

Κατανεμημένα Συστήματα I

Μάθημα Βασικής Επιλογής, Χειμερινού Εξαμήνου Τομέας Εφαρμογών και Θεμελιώσεων

Ιωάννης Χατζηγιαννάκης

Παρασκευή, 18 Δεκεμβρίου, 2009
Αίθουσα Β3



Προηγούμενο Μάθημα

- ▶ Ασύγχρονα Κατανεμημένα Συστήματα
- ▶ Καθολικές Καταστάσεις
- ▶ Κατασκευή Καθολικών Καταστάσεων
 - ▶ Παθητική Παρατήρηση με Φυσικά Ρολόγια
 - ▶ Παθητική Παρατήρηση με Λογικά Ρολόγια
- ▶ Συνεπή Ολικά Σημιότυπα
 - ▶ Συνεπή Ολικά Σημιότυπα με Φυσικά Ρολόγια
 - ▶ Συνεπή Ολικά Σημιότυπα με Λογικά Ρολόγια
 - ▶ Ο Αλγόριθμος των Chandy και Lamport



Μοντελοποίηση Ασύγχρονων Συστημάτων

Μελετάμε κατανεμημένα συστήματα, όπου

1. τα επιμέρους συστήματα εκτελούν τις εργασίες τους με **οποιαδήποτε, ακαθόριστη, ταχύτητα** σε σχέση με τα υπόλοιπα υποσυστήματα
2. τα κανάλια επικοινωνίας παραδίδουν τα μηνύματα με **οποιαδήποτε, ακαθόριστη, ταχύτητα** σε σχέση με τα υπόλοιπα κανάλια

Μοντελοποιούμε αυτή την **απροσδιόριστη χρονικά συμπεριφορά** χρησιμοποιώντας **αυτόματα εισόδου/εξόδου**

- ▶ Κάθε διεργασία μοντελοποιείτε από ένα αυτόματο εισόδου/εξόδου
- ▶ Κάθε κανάλι επικοινωνίας μοντελοποιείτε από ένα αυτόματο εισόδου/εξόδου



Ιδιότητες Συστήματος

Μέτρηση πολυπλοκότητας

- ▶ **Πολυπλοκότητα Επικοινωνίας** – συνολικός αριθμός μηνυμάτων που στάλθηκαν ή παραλήφθηκαν
- ▶ **Χρονική πολυπλοκότητα** – η χρονική απροσδιοριστία δεν μας επιτρέπει να μετρήσουμε με ευκολία
 1. Θέτουμε ένα άνω φράγμα l στον χρόνο εκτέλεσης κάθε ενέργειας e σε κάποια κατάσταση k :
 2. Θέτουμε ένα άνω φράγμα d στον χρόνο παράδοσης του παλαιότερου μηνύματος που βρίσκεται σε κάποιο κανάλι επικοινωνίας



- ▶ **Κατασκευή Επικαλυπτικού Δέντρου** – κάθε κόμβος u κατασκευάζει ένα επικαλυπτικό δέντρο $T_u(G)$, με ρίζα τον u
- ▶ **Αλγόριθμος** -- AsynchSpanningTree
 - ▶ Πολυπλοκότητα επικοινωνίας -- $\mathcal{O}(n \cdot m)$
 - ▶ Χρονική πολυπλοκότητα -- $\mathcal{O}(\delta(l + d))$
- ▶ **Αλγόριθμος** -- AsynchBFS
 - ▶ Πολυπλοκότητα επικοινωνίας -- $\mathcal{O}(n \cdot m)$
 - ▶ Χρονική πολυπλοκότητα -- $\mathcal{O}(n \cdot \delta(l + d))$



- ▶ **Κατασκευή Επικαλυπτικού Δέντρου** – κάθε κόμβος u κατασκευάζει ένα επικαλυπτικό δέντρο $T_u(G)$, με ρίζα τον u
- ▶ **Αλγόριθμος** -- AsynchSpanningTree
 - ▶ Πολυπλοκότητα επικοινωνίας -- $\mathcal{O}(n \cdot m)$
 - ▶ Χρονική πολυπλοκότητα -- $\mathcal{O}(\delta(l + d))$
- ▶ **Αλγόριθμος** -- AsynchBFS / SyncBFS
 - ▶ Πολυπλοκότητα επικοινωνίας -- $\mathcal{O}(n \cdot m)$ / $\mathcal{O}(n \cdot m)$
 - ▶ Χρονική πολυπλοκότητα -- $\mathcal{O}(n \cdot \delta(l + d))$ / $\mathcal{O}(\delta)$



- ▶ **Εύρεση Συντομότερων Μονοπατιών** – κάθε κόμβος u υπολογίζει την απόσταση με κάθε άλλο κόμβο v του δικτύου
- ▶ **Αλγόριθμος** -- AsynchBellmanFord
 - ▶ Πολυπλοκότητα επικοινωνίας -- $\mathcal{O}(n^2 \cdot m)$
 - ▶ Χρονική πολυπλοκότητα -- $\mathcal{O}(n^{n+1}(l + d))$



