Συστήματα Παράλληλης & Κατανεμημένης Επεξεργασίας

Ενότητα 11: Συγχρονισμός Ρολογιού

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων
http://arch.ece.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σκοπός της Ενότητας

- Η κατανόηση του προβλήματος του συγχρονισμού ρολογιού στα παράλληλα συστήματα και η επίλυσή του.

Προβλήματα σε Κατανεμημένα Συστήματα

- **Διάταξη χρόνου και Συγχρονισμός ρολογιού.**

- Επιλογή αρχηγού.
- Αμοιβαίος αποκλεισμός.
- Κατανεμημένες συναλλαγές.
- Ανακάλυψη αδιέξοδου.

Το πρόβλημα ξεκινάει από το γεγονός ότι τα ρολόγια δεν είναι μεγάλης ακρίβειας

- Κάθε κόμβος έχει ένα δικό του εσωτερικό ρολόι.
- Τα ρολόγια των υπολογιστών δεν είναι μεγάλης ακρίβειας, οπότε με τον καιρό υπάρχει απόκλιση.
- Η χρονική απόκλιση προκαλεί μεγάλα προβλήματα σε κόμβους!!!

Ο συγχρονισμός ρολογιού είναι ένα σημαντικό πρόβλημα σε ΚΣ

- Ο χρόνος είναι μη-διφορούμενος σε **συγκεντρωτικά** συστήματα:
  - Το ρολόι του συστήματος κρατάει χρόνο, όλες οι οντότητες χρησιμοποιούν αυτόν για χρόνο.
  - Μια διεργασία θα κάνει μια κλήση συστήματος και ο πυρήνας θα του “πει” τον χρόνο.
  - Εάν μια διεργασία Α ζητήσει κάποιο χρόνο, και μετά από λίγο ζητήσει και μια διεργασία Β κάποιο χρόνο, η τιμή που θα πάρει η Β θα είναι μεγαλύτερη από ή πιθανόν και ίση με την τιμή που πήρε η Α.
- **Κατανεμημένα συστήματα:** κάθε κόμβος έχει το δικό του ρολόι συστήματος:
  - Τα ρολόγια βασισμένα σε κρυστάλλους είναι λιγότερο ακριβή (1 part in million).
  - *Πρόβλημα:* Σε ένα γεγονός που έγινε μετά από ένα άλλο μπορεί να του αποδοθεί ένας περασμένος χρόνος.

Ο συγχρονισμός των διεργασιών είναι ένα σημαντικό πρόβλημα στα ΚΛΣ

- Σε ένα σύστημα με έναν μοναδικό επεξεργαστή, οι κρίσιμες περιοχές, ο αμοιβαίος αποκλεισμός και άλλα προβλήματα συγχρονισμού είναι γενικώς επιλύσημα με τη χρησιμοποίηση μεθόδων όπως σημαφόρους και monitors και λαμβάνοντας πολύ υπόψη τη κοινή μνήμη.

- Μη αληθές για τα κατανεμημένα συστήματα. Ακόμη και το πιο **απλό πράγμα** όπως η απόφαση εάν το γεγονός Α έγινε πριν ή μετά το γεγονός Β απαιτεί προσεκτική σκέψη.

Χαρακτηριστικά πολυ-υπολογιστών

- Γενικώς, τα κατανεμημένα συστήματα έχουν τις ακόλουθες **ιδιότητες:**
  - Οι σχετικές πληροφορίες διασκορπίζονται σε πολλαπλά μηχανήματα.
  - Οι διεργασίες παίρνουν αποφάσεις βασισμένες σε τοπικές πληροφορίες.
  - Ένα μοναδικό σημείο αποτυχίας σε ένα σύστημα πρέπει να αποφεύγεται.
  - **Δεν υπάρχει κοινό ρολόι ή άλλη πηγή παγκόσμιας ακριβής ώρας.**

Συγχρονισμός Ρολογιών

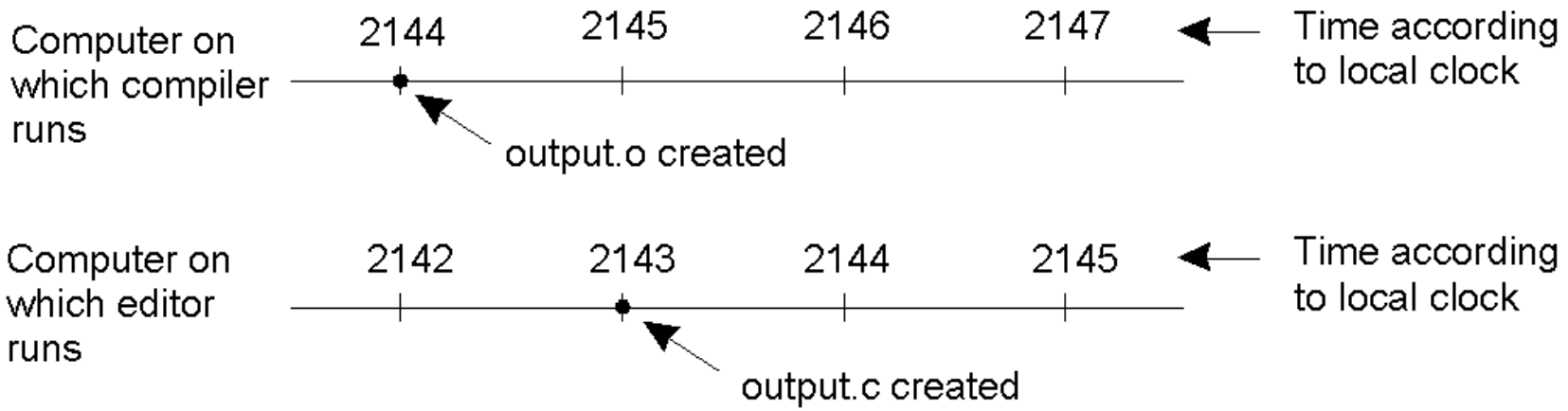
- Η πλειοψηφία των εργασιών που εκτελούνται σε κατανεμημένα συστήματα απαιτούν ένα είδος συγχρονισμού
- Ο συγχρονισμός ρολογιών είναι μία πολύ βασική λειτουργία και μπορεί να βοηθήσει σε κάτι τέτοιο

Είναι πολύ σημαντικό να υπάρχει ένας μηχανισμός για τον συγχρονισμό των ρολογιών ο οποίος να εκτελείται στο παρασκήνιο και **ο οποίος να είναι ανεκτικός σε σφάλματα** (fault tolerant)

Γιατί χρειάζεται συγχρονισμός;

- Προσωρινή διάταξη των γεγονότων που παράγονται από περιστασιακές διεργασίες.
- Συγχρονισμός μεταξύ αποστολέων και παραληπτών μηνυμάτων.
- **Συντονισμός (coordination)** κοινής δραστηριότητας.
- **Σειριοποίηση (serialization)** της περισταστιακής πρόσβασης για κοινά αντικείμενα.

Είναι σημαντικό όλοι οι κόμβοι να συμφωνούν ως προς την ώρα



Αν το output.o έχει μεγαλύτερη ώρα δημιουργίας από το Output.c τότε δε θα γίνει compile το output.c (ακόμη και αν έχει αλλάξει).

Φυσικά ρολόγια....

- **Υπόθεση:** Όλοι οι κόμβοι έχουν ένα φυσικό ρολόι υλικού (hardware) ή καλύτερα χρονιστή (timer).

Όλοι οι κόμβοι έχουν ένα «φυσικό» ρολόι (=hardware clock)

- Τα ακριβής ρολόγια είναι **ατομικοί ταλαντωτές** (one part in 10^{13}).
- Τα περισσότερα ρολόγια είναι λιγότερο ακριβής (π.χ. Τα μηχανικά ρολόγια).
  - Οι υπολογιστές χρησιμοποιούν ρολόγια βασισμένα σε κρυστάλλους (one part in million).
  - Καταλήγει σε απόκλιση ρολογιού (clock drift).
- **Πως μας λέει την ώρα;**
  - Χρησιμοποιεί αστρονομικό σύστημα μέτρησης (ηλιακή μέρα).
- ***Συντονισμένη παγκόσμια ώρα (UTC) – παγκοσμίως καθιερωμένη βασιζόμενη στην ατομική ώρα:***
  - Προσθέτει τα παλμικά δευτερόλεπτα για να είναι σύμφωνο με την αστρονομική ώρα.
  - UTC αναμεταδίδει στο ραδιόφωνο (δορυφόρο και γη).
  - Παραλήπτες ακριβής από 0.1 – 10 ms.
- **Πρέπει να συγχρονίζει τις μηχανές με μία κύρια ή την κάθε μία με την άλλη.**

2 ειδών ρολόγια:
(α) Φυσικά, (β) Λογικά

- **Λογικά ρολόγια** παρακολουθεί την διάταξη των γεγονότων:
  - => ανάμεσα σε αιτιατά (causal) γεγονότα.

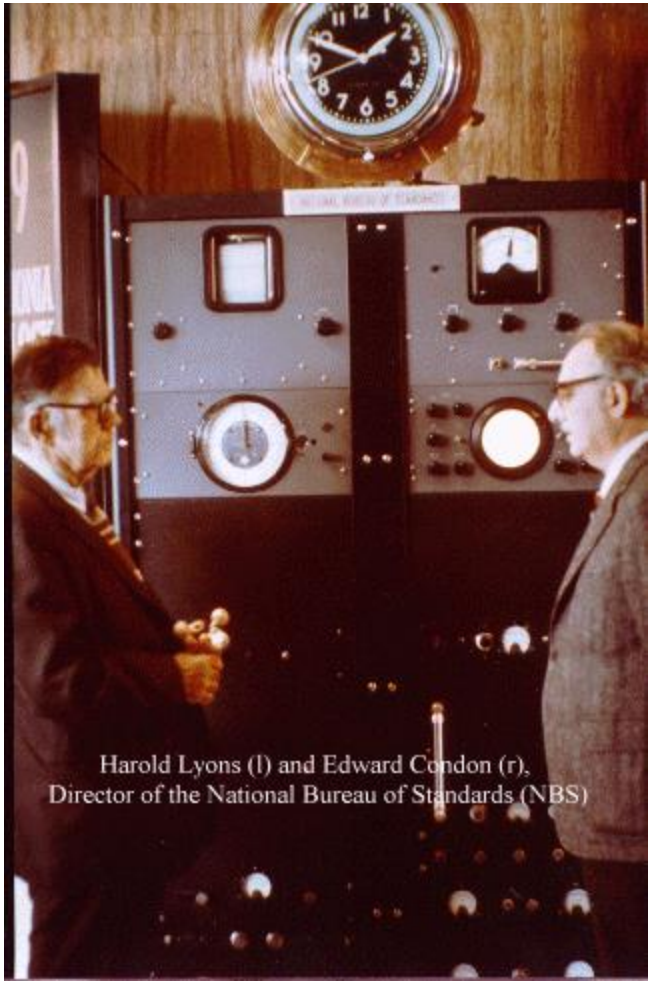
- **Φυσικά ρολόγια:** κρατάνε την ώρα της ημέρας:
  - => συμβατά με όλα τα συστήματα.

Ατομικά Ρολόγια

- Ένα δευτερόλεπτο ορίζεται ως 9,192,631,770 περίοδοι ακτινοβολίας που ανταποκρίνονται στη μετάβαση μεταξύ δύο hyperfine επιπέδων cesium-133.
- Ακρίβεια: καλύτερη από 1 δευτερόλεπτο σε έξι εκατομμύρια χρόνια.
- NIST standard since 1960.

Ατομικοί Ταλαντωτές

- Όλα τα άτομα ενός συγκεκριμένου στοιχείου είναι παρόμοια, θα πρέπει να παράγουν ακριβώς την ίδια συχνότητα.
- Οι ταλαντώσεις είναι για να μετράνε το χρονικό διάστημα.
- 1949 @ NIST ο πρώτος ατομικός ταλαντωτής.
- Χρησιμοποιείται το στοιχείο Rubidium.



Φυσικά ρολόγια σε υπολογιστές

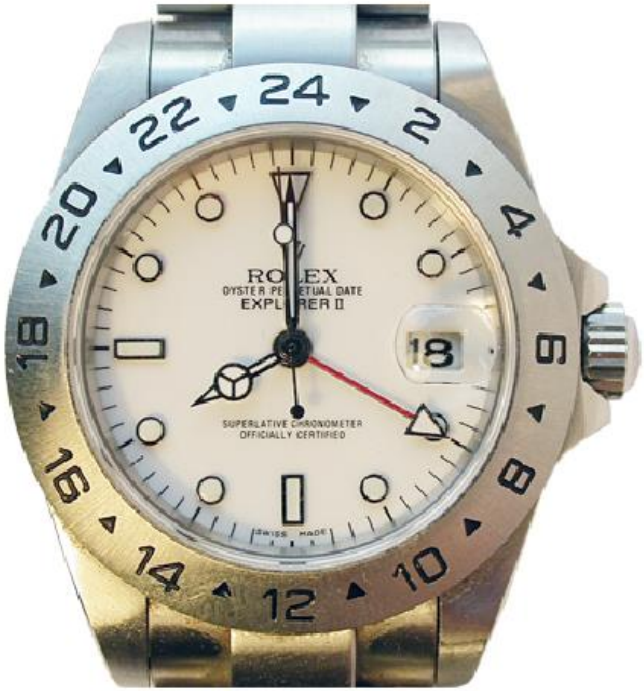
- Real-time Clock: CMOS clock (counter) κύκλωμα οδηγούμενο από έναν ταλαντωτή quartz.
- Εφεδρική μπαταρία για να συνεχίσει να μετράει την ώρα όταν ο υπολογιστής είναι κλειστός.
- Το λειτουργικό σύστημα γενικά προγραμματίζει ένα κύκλωμα χρονιστή να παράγει μια διακοπή περιοδικά.
- π.χ, 60, 100, 250, 1000 διακοπές το δευτερόλεπτο.
- (Linux 2.6+ adjustable up to 1000 Hz)
- Programmable Interval Timer (PIT) – Intel 8253, 8254
- Η διαδικασία της υπηρεσίας διακοπής προσθέτει 1 σε έναν μετρητή στη μνήμη.

Η ώρα συνδέεται με τα εξής προβλήματα

- Παίρνοντας δύο συστήματα για να δούμε αν συμφωνούν στην ώρα:
- Δύο ρολόγια δύσκολα συμβαδίζουν.
- Οι ταλαντωτές Quartz ταλαντώνονται σε ελαφρώς διαφορετικές συχνότητες.
  - ==>Τα ρολόγια χτυπάνε σε διαφορετικούς ρυθμούς.
  - ==>Δημιουργούν όλο και μεγαλύτερο κενό στον αντιλαμβανόμενο χρόνο.

