

Κατανεμημένα Συστήματα I

Μάθημα Βασικής Επιλογής, Χειμερινού Εξαμήνου

Τομέας Εφαρμογών και Θεμελιώσεων

Ιωάννης Χατζηγιαννάκης

Δευτέρα, 16 Νοεμβρίου, 2009
Αίθουσα Β3

Προηγούμενο Μάθημα

- Ασύγχρονα Κατανεμημένα Συστήματα
- Διάταξη Γεγονότων
 - Σχέση **συνέβη-πριν**
- Λογικός Χρόνος
 - Λογικά Ρολόγια Lamport
- Αμοιβαίος Αποκλεισμός
 - Συγκεντρωτική Λύση
 - Χρήση Λογικών Ρολογιών
 - Χρήση Λογικών Δομών

Διάταξη Γεγονότων

- Ο συγχρονισμός ρολογιών σε ένα κατανομημένο σύστημα δεν είναι καθόλου εύκολο πρόβλημα
- Ο συγχρονισμός διεργασιών μπορεί να λυθεί και με 'ασθενέστερες' συνθήκες
 - Ο Lamport παρατηρεί ότι αρκεί να εξασφαλίσουμε ότι όλα τα γεγονότα είναι πλήρως διατεταγμένα κατά έναν απολύτως συνεπή τρόπο
- Από το μοντέλλο των Ασύγχρονων Κατανομημένων Συστημάτων:
 - Κάθε διεργασία \mathcal{P}_u χαρακτηρίζεται από την κατάσταση της $state_u$
 - Ορίζουμε ως συμβάν σ^u μια ενέργεια ϵ που επιφέρει μια αλλαγή στην κατάσταση κάποιας διεργασίας \mathcal{P}_u
 - Υποθέτουμε ότι οι διεργασίες εξελίσσονται σειριακά

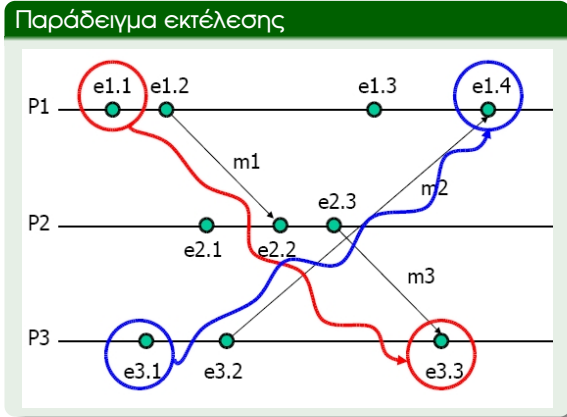
Η σχέση συνέβη-πριν

- Έστω α μία εκτέλεση του συστήματος \mathcal{S}
- Η εκτέλεση α αποτελείται από μια ακολουθία συμβάντων $\sigma_1, \sigma_2, \dots, \sigma_k, \dots$
- Θέλουμε να διατάξουμε τα συμβάντα που συμβαίνουν σε διαφορετικές διεργασίες
- Η σχέση **‘συνέβη-πριν’** \rightsquigarrow μοντελοποιεί την λογική σχέση εξάρτησης των συμβάντων
- Αν $\sigma_i \rightsquigarrow \sigma_j$ τότε το συμβάν σ_i συνέβη πριν το σ_j
- Η διάταξη βασίζεται στα παρακάτω:
 - 1 αν δύο συμβάντα συνέβησαν στην ίδια διεργασία \mathcal{P}_u τότε συνέβησαν με τη σειρά που τα παρατήρησε η \mathcal{P}_u – όπως προκύπτει απο την ιστορία \mathcal{H}_u
 - 2 αν η \mathcal{P}_u στείλει ένα μήνυμα m στην \mathcal{P}_v όπου $\sigma_i^u = send(m)$ και $\sigma_j^v = receive(m)$ τότε η αποστολή του μηνύματος (συμβάν στον αποστολέα) προηγείται λογικά της παραλαβής του (συμβάν στον παραλήπτη)

Παράδειγμα Εκτέλεσης

Η διάταξη ικανοποιεί την
Μεταβατική ιδιότητα αν
 $\sigma_i \rightsquigarrow \sigma_j$ και (στην
 συνέχεια) $\sigma_j \rightsquigarrow \sigma_k$, τότε
 $\sigma_i \rightsquigarrow \sigma_k$
 Επομένως

- $e_{1.1} \rightsquigarrow e_{3.3}$ εφόσον
 $e_{1.1} \rightsquigarrow e_{1.2} \rightsquigarrow$
 $e_{2.2} \rightsquigarrow e_{2.3} \rightsquigarrow e_{3.3}$
- $e_{3.1} \rightsquigarrow e_{1.4}$ εφόσον
 $e_{3.1} \rightsquigarrow e_{3.2} \rightsquigarrow e_{1.4}$



Λογικά Ανεξάρτητα Συμβάντα

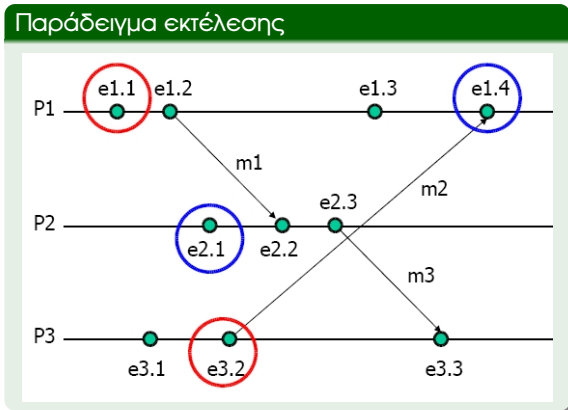
Δύο συμβάντα που δεν συνδέονται μεταξύ τους μέσω της σχέσης \rightsquigarrow ονομάζονται 'λογικά ανεξάρτητα' ή 'λογικά ταυτόχρονα':

$$\sigma_i^u \parallel \sigma_j^v \Leftrightarrow$$

$$(\sigma_i^u \not\rightsquigarrow \sigma_j^v) \wedge (\sigma_j^v \not\rightsquigarrow \sigma_i^u)$$

Επομένως

- $e_{1.1} \parallel e_{3.2}$
- $e_{2.1} \parallel e_{1.4}$



Λογικός Χρόνος

- Ο Lamport προτείνει τον μηχανισμό **‘λογικό ρολόι’** για τον ορισμό των χρονοσφραγίδων που ικανοποιούν την σχέση ‘συνέβη-πριν’
- Ένα λογικό ρολόι είναι ένας μονότονα αυξανόμενος μετρητής, του οποίου η τιμή δεν χρειάζεται να έχει κάποια ιδιαίτερη σχέση με κάποιο φυσικό ρολόι
- Κάθε διεργασία \mathcal{P}_u διατηρεί το δικό της λογικό ρολόι LC_u
- Δίνει σε κάθε συμβάν σ μια χρονοσφραγίδα $TS(\sigma) \in \mathcal{T}$, όπου \mathcal{T} είναι ένα πλήρως διατεταγμένο σύνολο
- Η χρονοσφραγίδα του συμβάντος σ_i όπου $\sigma_i \rightsquigarrow \sigma_j$ έχει μικρότερη από την χρονοσφραγίδα του σ_j δηλ. $TS(\sigma_i) < TS(\sigma_j)$
- Χρησιμοποιούμε την τιμή του λογικού ρολογιού ως χρονοσφραγίδα για κάθε συμβάν