- ▶ **Εύρεση Συντομότερων Μονοπατιών** – κάθε κόμβος u υπολογίζει την απόσταση με κάθε άλλο κόμβο v του δικτύου
- ▶ **Αλγόριθμος** -- AsynchBellmanFord / BellmanFord
 - ▶ Πολυπλοκότητα επικοινωνίας -- $\mathcal{O}(n^2 \cdot m)$ / $\mathcal{O}(n^2 \cdot m)$
 - ▶ Χρονική πολυπλοκότητα -- $\mathcal{O}(n^{n+1}(l + d))$ / $\mathcal{O}(n)$



- ▶ Παρατηρούμε ότι ορισμένοι αλγόριθμοι που σχεδιάστηκαν για τα σύγχρονα συστήματα **επιτυγχάνουν καλύτερη απόδοση** ως προς την χρονική πολυπλοκότητα αλλά και την πολυπλοκότητα επικοινωνίας
 - ▶ Μπορούμε να τους προσαρμόσουμε στα ασύγχρονα συστήματα;
- ▶ Η παρουσία ενός ρολογιού μπορεί να χρησιμοποιηθεί για την αντιμετώπιση πολλών προβλημάτων
 - ▶ Πρόβλημα συγχρονισμού
 - ▶ Πρόβλημα επικύρωσης δοσοληψιών
 - ▶ Πρόβλημα πιστοποίησης
 - ▶ ... (β. *Liskov, PODC '91*)



Προηγούμενο Μάθημα

Προηγούμενο Μάθημα
Ασύγχρονα Κατανεμημένα Συστήματα
Κατανεμημένες Δομές

Ασύγχρονα Κατανεμημένα Συστήματα

Πρόβλημα Συγχρονισμού
Συγχρονιστές
Συγχρονισμός Φυσικών Ρολογιών

Σύνοψη Μαθήματος

Σύνοψη Μαθήματος
Βιβλιογραφία
Επόμενο Μάθημα



Θέματα Σχεδιασμού (1)

Διαδικασία Σχεδιασμού:

1. Θέτουμε το πρόβλημα – κατασκευή επικαλυπτικού δέντρου, αναζήτηση κατά εύρος, αμοιβαίος αποκλεισμός, ...
2. Προσδιορίζουμε το σύστημα – κάποια εκδοχή του ασύγχρονου μοντέλου
3. Σχεδιάζουμε έναν νέο αλγόριθμο ή προσαρμόζουμε κάποιον από τους υπάρχοντες

Εναλλακτική προσέγγιση:

- ▶ Παρεμβάλουμε ένα 'ενδιάμεσο' επίπεδο μεταξύ του υλικού (κόμβοι, κανάλια) και του λογισμικού (διεργασίες)
- ▶ **Πλαίσιο Εργασίας:** Το 'ενδιάμεσο' επίπεδο *παρουσιάζει* το υποκείμενο σύστημα ως ένα σύγχρονο σύστημα



Θέματα Σχεδιασμού (2)

Πλαίσιο Εργασίας – Διαδικασία Σχεδιασμού

1. Θέτουμε το πρόβλημα – δρομολόγηση, κατασκευή επικαλυπτικού δέντρου, αναζήτηση κατά εύρος, ...
2. Προσδιορίζουμε το σύστημα – κάποια εκδοχή του ασύγχρονου μοντέλου
3. Παρεμβάλουμε το 'ενδιάμεσο' επίπεδο 'συγχρονισμού'
4. Σχεδιάζουμε έναν νέο αλγόριθμο ή προσαρμόζουμε κάποιον από τους υπάρχοντες – για το σύγχρονο μοντέλο

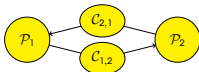
Μπορούμε να χαρακτηρίσουμε την παραπάνω διαδικασία ως ένα *τρόπο μετατροπής αλγορίθμων* για σύγχρονα συστήματα σε αλγόριθμους για ασύγχρονα συστήματα



Θέματα Σχεδιασμού (3)

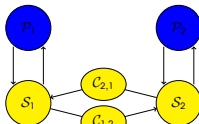
1^η προσέγγιση

algo-asynch.nc
tosmake.sh pc



2^η προσέγγιση

algo-synch.nc
tosmake.sh synch-pc
tosmake.sh pc



Η βασική ιδέα: σε κάθε κόμβο u η διεργασία επικοινωνεί με το υπόλοιπο σύστημα διαμέσου ενός συγχρονιστή - ο συγχρονιστής 'κρύβει' από τις ασύγχρονες διεργασίες το ασύγχρονο σύστημα



Το πρόβλημα του Συγχρονισμού

- ▶ Θεωρούμε ότι σε κάθε κόμβο εκτελούνται 2 διεργασίες
 1. Η διεργασία \mathcal{P} που αντιστοιχεί στο αυτόματο εισόδου/εξόδου του ασύγχρονου πρωτοκόλλου
 2. Η διεργασία \mathcal{S} που αντιστοιχεί στο αυτόματο εισόδου/εξόδου του συγχρονιστή

Πρόβλημα Συγχρονισμού

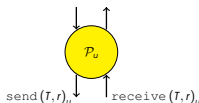
Το πρόβλημα του συγχρονισμού λύνεται από τον αλγόριθμο \mathcal{A} αν προσφέρει ένα περιβάλλον εκτέλεσης τέτοιο ώστε μια διεργασία \mathcal{P} να μην μπορεί να διαχωρίσει αν εκτελείται σε ένα ασύγχρονο σύστημα (δηλ. ο κατανεμημένος αλγόριθμος μαζί με τον συγχρονιστή) ή απ' ευθείας σε ένα σύγχρονο σύστημα.