- **Clock Drift (απόκλιση):**
- ==> Η διαφορά μεταξύ σε δύο ρολόγια σε ένα συγκεκριμένο σημείο της ώρας.
- **Clock Skew (λόξωση).**

Δύο ρολόγια (1/2)



8:00:00

8:00:00

Sept 18, 2006
8:00:00

Δύο ρολόγια (2/2)

8:01:24

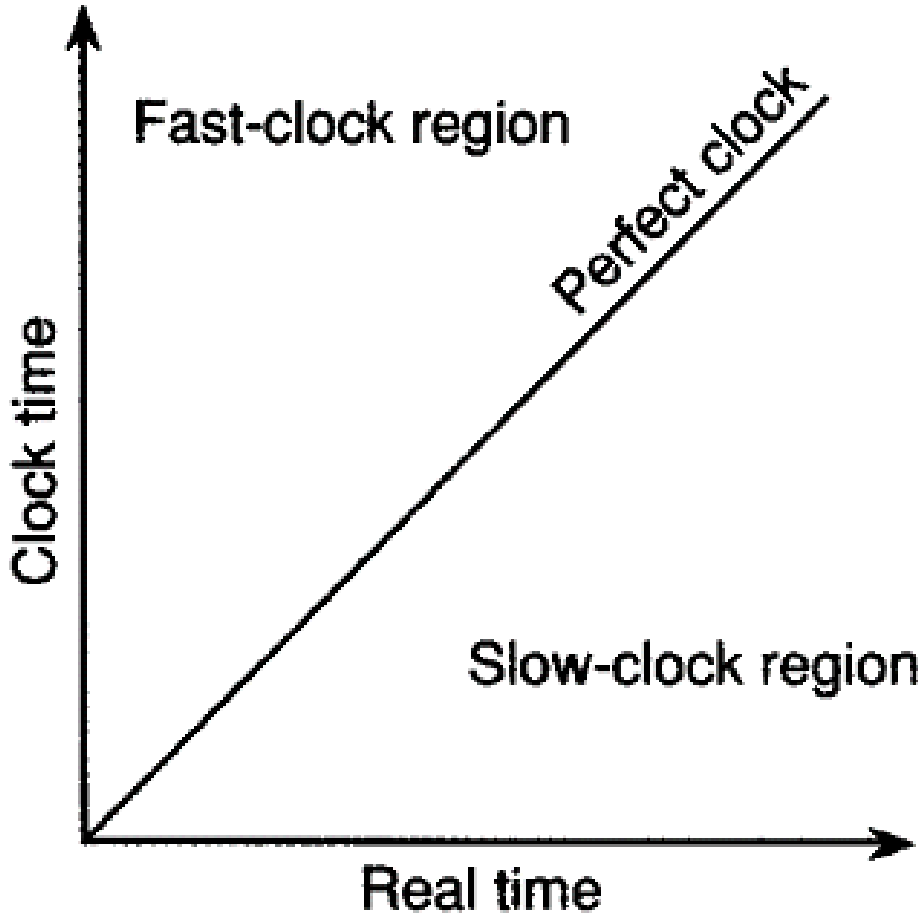
Skew = +84 seconds
+84 seconds/35 days
Drift = +2.4 sec/day

Oct 23, 2006
8:00:00

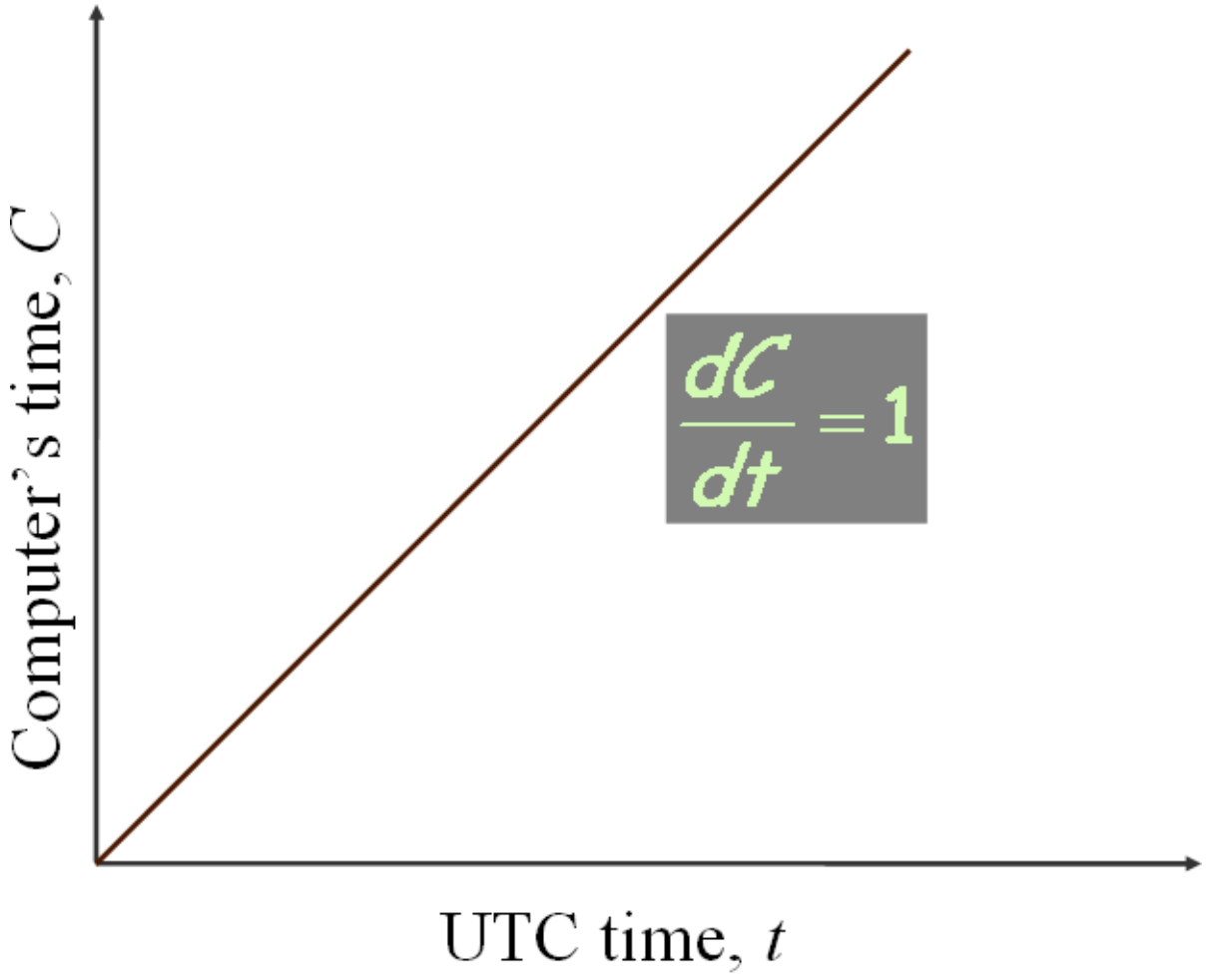
8:01:48

Skew = +108 seconds
+108 seconds/35 days
Drift = +3.1 sec/day

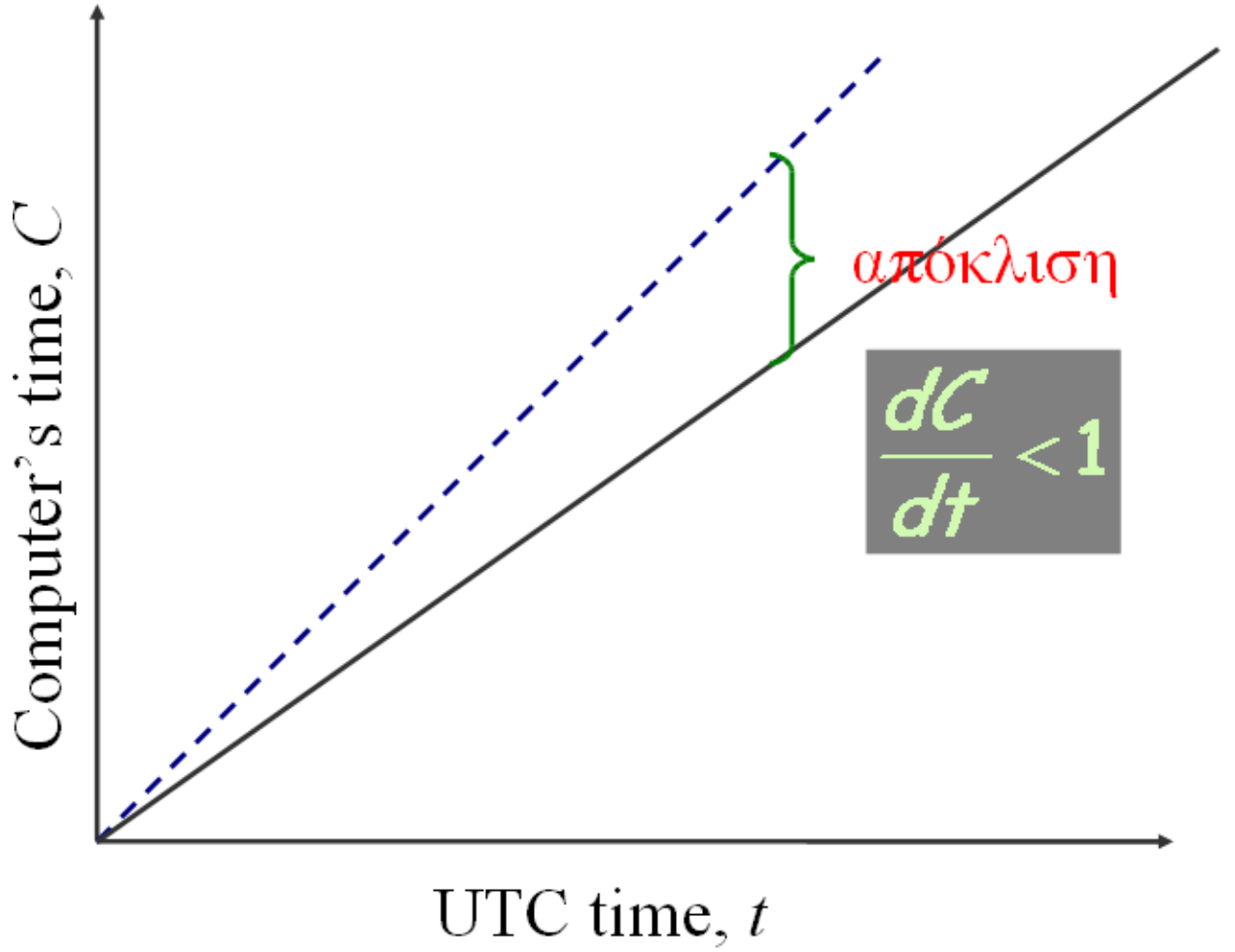
Η σχέση του χρονιστή του κόμβου (=ρολόι) ως προς την πραγματική ώρα είναι μονοτονική



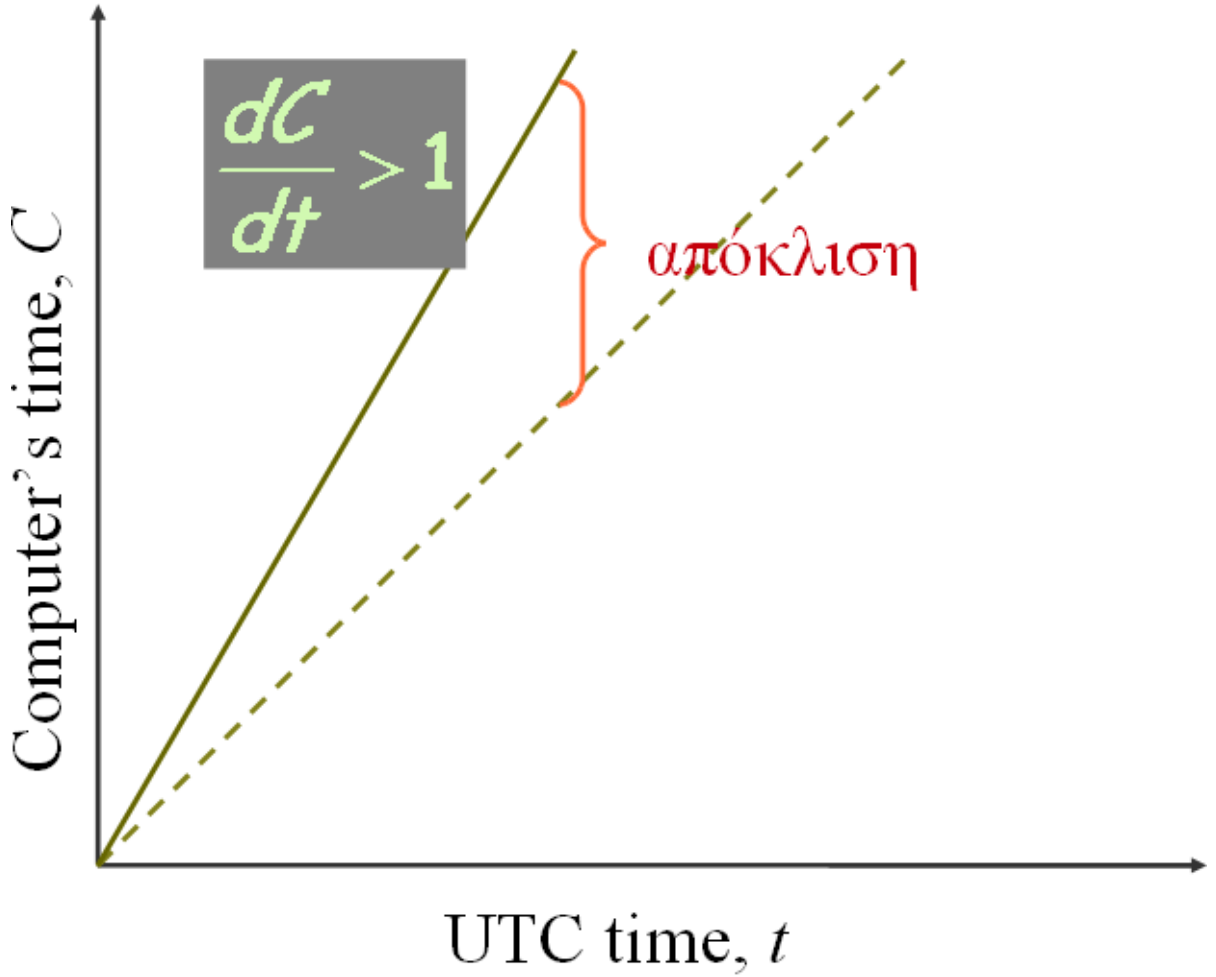
Το τέλειο ρολόι



Λόξωση σε αργό ρολόι

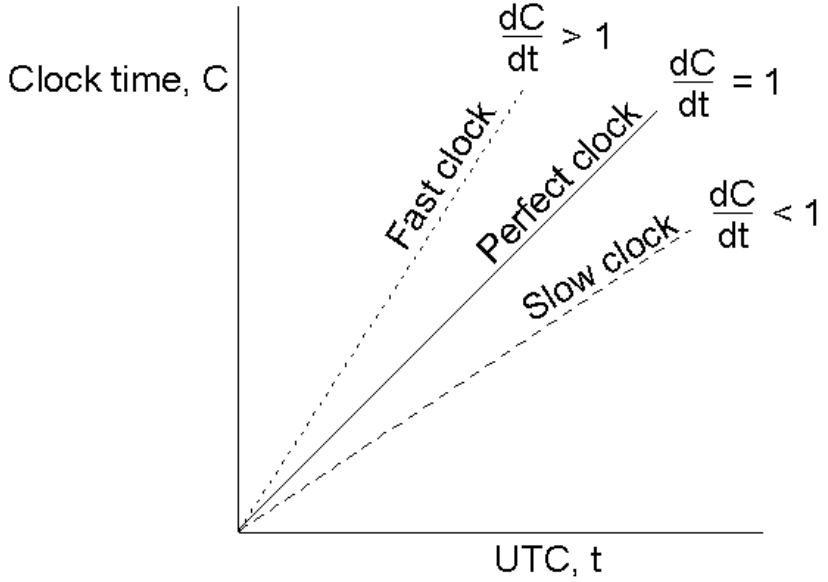


Λόξωση σε γρήγορο ρολόι



Η μέγιστη λόξωση δύο ρολογιών

- Κάθε ρολόι έχει ένα μέγιστο ποσοστό λόξωσης ρ όπου, 1-ρ<= dC/dt <= 1+ρ.
- Δύο ρολόγια μπορούν να έχουν λόξωση της τάξεως του 2ρ Δt σε χρόνο Δt.
- Για να οριοθετήσουμε τη λόξωση σε δ=> επανασυγχρονίζουμε κάθε δ/2ρ δευτερόλεπτα.