Ο Αλγόριθμος του Lamport (1)

Αλγόριθμος LamportTime

Οι διεργασίες διατηρούν μια μεταβλητή LC η οποία αρχικά είναι 0. Για κάθε εσωτερικό συμβάν ή ένα συμβάν $send(m)$ θέτουν $LC++$. Μαζί με κάθε μήνυμα m που στέλνουν οι διεργασίες, επισυνάπτουν την τιμή της μεταβλητής LC . Μόλις λάβουν ένα μήνυμα m με χρονοσφραγίδα $TS(m)$ θέτουν $LC = \max\{LC, TS(m)\} + 1$.

- Αν για δύο συμβάντα σ_i και σ_j ισχύει $\sigma_i \rightsquigarrow \sigma_j$ ο αλγόριθμος LamportTime εξασφαλίζει ότι $TS(\sigma_i) < TS(\sigma_j)$
- Αν $TS(\sigma_i) < TS(\sigma_j)$ δεν ισχύει αναγκαστικά ότι $\sigma_i \rightsquigarrow \sigma_j$

Ο Αλγόριθμος του Lamport (2)

- Είναι πιθανό δύο συμβάντα σ_i και σ_j που πραγματοποιούνται σε δύο διαφορετικές διεργασίες να μην συνδέονται με την σχέση και να έχουν τις ίδιες χρονοσφραγίδες
- Όμως δεν επιτρέπουμε σε ένα συμβάν να συμβεί ακριβώς την ίδια χρονική στιγμή με κάποιο άλλο
- Για να ικανοποιούν τα λογικά ρολόγια του Lamport αυτή την συνθήκη, μια απλή λύση είναι η χρήση των ταυτοτήτων των διεργασιών για τον υπολογισμό των χρονοσφραγίδων
 - εδώ υποθέτουμε ότι οι διεργασίες διαθέτουν μοναδικές ταυτότητες
- Επομένως οι χρονοσφραγίδες των γεγονότων σ_i^u και σ_j^v θα είναι $i.u$ και $j.v$ -- π.χ. 11.15 και 11.306

Αμοιβαίος Αποκλεισμός

- Εστιάσαμε στο πρόβλημα του Αμοιβαίου Αποκλεισμού
 - Βασικό πρόβλημα – έχει μελετηθεί σε προηγούμενα μαθήματα
- Υποθέσαμε ότι:
 - Οι διεργασίες έχουν μοναδικές ταυτότητες
 - Κάθε διεργασία έχει μέρη του κώδικα της με κρίσιμα τμήματα
 - Δεν υπάρχει κάποιο καθολικό ρολόι
 - Το δίκτυο είναι πλήρες
 - Υποθέτουμε ότι τα κανάλια είναι **αξιόπιστα, FIFO**
- Είδαμε ορισμένους αλγόριθμους που βασίζονται στον λογικό χρόνο – στην σχέση συνέβη-πρίν
- Ο Lamport πρότεινε έναν αλγόριθμο που χρησιμοποιεί λογικά ρολόγια

Ο Αλγόριθμος του Lamport

Αλγόριθμος LamportME

Οι διεργασίες διατηρούν ένα τοπικό λογικό ρολόι σύμφωνα με τον αλγόριθμο LamportTime. Οι διεργασίες διατηρούν μια **ουρά** για τις εισερχόμενες αιτήσεις. Κάθε διεργασία που επιθυμεί να μπει σε ΚΤ στέλνει μια αίτηση (με χρονοσφραγίδα) σε όλες τις άλλες διεργασίες, και τοποθετεί την αίτηση στην ουρά της. Όταν μια διεργασία παραλάβει μια αίτηση την προσθέτει στην ουρά και απαντάει επιβεβαιώνοντας την παραλαβή. Η διεργασία με μικρότερη χρονοσφραγίδα εισέρχεται στο ΚΤ. Όταν η διεργασία βγει από το ΚΤ στέλνει ένα μήνυμα εξόδου σε όλες τις διεργασίες και αφαιρεί την αίτηση από την ουρά της – αντίστοιχα διαγράφουν και οι άλλες διεργασίες την αίτηση από την ουρά που διατηρούν.

- Πρώτη κατανεμημένη προσέγγιση – 1978
- Πολύ εύκολη υλοποίηση – 3 είδη μηνυμάτων: `request`, `reply`, `release`

Απαραίτητες ιδιότητες

Θεωρούμε ότι ένας αλγόριθμος λύνει το πρόβλημα του Αμοιβαίου Αποκλεισμού αν ικανοποιεί τις παρακάτω συνθήκες :

- 1 **Ασφάλεια (safety)** -- μια μόνο από τις διεργασίες μπορεί να αποκτήσει πρόσβαση στον κοινό πόρο σε ένα ορισμένο χρονικό διάστημα
- 2 **Βιωσιμότητα (liveness)** --
 - αν μια διεργασία επιθυμεί να εισέλθει στο κρίσιμο τμήμα τελικά θα το καταφέρει
 - αν ο κοινός πόρος δεν χρησιμοποιείται, τότε όποια διεργασία ζητήσει πρόσβαση θα πρέπει να την αποκτήσει σε πεπερασμένο χρονικό διάστημα
- 3 **Διάταξη (ordering)** -- η άδεια εισόδου στο κρίσιμο τμήμα πρέπει να παραχωρηθεί σύμφωνα με τη σχέση **συνέβη-πριν**: οι αιτήσεις των διεργασιών εξυπηρετούνται με τη σειρά που έχουν εκδοθεί

Ορθότητα του Αλγόριθμου LamportME (1)

Λήμμα (LamportME.1)

Ο αλγόριθμος LamportME ικανοποιεί την ιδιότητα της ασφάλειας.

Απόδειξη: Με εις άτοπο απαγωγή στον τρόπο λειτουργίας του αλγόριθμου.

- Έστω σε κάποια εκτέλεση του συστήματος όπου δύο διεργασίες u και v βρίσκονται στο ΚΤ την ίδια στιγμή.
- Έστω το μήνυμα $request_u$, το έστειλε η u τον λογικό χρόνο t_u και το μήνυμα $request_v$ το έστειλε η v τον λογικό χρόνο t_v .
- Ας υποθέσουμε ότι $t_u < t_v$.
- Άρα για να μπορέσει η v να μπει στο ΚΤ, η ουρά της v θα πρέπει να περιέχει κάποιο μήνυμα από την u με χρονοσφραγίδα μεγαλύτερη του t_v και άρα μεγαλύτερη του t_u .