Προδιαγραφές Αυτόματου Εισόδου/Εξόδου \mathcal{P} (1)

Η διεργασία \mathcal{P}_u

- ▶ Έστω \mathcal{M} το αλφάβητο μηνυμάτων που χρησιμοποιεί ο αλγόριθμος κατά την εκτέλεση σε σύγχρονο σύστημα
- ▶ Αντιστοιχούμε σε κάθε μήνυμα m μια επικέτα v που προσδιορίζει τον παραλήπτη του μηνύματος



- ▶ Η ενέργεια εξόδου του αυτόματου \mathcal{P}_u είναι της μορφής $\text{send}(T, r)_u$ όπου
 - ▶ T - ένα σύνολο από μηνύματα με επικέτες (π.χ. $\{(m, v)\}$)
 - ▶ $r \in \mathcal{N}^+$ - ο γύρος του ασύγχρονου συστήματος κατά τον οποίο πραγματοποιείται η ενέργεια



Προδιαγραφές Αυτόματου Εισόδου/Εξόδου \mathcal{P} (2)

- ▶ Η ενέργεια εισόδου του αυτόματου \mathcal{P}_u είναι της μορφής $\text{receive}(T, r)_u$ όπου
 - ▶ T - ένα σύνολο από μηνύματα με επικέτες (π.χ. $\{(m, v)\}$)
 - ▶ $r \in \mathcal{N}^+$ - ο γύρος του ασύγχρονου συστήματος κατά τον οποίο πραγματοποιείται η ενέργεια
- ▶ Αν η \mathcal{P}_u δεν έχει να στείλει κάποιο μήνυμα κατά τον γύρο r τότε πραγματοποιεί την ενέργεια $\text{send}(\text{null}, r)_u$

Παράδειγμα Ενέργειών Εισόδου / Εξόδου για το αυτόματο \mathcal{P}

Έστω $n = 3$. Η ενέργεια $\text{send}(\{(m_1, 1), (m_2, 2)\}, 4)_3$ υποδεικνύει ότι στον γύρο 4, η διεργασία \mathcal{P}_3 στέλνει το μήνυμα m_1 στην \mathcal{P}_1 και το m_2 στην \mathcal{P}_2 . Αντίστοιχα, η $\text{receive}(\{(m_1, 1), (m_2, 2)\}, 4)_3$ υποδεικνύει ότι στον γύρο 4, η διεργασία \mathcal{P}_3 παραλαμβάνει το μήνυμα m_1 από την \mathcal{P}_1 και το m_2 από την \mathcal{P}_2 .



Αλγόριθμος SimpleSynch

Για κάθε γύρο r η διεργασία S_u αφού λάβει $\text{send}(T, r)_u$ από την \mathcal{P}_u , για κάθε $\langle m, v \rangle \in T$ στέλνει το μήνυμα $\langle m, r \rangle$ στην S_v . Για κάθε μήνυμα $\langle m, r \rangle$ που λαμβάνει από την S_v , εισάγει το $\langle m, r \rangle$ στο διάνυσμα T . Όταν λάβει μήνυμα από κάθε γειτονική S_v για το γύρο r παραδίδει το τελικό $\text{receive}(T, r)_u$.

- ▶ Αν η \mathcal{P}_u δεν έχει να στείλει κάποιο μήνυμα κατά τον γύρο r στην διεργασία v θεωρούμε ότι 'συμπληρώνει' το T με $\langle \text{null}, v \rangle$
- ▶ Μια πρώτη, απλή υλοποίηση του συγχρονιστή S
- ▶ Η SimpleSynch λειτουργεί σε 'τοπικό επίπεδο' – οι διεργασίες συνεργάζονται για να συντονίσουν τα βήματα των υποκείμενων αλγόριθμων



Ενέργειες:

- ▶ Ενέργειες Εισόδου $\text{in}(\text{SimpleSynch}_u)$
 1. $\text{send}(T, r)_u$ – όπου T ένα σύνολο απο μηνύματα με ετικέτες, $r \in \mathcal{N}^+$
 2. $\text{net-recv}(N, r)_{v,u}$ – όπου N ένα σύνολο απο μηνύματα, $r \in \mathcal{N}^+$, $v \in \text{nbrs}_u$
- ▶ Ενέργειες Εξόδου $\text{out}(\text{SimpleSynch}_u)$
 1. $\text{receive}(T, r)_u$ – όπου T ένα σύνολο απο μηνύματα με ετικέτες, $r \in \mathcal{N}^+$
 2. $\text{net-send}(N, r)_{u,v}$ – όπου N ένα σύνολο απο μηνύματα, $r \in \mathcal{N}^+$, $v \in \text{nbrs}_u$



Καταστάσεις:

- ▶ $\text{proc-sent}, \text{proc-rcvd}$ – διανύσματα από booleans που δεικτοδοτούνται από το \mathcal{N}^+ , αρχικά όλα τα στοιχεία είναι false
- ▶ $\text{net-sent}, \text{net-rcvd}$ – πίνακες από booleans που δεικτοδοτούνται από $\text{nbrs}_u \times \mathcal{N}^+$, αρχικά όλα τα στοιχεία είναι false
- ▶ outbox – ένας πίνακας από σύνολα μηνυμάτων, που δεικτοδοτείται από το $\text{nbrs}_u \times \mathcal{N}^+$, αρχικά όλα τα στοιχεία είναι null
- ▶ inbox – ένας πίνακας από σύνολα μηνυμάτων με ετικέτες, που δεικτοδοτείται από το \mathcal{N}^+ , αρχικά όλα τα στοιχεία είναι null