Πιθανές λύσεις για το συγχρονισμό ρολογιών

- **Κεντρικοποιημένοι αλγόριθμοι**: Ένας κεντρικός υπολογιστής υπαγορεύει την ώρα.
  - Αλγόριθμος christian.
  - Αλγόριθμος berkeley.
- **Κατανεμημένοι αλγόριθμοι**: Στρωματοποιημένοι client-server αλγόριθμοι.
  - NTP.
  - Lamport Χρονοσφραγίδα (Timestamp).
  - Διανυσματικά Ρολόγια (Vector Clocks).

Επιτρέπεται ο συγχρονισμός της ώρας με επαναφορά;

- Πρέπει να αποφεύγεται να ρυθμίζεται η ώρα προς ώρα που έχει περάσει από τον τρέχων κόμβο.
- Κάτι τέτοιο παραβιάζει την ιδιότητα της μονοτονικής ώρας που εννοείται σχεδόν σε όλες τις εφαρμογές (π.χ. Make).
- Η προφανής λύση να σταματήσουμε την ώρα μέχρι να έρθει η κατάλληλη στιγμή απορρίπτεται.
  - **Η ενδεδειγμένη λύση:** Αργή λόξωση για μεγάλο χρονικό διάστημα.

Αντιμετωπίζοντας τη Λόξωση (1/2)

- Θεωρούμε ότι ορίζουμε τον υπολογιστή με πραγματική ώρα.
- **Καθόλου καλή ιδέα** να γυρίσουμε το ρολόι πίσω.
- Η ψευδαίσθηση του χρόνου που γυρνάει πίσω μπορεί να μπερδέψει τη διάταξη των μηνυμάτων και τα λογισμικά περιβάλλοντα ανάπτυξης.

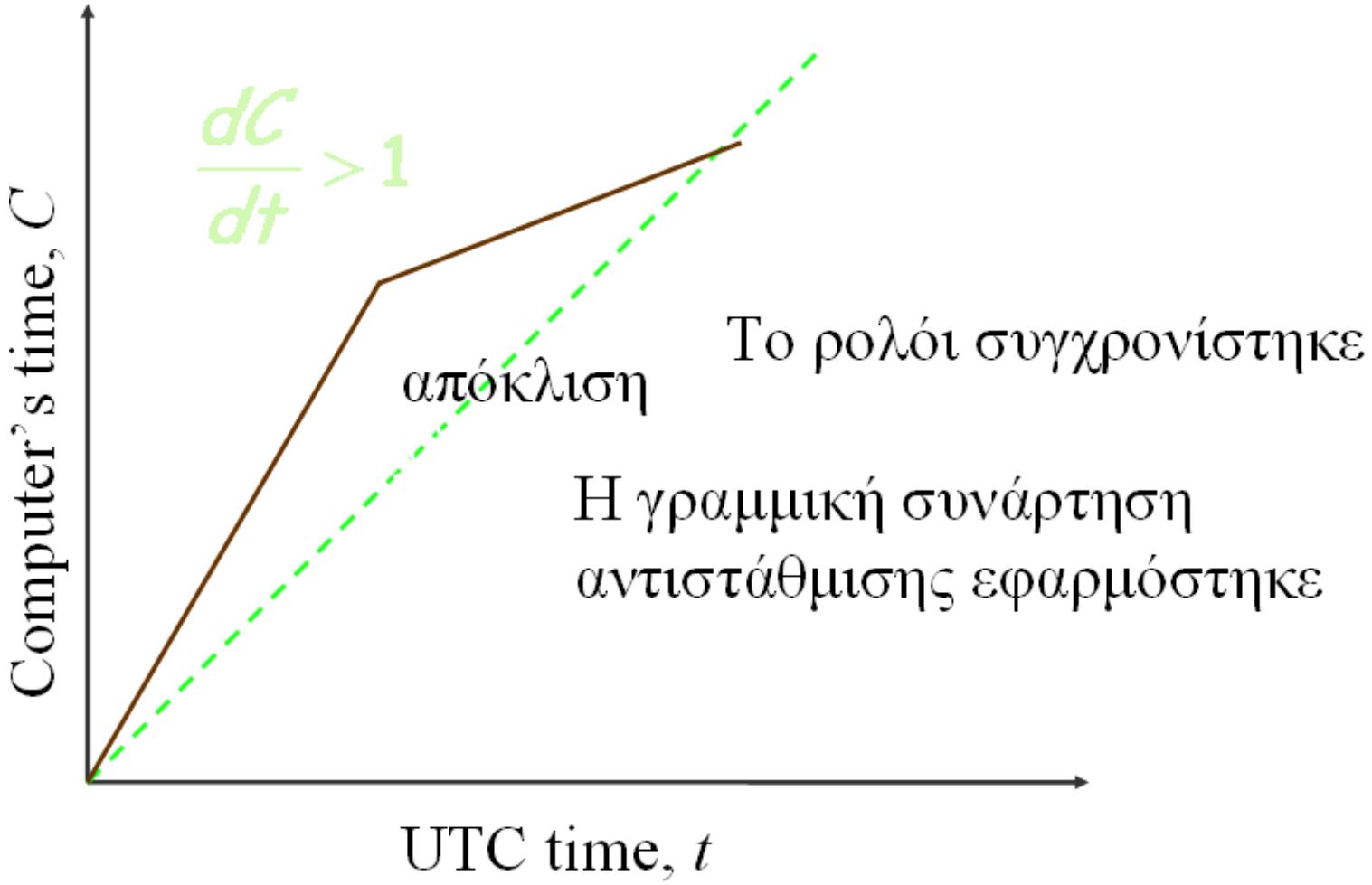
Αντιμετωπίζοντας τη Λόξωση (2/2)

- Με σταδιακή διόρθωση του ρολογιού,
  - **αν πάει πιο γρήγορα:**
    - Κάνουμε το ρολόι να πηγαίνει πιο αργά μέχρι να συγχρονίσει.
  - **αν πάει πιο αργά:**
    - Κάνουμε το ρολόι να πηγαίνει πιο γρήγορα μέχρι να συγχρονίσει.

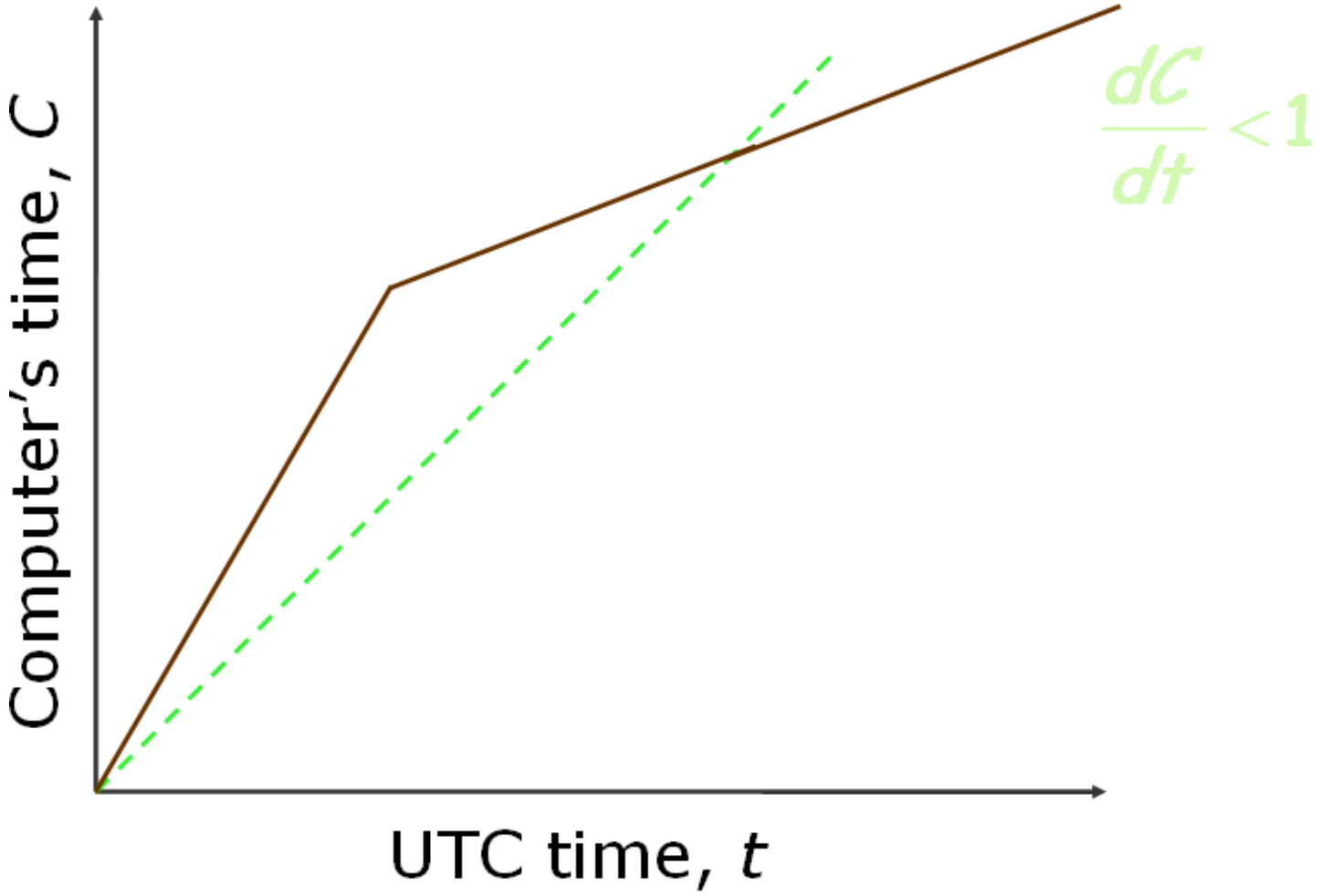
Αντιμετωπίζοντας τη Λόξωση (3/3)

- Το λειτουργικό σύστημα μπορεί να αλλάξει το ρυθμό στον οποίο απαιτεί διακοπές.
  - π.χ. εάν το σύστημα απαιτεί διακοπές κάθε 17 msec αλλά το ρολόι είναι πολύ αργό, το λειτουργικό απαιτεί διακοπές κάθε (π.χ.) 15 msec.
  - Είτε υπάρχει και η διόρθωση από λογισμικό: επαναπροσδιορισμός των διαστημάτων.
- Η ρύθμιση αλλάζει το slope (κλίση) της ώρας του συστήματος:
  - **Γραμμική συνάρτηση αντιστάθμισης.**

Αντιστάθμιση ενός γρήγορου ρολογιού (1/2)



Αντιστάθμιση ενός γρήγορου ρολογιού (2/2)



Επανασυγχρονισμός

- **Όταν έρθει η περίοδος συγχρονισμού:**
  - Επανασυγχρονίζεται περιοδικά.
  - Η επιτυχής εφαρμογή μιας δεύτερης γραμμικής συνάρτησης αντιστάθμισης μπορεί να μας φέρει πιο κοντά στη πραγματική κλίση.
  - Παρακολουθεί τις τροποποιήσεις και εφαρμόζεται συνέχεια.
  - π.χ., κλίση συστήματος της UNIX adjtime.

Για να πάρουμε την ακριβή ώρα (1/2)

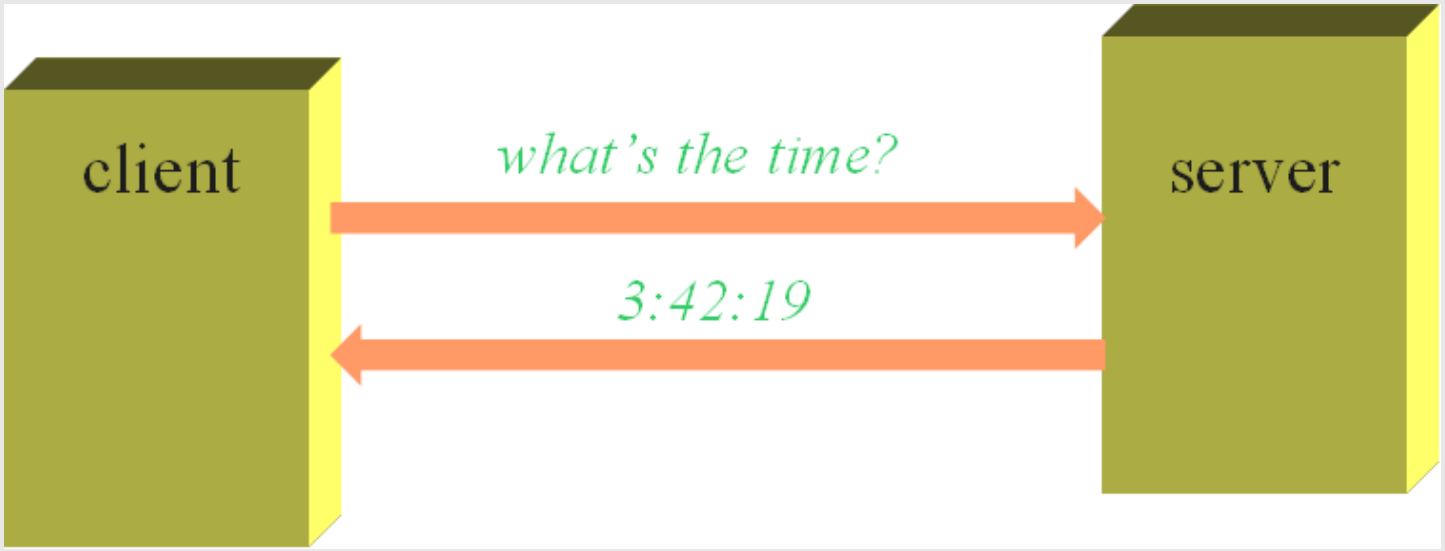
- Συνδέοντας GPS δέκτη σε κάθε υπολογιστή =>± 1 msec of UTC.
- Συνδέοντας ένα ραδιοφωνικό δέκτη ρολογιού (radio controlled clock).
- Παίρνοντας χρονικές αναμεταδώσεις από το Boulder ή το DC =>± 3 msec of UTC (depending on distance).
- Συνδέοντας GOES δέκτη =>± 0.1 msec of UTC
- Μη πρακτική λύση για κάθε μηχάνημα (Κόστος, μέγεθος, ευκολία, περιβάλλον).

