' ή 'μη-γονέας' από όλες τις διεργασίες στις οποίες έστειλε μηνύματα 'αναζήτησης'



Τερματισμός (3)

Επομένως τα μηνύματα 'γονέας' ή 'μη-γονέας' έχουν δύο χρήσεις

1. **Παροχή πλήρους γνώσης** – κάθε διεργασία γνωρίζει και τα 'παιδιά' της
2. **Ενημέρωση τερματισμού** – κάθε διεργασία γνωρίζει πότε τα υποδέντρα των 'παιδιών' της έχουν κατασκευαστεί

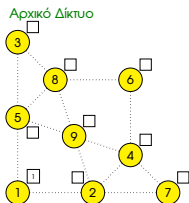
Ο αλγόριθμος SynchronBFS_T απαιτεί το πολύ $2 \cdot \text{diam}(G) + 2$ γύρους και χρησιμοποιεί $2 \cdot m$ μηνύματα



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

Αρχικό Δίκτυο

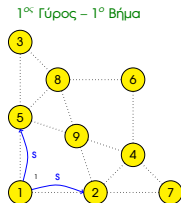
Το δίκτυο έχει 9 μονάδες, 14 κανάλια
 Η διεργασία 1 ξεκινά την εκτέλεση του αλγορίθμου.
 Η διεργασία 1 θεωρείται μαρκιαρισμένη.
 Όλες οι άλλες διεργασίες δεν είναι μαρκιαρισμένες.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

1^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα 'αναζήτησης' σε όλους του γείτονες της.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

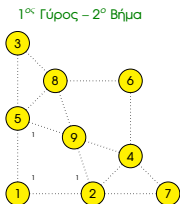
1^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα 'αναζήτησης' σε όλους του γείτονες της.

1^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες 2, 5 μαρκάρονται.

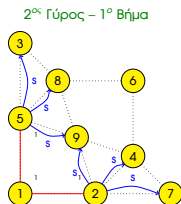
Οι διεργασίες 2, 5 θέτουν την 1 ως γονέα στο δέντρο.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

2^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες 2, 5 στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

2^{ος} Γύρος - 1^ο Βήμα

Οι διεργασίες **2, 5** στέλνουν μήνυμα "αναζήτησης" σε όλους τους γείτονες τους.

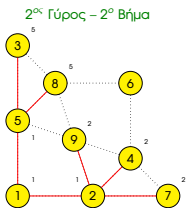
2^{ος} Γύρος - 2^ο Βήμα

Οι διεργασίες **3, 4, 7, 8, 9** μαρκάρονται.

Οι διεργασίες **3, 8** θέτουν την **5** ως γονέα στο δέντρο.

Οι διεργασίες **4, 7** θέτουν την **2** ως γονέα στο δέντρο.

Η διεργασία **9** διαλέγει τυχαία την **2** ως γονέα στο δέντρο.



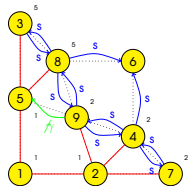
Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

3^{ος} Γύρος - 1^ο Βήμα

3^{ος} Γύρος - 1^ο Βήμα

Οι διεργασίες **3, 4, 7, 8, 9** στέλνουν μήνυμα "αναζήτησης" σε όλους τους γείτονες τους.

Η διεργασία **9** στέλνει μήνυμα "μη-γονέας" στην **5**



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

3^{ος} Γύρος - 1^ο Βήμα

Οι διεργασίες **3, 4, 7, 8, 9** στέλνουν μήνυμα "αναζήτησης" σε όλους τους γείτονες τους.

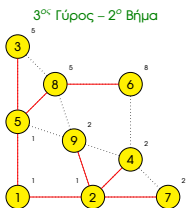
Η διεργασία **9** στέλνει μήνυμα "μη-γονέας" στην **5**

3^{ος} Γύρος - 2^ο Βήμα

Οι διεργασίες **2, 3, 4, 5, 7, 8, 9** αγνοούν τα μηνύματα "αναζήτησης" που έλαβαν.

Η διεργασία **6** μαρκάρεται.