Μεταβάσεις:

- ▶ $\text{send}(T, r)_u$
 - ▶ αποτέλεσμα:
 - $\text{proc-sent}(x) = \text{true}$
 - για κάθε $v \in \text{nbrs}_u$, $\text{outbox}(v, r) = \{m | \langle m, v \rangle \in T\}$
- ▶ $\text{net-send}(N, r)_{u,v}$
 - ▶ προϋπόθεση:
 - $\text{proc-sent}(x) == \text{true}$
 - $\text{net-sent}(v, r) = \text{false}$
 - $N = \text{outbox}(v, r)$
 - ▶ αποτέλεσμα:
 - $\text{net-sent}(v, r) = \text{true}$



Αυτόματο SimpleSynch_u (4)

Μεταβάσεις (συνέχεια):

- ▶ $\text{net-recv}(N, r)_{v,u}$
 - ▶ **αποτέλεσμα:**
 $\text{inbox}(r) = \text{inbox}(r) \cup \{(m, v) | m \in N\}$
 $\text{net-rcvd}(v, r) = \text{true}$
- ▶ $\text{receive}(T, r)_u$
 - ▶ **προϋπόθεση:**
 $\text{proc-sent}(r) == \text{true}$
για κάθε $v \in \text{nbrs}_u, \text{net-rcvd}(v, r) == \text{true}$
 $T = \text{inbox}(r)$
 $\text{proc-rcvd}(r) == \text{false}$
 - ▶ **αποτέλεσμα:**
 $\text{proc-rcvd}(r) = \text{true}$



Χαρακτηριστικά του Αλγόριθμου SimpleSynch

Για κάθε γύρο:

- ▶ ανταλλάσσονται $2m$ μηνύματα
- ▶ Η διεργασία \mathcal{P}_u απαιτεί χρόνο $\mathcal{O}(l)$
- ▶ Η διεργασία \mathcal{S}_u απαιτεί χρόνο $\mathcal{O}(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r(d + \mathcal{O}(l))$

Παράδειγμα εκτέλεσης



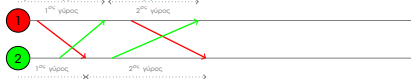
Χαρακτηριστικά του Αλγόριθμου SimpleSynch

Για κάθε γύρο:

- ▶ ανταλλάσσονται $2m$ μηνύματα
- ▶ Η διεργασία \mathcal{P}_u απαιτεί χρόνο $\mathcal{O}(l)$
- ▶ Η διεργασία \mathcal{S}_u απαιτεί χρόνο $\mathcal{O}(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r(d + \mathcal{O}(l))$

Παράδειγμα εκτέλεσης



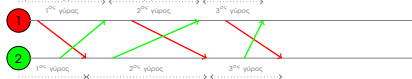
Χαρακτηριστικά του Αλγόριθμου SimpleSynch

Για κάθε γύρο:

- ▶ ανταλλάσσονται $2m$ μηνύματα
- ▶ Η διεργασία \mathcal{P}_u απαιτεί χρόνο $\mathcal{O}(l)$
- ▶ Η διεργασία \mathcal{S}_u απαιτεί χρόνο $\mathcal{O}(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r(d + \mathcal{O}(l))$

Παράδειγμα εκτέλεσης

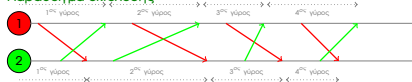


Για κάθε γύρο:

- ▶ ανταλλάσσονται $2m$ μηνύματα
- ▶ Η διεργασία \mathcal{P}_u απαιτεί χρόνο $\mathcal{O}(l)$
- ▶ Η διεργασία \mathcal{S}_u απαιτεί χρόνο $\mathcal{O}(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r(d + \mathcal{O}(l))$

Παράδειγμα εκτέλεσης

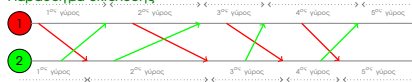


Για κάθε γύρο:

- ▶ ανταλλάσσονται $2m$ μηνύματα
- ▶ Η διεργασία \mathcal{P}_u απαιτεί χρόνο $\mathcal{O}(l)$
- ▶ Η διεργασία \mathcal{S}_u απαιτεί χρόνο $\mathcal{O}(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r(d + \mathcal{O}(l))$

Παράδειγμα εκτέλεσης



Ο Συγχρονιστής των Tel και Lseuwen (1)

- ▶ Ο συγχρονιστής των Tel και Lseuwen βασίζεται στις εξής παραδοχές
 1. Υπάρχει ένα άνω φράγμα l στον χρόνο εκτέλεσης κάθε ενέργειας ϵ σε κάποια κατάσταση κ
 2. Υπάρχει ένα άνω φράγμα d στον χρόνο παράδοσης του παλαιότερου μηνύματος που βρίσκεται σε κάποιο κανάλι επικοινωνίας
- ▶ Ονομάζουν τα ασύγχρονα συστήματα που ικανοποιούν τις δύο συνθήκες ως *Asynchronous Bounded-Delay Networks*
- ▶ Υπό αυτές τις συνθήκες είναι εύκολο να υλοποιήσεις έναν συγχρονιστή
- ▶ Το βασικό πρόβλημα στο σχεδιασμό του συγχρονιστή είναι η εξασφάλιση ότι όλα τα μηνύματα του γύρου r παραλαμβάνονται πριν ξεκινήσει ο γύρος $r + 1$