Για να πάρουμε την ακριβή ώρα (2/2)

- Συγχρονίζοντας από μία άλλη μηχανή,
- Μία με πιο ακριβές ρολόι.
- Μηχανή/υπηρεσία που παρέχει πληροφορίες ώρας:
- ***Time server***

RPC

- **Η πιο απλή τεχνική συγχρονισμού:**
  - Ρωτάει το RPC για να πάρει την ώρα.
  - Ορίζει την ώρα.



- Δεν ισχύει για καθυστέρηση δικτύου ή διεργασιών.

Ο αλγόριθμος του Christian (1/5)

- Βασίζεται στην ύπαρξη ενός διακομιστή ώρας.
- Ο time server διατηρεί ακριβή ώρα χρησιμοποιώντας ένα radio clock ή κάποια άλλη έγκυρη πηγή ώρας.
- Όλοι οι υπόλοιποι υπολογιστές συγχρονίζονται από αυτόν.
- Χρησιμοποιούνται procedure calls.
- Διαφοροποιημένος αλγόριθμος λαμβάνει υπόψιν και την καθυστέρηση του δικτύου.

Ο αλγόριθμος του Christian (1989)

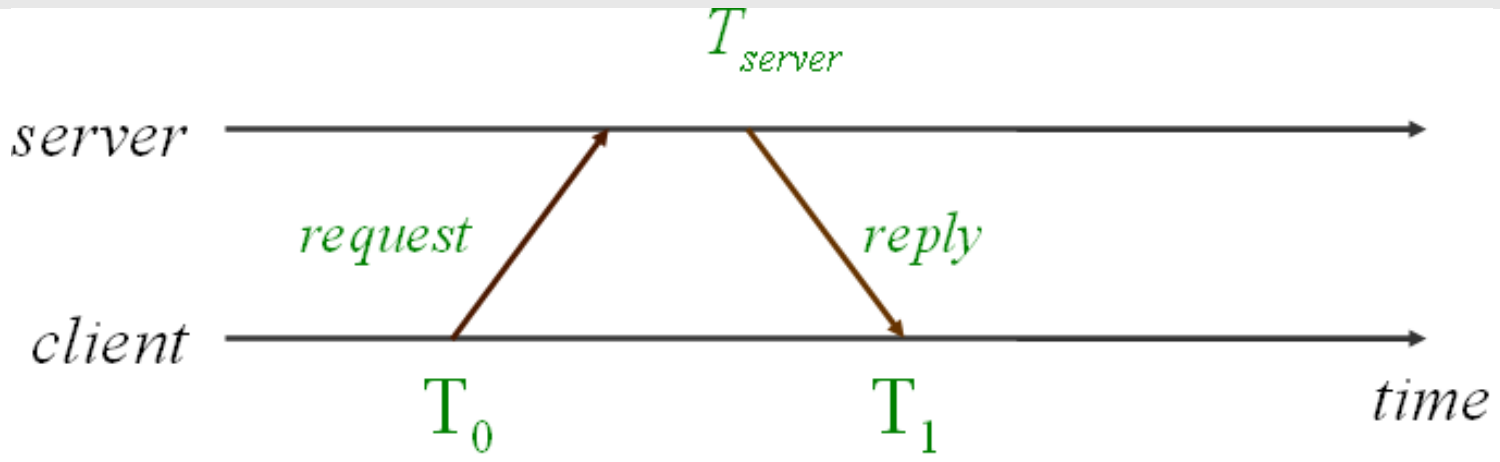
- Επιτυγχάνει συγχρονισμό μόνο αν το round-trip time (RTT) του αιτήματος είναι σύντομο σε σχέση με την απαιτούμενη ακρίβεια.
- Επίσης υπόκειται σε implementations χρησιμοποιώντας έναν μοναδικό server (χωρίς redundancy):
  - Το P απαιτεί την ώρα από το S.
  - Αφού λάβει το αίτημα από το P, το S ετοιμάζει μια απάντηση και συμπληρώνει την ώρα στο T από το δικό του ρολόι.
  - Το P μετά ορίζει την ώρα του να είναι T + RTT/2.

Ο αλγόριθμος του Christian (2/5)

- Εάν είναι γνωστός ο ελάχιστος χρόνος μετάδοσης του μηνύματος (Tmin)...
  - Τοποθετεί όρια στην ακρίβεια του αποτελέσματος.

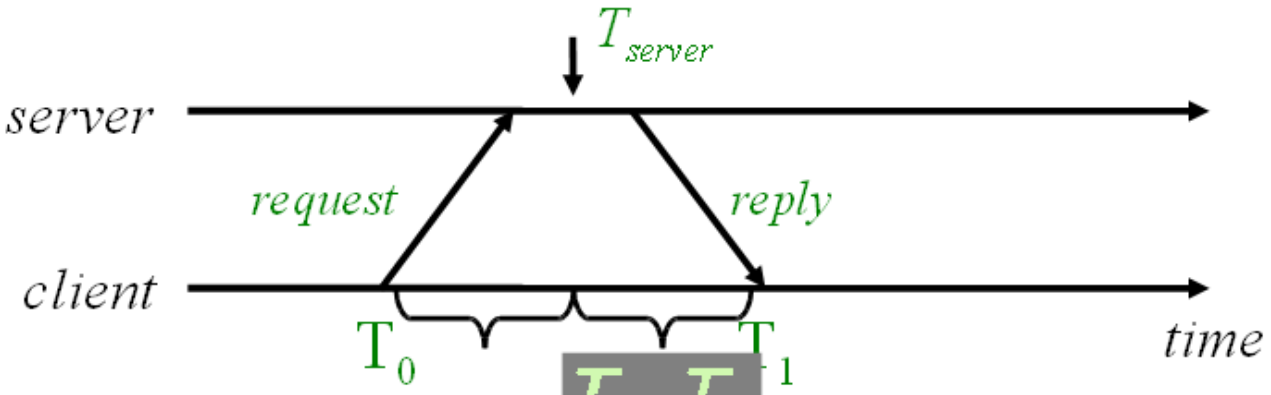
Ο αλγόριθμος του Christian (3/5)

- Αποκατάσταση των καθυστερήσεων (delays):
- Σημειώνω τους χρόνους,
  - Το αίτημα στάλθηκε: T_0
  - Η απάντηση λήφθηκε: T_1
- Θεωρώ ότι οι καθυστερήσεις του δικτύου είναι συμμετρικές.



Ο αλγόριθμος του Christian (4/5)

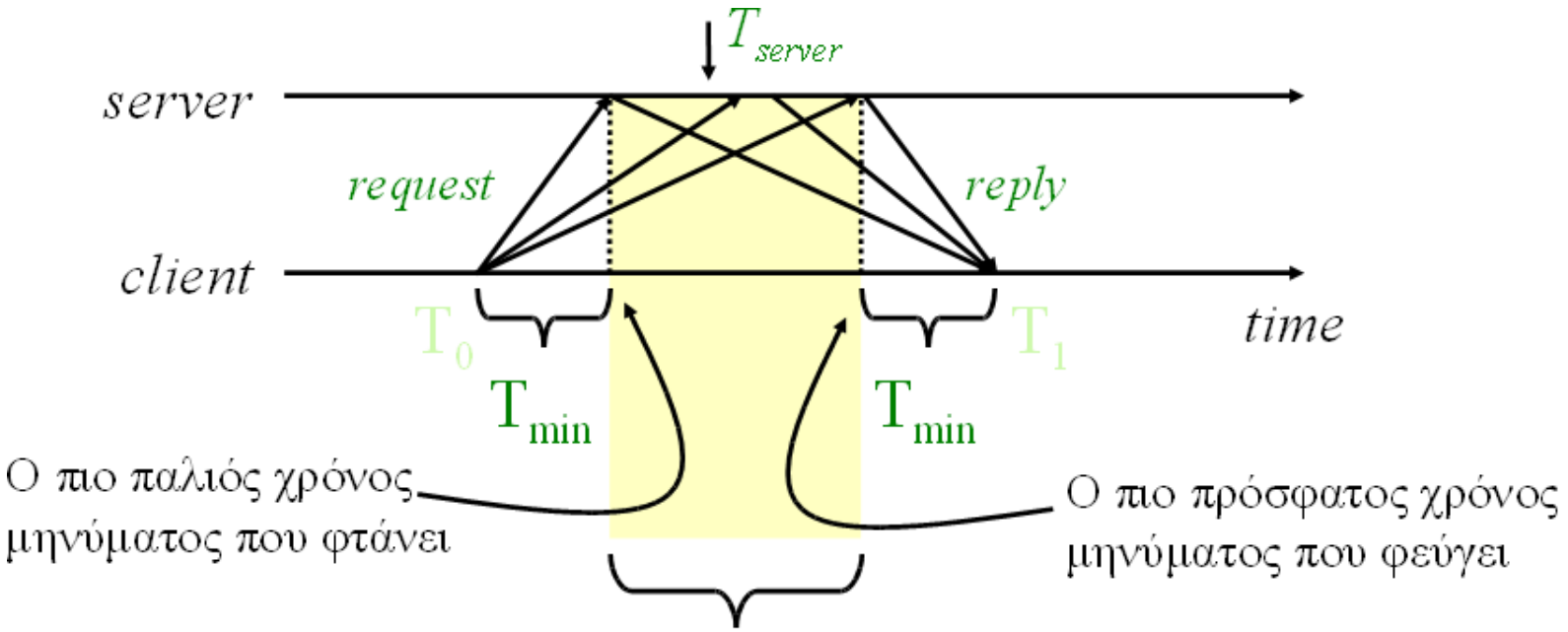
Ο Client ορίζει την ώρα σε:



$\frac{T_1 - T_0}{2}$ = εκτιμώμενο overhead σε κάθε κατεύθυνση

$$T_{new} = T_{server} + \frac{T_1 - T_0}{2}$$

Όρια σφάλματος στον αλγόριθμο του Christian



εύρος = $T_1 - T_0 - 2T_{min}$

Ακρίβεια του αποτελέσματος= $\pm \frac{T_1 - T_0}{2} - T_{min}$

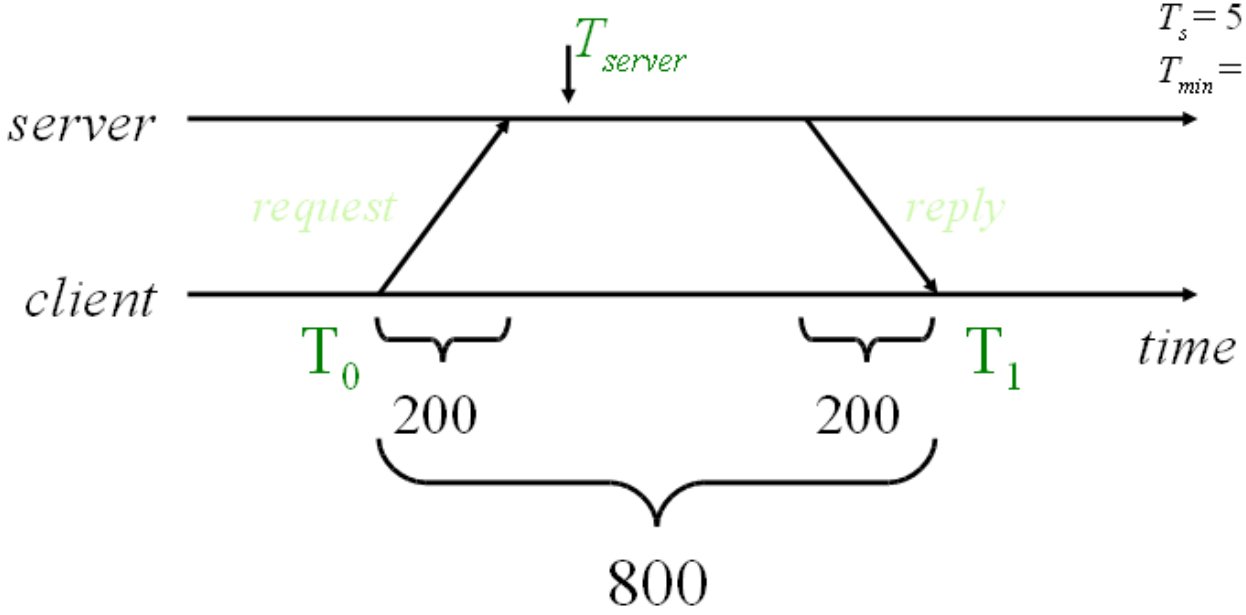
Παράδειγμα αλγορίθμου Christian (1/2)

- Στάλθηκε αίτημα στις 5:08:15.100 (T0).
- Λήφθηκε απάντηση στις 5:08:15.900 (T1).
- Η απάντηση περιέχει 5:09:25.300 (Tserver).
- Ο χρόνος που πέρασε (elapsed time) είναι T1 -T0:
  - 5:08:15.900 - 5:08:15.100 = 800 msec.
- Καλύτερη εικασία: η χρονοσφραγίδα ενεργοποιήθηκε:
  - 400 msec ago.
- Ορίζει την ώρα στο Tserver + elapsed time:
  - 5:09:25.300 + 400 = 5:09.25.700

Παράδειγμα αλγορίθμου Christian (2/2)

Εάν ο χρόνος μηνύματος στη καλύτερη περίπτωση είναι = 200 msec

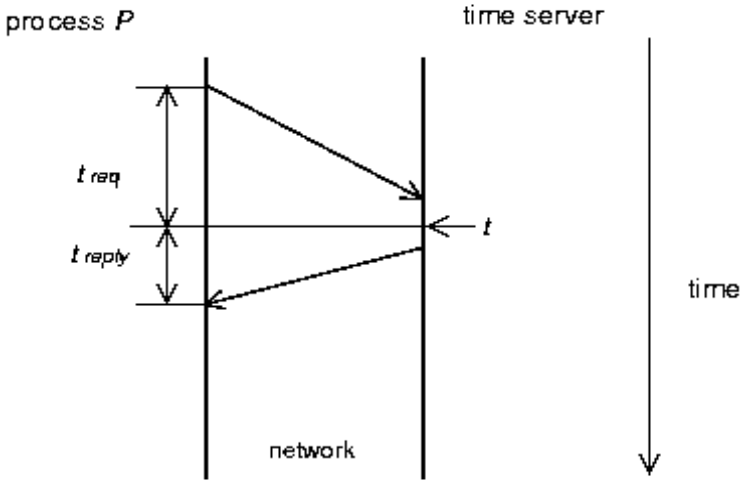
$T_0 = 5:08:15.100$
$T_1 = 5:08:15.900$
$T_s = 5:09:25:300$
$T_{min} = 200\text{msec}$



$$\Sigma\varphi\acute{\alpha}\lambda\mu\alpha = \pm\frac{900-100}{2} - 200 = \pm\frac{800}{2} - 200 = \pm 200$$

Ο αλγόριθμος του Christian (5/5)

- Συγχρονίζει τις μηχανές με βάση έναν *time server με* UTC δέκτη.
- Η μηχανή P απαιτεί την ώρα από τον server κάθε δ/2ρ seconds.
- Λαμβάνει τον χρόνο *t από* server, η P ορίζει το ρολόι σε $t+t_{reply}$ *όπου* t_{reply} *είναι ο χρόνος για να σταλεί απάντηση στην P.*
- Χρησιμοποιείται $(t_{req}+t_{reply})/2$ *για να υπολογίσουμε το* $t_{reply.}$
- Βελτιώνει την ακρίβεια με το να κάνει μια σειρά από μετρήσεις.