Ορθότητα του Αλγόριθμου LamportME (2)

- Εφόσον τα κανάλια είναι FIFO για να συμβεί αυτό θα πρέπει η ν να έλαβε το μήνυμα $request_u$ όταν εκτελούσε το ΚΤ.
- Όμως για να μπορέσει να εκτελέσει η ν το ΚΤ θα πρέπει να είχε λάβει το μήνυμα $release_u$.
- Άρα για να συμβεί αυτό, η υ είχε ήδη βγει από το ΚΤ την στιγμή που η ν εκτέλεσε το ΚΤ.
- Όμως υποθέσαμε ότι η υ και η ν βρίσκονται στο ΚΤ την ίδια στιγμή.



Ορθότητα του Αλγόριθμου LamportME (3)

Λήμμα (LamportME.2)

Ο αλγόριθμος LamportME ικανοποιεί την ιδιότητα της βιωσιμότητας.

Απόδειξη: Η ιδιότητα της βιωσιμότητας προκύπτει από την χρήση των λογικών ρολογιών και την εξυπηρέτηση των αιτήσεων με βάση τον λογικό χρόνο των μηνυμάτων `request`.

- Αρκεί να δείξουμε ότι η διεργασία που έστειλε το μήνυμα `request` με την μικρότερη χρονοσφραγίδα είναι αυτή που εισέρχεται πρώτη στο ΚΤ.
- Βασιζόμαστε στο γεγονός ότι τα κανάλια είναι αξιόπιστα και FIFO.
- Εφόσον το σύνολο των μηνυμάτων `request` που έχουν χρονοσφραγίδα μικρότερη από ένα γεγονός είναι πεπερασμένο, με επαγωγικό τρόπο μπορούμε να δείξουμε ότι όλες οι αιτήσεις ικανοποιούνται.

Ορθότητα του Αλγόριθμου LamportME (4)

Λήμμα (LamportME.3)

Ο αλγόριθμος LamportME ικανοποιεί την ιδιότητα της διάταξης.

Απόδειξη: Η ιδιότητα της διάταξης προκύπτει από την χρήση των λογικών ρολογιών και την εξυπηρέτηση των αιτήσεων με βάση τον λογικό χρόνο των μηνυμάτων `request` καθώς και την υπόθεση ότι τα κανάλια είναι αξιόπιστα και FIFO.

Θεώρημα (LamportME.4)

Ο αλγόριθμος LamportME λύνει το πρόβλημα του αμοιβαίου αποκλεισμού.

Απόδειξη: Προκύπτει από τα παραπάνω 3 λήμματα.

Σύνοψη 7^{ης} Διάλεξης

- 1 Προηγούμενο Μάθημα
 - Προηγούμενο Μάθημα
 - Διάταξη Γεγονότων – Λογικός Χρόνος
- 2 Ασύγχρονα Κατανεμημένα Συστήματα
 - Καθολικές Καταστάσεις
 - Κατασκευή Καθολικών Καταστάσεων
 - Συνεπή Ολικά Στιγμιότυπα
- 3 Σύνοψη Μαθήματος
 - Σύνοψη Μαθήματος
 - Βιβλιογραφία
 - Επόμενη Διάλεξη

Γενικά (1)

- Η κατάσταση ενός **κεντριοποιημένου συστήματος** μια δεδομένη στιγμή χαρακτηρίζεται από την κατάσταση όλων των διεργασιών που περιλαμβάνει
 - Έστω n διεργασίες
 - Κάθε διεργασία \mathcal{P}_u την χρονική στιγμή i βρίσκεται στην κατάσταση k_i^u
- Η περιγραφή της κατάστασης του συστήματος, την χρονική στιγμή i απαιτεί την καταγραφή των καταστάσεων όλων των διεργασιών, την χρονική στιγμή i
- Σε ένα κεντριοποιημένο σύστημα η καταγραφή της κατάστασης όλων των διεργασιών είναι απλή:
 - Ταυτόχρονη προσωρινή παύση των διεργασιών
 - Καταγραφή της μνήμης του συστήματος

Γενικά (2)

- Πως μπορούμε να καταγράψουμε την κατάσταση ενός **κατανεμημένου συστήματος;**
 - Η προσωρινή παύση των διεργασιών δεν μπορεί να γίνει ταυτόχρονα
 - Οι διεργασίες εκτελούνται σε διαφορετικές μονάδες
 - Τα μηνύματα ελέγχου δεν παραδίδονται ταυτόχρονα
 - Δεν είναι εύκολο να συντονίσουμε τις διεργασίες σε μια προκαθορισμένη χρονική στιγμή (θέματα συγχρονισμού ρολογιών κλπ.)
- Σε πραγματικές συνθήκες δεν είναι επιθυμητό να γίνει παύση της εκτέλεσης (έστω και προσωρινά) – π.χ. για λόγους απόδοσης, ή δεν είναι εφικτό

Εφαρμογές

Η καταγραφή της κατάσταση ενός **κατανεμημένου συστήματος** εφαρμόζεται:

- Μέτρηση ποσοτήτων που αλλάζουν δυναμικά (π.χ. χρήματα σε τραπεζικό λογαριασμό)
- Αντίγραφα ασφαλείας
- Ανίχνευση τερματισμού
- Ανίχνευση deadlock / livelock
- Αποσφαλμάτωση ενός κατανεμημένου αλγορίθμου ελέγχοντας αν έχει παραβιαστεί μια συνθήκη / ιδιότητα

Ορισμοί – Τοπική ιστορία

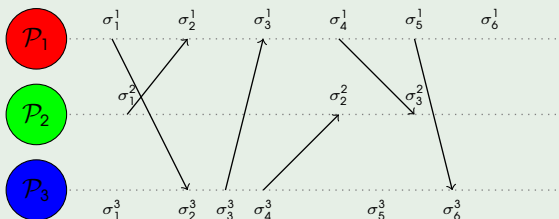
- Ορίζουμε ως συμβάν σ^u μια ενέργεια ϵ που επιφέρει μια αλλαγή στην κατάσταση κάποιας διεργασίας \mathcal{P}_u
- Υποθέτουμε ότι οι διεργασίες εξελίσσονται σειριακά
- Υπάρχει μια αυστηρή διάταξη των συμβάντων $\sigma_1^u, \sigma_2^u, \dots$ κάθε διεργασίας \mathcal{P}_u όπου $\sigma_k^u, \sigma_{k+1}^u$ σημαίνει ότι το σ_k^u συνέβη πριν το σ_{k+1}^u

Τοπική ιστορία (local history)

Η τοπική ιστορία της διεργασίας \mathcal{P}_u συμβολίζεται με h_u και αποτελεί την ακολουθία των συμβάντων που πραγματοποιήθηκαν στην διεργασία, π.χ. $h_u = \sigma_1^u, \sigma_2^u, \sigma_3^u, \sigma_4^u$.