Ο Συγχρονιστής των Tel και Lseuwen (2)

- ▶ Θεωρούμε ότι οι κόμβοι διαθέτουν ένα ρολόι
- ▶ Τα ρολόγια των κόμβων είναι συγχρονισμένα
- ▶ Υπάρχει ένα άνω φράγμα $\mu \geq l + d$ το οποίο είναι γνωστό
- ▶ Δεν υποχρεώνουμε την \mathcal{P}_u να στείλει μήνυμα null κατά τον γύρο r αν δεν έχει κάποιο μήνυμα

Αλγόριθμος ABD

Για κάθε γύρο r η διεργασία \mathcal{S}_u αφού λάβει $\text{send}(T, r)_u$ από την \mathcal{P}_u , για κάθε $\langle m, v \rangle \in T$ στέλνει το μήνυμα $\langle m, r \rangle$ στην \mathcal{S}_v . Για κάθε μήνυμα $\langle m, r \rangle$ που λαμβάνει από την \mathcal{S}_v , εισάγει το $\langle m, r \rangle$ στο διάγραμμα T_r . Όταν το ρολόι δείξει $r \cdot 2 \cdot \mu$, παραδίδει το τελικό $\text{receive}(T, r)_u$.

Αυτόματο ABD_u (1)

Ενέργειες:

- ▶ Ενέργειες Εισόδου $in(ABD_u)$
 1. $send(T, r)_u$ – όπου T ένα σύνολο απο μηνύματα με επικέτες, $r \in \mathcal{N}^+$
 2. $net-receive(N, r)_{v,u}$ – όπου N ένα σύνολο απο μηνύματα, $r \in \mathcal{N}^+$, $v \in nbrs_u$
- ▶ Ενέργειες Εξόδου $out(ABD_u)$
 1. $receive(T, r)_u$ – όπου T ένα σύνολο απο μηνύματα με επικέτες, $r \in \mathcal{N}^+$
 2. $net-send(N, r)_{u,v}$ – όπου N ένα σύνολο απο μηνύματα, $r \in \mathcal{N}^+$, $v \in nbrs_u$



Αυτόματο ABD_u (2)

Καταστάσεις:

- ▶ $clock_u$ – ένα ρολόι
- ▶ $round$ – μεταβλητή τύπου $integer$, αρχικά 1
- ▶ $pulse$ – μεταβλητή τύπου $time$, αρχικά η στιγμή έναρξης
- ▶ $proc-sent, proc-rcvd$ – διανύσματα από booleans που δεικτοδοτούνται από το \mathcal{N}^+ , αρχικά όλα τα στοιχεία είναι $false$
- ▶ $outbox$ – ένας πίνακας από σύνολα μηνυμάτων, που δεικτοδοτείται από το $nbrs_u \times \mathcal{N}^+$ αρχικά όλα τα στοιχεία είναι $null$
- ▶ $inbox$ – ένας πίνακας από σύνολα μηνυμάτων με επικέτες, που δεικτοδοτείται από το \mathcal{N}^+ , αρχικά όλα τα στοιχεία είναι $null$



Αυτόματο ABD_u (3)

Μεταβάσεις:

- ▶ $send(T, r)_u$
 - ▶ αποτέλεσμα:
 $proc-sent(x) = true$
για κάθε $v \in nbrs_u$, $outbox(v, r) = \{m \mid (m, v) \in T\}$
- ▶ $net-send(N, r)_{u,v}$
 - ▶ προϋπόθεση:
 $proc-sent(x) == true$
 $N = outbox(v, r)$



Αυτόματο ABD_u (4)

Μεταβάσεις (συνέχεια):

- ▶ $net-receive(N, r)_{v,u}$
 - ▶ αποτέλεσμα:
 $inbox(x) = inbox(r) \cup \{(m, v) \mid m \in N\}$
- ▶ $receive(T, r)_u$
 - ▶ προϋπόθεση:
 $clock_u - pulse == 2 \cdot round \cdot \mu$
 $T = inbox(r)$
 $proc-rcvd(x) == false$
 - ▶ αποτέλεσμα:
 $proc-rcvd(x) = true$



Χαρακτηριστικά του Αλγόριθμου ABD

Για κάθε γύρο:

- ▶ δεν ανταλλάσσονται μηνύματα από την διεργασία S_U
- ▶ Η διεργασία P_U απαιτεί χρόνο $\mathcal{O}(l)$
- ▶ Η διεργασία S_U απαιτεί χρόνο $\mathcal{O}(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r \cdot \mathcal{O}(d + l)$

Παράδειγμα εκτέλεσης



Χαρακτηριστικά του Αλγόριθμου ABD

Για κάθε γύρο:

- ▶ δεν ανταλλάσσονται μηνύματα από την διεργασία S_U
- ▶ Η διεργασία P_U απαιτεί χρόνο $\mathcal{O}(l)$
- ▶ Η διεργασία S_U απαιτεί χρόνο $\mathcal{O}(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r \cdot \mathcal{O}(d + l)$