Ο αλγόριθμος του Berkeley (1/6)

- Δεν υπάρχει ακριβής ώρα σε κανένα κόμβο.
- Διατηρείται μια μέση ώρα.
- Ένας διακομιστής ρωτάει τους υπολοίπους για την ώρα, βρίσκει τη μέση ώρα και ενημερώνει για τις ρυθμίσεις που πρέπει να κάνει ο κάθε κόμβος.
- Υπάρχει η δυνατότητα να αγνοηθεί κατά τον υπολογισμό κάποια ώρα που έχει πολύ μεγάλη απόσταση από το μέσο όρο.

Ο αλγόριθμος του Berkeley (2/6)

- Χρησιμοποιείται σε συστήματα χωρίς UTC δέκτη:
  - Κρατάει τα ρολόγια συγχρονισμένα το ένα με το άλλο.
  - Ένας υπολογιστής είναι *master,* και οι άλλοι *slaves.*
  - Ο Master περιοδικά σταθμοσκοπεί (polls) τους slaves για τους χρόνους τους:
    - Βρίσκει τους μέσους χρόνους και επιστρέφει τις διαφορές στους slaves.
    - Οι επικοινωνιακές **καθυστερήσεις αντισταθμίζονται** όπως στον Cristian's αλγόριθμο.
  - Αποτυχία ενός master => Εκλογή ενός καινούριου master.
- 15 υπολογιστές συγχρονίστηκαν σε 25 ms max (1989).

Ο αλγόριθμος του Berkeley (3/6)

- Οι μηχανές τρέχουν χρόνο δαίμονα (**time daemon**).
- Διεργασία η οποία εφαρμόζει πρωτόκολλο.
- Μία μηχανή επιλέγεται (ή σχεδιάζεται) σαν server (**master**).
- Οι άλλες είναι **slaves.**

Ο αλγόριθμος του Berkeley (4/6)

- Ο master σταθμοσκοπεί κάθε μηχανή περιοδικά:
  - Ρωτάει κάθε μηχανή για την ώρα.
  - Μπορεί να χρησιμοποιήσει τον Cristian's αλγόριθμο για να αντισταθμίζει τη καθυστέρηση του δικτύου.
  - Όταν πάρει τα αποτελέσματα, υπολογίζει το μέσο όρο συμπεριλαμβάνοντας και την ώρα του master.
- *Ελπίδα: ο μέσος όρος ακυρώνει τις τάσεις που έχουν τα προσωπικά ρολόγια να τρέχουν γρήγορα ή αργά.*
  - Στέλνει offset με το οποίο κάθε ρολόι χρειάζεται ρύθμιση σε κάθε slave.
  - Αποφεύγει προβλήματα με τις καθυστερήσεις δικτύου εάν στείλουμε μια χρονοσφραγίδα.

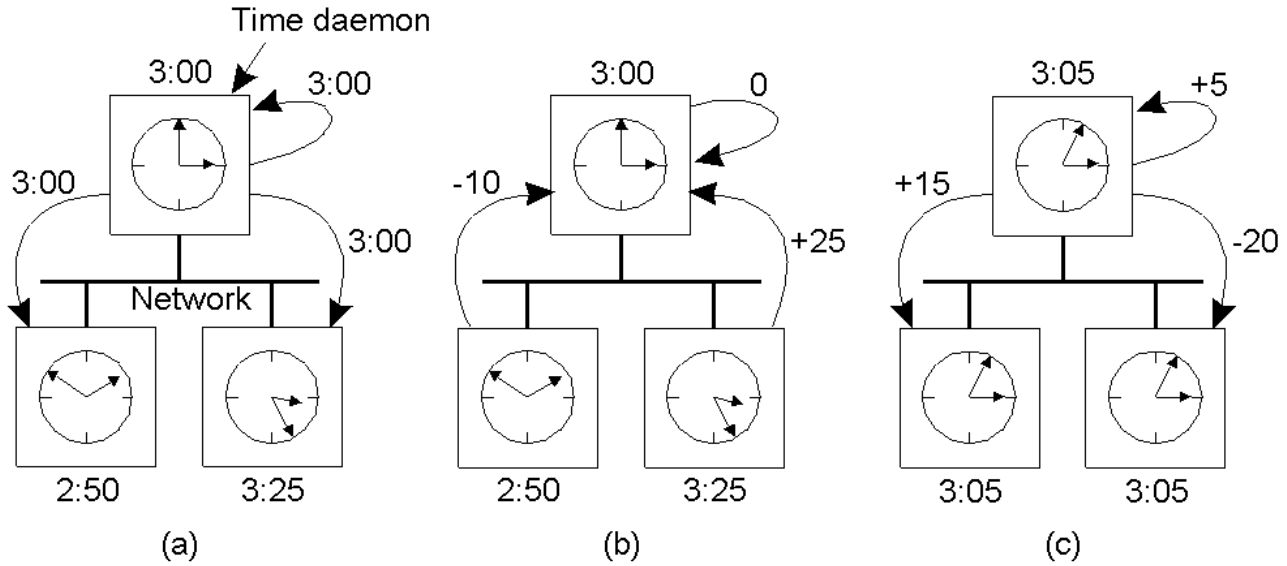
Ο αλγόριθμος του Berkeley (5/6)

- Ο αλγόριθμος είναι σχεδιασμένος να αγνοεί αναγνώσεις από ρολόγια στα οποία η απόκλιση τους είναι πολύ μεγάλη.
- Υπολογίζει έναν μέσο όρο ανοχής σφαλμάτων (fault-tolerant average)
- Αν ο master αποτύχει:
  - Μπορεί να αναλάβει οποιοσδήποτε slave.

Ο αλγόριθμος του Berkeley (6/6)

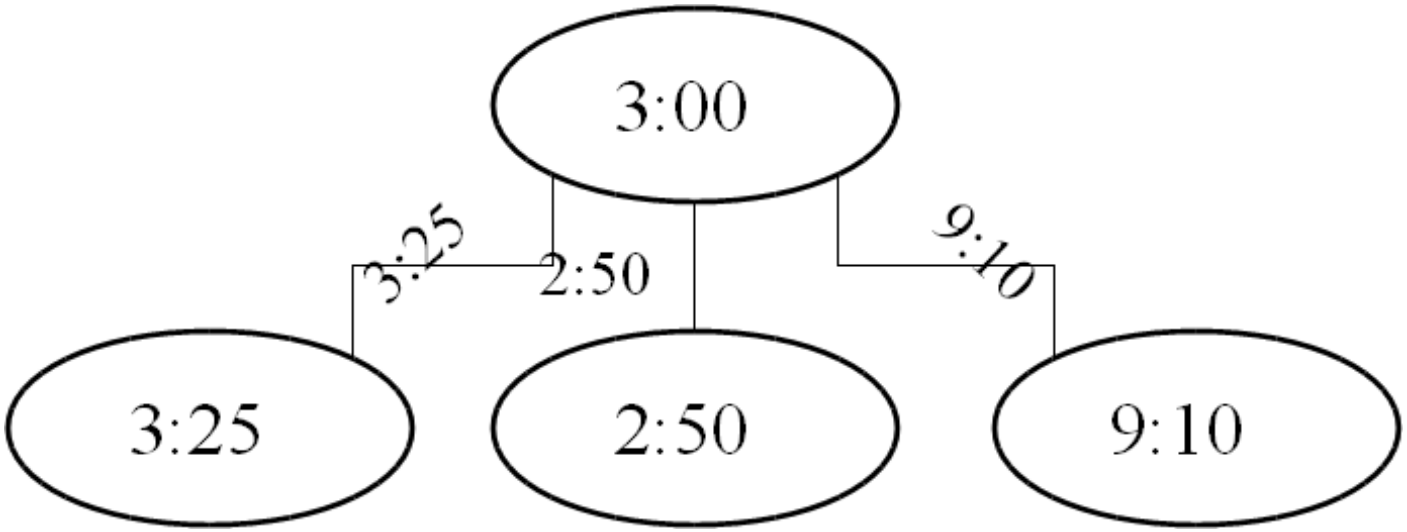
1. Επιλέγεται ένας master μέσω της διαδικασίας εκλογής.
2. Ο master σταθμοσκοπεί τους slaves που απαντούν με τον χρόνο τους σε παρόμοιο τρόπο όπως και στον Cristian's αλγόριθμο.
3. Ο master παρακολουθεί το round-trip time (RTT) των μηνυμάτων και υπολογίζει τον χρόνο κάθε slave καθώς και τον δικό του.
4. Μετά ο master κάνει τον μέσο όρο των χρόνων των ρολογιών, αγνοώντας τιμές που λαμβάνει και απέχουν πολύ από τις τιμές των άλλων.
5. Αντί να στείλει την αναβαθμισμένη τωρινή ώρα πίσω στην άλλη διεργασία, ο master στέλνει το ποσό (θετικό ή αρνητικό) με το οποίο κάθε slave πρέπει να προσαρμόσει το ρολόι του. Με αυτό αποφεύγει περεταίρω αβεβαιότητα εξαιτίας του RTT στις διεργασίες του slave.

Ο αλγόριθμος του Berkeley (1989)



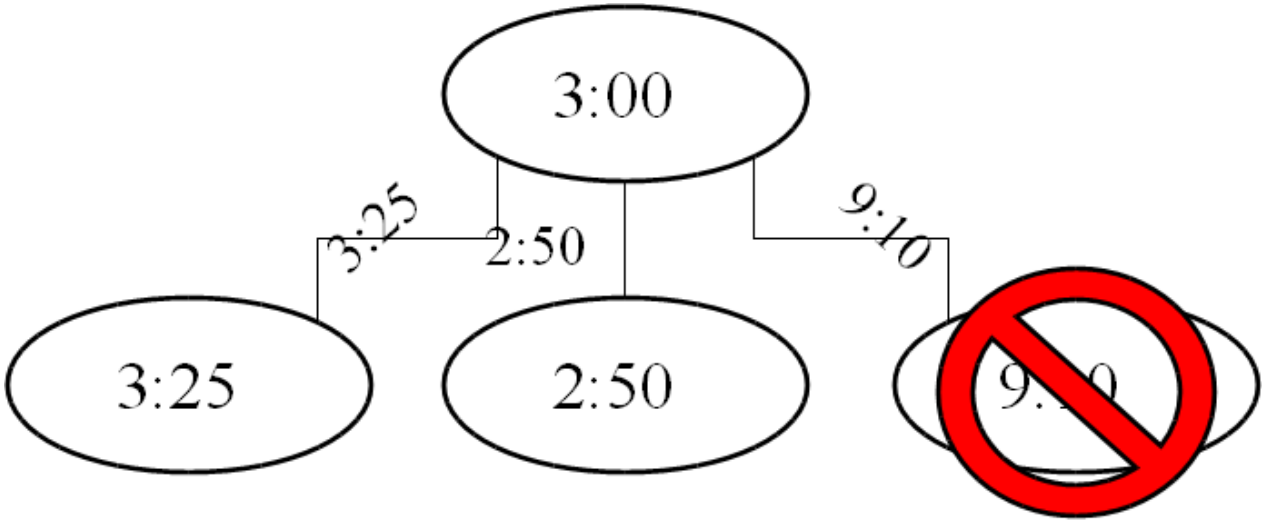
- Ο time daemon ρωτάει όλες τις άλλες μηχανές για τις τιμές των ρολογιών τους.
- Οι μηχανές απαντούν.
- Ο time daemon λέει σε όλους πως να ρυθμίσουν τα ρολόγια τους.

Ο αλγόριθμος του Berkeley: Παράδειγμα (1/3)



1. Απαιτεί χρονοσφραγίδες από όλους τους slaves.

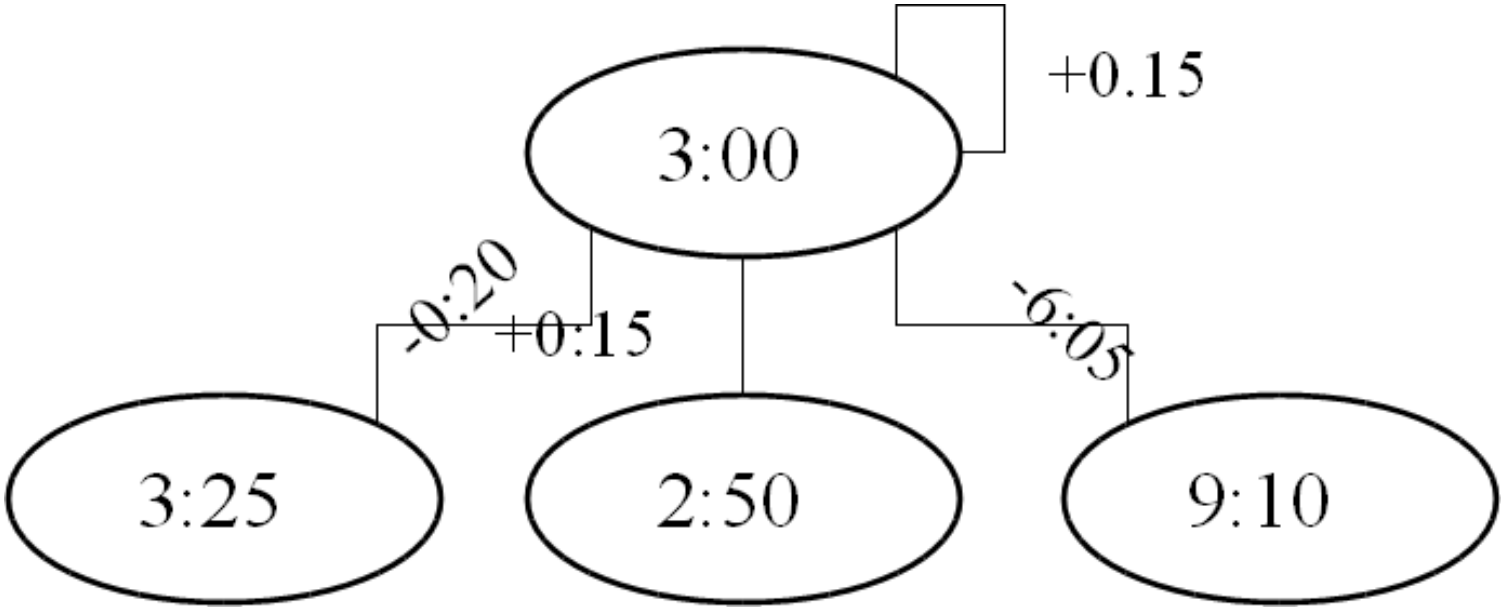
Ο αλγόριθμος του Berkeley: Παράδειγμα (2/3)



- Υπολογισμός του μέσου όρου ανοχής σφάλματος:

$$\frac{3{:}25 + 2{:}50 + 3{:}00}{3} = 3{:}05$$

Ο αλγόριθμος του Berkeley: Παράδειγμα (3/3)



3. Στέλνει offset σε κάθε client.