Παράδειγμα – Τοπική ιστορία

Εκτέλεση συστήματος – διάγραμμα μηνυμάτων



- Τοπική ιστορία – $h_1 = \sigma_1^1, \sigma_2^1, \sigma_3^1, \sigma_4^1, \sigma_5^1, \sigma_6^1$
- Τοπική ιστορία – $h_2 = \sigma_1^2, \sigma_2^2, \sigma_3^2$
- Τοπική ιστορία – $h_3 = \sigma_1^3, \sigma_2^3, \sigma_3^3, \sigma_4^3, \sigma_5^3, \sigma_6^3$
- Η τοπική ιστορία κάθε διεργασίας είναι **χρονικά πλήρως διατεταγμένη**

Ορισμοί – Καθολική ιστορία / Καθολική κατάσταση

Καθολική ιστορία (global history)

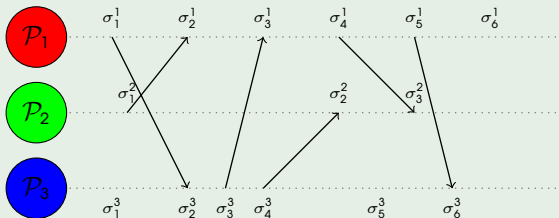
Η *καθολική ιστορία* \mathcal{H} ενός κατανεμημένου συστήματος ορίζεται ως η ένωση των τοπικών ιστοριών όλων των διεργασιών που συμμετέχουν σε αυτό, δηλ. $\mathcal{H} = h_1 \cup \dots \cup h_n$.

Καθολική κατάσταση (global state)

Η *καθολική κατάσταση* ενός κατανεμημένου συστήματος συμβολίζεται με Σ και ορίζεται ως η ένωση των τοπικών καταστάσεων όλων των διεργασιών που συμμετέχουν σε αυτό, δηλ. $\Sigma = \{k^1, \dots, k^n\}$.

Παράδειγμα – Καθολική ιστορία / Καθολική κατάσταση

Εκτέλεση συστήματος – διάγραμμα μηνυμάτων



- Καθολική ιστορία – $\mathcal{H} = h_1 \cup h_2 \cup h_3$
- Το σύνολο αυτό είναι μόνο μερικώς διατεταμένο.
- Καθολική κατάσταση – $\Sigma_1 = \{\kappa_2^1, \kappa_1^2, \kappa_2^3\}$
- Καθολική κατάσταση – $\Sigma_2 = \{\kappa_3^1, \kappa_2^2, \kappa_4^3\}$

Ορισμοί – Τομή / Σύνορο τομής

Τομή (cut)

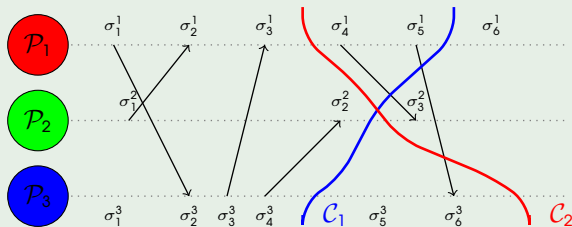
Μια *τομή* \mathcal{C} ενός κατανεμημένου συστήματος είναι ένα υποσύνολο της καθολικής ιστορίας \mathcal{H} που αποτελείται από $\sigma^u \geq 0$ αρχικά συμβάντα από κάθε διεργασία \mathcal{P}_u , δηλ. $\mathcal{C} = h_1^{\sigma^1} \cup \dots \cup h_n^{\sigma^n}$. Επομένως, μια τομή προσδιορίζεται μέσω του διανύσματος $\{\sigma^1, \dots, \sigma^n\}$.

Σύνορο τομής (cut frontier)

Το σύνολο των τελευταίων συμβάντων $\{\max(\sigma^1), \dots, \max(\sigma^n)\}$ που περιλαμβάνονται στην τομή \mathcal{C} καλείται σύνορο της τομής.

Παράδειγμα – Τομή / Σύνορο τομής

Εκτέλεση συστήματος – διάγραμμα μηνυμάτων



- $C_1 = h_1^{\sigma_1^1} \cup h_2^{\sigma_2^2} \cup h_3^{\sigma_3^3} = \{\sigma_1^1, \dots, \sigma_5^1, \sigma_2^2, \sigma_2^2, \sigma_3^3, \dots, \sigma_4^3\}$
- Σύνορο τομής της C_1 είναι το $\{\sigma_5^1, \sigma_2^2, \sigma_4^3\}$
- $C_2 = h_1^{\sigma_1^1} \cup h_2^{\sigma_2^2} \cup h_3^{\sigma_3^3} = \{\sigma_1^1, \dots, \sigma_3^1, \sigma_2^2, \sigma_2^2, \sigma_3^3, \dots, \sigma_6^3\}$
- Σύνορο τομής της C_2 είναι το $\{\sigma_3^1, \sigma_2^2, \sigma_6^3\}$

Συζήτηση

- Κάθε τομή ενός κατανεμημένου συστήματος αντιστοιχεί σε μία καθολική κατάσταση
- Ορισμένες μόνο τομές αντιστοιχούν σε καθολικές καταστάσεις που θα μπορούσαν να συμβούν κατά τη διάρκεια μιας εκτέλεσης
- Στην εκτέλεση του προηγούμενου παραδείγματος, η τομή C_1 είναι μια 'εφικτή' καθολική κατάσταση
- Η τομή C_2 δεν μπορεί να συμβεί εφόσον η διεργασία P_3 λαμβάνει ένα μήνυμα από την P_1 , το οποίο η P_1 δεν έχει στείλει μέσα στα χρονικά όρια που ορίζονται από την τομή

Ορισμός – Συνεπής Τομή

Συνεπής Τομή (consistent cut)

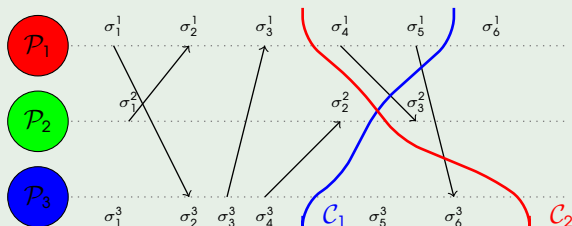
Μια τομή \mathcal{C} είναι συνεπής αν για όλα τα συμβάντα σ και σ' ισχύει ότι:

$$\sigma \in \mathcal{C} \wedge (\sigma' \rightsquigarrow \sigma) \Rightarrow \sigma' \in \mathcal{C}$$

- Μία καθολική κατάσταση είναι **συνεπής** όταν αντιστοιχεί σε μια συνεπή τομή
- Οι συνεπής καθολικές καταστάσεις είναι εκείνες που μπορούν να συμβούν σε μια πραγματική εκτέλεση ενός κατανομημένου συστήματος