Η διεργασία **6** διαλέγει τυχαία την **8** ως γονέα στο δέντρο.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

4^{ος} Γύρος - 1^ο Βήμα

Η διεργασία **3** στέλνει μήνυμα

"μη-γονέας" στην **8**

Η διεργασία **8** στέλνει μήνυμα "μη-γονέας" στην **3**

Η διεργασία **8** στέλνει μήνυμα "μη-γονέας" στην **9**

Η διεργασία **9** στέλνει μήνυμα "μη-γονέας" στην **4**

Η διεργασία **4** στέλνει μήνυμα "μη-γονέας" στην **7**

Η διεργασία **7** στέλνει μήνυμα "μη-γονέας" στην **4**

Η διεργασία **6** στέλνει μήνυμα "μη-γονέας" στην **4**

Η διεργασία **6** στέλνει μήνυμα "γονέας" στην **8**

Η διεργασία **6** στέλνει μήνυμα "γονέας" στην **8**

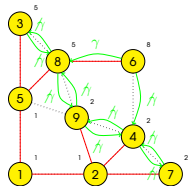
Η διεργασία **6** στέλνει μήνυμα "γονέας" στην **8**

Η διεργασία **6** στέλνει μήνυμα "γονέας" στην **8**

Η διεργασία **6** στέλνει μήνυμα "γονέας" στην **8**

Η διεργασία **6** στέλνει μήνυμα "γονέας" στην **8**

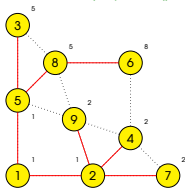
4^{ος} Γύρος - 1^ο Βήμα



6^{ος} Γύρος - 2^ο Βήμα

6^{ος} Γύρος - 1^ο Βήμα Οι διεργασίες 2, 5 στέλνουν μήνυμα "γονέας" στην 1
6^{ος} Γύρος - 2^ο Βήμα

Η διεργασία 1 διαπιστώνει ότι η κατασκευή των υποδέντρων των 2, 5 ολοκληρώθηκε
Η διεργασία 1 τερματίζει

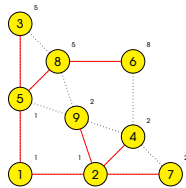


Τελικό Δίκτυο

Τελικό Δίκτυο

Το δέντρο αναζήτησης κατά εύρος έχει κατασκευαστεί.

Ο αλγόριθμος εκτελέστηκε σε 6 γύρους.
Συνολικά μεταδόθηκαν 36 μηνύματα.



Γενικά (1)

Συνοτότερα Μονοπάτια

Σε ένα σύγχρονο δίκτυο G , όπου οι ακμές έχουν βάρη, η εύρεση όλων των συνοτότερων μονοπατιών από μια διεργασία u_i προς όλες τις υπόλοιπες διεργασίες, απαιτεί την κατασκευή ενός επικαλυπτικού δέντρου $T(G)$, με ρίζα την διεργασία u_i όπου οι κορυφές που είναι σε απόσταση d από την u_i στο G , είναι σε απόσταση d στο δέντρο $T(G)$.

- ▶ Η κατασκευή αυτής της δομής είναι χρήσιμη σε πολλές περιπτώσεις
 - ▶ Ελαχιστοποιεί το μέγιστο κόστος μετάδοσης πληροφορίας
- ▶ Τα βάρη στις ακμές μπορεί να απεικονίζονται
 - ▶ Ταχύτητα μετάδοσης
 - ▶ Κόστος μετάδοσης



Γενικά (2)

- ▶ Υποθέτουμε ότι κάθε διεργασία γνωρίζει τα κόστη των γεινιάζουσων ακμών
 - ▶ Είναι μια εσωτερική μεταβλητή **βάρος**
- ▶ Κάθε διεργασία γνωρίζει το πλήθος των διεργασιών n
- ▶ Εάν όλα τα βάρη των ακμών είναι ίσα τότε ένα δέντρο αναζήτησης κατά εύρος είναι επίσης και ένα δέντρο συνοτότερων μονοπατιών
 - ▶ μια απλή τροποποίηση του SyncBFS_T μπορεί να υπολογίσει μέσω των μηνυμάτων "γονέας" την απόσταση από την διεργασία u_i