Κατανεμημένες Προσεγγίσεις

- Και οι δύο προσεγγίσεις που μελετήθηκαν ως τώρα είναι συγκεντρωτικές **(centralized).**
- **Μη-συγκεντρωτικοί αλγόριθμοι:** χρησιμοποιούν διαστήματα επανασυγχρονισμού.
  - Μεταδίδουν την ώρα στην αρχή κάθε διαστήματος.
  - Μαζεύουν όλες τις άλλες μεταδόσεις που φτάνουν με περίοδο *S.*
  - Χρησιμοποιούν τον μέσο όρο όλων των τιμών των αναφερόμενων χρόνων.
  - Μπορούν να πετάξουν μερικές υψηλές ή χαμηλές τιμές.
- **Προσεγγίσεις που χρησιμοποιούνται σήμερα:**
  - ***rdate***: συγχρονίζει μια μηχανή με μία άλλη συγκεκριμένη μηχανή.
  - **Network Time Protocol (NTP)** (συνιστώμενο): χρησιμοποιεί προχωρημένες τεχνικές για την ακρίβεια της τάξεως των 1-50 ms.

rdate

- Το **rdate** είναι μια εντολή Linux και το πρωτόκολλο της ώρας δικτύου για άμεση ρύθμιση ώρας και ημερομηνίας από άλλη μηχανή.
- Αντικαταστήθηκε από το **ntp/ntpd**.
- Αντίθετα με το ntp, το rdate ορίζει τη νέα ώρα αμέσως και είναι πιο κατάλληλο για καταστάσεις όπως της αρχικής ρύθμισης (initial setup).
- Χρησιμοποιεί τη θύρα (port) 37 και μπορεί να δουλέψει μέσω του TCP ή UDP (με -u switch).

Το πρωτόκολλο NTP (1985)

- Από τα πιο παλιά πρωτόκολλα του Internet.
- Όλοι οι κόμβοι συμμετέχουν στο συγχρονισμό του δικτύου.
- Ακρίβεια σε μερικά microseconds.
- Χρησιμοποιεί τη θύρα UDP 123.
- Λαμβάνεται υπόψη η διαφοροποίηση απόκρισης λόγω δικτύου.
- Το NTP χρησιμοποιεί UTC.
- Internet Standard, version 3: RFC 1305.

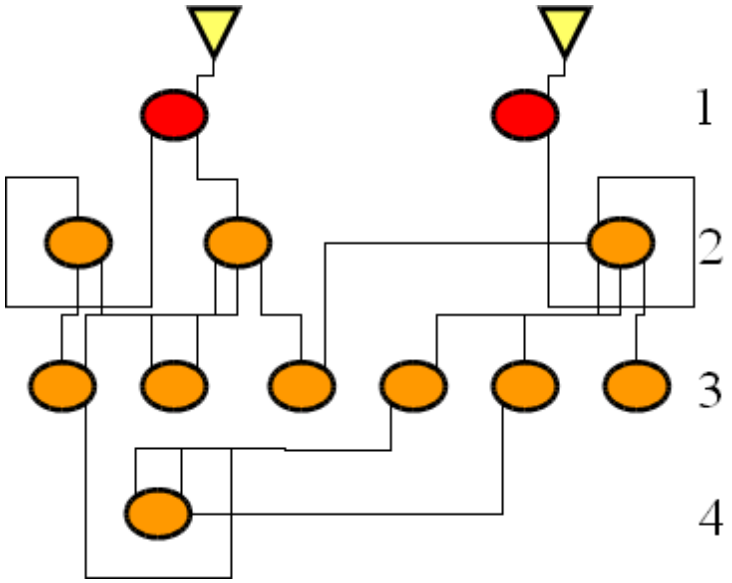
Στόχοι του NTP

- Δίνει τη δυνατότητα σε clients όλου του Διαδικτύου να είναι συγχρονισμένοι με την ίδια ακρίβεια παρόλη τη καθυστέρηση των μηνυμάτων.
  - Χρησιμοποιεί στατιστικές τεχνικές για να φιλτράρει δεδομένα και να υπολογίσει την ποιότητα των αποτελεσμάτων.
- Παρέχει αξιοπιστία.
  - Επιβιώνει σε μεγάλες απώλειες συνδεσιμότητας.
  - Επιπρόσθετα μονοπάτια.
  - Επιπρόσθετοι servers.
- Δίνει τη δυνατότητα σε clients να συγχρονίζουν συχνά.
  - offset effects of clock drift.
- Παρέχει προστασία σε παρεμβολές.
  - Πιστοποιεί τη πηγή των δεδομένων.

NTP servers

- Στοιχισμένοι σε strata.
- 1st stratum: Οι μηχανές είναι συνδεδεμένες απευθείας στην ακριβή πηγή ώρας.
- 2nd stratum: Οι μηχανές συγχρονίζουν από τις μηχανές του 1st stratum.
- …

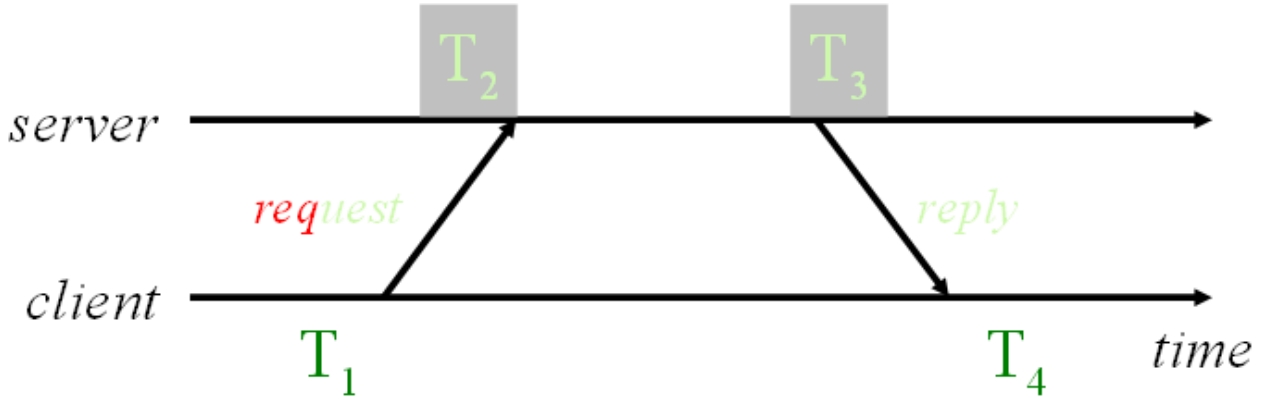
Υποδίκτυο Συγχρονισμού.



Μέθοδοι Συγχρονισμού του NTP

- Μέθοδος πολυδιανομής (Multicast):
  - για υψηλής ταχύτητας LANs.
  - χαμηλότερης ακρίβειας αλλά πιο αποτελεσματική.
- Μέθοδος Procedure call:
  - Παρόμοια με αυτή του αλγόριθμου του Cristian.
- Συμμετρική μέθοδος:
  - Στοχεύει σε master servers.
  - Ζευγάρια servers ανταλλάσσουν μηνύματα και διατηρούν τα δεδομένα για να βελτιώσουν το συγχρονισμό με την ώρα.
- Όλα τα μηνύματα παραδίδονται αναξιόπιστα με UDP.

Παράδειγμα SNTP (1/2)



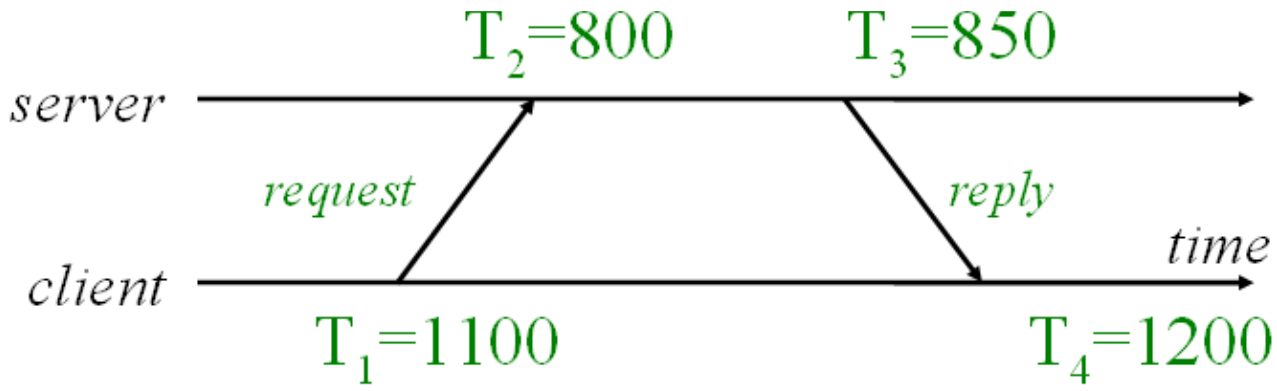
Καθυστέρηση Roundtrip:

$d = (T_4 - T_1) - (T_2 - T_3)$

Time offset:

$$t = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

Παράδειγμα SNTP (2/2)



Offset =

((800 - 1100) + (850 - 1200))/2

=((-300) + (-350))/2

= -650/2 = **-325**

Time offset:

$$t = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

Ορίζει την ώρα σε $T_4 + t$

= 1200 - 325 = **875**

pool.ntp.org

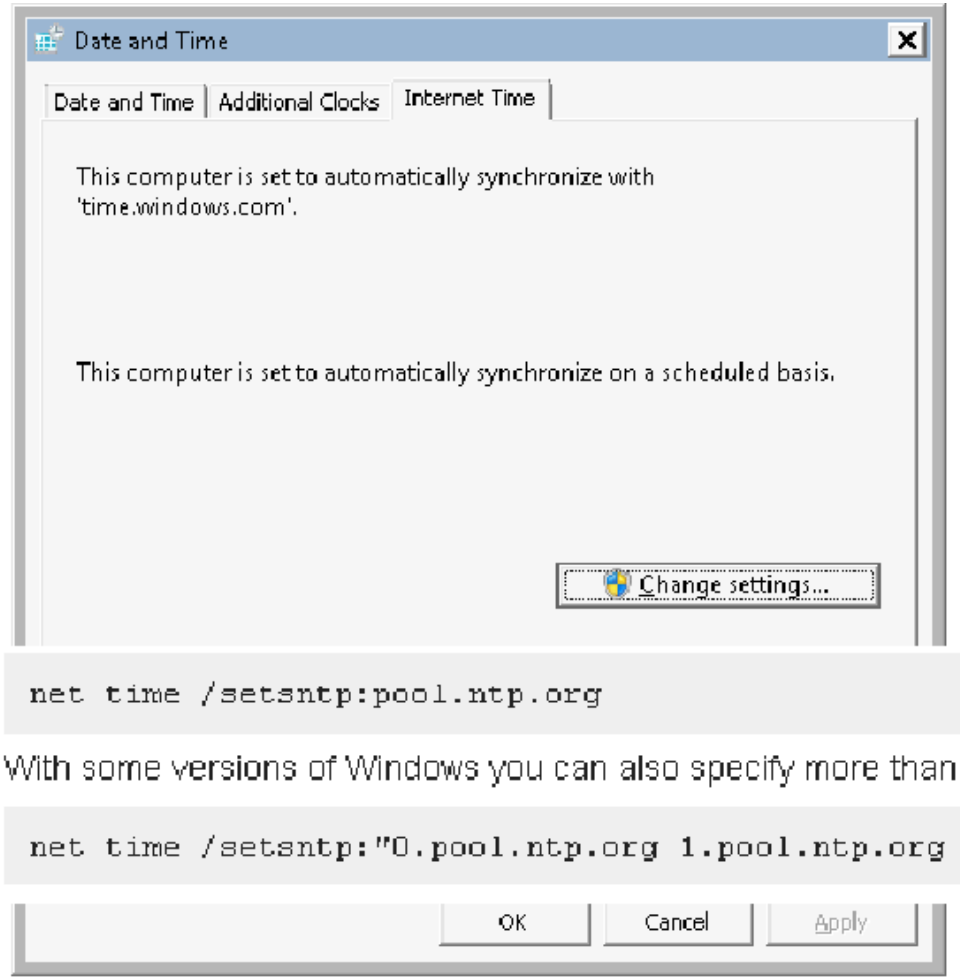
pool.ntp.org: public ntp time server for everyone

Πως χρησιμοποιείται το SNMP.. (1/2)

- linux,bsd,unix...
  - driftfile /var/lib/ntp/ntp.drift
  - “server 0.pool.ntp.org
  - server 1.pool.ntp.org
  - server 2.pool.ntp.org
  - server 3.pool.ntp.org”
  - ----
- Σιγουρέψτε ότι το ρολόι του υπολογιστή σας είναι ορισμένο σε κάτι λογικό.

```
avbidder:~$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+81.6.42.224     193.5.216.14     2 u   68 1024  377  158.995   51.220  50.287
*217.162.232.173 130.149.17.8     2 u  191 1024  176   79.245    3.589  27.454
-129.132.57.95   131.188.3.222    3 u  766 1024  377   22.302   -2.928   0.508
```

Πως χρησιμοποιείται το SNMP.. (2/2)




```
net time /setsntp:pool.ntp.org
```

With some versions of Windows you can also specify more than one server

```
net time /setsntp:"0.pool.ntp.org 1.pool.ntp.org 2.pool.ntp.org"
```

Μια καλή τακτική είναι να συγχρονίζουμε την ώρα στο διακομιστή μας..

```
bigbrother@bigb3:[ ./bigbrother ]{14}> ntptrace ntp.forthnet.gr
ntp.forthnet.gr: stratum 2, offset 0.000566, root distance 0.002830
ntpq: write to 10.24.200.196 failed: Permission denied
```

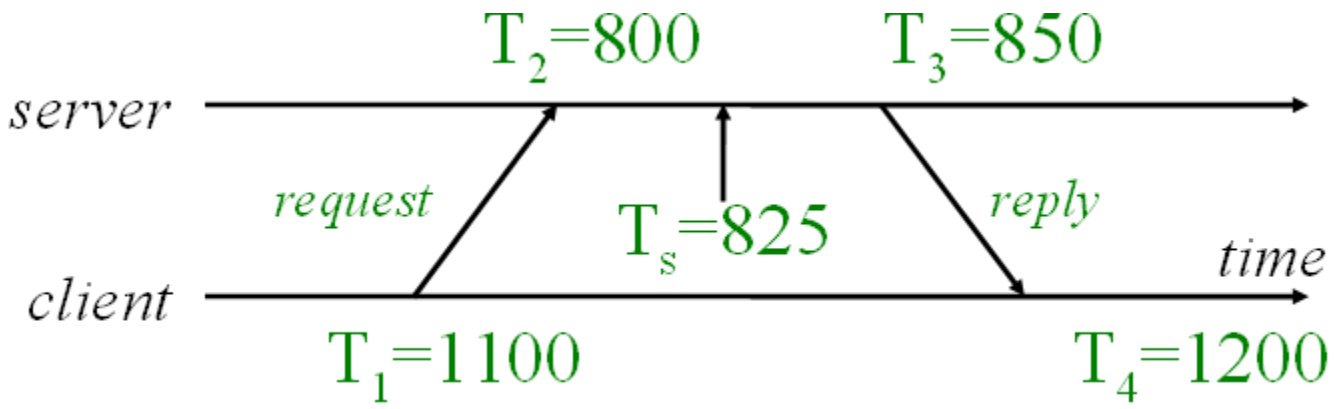
Στο crontab για ημερήσιο update της ώρας:

```
@daily     /usr/sbin/ntpdate -u -s ntp.duth.gr
ntp.forthnet.gr ntp.ntua.gr ntp.grnet.gr
```

Στο /var/log/messages.log καταγράφονται:

adjust time server 193.92.150.3 offset -0.263939 sec

Αλγόριθμος του Cristian



Offset = (1200 - 1100)/2 = **50**

Ορίζει την ώρα σε T_s + *offset*
= 825 + 50 = **875**

Σημεία κλειδιά: Φυσικά Ρολόγια

- Αλγόριθμος του Cristian & SNTP:
  - Ορίζουν το ρολόι από τον server.
- Αλλά μετράνε στις καθυστερήσεις δικτύου.
- Σφάλμα: αβεβαιότητα εξαιτίας της καθυστέρησης του δικτύου/επεξεργαστή:
  - τα σφάλματα είναι προσθετικά δηλαδή:±10 msec και ±20 msec = ±30 msec.
- Ρύθμιση για την απόκλιση τοπικού ρολογιού:
  - Γραμμική συνάρτηση αντιστάθμισης.