Παράδειγμα εκτέλεσης



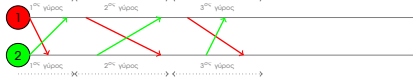
Χαρακτηριστικά του Αλγόριθμου ABD

Για κάθε γύρο:

- ▶ δεν ανταλλάσσονται μηνύματα από την διεργασία S_U
- ▶ Η διεργασία P_U απαιτεί χρόνο $\mathcal{O}(l)$
- ▶ Η διεργασία S_U απαιτεί χρόνο $\mathcal{O}(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r \cdot \mathcal{O}(d + l)$

Παράδειγμα εκτέλεσης



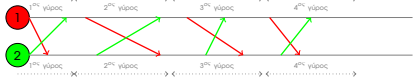
Χαρακτηριστικά του Αλγόριθμου ABD

Για κάθε γύρο:

- ▶ δεν ανταλλάσσονται μηνύματα από την διεργασία S_U
- ▶ Η διεργασία P_U απαιτεί χρόνο $\mathcal{O}(l)$
- ▶ Η διεργασία S_U απαιτεί χρόνο $\mathcal{O}(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r \cdot \mathcal{O}(d + l)$

Παράδειγμα εκτέλεσης



Χαρακτηριστικά του Αλγόριθμου ABD

Για κάθε γύρο:

- ▶ Δεν ανταλλάσσονται μηνύματα από την διεργασία S_u
- ▶ Η διεργασία P_u απαιτεί χρόνο $O(l)$
- ▶ Η διεργασία S_u απαιτεί χρόνο $O(l)$

Για την προσομοίωση r γύρων απαιτείται χρόνος $r \cdot O(d + l)$

Παράδειγμα εκτέλεσης



Συζήτηση

Αναζήτηση κατά Εύρος

αλγόριθμος	AsynchBFS	SynchBFS SimpleSync	SynchBFS ABD
χρόνος	$O(\delta \cdot n(d + l))$	$O(\delta(d + l))$	$O(\delta \cdot \mu)$
μηνύματα	$O(n \cdot m)$	$O(n \cdot m^2)$	$O(n \cdot m)$

Συντομότερα μονοπάτια

αλγόριθμος	AsynchBellmanFord	BellmanFord SimpleSync	BellmanFord ABD
χρόνος	$O(n^{n+1}(l + d))$	$O(n(l + d))$	$O(n \cdot \mu)$
μηνύματα	$O(n^n \cdot m)$	$O(n^2 \cdot m^2)$	$O(n^2 \cdot m)$

- ▶ Η χρονική πολυπλοκότητα του ABD είναι άνω φράγμα για την απόδοση του SimpleSync
- ▶ **Συμπέρασμα:** Αν έχουμε επιπλέον γνώση, μπορούμε να μειώσουμε την πολυπλοκότητα επικοινωνίας

Φυσικά Ρολόγια

- ▶ Σε επίπεδο λογισμικού, η ώρα την στιγμή t είναι $C(t) = H(t) \cdot \alpha + \beta$, όπου
 - ▶ $H(t)$ – η τιμή του φυσικού ρολογιού
 - ▶ α – η μονάδα μέτρησης του φυσικού ρολογιού
 - ▶ β – η τιμή της 'ώρας 0' (διόρθωση)
- ▶ Τα ρολόγια δεν είναι τέλεια, δηλ. $C(t) \neq t$
- ▶ Ακρίβεια – Η ακρίβεια του φυσικού ρολογιού (επίπεδο υλικό) χαρακτηρίζεται ως προς:
 - ▶ Ρυθμός απόκλισης (*drift*) – ο ρυθμός με τον οποίο το ρολόι 'επιβραδύνει' ή 'επιταχύνει' κατά την μέτρηση του χρόνου
 - ▶ Διαφορά (*skew*) – Η απόλυτη διαφορά $|C_1(t) - C_2(t)|$ μεταξύ των μετρήσεων δύο ρολογιών την χρονική στιγμή t

Συγχρονισμός Ρολογιών (1)

- ▶ Ένα ρολόι που αποκλίνει πρέπει να διορθωθεί σύμφωνα με μια τιμή που θεωρείται σωστή – π.χ. λαμβάνεται από κάποια άλλη διεργασία
- ▶ Αν το ρολόι έχει θετικό ρυθμό απόκλισης, δηλ. επιταχύνει – η διόρθωση θα θέσει το τοπικό ρολόι σε κάποια προηγούμενη τιμή
 - ▶ Ακυρώνεται η μονοτονική ιδιότητα του ρολογιού
- ▶ Η διόρθωση ενός ρολογιού πρέπει να γίνεται με προσοχή
 - ▶ *Σωστή λύση:* Αλλάζουμε καταλληλώς την συχνότητα μέτρησης του υλικού του ρολογιού ώστε να περιοριστεί η τάση απόκλισης που παρουσιάζει

Συγχρονισμός Ρολογιών (2)

Εξωτερικός συγχρονισμός: – σε σχέση με ένα εξωτερικό ρολόι C_{global} , η διεργασία \mathcal{P}_u έχει ακρίβεια δ , όταν για κάθε χρονική στιγμή t ισχύει:

$$|C_u(t) - C_{global}(t)| \leq \delta, \delta \geq 0, u \in \{1, \dots, n\}$$

Εσωτερικός συγχρονισμός – οι διεργασίες $\mathcal{P}_u, u \in \{1, \dots, n\}$ με διαφορά δ , όταν για κάθε χρονική στιγμή t ισχύει:

$$|C_u(t) - C_v(t)| \leq \delta, \delta \geq 0, \forall u, v \in \{1, \dots, n\}$$