Συνεπής Τομή – Συζήτηση

Εκτέλεση συστήματος – διάγραμμα μηνυμάτων



Εύκολος τρόπος να προσδιορίσουμε αν μία τομή είναι συνεπής :

- Όλα τα βέλη που 'κόβουν' την τομή ξεκινούν από τα αριστερά και καταλήγουν στα δεξιά της τομής

C_1 είναι συνεπής – τα βέλη $\sigma_4^1 \rightarrow \sigma_3^3$, $\sigma_5^1 \rightarrow \sigma_6^3$ κόβουν την τομή από αριστερά προς δεξιά

C_2 δεν είναι – το βέλος $\sigma_5^2 \rightarrow \sigma_4^3$ κόβει από δεξιά προς αριστερά

Γενικά

- Μια διεργασία θέλει να μάθει την καθολική κατάσταση του καταναημεμένου συστήματος
 - Ονομάζουμε την διεργασία monitor
- Πρέπει να 'συλλέξει' τις τοπικές καταστάσεις όλων των διεργασιών του συστήματος
- Λόγω της χρονικής αβεβαιότητας (ως προς την εκτέλεση και την παράδοση μηνυμάτων) που χαρακτηρίζει ένα ασύγχρονο καταναημεμένο σύστημα, η κατασκευή καθολικών καταστάσεων δεν είναι καθόλου εύκολη.
- Θεμελιώδες πρόβλημα

Παθητική Κατασκευή Καθολικών Καταστάσεων

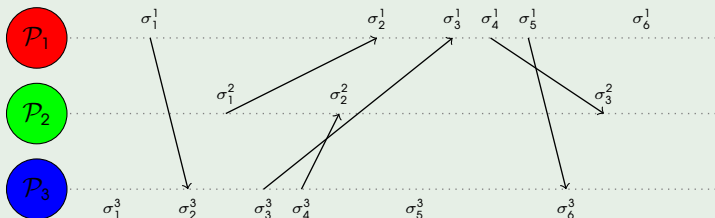
- Έστω ότι η διεργασία \mathcal{P}_0 είναι monitor του συστήματος και θέλει να κατασκευάσει την καθολική κατάσταση του συστήματος
 - Δεν στέλνει κανένα μήνυμα – παθητικά παρακολουθεί το σύστημα
- Οι άλλες διεργασίες μόλις επεξεργαστούν ένα συμβάν, στέλνουν ένα μήνυμα στην \mathcal{P}_0 περιγράφοντας το
- Η \mathcal{P}_0 κατασκευάζει μια παρατήρηση (observation) της συγκεκριμένης εκτέλεσης (run) του κατανεμημένου συστήματος
- Η κατασκευή της παρατήρησης προκύπτει από την ακολουθία των γεγονότων, με την σειρά με την οποία έλαβε η \mathcal{P}_0 τα αντίστοιχα μηνύματα

Ιδιότητες Παρατηρήσεων

- Εξαιτίας των απροσδιόριστων καθυστερήσεων κατά την αποστολή των μηνυμάτων, δυο διαφορετικές διεργασίες monitor μπορεί να κατασκευάσουν δυο διαφορετικές παρατηρήσεις για την ίδια εκτέλεση
- Μια παρατήρηση μπορεί να μην αποτυπώνει σωστά την εκτέλεση

Παράδειγμα Εκτέλεσης

$$\mathcal{R} = \{\sigma_1^3, \sigma_1^1, \sigma_2^3, \sigma_1^2, \sigma_3^3, \sigma_4^3, \sigma_2^2, \sigma_2^1, \sigma_5^3, \sigma_3^1, \sigma_4^1, \sigma_5^1, \sigma_6^3, \sigma_3^2, \sigma_6^1\}$$

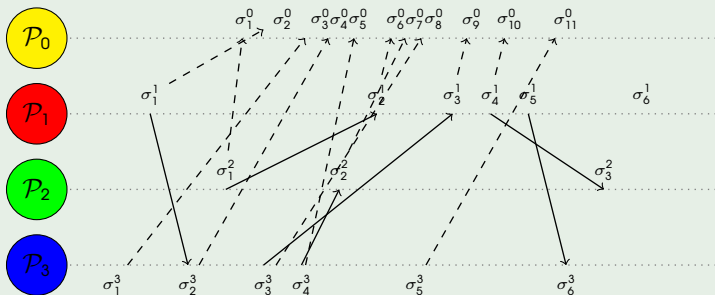


Οι παρακάτω είναι πιθανές παρατηρήσεις της \mathcal{R}

- $\mathcal{O}_1 = \{\sigma_1^2, \sigma_1^1, \sigma_1^3, \sigma_2^3, \sigma_4^3, \sigma_2^1, \sigma_2^2, \sigma_3^3, \sigma_3^1, \sigma_4^1, \sigma_5^3, \dots\}$
- $\mathcal{O}_2 = \{\sigma_1^1, \sigma_1^3, \sigma_1^2, \sigma_2^3, \sigma_2^1, \sigma_3^1, \sigma_3^3, \sigma_4^3, \sigma_2^2, \sigma_5^3, \sigma_6^3, \dots\}$
- $\mathcal{O}_3 = \{\sigma_1^3, \sigma_1^2, \sigma_1^1, \sigma_2^1, \sigma_2^3, \sigma_3^3, \sigma_3^1, \sigma_4^3, \sigma_4^1, \sigma_2^2, \sigma_5^1, \dots\}$

Παράδειγμα Παρατήρησης

$$\mathcal{O}_1 = \{\sigma_1^2, \sigma_1^1, \sigma_1^3, \sigma_2^3, \sigma_4^3, \sigma_2^1, \sigma_2^2, \sigma_3^3, \sigma_3^1, \sigma_4^1, \sigma_5^3, \dots\}$$



- Δεν αντιστοιχεί σε εκτέλεση
- Η διάταξη των συμβάντων της \mathcal{P}_3 παραβιάζει τη διάταξή τους στο τοπικό ιστορικό της διεργασίας
- Το σ_4^3 εμφανίζεται πριν από το σ_3^3

Παθητική Παρατήρηση με Φυσικά Ρολόγια

- 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
- Η διεργασία monitor, τη χρονική στιγμή t καταγράφει όλα τα μηνύματα που έχει λάβει με σφραγίδες χρόνου μέχρι $t - \mu$ με αύξουσα διάταξη σφραγίδων χρόνου
- Οι παρατηρήσεις της διεργασίας monitor μπορούν να χρησιμοποιηθούν για την κατασκευή συνεπών καθολικών καταστάσεων
- Βασισμένο στον σχεδιασμό του Συγχρονιστή των Tel και Leeuwen