Μήπως όμως μπορούμε να συγχρονίσουμε τους κόμβους διαφορετικά;

- Ο χρόνος **δεν** είναι αξιόπιστη μέθοδος συγχρονισμού.
  - Οι χρήστες ανακατεύουν τα ρολόγια (και ξεχνάνε να ορίσουν τις ζώνες ώρας τους!)
  - Έχει απρόβλεπτες καθυστερήσεις στο Διαδίκτυο.
  - Σχετικιστικά ζητήματα:
    - Εάν *A* και *B* έχουν μεγάλη φυσική απόσταση, και δύο γεγονότα T_A *και* T_B έχουν πολύ μικρή χρονική απόσταση, τότε ποιο γεγονός έρχεται πρώτο;
    - Πως μπορούμε να ξέρουμε;

Φυσικά Ρολόγια

- Για πολλά προβλήματα, η εσωτερική σταθερότητα των ρολογιών είναι σημαντική.
  - Η τέλεια ώρα είναι λιγότερο σημαντική.
  - Χρήση λογικών ρολογιών.
- **Βασική Ιδέα:**
  - Ο συγχρονισμός του ρολογιού δε χρειάζεται να είναι τέλειος.
  - Αν δύο μηχανές δεν αλληλεπιδρούν, δε χρειάζεται να τις συγχρονίσουμε.
  - **Οι διεργασίες πρέπει κυρίως να συμφωνούν στη σειρά που συμβαίνουν τα γεγονότα παρά στην ώρα την οποία συμβαίνουν.**

Λογικά Ρολόγια

- Αποδίδουν αριθμούς ακολουθίας (**sequence number**) στα μηνύματα.
  - Όλες οι διεργασίες που συνεργάζονται μπορούν να συμφωνούν στην σειρά των γεγονότων.
  - vs. **φυσικά ρολόγια**: ώρα της ημέρας.
- Υποθέτουμε **μη κεντρική** πηγή ώρας:
  - Κάθε σύστημα διατηρεί το δικό του τοπικό ρολόι.
  - Δεν υπάρχει καθολική διάταξη των γεγονότων.
    - Δεν υπάρχει η έννοια του “πότε έγινε”.

Διάταξη Γεγονότων

- **Πρόβλημα:** ο **προσδιορισμός της καθολικής διάταξης** όλων των γεγονότων που συμβαίνουν στο σύστημα.
  - Γεγονότα σε μία μηχανή μονού επεξεργαστή ταξινομούνται πλήρως.
  - Προβλήματα στα κατανεμημένα συστήματα:
    - Δεν υπάρχει παγκόσμιο ρολόι, τα τοπικά ρολόγια μπορεί να μην είναι συγχρονισμένα.
    - Δεν μπορούν να ταξινομηθούν τα γεγονότα σε διαφορετικές μηχανές χρησιμοποιώντας τοπικές ώρες.
- **=> Βασική Ιδέα** [Lamport]
  - Οι διεργασίες να ανταλλάσσουν μηνύματα.
  - Το μήνυμα θα πρέπει να στέλνεται πριν παραληφθεί κάποιο άλλο.
  - Η αποστολή/λήψη χρησιμοποιείται για να ταξινομήσει τα γεγονότα (και να συγχρονίσει τα ρολόγια).

Happened-before

- Η σημείωση “Happened-before” του Lamport.
- **a -> *b*** , *το γεγονός* **a** *έγινε πριν το γεγονός* ***b***
- π.χ.: **a**: το μήνυμα στέλνεται, **b**: το μήνυμα λαμβάνεται.
- Αν **a -> *b*** και ***b* -> *c*** τότε και **a -> *c*.**

Λογικά ρολόγια & Ταύτιση

- Αποδίδοντας τιμή “ρολογιού” σε κάθε γεγονός:
  - Αν **a -> *b*** τότε **ρολόι(a) < ρολόι(*b*)**, αφού ο χρόνος δε μπορεί να πάει πίσω.
  - Αν **a** και **b** συμβαίνουν σε διαφορετικές διεργασίες που δεν ανταλλάσσουν μηνύματα, τότε ούτε το **a -> *b*** είναι αληθές ούτε το ***b* -> a** .
  - Αυτά τα γεγονότα είναι **ταυτόχρονα (concurrent).**

Σχέση Happened Before

- Αν ***A*** *και* ***B*** γεγονότα σε μία ίδια διεργασία και το ***A*** εκτελείται πριν το ***B***, τότε ***A*** -> ***B.***
- Αν το ***A*** αντιπροσωπεύει την αποστολή ενός μηνύματος και το ***B*** είναι η λήψη αυτού του μηνύματος, τότε ***A*** -> ***B.***
- Η σχέση είναι transitive:
- A -> B και B -> C => A -> C
- Η σχέση είναι **αόριστη** σε διεργασίες οι οποίες δεν ανταλλάσσουν μηνύματα:
- ➔ Μερική διάταξη γεγονότων.

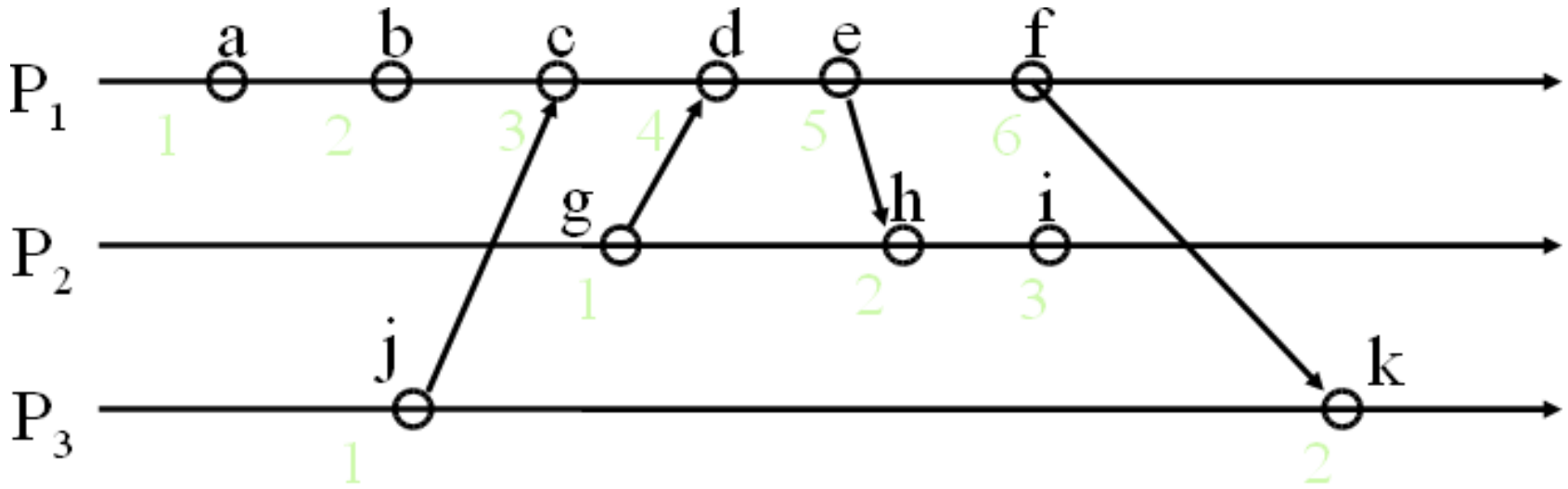
Διάταξη Γεγονότων με τη χρήση HB

- Στόχος: Να ορίσει την αντίληψη του χρόνου σαν ένα γεγονός όπως το παρακάτω:
  - Αν A-> B τότε C(A) < C(B).
  - Αν A και B συντρέχοντα, τότε C(A) < ή = ή > C(B).
- Λύση:
  - Κάθε επεξεργαστής διατηρεί ένα λογικό ρολόι LCi .
  - Όποτε συμβαίνει ένα γεγονός τοπικά στο I, LCi = LCi+1 .
  - Όταν ο *i* στέλνει ένα μήνυμα στον *j,* piggyback Lci .
  - Όταν ο j λαμβάνει ένα μήνυμα από τον i.
  - ➔ Αν LCj < LCi τότε LCj = LCi +1 , διαφορετικά δεν κάνει τίποτα.
  - Ισχυρισμός: Ο αλγόριθμος να πληρεί τους παραπάνω στόχους.

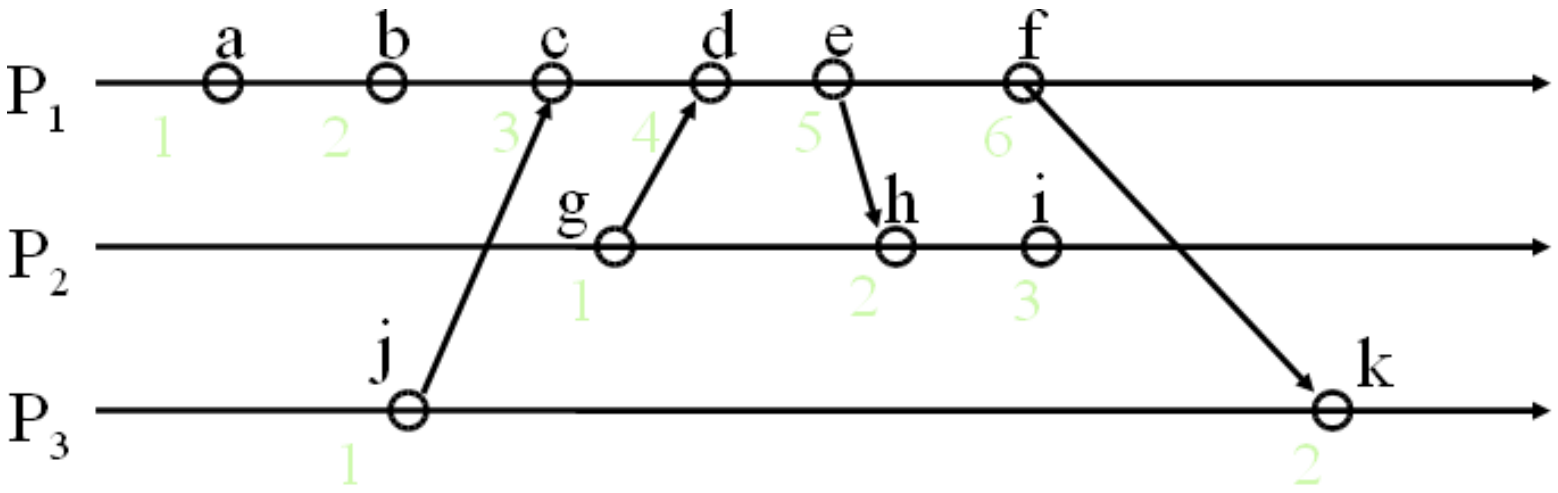
Παράδειγμα υπολογισμού γεγονότων (1/4)

- Τρία συστήματα: P0, P1, P2.
- Γεγονότα a, *b*, *c*, …
- Το τοπικό γεγονός υπολογίζεται σε κάθε σύστημα.
- Τα συστήματα επικοινωνούν περιστασιακά.

Παράδειγμα υπολογισμού γεγονότων (2/4)



Παράδειγμα υπολογισμού γεγονότων (3/4)



Κακή Διάταξη:

e → h (5->2)

f → k

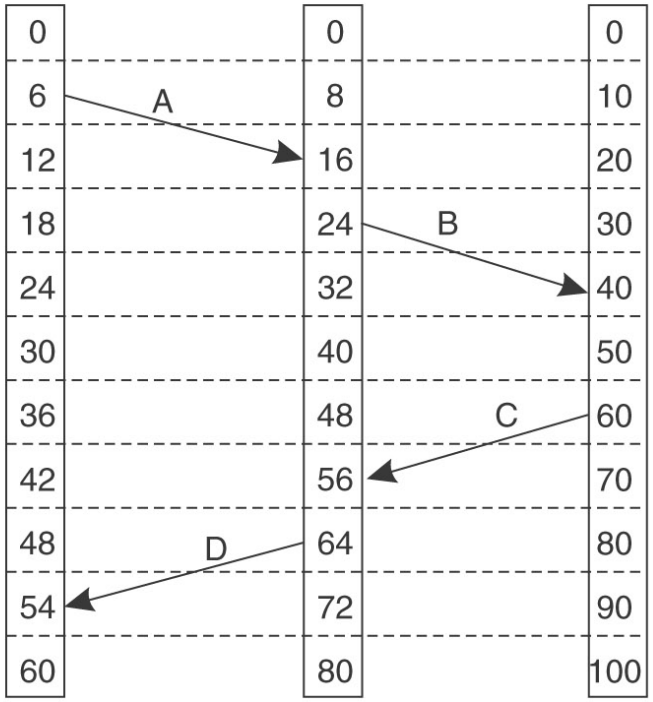
Ο Αλγόριθμος του Lamport (1/2)

- Κάθε μήνυμα κουβαλάει μία χρονοσφραγίδα από το ρολόι του αποστολέα.
- **Όταν φτάνει ένα μήνυμα:**
  - Αν το ρολόι του παραλήπτη < της χρονοσφραγίδας του μηνύματος τότε:
    - ορίζεται το ρολόι του συστήματος σε “χρονοσφραγίδα μηνύματος + 1”,
    - διαφορετικά δε κάνει τίποτα.
  - Το ρολόι πρέπει να είναι advanced ανάμεσα σε δύο οποιαδήποτε γεγονότα μέσα στην ίδια διεργασία.