Παρατηρήσεις:

- ▶ Αν όλες οι διεργασίες $\mathcal{P}_u, u \in \{1, \dots, n\}$ είναι εξωτερικά συγχρονισμένες με ακρίβεια δ , τότε συνεπάγεται πως είναι εσωτερικά συγχρονισμένες με μέγιστη διαφορά $2 \cdot \delta$
- ▶ Αν γνωρίζουμε ότι οι διεργασίες είναι εσωτερικά συγχρονισμένες, δεν μπορούμε να βγάλουμε κάποιο συμπέρασμα για τον εξωτερικό συγχρονισμό τους



Συγχρονισμός Ρολογιών (3)

- ▶ Ακόμα και όταν συγχρονίσουμε δύο ρολόγια την χρονική στιγμή t με καλή ακρίβεια, λόγω του ρυθμού απόκλισης, μια επόμενη χρονική στιγμή $t' > t$, τα δυο ρολόγια μπορεί να παρουσιάσουν διαφορά
- ▶ Εφόσον επιθυμούμε τα τοπικά ρολόγια να είναι συγχρονισμένα κατά την λειτουργία του συστήματος, πρέπει να συγχρονίζουμε τα τοπικά ρολόγια περιοδικά, τουλάχιστον κάθε $\frac{\delta}{2 \cdot \rho}$ μονάδες χρόνου
- ▶ Επίσης πρέπει να δεχθούμε τις παρακάτω αποκλίσεις:
 1. Η διαδικασία συγχρονισμού απαιτεί κάποιο χρονικό διάστημα για να ολοκληρωθεί και πιθανόν να προϋποθέτει ένα πλήθος μηνυμάτων
 2. Η διαδικασία μπορεί να μην είναι απόλυτως σωστή – επομένως δεν επιτυγχάνει τέλει συγχρονισμό



Συγχρονισμός μέσω Εξωτερικού Ρολογιού

Δορυφόρος Navstar

Υπάρχουν συστήματα που προσφέρουν υπηρεσίες μέτρησης του χρόνου

- ▶ **Ραδιο-Φάροι** – εκπέμπουν την τοπική ώρα
- ▶ **Συστήματα Παγκόσμιας Γεωγραφικής Θέσης (GPS)** εκπέμπουν την παγκόσμια ώρα



Αν υπάρχει δυνατότητα επικοινωνίας με τέτοια δίκτυα, τα τοπικά ρολόγια συγχρονίζονται εύκολα

- ▶ Μπορεί να μην είναι εφικτό – κόστος, κατανάλωση, μέγεθος

Ένα μέρος των κόμβων έχουν πρόσβαση – ετερογενές δίκτυο



Η Μέθοδος του Cristian για Συγχρονισμό Ρολογιών (1)

Ένα μέρος των κόμβων έχουν πρόσβαση σε δίκτυα παροχής υπηρεσιών μέτρησης χρόνου υψηλής ποιότητας – **ετερογενές σύστημα**

- ▶ Όταν η διεργασία \mathcal{P}_u βρίσκεται στην φάση συγχρονισμού του τοπικού ρολογιού της, στέλνει ένα μήνυμα `synch` στην διεργασία \mathcal{P}_{server}
- ▶ Η διεργασία \mathcal{P}_{server} απαντά με ένα μήνυμα `time (t)`
- ▶ Η διεργασία \mathcal{P}_u σημειώνει την χρονική στιγμή t_{send} που έστειλε το μήνυμα `synch` και την χρονική στιγμή t_{cv} που έλαβε το μήνυμα από την \mathcal{P}_{server} – μπορεί να υπολογίσει την συνολική καθυστέρηση $T_{roundtrip} = t_{cv} - t_{send}$



Η Μέθοδος του Cristian για Συγχρονισμό Ρολογιών (2)

- ▶ Η μέτρηση της $T_{roundtrip}$, σε πραγματικές συνθήκες, είναι αρκετά ακριβής, δηλ. μικρός ρυθμός απόκλισης ρ
- ▶ Η διεργασία \mathcal{P}_U θέτει το τοπικό ρολόι στην τιμή $t' = t_{server} + \frac{T_{roundtrip}}{2}$
- ▶ Η διεργασία \mathcal{P}_U μπορεί να υπολογίσει τα άνω και κάτω όρια για την μετάδοση ενός μηνύματος από ένα κανάλι (t_{max}, t_{min})
- ▶ Η διεργασία \mathcal{P}_U μπορεί να επιλέξει ρυθμό συγχρονισμού $t_d = \frac{t_{max} + t_{min}}{2}$, όπου η μέγιστη διαφορά είναι $\frac{t_{max} + t_{min}}{2}$
- ▶ Με τη μέθοδο Cristian μπορεί να υπολογιστεί $t_d = \frac{T_{roundtrip}}{2} - t_{min}$