Θέματα Συγχρονισμού Ρολογιών

- Ο συγχρονισμός ρολογιών σε ένα καταμεμημένο σύστημα δεν είναι καθόλου εύκολο πρόβλημα
- Η κατασκευή συνεπών καθολικών καταστάσεων μπορεί να βασιστεί και σε `ασθενέστερες` συνθήκες
 - Αρκεί να μπορούμε να εγγυηθούμε την χρονολογική σειρά των γεγονότων
- Χρησιμοποιούμε τον αλγόριθμο LamportTime για τον ορισμό των χρονοσφραγίδων που ικανοποιούν την σχέση `συνέβη-πριν`

Παθητική Παρατήρηση με Λογικά Ρολόγια (1)

- Θεωρούμε ότι οι κόμβοι έχουν πρόσβαση σε ένα **λογικό ρολόι** που διατηρεί μια τοπική διεργασία που εκτελεί τον αλγόριθμο LamportTime
- Η διεργασία monitor, τη χρονική στιγμή t καταγράφει όλα τα μηνύματα που έχει λάβει με αύξουσα διάταξη χρονοσφραγίδων
- Οι παρατηρήσεις της διεργασίας monitor μπορούν να χρησιμοποιηθούν για την κατασκευή συνεπών καθολικών καταστάσεων
- π.χ. $O_4 = \{\sigma_1^1, \sigma_1^2, \sigma_1^3, \sigma_2^1, \sigma_2^3, \sigma_3^3, \sigma_3^1, \sigma_4^3, \sigma_4^1, \sigma_2^2, \sigma_5^3, \dots\}$
 - είναι συνεπής

Παθητική Παρατήρηση με Λογικά Ρολόγια (2)

- Ο παραπάνω αλγόριθμος δεν είναι απολύτως σωστός
- Μπορεί να παραλάβουμε ένα μήνυμα που αφορά το συμβάν σ'' μετά από το μήνυμα που αφορά το συμβάν σ' ενώ $LC(\sigma'') < LC(\sigma')$
- Αυτό συμβαίνει, διότι τα λογικά ρολόγια δεν μπορούν να ανιχνεύσουν το χάσμα (gap detection) μεταξύ χρονοσφραγίδων διαφορετικών διεργασιών

Ανίχνευση Χάσματος

Δεδομένων δύο συμβάντων σ και σ' με χρονοσφραγίδες $LC(\sigma)$ και $LC(\sigma')$ για τις οποίες ισχύει $LC(\sigma) < LC(\sigma')$, αποφάσισε αν υπάρχει κάποιο άλλο συμβάν σ'' τέτοιο ώστε $LC(\sigma) < LC(\sigma'') < LC(\sigma')$

Παθητική Παρατήρηση με Λογικά Ρολόγια (3)

- Υποθέτουμε ότι τα κανάλια επικοινωνίας παραδίδουν τα μηνύματα με σειρά FIFO.
- Τότε, αν η \mathcal{P}_0 παραλάβει ένα μήνυμα m από την διεργασία \mathcal{P}_u με χρονοσφραγίδα $LC(m)$ είναι βέβαιο ότι κανένα άλλο μήνυμα m' δεν μπορεί να ληφθεί από την \mathcal{P}_u με χρονοσφραγίδα $LC(m') < LC(m)$
- Αυτό συμβαίνει, διότι τα λογικά ρολόγια δεν μπορούν να ανιχνεύσουν το χάσμα (gap detection) μεταξύ χρονοσφραγίδων διαφορετικών διεργασιών
 - Το μήνυμα m χαρακτηρίζεται **ευσταθές**
- Επομένως, η διεργασία monitor, τη χρονική στιγμή t καταγράφει όλα τα **ευσταθή** μηνύματα που έχει λάβει με αύξουσα διάταξη χρονοσφραγίδων

Συζήτηση

- Ο αλγόριθμος είναι γνωστός ως *LogicalTimeSnapshot*
- Παρατηρούμε ότι τα φυσικά ρολόγια στερούνται επίσης της ιδιότητας ανίχνευσης χάσματος
- Όμως, λόγω τη υπόθεσης ενός άνω φράγματος $\mu \geq l + d$ το οποίο είναι γνωστό στις διεργασίες, οδηγούμαστε σε μια ισοδύναμη υπόθεση:
 - τη χρονική στιγμή t , όλα τα μηνύματα με χρονοσφραγίδες μικρότερες από $t - \mu$ είναι ευσταθή

Στιγμιότυπα

- Οι δύο αλγόριθμοι που είδαμε βασίζονται στον παθητικό ρόλο της διεργασίας monitor
- Όλες οι άλλες διεργασίες που συμμετείχαν στο κατανεμημένο σύστημα ενημέρωναν την \mathcal{P}_0
- Η προσέγγιση των στιγμιότυπων δίνει ενεργό ρόλο στην διεργασία monitor η οποία καθορίζει πότε θα εκτελεστεί η διαδικασία της σύνθεσης της καθολικής κατάστασης
- Με άλλα λόγια η \mathcal{P}_0 λαμβάνει 'φωτογραφίες' του συστήματος, τις οποίες αποκαλούμε *στιγμιότυπα* (snapshots)

Ορισμοί

Κατάσταση καναλιού (channel state)

Η κατάσταση κάθε καναλιού C_{uv} που συνδέει την διεργασία \mathcal{P}_u με την διεργασία \mathcal{P}_v , είναι όλα τα μηνύματα που έχει στείλει η \mathcal{P}_u στην διεργασία \mathcal{P}_v , και η \mathcal{P}_v δεν έχει ακόμη λάβει.

- Συμβολίζουμε ως $nbrs_u^{in} = \{v | (v, u) \in E\}$ όλες τις κορυφές που είναι εισερχόμενες γειτονικές της u
- Συμβολίζουμε ως $nbrs_u^{out} = \{v | (u, v) \in E\}$ όλες τις κορυφές που είναι εξερχόμενες γειτονικές της κορυφής u

Συνεπή Ολικά Στιγμιότυπα με Φυσικά Ρολόγια (1)

- 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
- Όπως και πριν, το πρωτόκολλο βασίζεται στο ότι όλες οι διεργασίες καταγράφουν την καθολική τους κατάσταση στον ίδιο πραγματικό χρόνο
- Η διεργασία monitor, επιλέγει μια χρονική στιγμή t_* αρκετά μεγάλη, έτσι ώστε να εγγυηθεί ότι ένα μήνυμα που εστάλη τώρα θα έχει ληφθεί από όλες τις υπόλοιπες διεργασίες πριν την t_*

Συνεπή Ολικά Στιγμιότυπα με Φυσικά Ρολόγια (2)