Ο Αλγόριθμος BellmanFord

Αλγόριθμος BellmanFord

Οι διεργασίες διατηρούν μια μεταβλητή **απόσταση** η οποία αρχικά είναι ∞ και μια μεταβλητή **γονέας** με αρχική τιμή 0. Σε κάθε γύρο, οι διεργασίες στέλνουν την τιμή της μεταβλητής **απόσταση** σε όλους τους γείτονες τους. Κάθε διεργασία u , επεξεργάζεται τα μηνύματα που έλαβε, και εντοπίζει το μήνυμα της v το οποίο έχει την ελάχιστη τιμή **απόσταση_v + βάρους_{uv}**. Εάν ένα τέτοιο μήνυμα βρεθεί, θέτει την μεταβλητή **απόσταση_u** στην τιμή **απόσταση_v + βάρους_{uv}** και την μεταβλητή **γονέας_u = v**. Μετά από $n - 1$ γύρους, η τιμή της μεταβλητή **απόσταση** είναι η ελάχιστη απόσταση της διεργασίας με την u_0 και η μεταβλητή **γονέας** δείχνει τον γονέα της διεργασίας στο δέντρο των συντομότερων μονοπατιών.

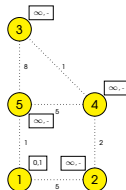


Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

Αρχικό Δίκτυο

Το δίκτυο έχει 5 μονάδες, 6 κανάλια
Η διεργασία 1 ξεκινά την εκτέλεση του αλγορίθμου.
Η διεργασία 1 είναι η ρίζα του δέντρου.

Αρχικό Δίκτυο



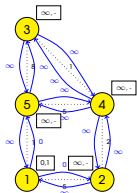
Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

1^{ος} Γύρος - 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Οι διεργασίες 2, 3, 4, 5 στέλνουν μήνυμα '∞' σε όλους τους γείτονες τους.

1^{ος} Γύρος - 1^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

1^{ος} Γύρος - 1^ο Βήμα

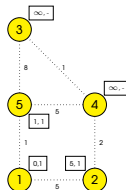
Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Οι διεργασίες 2, 3, 4, 5 στέλνουν μήνυμα '∞' σε όλους τους γείτονες τους.

1^{ος} Γύρος - 2^ο Βήμα

Οι διεργασίες 2, 5 θέτουν την 1 ως γονέα με απόσταση 5 και 1 αντίστοιχα.

1^{ος} Γύρος - 2^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

2^{ος} Γύρος – 1^ο Βήμα

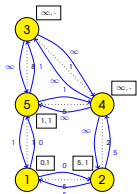
Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.

Οι διεργασίες 3, 4 στέλνουν μήνυμα '∞' σε όλους τους γείτονες τους.

2^{ος} Γύρος – 1^ο Βήμα



2^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

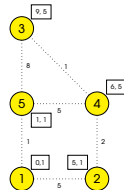
Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.

Οι διεργασίες 3, 4 στέλνουν μήνυμα '∞' σε όλους τους γείτονες τους.

2^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες 3, 4 θέτουν την 5 ως γονέα με απόσταση 9 και 6 αντίστοιχα.

2^{ος} Γύρος – 2^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

3^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

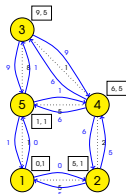
Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

Η διεργασία 3 στέλνει μήνυμα '9' σε όλους του γείτονες της.

Η διεργασία 4 στέλνει μήνυμα '6' σε όλους του γείτονες της.

Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.

3^{ος} Γύρος – 1^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

3^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

Η διεργασία 3 στέλνει μήνυμα '9' σε όλους του γείτονες της.

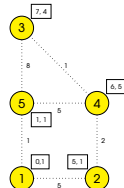
Η διεργασία 4 στέλνει μήνυμα '6' σε όλους του γείτονες της.

Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.

3^{ος} Γύρος – 2^ο Βήμα

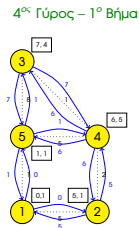
Η διεργασία 3 θέτει την 4 ως γονέα με απόσταση 7.

3^{ος} Γύρος – 2^ο Βήμα



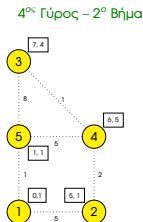
Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

4^{ος} Γύρος – 1^ο Βήμα Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.
Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.
Η διεργασία 3 στέλνει μήνυμα '7' σε όλους του γείτονες της.
Η διεργασία 4 στέλνει μήνυμα '6' σε όλους του γείτονες της.
Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

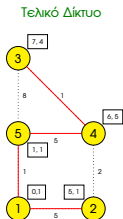
4^{ος} Γύρος – 1^ο Βήμα Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.
Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.
Η διεργασία 3 στέλνει μήνυμα '7' σε όλους του γείτονες της.
Η διεργασία 4 στέλνει μήνυμα '6' σε όλους του γείτονες της.
Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.
4^{ος} Γύρος – 2^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

Τελικό Δίκτυο

Το δέντρο ελάχιστων μονοπατιών έχει κατασκευαστεί.
Ο αλγόριθμος εκτελέστηκε σε 4 γύρους.
Συνολικά μεταδόθηκαν 48 μηνύματα.