Χαρακτηριστικά μεθόδου Cristian

- ▶ Η ανάλυση της απόδοσης της μεθόδου μπορεί να γίνει με την χρήση πιθανοτικών τεχνικών – οι $t_{min}, t_{max}, T_{roundtrip}$ είναι τυχαίες μεταβλητές
- ▶ Η μέθοδος **προσαρμόζεται** στις τοπικές συνθήκες του δικτύου αναλόγως με τις μέτρησης των $t_{min}, t_{max}, T_{roundtrip}$ από την \mathcal{P}_U
- ▶ Τι γίνεται με την διεργασία \mathcal{P}_{server} :
 - ▶ Το σύστημα μπορεί να παρουσιάσει **περιορισμένη κλιμάκωση**
 - ▶ Τα κανάλια επικοινωνίας του \mathcal{P}_{server} λαμβάνουν μεγάλο όγκο μηνυμάτων
 - ▶ Η διεργασία \mathcal{P}_{server} μπορεί να παρουσιάσει καθυστερήσεις σε περιπτώσεις ταυτόχρονων αιτήσεων



Η Μέθοδος των Gusella και Zatti

- ▶ Υλοποιείται στο λειτουργικό Berkeley UNIX
- ▶ Λειτουργεί 'ανάποδα' από τη μέθοδο Cristian
- ▶ Η διεργασία \mathcal{P}_{master} επικοινωνεί με την διεργασία \mathcal{P}_U σε τακτά χρονικά διαστήματα, ενημερώνοντας την με την τρέχουσα ώρα
- ▶ Η διεργασία \mathcal{P}_{master} χρησιμοποιεί την μέθοδο Cristian για να υπολογίσει την τρέχουσα τιμή του ρολογιού της διεργασίας \mathcal{P}_U
- ▶ Σηματοποιεί τον μέσο όρο από όλα τα ρολόγια συμπεριλαμβανοντας και την τιμή του δικού της
- ▶ Στέλνει σε κάθε διεργασία \mathcal{P}_U την **διαφορά της τιμής του ρολογιού της** από τον μέσο όρο, και η \mathcal{P}_U προσαρμόζει το ρολόι της ανάλογα



RFC 958: Network Time Protocol (NTP)

- ▶ Οι προηγούμενες μέθοδοι λειτουργούν ικανοποιητικά σε συστήματα
 - ▶ με μικρό αριθμό κόμβων
 - ▶ μικρή ανοχή σφαλμάτων
- ▶ Συγχρονίζει τα τοπικά ρολόγια διεργασιών που συνδέονται μέσω του διαδικτύου – με τις ακόλουθες προδιαγραφές
 - ▶ ανοχή σε σφάλματα επικοινωνίας ακόμα και για μεγάλα χρονικά διαστήματα
 - ▶ κλιμάκωση για πολύ μεγάλο αριθμό κόμβων
 - ▶ ακρίβεια συγχρονισμού σε μερικές δεκάδες ms σε κανάλια του διαδικτύου και λίγα ms σε τοπικά δίκτυα

<http://www.faqs.org/rfcs/rfc958.html>



Προηγούμενο Μάθημα

- Προηγούμενο Μάθημα
- Ασύγχρονα Καταναμημένα Συστήματα
- Καταναμημένες Δομές

Ασύγχρονα Καταναμημένα Συστήματα

- Πρόβλημα Συγχρονισμού
- Συγχρονιστές
- Συγχρονισμός Φυσικών Ρολογιών

Σύνοψη Μαθήματος

- Σύνοψη Μαθήματος
- Βιβλιογραφία
- Επόμενο Μάθημα



- ▶ Ασύγχρονα Καταναμημένα Συστήματα
- ▶ Πρόβλημα Δρομολόγησης
- ▶ Συγχρονιστές
 - ▶ Καταναμημένος Αλγόριθμος SimpleSync
 - ▶ Καταναμημένος Αλγόριθμος ABD
- ▶ Συγχρονισμός Ρολογιών
 - ▶ Μέθοδος Cristian
 - ▶ Μέθοδος Gusella και Zatti
 - ▶ Πρωτόκολλο NTP



Βιβλιογραφία (1)

- ▶ Βιβλίο "Distributed Algorithms" (N.Lynch)
 1. Κεφάλαιο 15: Basic Asynchronous Network Algorithms
 2. Κεφάλαιο 16: Modelling II: Synchronizers
- ▶ Βιβλίο "Distributed Computing Fundamentals, Simulations, and Advanced Topics" (H.Attiya, J.Welch)
 1. Κεφάλαιο 11: Simulating Synchrony



Βιβλιογραφία (2)

- ▶ Βιβλίο "Distributed Systems, Concepts and Design" (G.Coulouris, J.Dollimore, T.Kindberg)
 1. Κεφάλαιο 2: Time and Global States – Μόνο 10.1, 10.2, 10.3
- ▶ Βιβλίο "Introduction to Distributed Algorithms" (G.Tel)
 1. Κεφάλαιο 12: Synchrony in Networks – Μόνο 12.3, 12.4
 2. Κεφάλαιο 15: Fault Tolerance In Synchronous Systems-- Μόνο 15.3
- ▶ Βιβλίο "Distributed Systems: Principles and Paradigms" (A.Tanenbaum, M.Steen)
 1. Κεφάλαιο 5: Synchronization – Μόνο 5.1



Επόμενο Μάθημα

- ▶ Ασύγχρονα Κατανεμημένα Συστήματα
- ▶ Πρόβλημα Συναίεσης
- ▶ Εντοπιστές Σφαλμάτων