- Αρχικά, η διεργασία \mathcal{P}_0 στέλνει το μήνυμα *ΠάρεΣτιγμιότυπο(t_*)* σε όλες τις διεργασίες.
- **Την στιγμή t_*** κάθε διεργασία \mathcal{P}_U
 - Καταγράφει την τοπική της κατάσταση σ_U
 - Στέλνει ένα κενό μήνυμα στις $nbrs_U^{out}$
 - Θέτει κάθε σύνολο *κατάσταση(C_{vU})* με το κενό σύνολο
 - Καταγράφει τα μηνύματα από τις $nbrs_U^{in}$
- Όταν η \mathcal{P}_U λάβει από την \mathcal{P}_V μήνυμα με *χρονοσφραγίδα(m)* $> t_*$
 - Σταματάει την καταγραφή των μηνυμάτων που δέχεται από την \mathcal{P}_V
 - Δηλώνει στην \mathcal{P}_0 την *κατάσταση(C_{vU})*

Συζήτηση – Παρατηρήσεις (1)

- Για κάθε $\mathcal{P}_v \in nbrs_U^{in}$ η κατάσταση του καναλιού C_{vu} περιέχει
 - το σύνολο των μηνυμάτων που έστειλε η \mathcal{P}_v πριν από τη χρονική στιγμή t_* και έλαβε η \mathcal{P}_u μετά από τη χρονική στιγμή t_*
 - Δηλαδή, όλα τα μηνύματα τα οποία την χρονική στιγμή t_* είναι σε μεταφορά
- Τα κενά μηνύματα εγγυώνται ότι η διεργασία \mathcal{P}_u θα λάβει τελικά ένα μήνυμα m για το οποίο $χρονοσφραγίδα(m) \geq t_*$

Συζήτηση – Παρατηρήσεις (2)

- Έστω ένα συμβάν σ που ανήκει στην τομή \mathcal{C}_* , η οποία σχετίζεται με την καθολική κατάσταση που έχει δημιουργηθεί, τότε $\text{χρονοσφραγίδα}(\sigma) < t_*$
- Άρα,

$$(\sigma \in \mathcal{C}_*) \wedge (\text{χρονοσφραγίδα}(\sigma') < \text{χρονοσφραγίδα}(\sigma)) \Rightarrow \sigma' \in \mathcal{C}_*$$

- Εφόσον τα φυσικά ρολόγια ικανοποιούν τη συνθήκη του ρολογιού, η παραπάνω σχέση αποδεικνύει ότι η τομή \mathcal{C}_* είναι συνεπής, άρα και η καθολική κατάσταση είναι συνεπής

Συνεπή Ολικά Στιγμιότυπα με Λογικά Ρολόγια (1)

- Εφόσον τα λογικά ρολόγια ικανοποιούν τη συνθήκη του ρολογιού, θα ήταν καλύτερο να χρησιμοποιήσουμε λογικά ρολόγια
- Όμως πως θα ορίσουμε την χρονική στιγμή t_* με την χρήση λογικών ρολογιών;
- Επίσης, στο προηγούμενο πρωτόκολλο υποθέσαμε ότι η \mathcal{P}_0 μπορεί να υπολογίσει το t_*
- Τώρα υποθέτουμε ότι η \mathcal{P}_0 μπορεί να υπολογίσει μια αντίστοιχη τιμή για το λογικό ρολόι ω_* αρκετά μεγάλη, ώστε κανένα λογικό ρολόι να μπορεί να φτάσει την τιμή αυτήν
 - Ασθενέστερη υπόθεση

Συνεπή Ολικά Στιγμιότυπα με Λογικά Ρολόγια (2)

- Αρχικά, η διεργασία \mathcal{P}_0 στέλνει το μήνυμα *ΠάρεΣτιγμιότυπο*(ω_*) σε όλες τις διεργασίες και θέτει το λογικό ρολόι της στην τιμή ω_*
- **Την στιγμή** ω_* κάθε διεργασία \mathcal{P}_u
 - Καταγράφει την τοπική της κατάσταση σ_u
 - Στέλνει ένα κενό μήνυμα στις $nbrs_u^{out}$
 - Αρχίζει να καταγράφει τα μηνύματα από τις $nbrs_u^{in}$
- Όταν η \mathcal{P}_u λάβει από την \mathcal{P}_v μήνυμα με *χρονοσφραγίδα*(m) $\geq \omega_*$
 - Σταματάει την καταγραφή των μηνυμάτων που δέχεται από την \mathcal{P}_v
 - Δηλώνει στην \mathcal{P}_0 την κατάσταση (C_{vu})

Ο Αλγόριθμος των Chandy και Lamport (1)

- Οι Chandy και Lamport παρατήρησαν ότι η διεργασία δεν κάνει τίποτα μεταξύ της λήψης του μηνύματος *ΠάρεΣτιγμιότυπο*(ω_*) και της λήψης ενός κενού μηνύματος από μια άλλη διεργασία
- Επομένως το λογικό ρολόι της διεργασίας εξαναγκάζεται να πάρει την τιμή ω_*
- Μπορούμε να αντικαταστήσουμε το μήνυμα *ΠάρεΣτιγμιότυπο*(ω_*) με ένα απλό μήνυμα *ΠάρεΣτιγμιότυπο*
 - η διεργασία καταγράφει την τοπική της κατάσταση μόλις λάβει το μήνυμα *ΠάρεΣτιγμιότυπο*
- Με αυτή την σκέψη οι Chandy και Lamport προτείνουν έναν αλγόριθμο που δεν χρησιμοποιεί λογικά ρολόγια

Ο Αλγόριθμος των Chandy και Lamport (2)

- Αρχικά, η διεργασία \mathcal{P}_0 στέλνει το μήνυμα *ΠάρεΣτιγμιότυπο* στον εαυτό της
- Όταν η διεργασία \mathcal{P}_u λάβει το μήνυμα *ΠάρεΣτιγμιότυπο* από την διεργασία \mathcal{P}_π για πρώτη φορά
 - Καταγράφει την τοπική της κατάσταση σ_u
 - Στέλνει το μήνυμα *ΠάρεΣτιγμιότυπο* στις $nbrs_u^{out}$
 - Θέτει κάθε σύνολο κατάσταση(C_{π_u}) με το κενό σύνολο
 - Αρχίζει να καταγράφει τα μηνύματα από τις $nbrs_u^{in}$ εκτός από την \mathcal{P}_π
- Όταν η \mathcal{P}_u λάβει ξανά από την \mathcal{P}_δ το μήνυμα *ΠάρεΣτιγμιότυπο*
 - Σταματάει την καταγραφή των μηνυμάτων που δέχεται από την \mathcal{P}_δ
 - Δηλώνει στην \mathcal{P}_0 την κατάσταση(C_{δ_u})

Συζήτηση – Παρατηρήσεις

- Το μήνυμα *ΠάρεΣτιγμιότυπο* μεταβιβάζεται στα εξερχόμενα κανάλια μια διεργασίας, μόλις η διεργασία αυτή λάβει το μήνυμα για πρώτη φορά
 - Αν το σύστημα είναι ισχυρά συνεκτικό, τότε είναι βέβαιο ότι το μήνυμα *ΠάρεΣτιγμιότυπο* θα διασχίσει κάθε κανάλι ακριβώς μία φορά
- Όταν μια διεργασία δέχεται ένα μήνυμα *ΠάρεΣτιγμιότυπο* από όλα τα εισερχόμενα κανάλια της, η συνεισφορά της στην κατασκευή της καθολικής κατάστασης έχει ολοκληρωθεί
 - Τερματίζει η συμμετοχή της στο πρωτόκολλο στιγμιότυπου

Ορθότητα του Αλγόριθμου των Chandy και Lamport (1)

Θεώρημα (ChandyLamportSnapshot.1)

Ο αλγόριθμος *ChandyLamportSnapshot* καταχωρεί ένα συνεπές ολικό στιγμιότυπο για τον A .