Χαρακτηριστικά του Αλγόριθμου BellmanFord

- ▶ Ο αλγόριθμος BellmanFord κατασκευάζει ένα δέντρο ελάχιστων μονοπατιών.
 - ▶ Με επαγωγή στον αριθμό των γύρων
 - ▶ Μετά από γ γύρους, οι μεταβλητές **απόσταση** και **γονέας** για τη διεργασία v αντιστοιχούν στο συντομότερο μονοπάτι που την ενώνει με την u_γ μήκους το πολύ γ
 - ▶ Εάν δεν υπάρχει κάποιο μονοπάτι μήκους το πολύ γ η μεταβλητή **απόσταση** = ∞ και η **γονέας** = 0
 - ▶ Επομένως μετά απο $n - 1$ γύρους, όλες οι διεργασίες θα έχουν εντοπίσει το συντομότερο μονοπάτι που τις ενώνει με την u_γ
- ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(n)$
- ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(n \cdot m)$

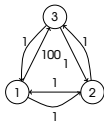
- ▶ Αναλογικά με τον αλγόριθμο SynchBFS η χρονική πολυπλοκότητα του BellmanFord περιμένουμε να είναι $O(mn)$

- ▶ Για το παράδειγμα:

- ▶ η ελάχιστη απόσταση από την διεργασία 1 στην 3 είναι 2
- ▶ Χρειάζονται 2 γύροι για να βρεθεί
- ▶ Η διάμετρος του δικτύου είναι 1

- ▶ Η χρονική πολυπλοκότητα είναι $O(n)$

Παράδειγμα



- ▶ Μπορούμε να εκτελέσουμε τον αλγόριθμο BellmanFord για β γύρους για να βρούμε όλα τα ελάχιστα μονοπάτια σε απόσταση β
- ▶ Η εύρεση του πλήθους των διεργασιών n μπορεί να γίνει σε συνδυασμό με τον αλγόριθμο SynchBFS

Σύνοψη 3^{ης} Διάλεξης

Προηγούμενο Μάθημα

Προηγούμενο Μάθημα
Σύγχρονα Καταμεμημένα Συστήματα
Εκλογή Αρχηγού

Καταμεμημένες Δομές

Αναζήτηση κατά Εύρος
Συντομότερα Μονοπάτια

Σύνοψη Μαθήματος

Σύνοψη Μαθήματος
Βιβλιογραφία
Επόμενο Μάθημα

Σύνοψη Μαθήματος

- ▶ Σύγχρονα Καταμεμημένα Συστήματα
- ▶ Πρόβλημα Εκλογής Αρχηγού
 - ▶ Δίκτυα Δακτυλίου
 - ▶ Αλγόριθμος TimeSlice
 - ▶ Αλγόριθμος των Itai και Rodeh
 - ▶ Γενικά Δίκτυα
 - ▶ Αλγόριθμος FloodMax
- ▶ Αναζήτηση κατά Εύρος
 - ▶ Αλγόριθμος SynchBFS
- ▶ Συντομότερα Μονοπάτια
 - ▶ Αλγόριθμος των Bellman και Ford

- ▶ Σημειώσεις του μαθήματος
- ▶ Βιβλίο "Distributed Algorithms" (N.Lynch)
 1. Κεφάλαιο 4: Algorithms in General Synchronous Networks
- ▶ Βιβλίο "Distributed Computing Fundamentals, Simulations, and Advanced Topics" (H.Attiya, J.Welch)
 1. Κεφάλαιο 2: Basic Algorithms in Message Passing Systems
- ▶ Βιβλίο "Introduction to Distributed Algorithms" (G.Tel)
 1. Κεφάλαιο 6: Wave and Traversal Algorithms
- ▶ Βιβλίο "Distributed Systems, Concepts and Design" (G.Coulouris, J.Dollimore, T.Kindberg)
 1. Κεφάλαιο 11: Coordination and Agreement

- ▶ Σύγχρονα Κατανεμημένα Συστήματα
- ▶ Συναίνεση Με Σφάλματα