Απόδειξη: Έστω η εκτέλεση α της διεργασίας του υψηλότερου επίπεδου A

- Έστω ότι κατά την διάρκεια της εκτέλεσης, στην κατάσταση Σ_ϵ ενεργοποιήθηκε ο αλγόριθμος *ChandyLamportSnapshot*, ο οποίος τερμάτισε στην κατάσταση Σ_τ και κατέγραψε την κατάσταση Σ_*
- Έστω α_1 το μέρος της α πριν από την κατάσταση Σ_ϵ
- Έστω α_2 το μέρος της α μετά από την κατάσταση Σ_τ

Ορθότητα του Αλγόριθμου των Chandy και Lamport (2)

- Το ολικό σιγμιότυπο Σ_* είναι **συνεπές** αν υπάρχει εκτέλεση α' τέτοια ώστε
 - καμία διεργασία δεν μπορεί να ξεχωρίσει την α από την α'
 - Η α' ξεκινά με α_1 και καταλήγει με α_2
 - Τα $\Sigma_\epsilon, \Sigma_*, \Sigma_T$ εμφανίζονται με αυτή την σειρά στην α'
- Ο στόχος μας είναι να επαναδιατάξουμε τα συμβάντα της α έτσι ώστε να καταλήξουμε σε μια εκτέλεση α' στην οποία τα $\Sigma_\epsilon, \Sigma_*, \Sigma_T$ εμφανίζονται με την ίδια σειρά.
- Στην ουσία επιλέγουμε λογικά ανεξάρτητα συμβάντα που πραγματοποιούνται σε διαφορετικές διεργασίες και τα επαναδιατάσσουμε.

Ορθότητα του Αλγόριθμου των Chandy και Lamport (3)

- Έστω σ_k και σ_{k+1} διαδοχικά συμβάντα στην α τα οποία έλαβαν χώρα στις διεργασίες \mathcal{P}_u και \mathcal{P}_v και είναι μετά και προ καταχώρισης αντίστοιχα
- Άρα, δεν μπορεί το σ_k να είναι η αποστολή ενός μηνύματος m και το σ_{k+1} να είναι η παραλαβή του m
 - Όταν η \mathcal{P}_u καταχώρησε την κατάσταση της έστειλε αμέσως το μήνυμα *P'areStigmi'oturo* στη \mathcal{P}_v
 - Εφόσον τα κανάλια είναι FIFO το μήνυμα έφτασε στην \mathcal{P}_v πριν το m γιατί το σ_{k+1} θα ήταν μετά την καταχώρηση, άτοπο.
- Επιπλέον, η κατάσταση της \mathcal{P}_v μετά το σ_{k+1} δεν επηρεάζεται από το σ_k γιατί αυτό συμβαίνει σε άλλη διεργασία
- Επίσης, η κατάσταση της \mathcal{P}_u μετά το σ_k δεν επηρεάζεται από το σ_{k+1}
- Επομένως, μπορούμε να εναλλάξουμε τα σ_k και σ_{k+1}

Ορθότητα του Αλγόριθμου των Chandy και Lamport (4)

- Με συνεχείς τέτοιες εναλλαγές παίρνουμε μια εκτέλεση α' όπου όλα τα συμβάντα προ καταχώρισης προηγούνται όλων των συμβάντων μετά καταχώρισης
- Η α' ξεκινά με α_1 και καταλήγει με α_2
- Η Σ_* εμφανίζεται στην α' αμέσως πριν την α_2
- Όλες οι εναλλαγές που έγιναν αφορούσαν συμβάντα μετά την Σ_ϵ και πριν την Σ_τ
- Το Σ_* είναι η κατάσταση του δικτύου μετά το τελευταίο προ καταχώρισης συμβάν στην εκτέλεση α' και πριν το πρώτο μετά καταχώρισης συμβάν
- Με αυτό τον τρόπο καταλήγουμε στην εκτέλεση α' όπου καμία διεργασία δεν μπορεί να ξεχωρίσει την α από την α' .

Χαρακτηριστικά του Αλγόριθμου των Chandy και Lamport

- Ο αλγόριθμος είναι σωστός – κατασκευάζει συνεπή ολικά στιγμιότυπα
- Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(|E|)$
- Η χρονική πολυπλοκότητα δεν είναι εύκολο να υπολογιστεί διότι ταυτόχρονα στέλνει μηνύματα και η διεργασία υψηλότερου επιπέδου
- Αν αγνοήσουμε πιθανές καθυστερήσεις που προκύπτουν από τις αποστολές των μηνυμάτων της διεργασίας του υψηλότερου επιπέδου, ο αλγόριθμος *ChandyLamportSnapshot* θα τερματίσει εντός χρόνου $\mathcal{O}(\delta(1+d))$

Σύνοψη 7^{ης} Διάλεξης

- 1 Προηγούμενο Μάθημα
 - Προηγούμενο Μάθημα
 - Διάταξη Γεγονότων – Λογικός Χρόνος
- 2 Ασύγχρονα Κατανεμημένα Συστήματα
 - Καθολικές Καταστάσεις
 - Κατασκευή Καθολικών Καταστάσεων
 - Συνεπή Ολικά Στιγμιότυπα
- 3 Σύνοψη Μαθήματος
 - Σύνοψη Μαθήματος
 - Βιβλιογραφία
 - Επόμενη Διάλεξη

Σύνοψη Μαθήματος

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

Βιβλιογραφία

- Βιβλίο "Distributed Algorithms" (N.Lynch)
 - ① Κεφάλαιο 19: Global Snapshots and Stable Properties
- Βιβλίο "Introduction to Distributed Algorithms" (G.Tel)
 - ① Κεφάλαιο 10: Snapshots
- Βιβλίο 'Κατανεμημένα Συστήματα με Java' (I.K.Κάβουρας, I.Z.Μήλης, Γ.Β.Ξυλωμένος, Α.Α.Ρουκουνάκη)
 - ① Κεφάλαιο 4: Καθολικές καταστάσεις

Επόμενη Διάλεξη

- Ασύγχρονα Κατανεμημένα Συστήματα
- Αποτίμηση καθολικού κατηγορήματος
- Ιδιότητες καθολικών κατηγορημάτων, αδιέξοδα, κατανεμημένος τερματισμός