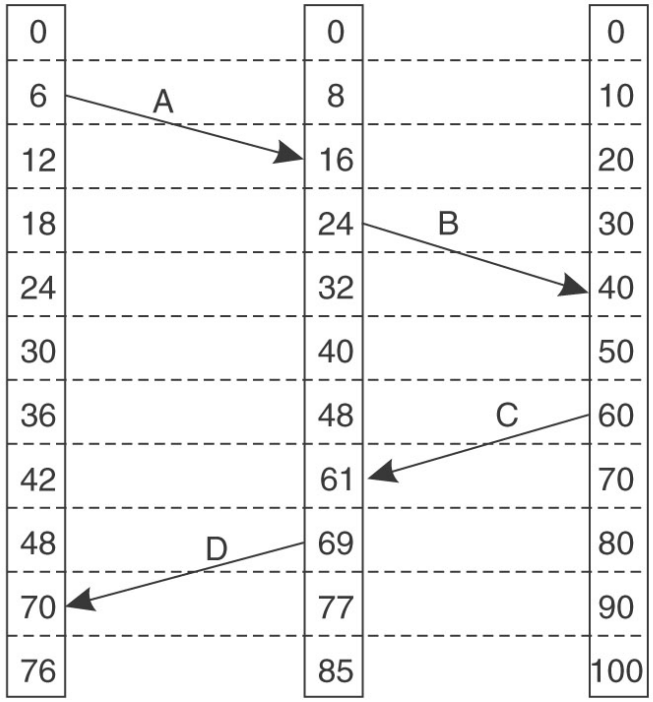
Ο Αλγόριθμος του Lamport (2/2)

- Ο αλγόριθμος αυτός μας επιτρέπει να διατηρούμε την διάταξη του χρόνου σε σχετικά μεταξύ τους γεγονότα.
- ***Μερική διάταξη.***

Τα λογικά ρολόγια του Lamport

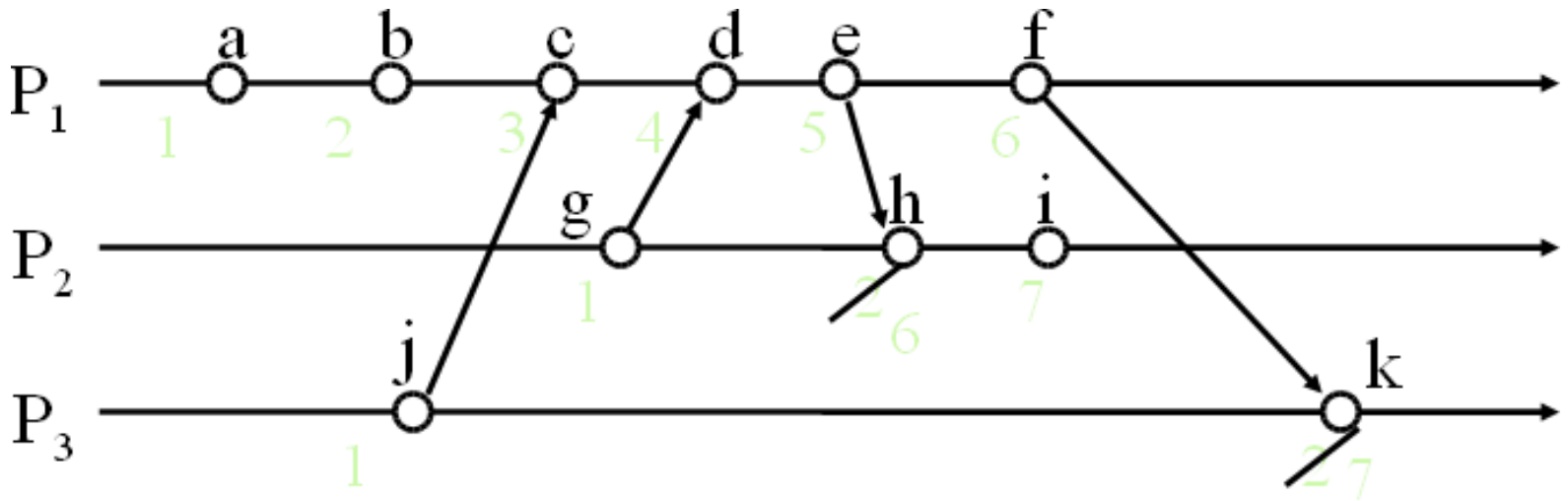


(a)



(b)

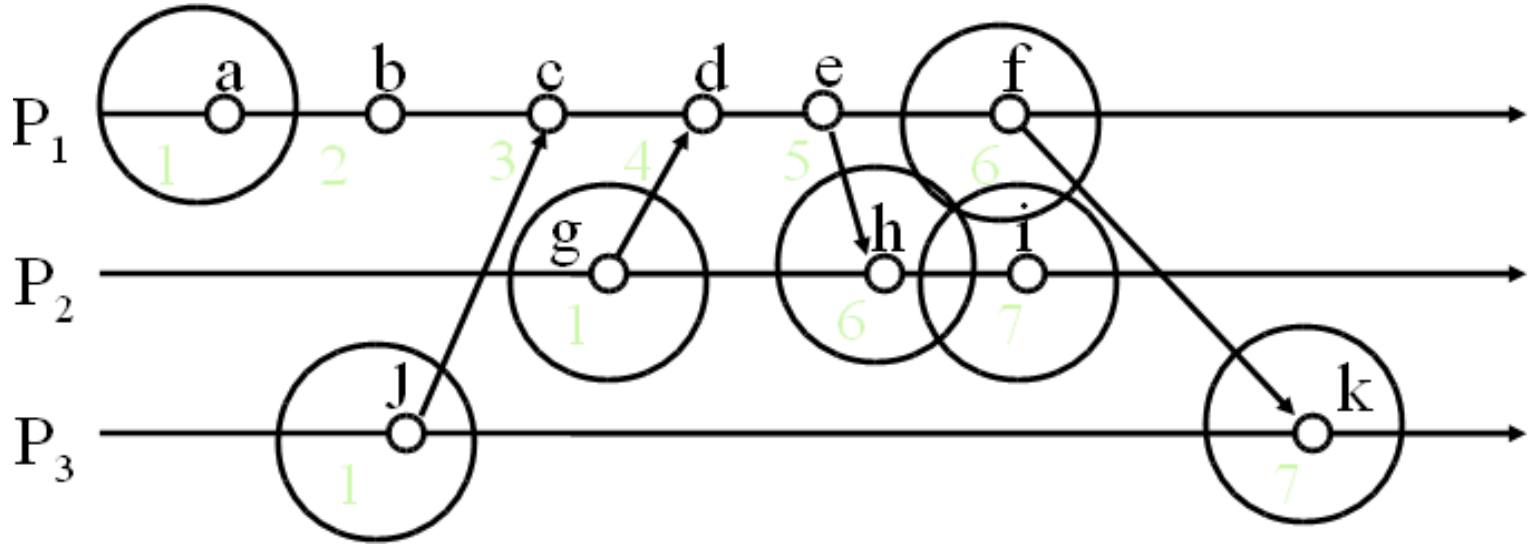
Παράδειγμα υπολογισμού γεγονότων (4/4)



Σύνοψη

- Ο αλγόριθμος χρειάζεται μονοτονική αύξηση του μετρητή του λογισμικού.
- Αυξάνεται τουλάχιστον κάθε φορά που πρέπει να χρονοσφραγιστούν γεγονότα.
- Κάθε γεγονός φέρει και μία **χρονοσφραγίδα Lamport**.
- Για δύο οποιαδήποτε γεγονότα, όπου a -> b:L(a) < L(b).

Πρόβλημα: Πανομοιότυπες Χρονοσφραγίδες

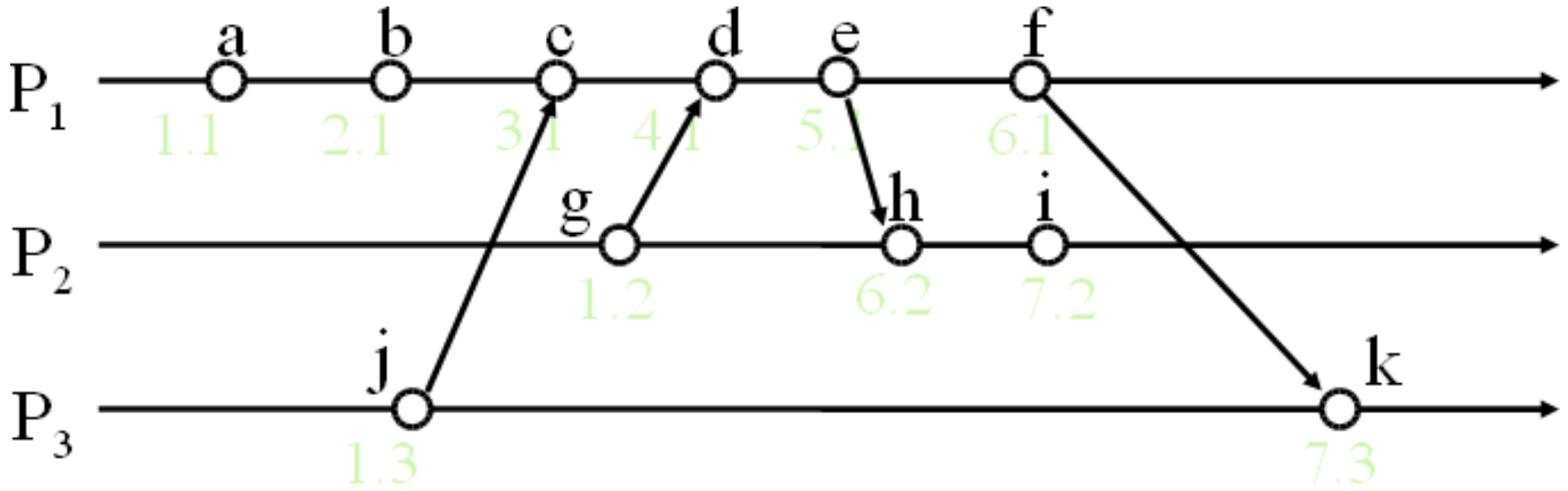


- a ->b, b->c, …: τοπικά γεγονότα σε ακολουθία.
- i->c, f->d , d->g, … :O Lamport εκμεταλλεύεται τη σχέση *send->receive*
- **Ταυτόχρονα γεγονότα (π.χ., a & i) *ίσως* να έχουν την ίδια χρονοσφραγίδα... ίσως και όχι!**

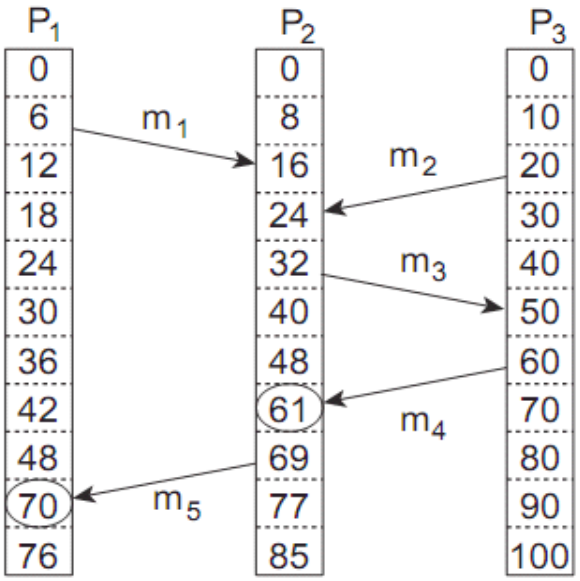
Μοναδικές χρονοσφραγίδες (καθολική διάταξη) (1/2)

- Μπορούμε να κάνουμε κάθε χρονοσφραγίδα να είναι μοναδική ορίζοντας **παγκόσμια λογική χρονοσφραγίδα** ($T_{i, i)}$:
  - Το T_i αντιπροσωπεύει τη τοπική χρονοσφραγίδα Lamport.
  - Το i αντιπροσωπεύει τον αριθμό διεργασίας (παγκοσμίως μοναδικό).
  - Π.χ. (host address, process ID).
- Σύγκριση Χρονοσφραγίδων:
  - $(T_i, i) < (T_j, j)$
  - Αν και μόνο αν
  - $T_i < T_j$ ή
  - $T_i = T_j$ και $i < j$
- **Δεν έχει να κάνει με τη διάταξη των γεγονότων.**

Μοναδικές χρονοσφραγίδες (καθολική διάταξη) (2/2)



Τα ρολόγια του Lamport δεν εγγυώνται ότι αν C(a)<C(b) τότε και το a αιτιολογικά προηγείται του b



Παρατήρηση

Γεγονός a: το m_1 λαμβάνεται σε χρόνο T=16.

Γεγονός b: το m_2 στέλνεται σε χρόνο T=20.

Note

Δε μπορούμε να συμπαιράνουμε ότι το a προηγείται αιτιολογικά του b.

Πρόβλημα: Ανακαλύπτοντας αιτιατές σχέσεις

- Αν L(e) < L(e’)
  - Δεν μπορούμε να συμπεράνουμε ότι και e->e’.
- Κοιτώντας τις χρονοσφραγίδες του Lamport:
  - Δεν μπορούμε να συμπεράνουμε ποια γεγονότα σχετίζονται αιτιατά.
- **Λύση:** η χρησιμοποίηση ενός **Διανυσματικού Ρολογιού.**

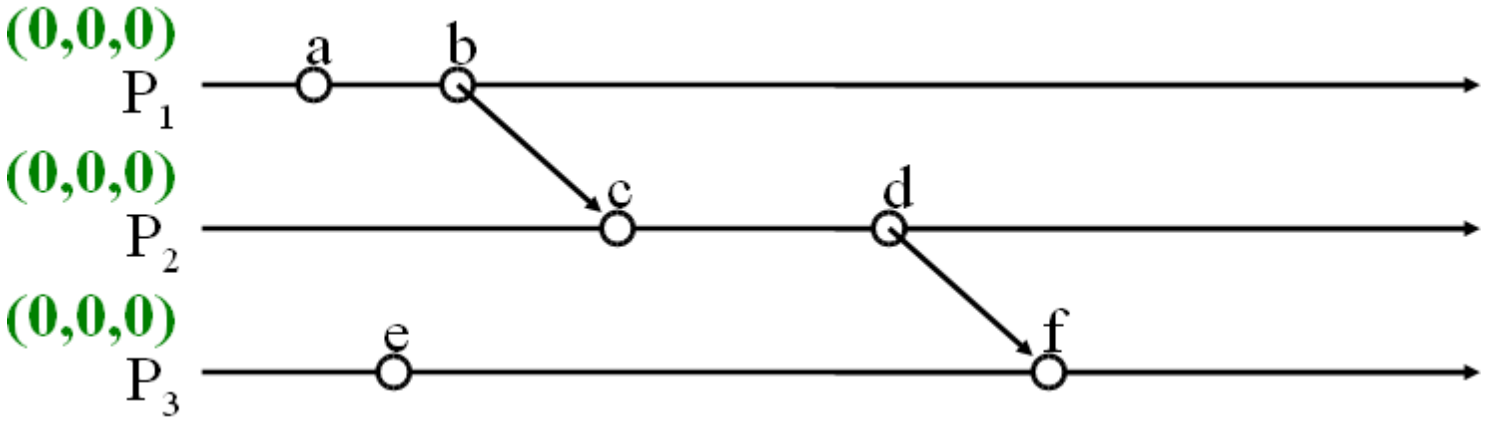
Διανυσματικά Ρολόγια (Vector Clocks)

- Το διάνυσμα αρχικοποιείται σε 0 σε κάθε διεργασία
  - $\mathbf{V_i[j] = 0}$ **for** $i, j = 1, \ldots, N$
- Η διεργασία επαυξάνει το στοιχείο του διανύσματος της σε τοπικό διάνυσμα πριν το γεγονός της χρονοσφράγισης.
- Το μήνυμα που στέλνεται από τη διεργασία P_i φέρει και διάνυσμα V_i.
- Όταν η Pj λαμβάνει το μήνυμα, συγκρίνει τα διανύσματα στοιχείο με στοιχείο και ορίζει τοπικό διάνυσμα μεγαλύτερο από αυτές τις δύο τιμές.
  - $\mathbf{V_j[i] = \max(V_i[i], V_j[i])}$ **for** $i = 1, \ldots, N$

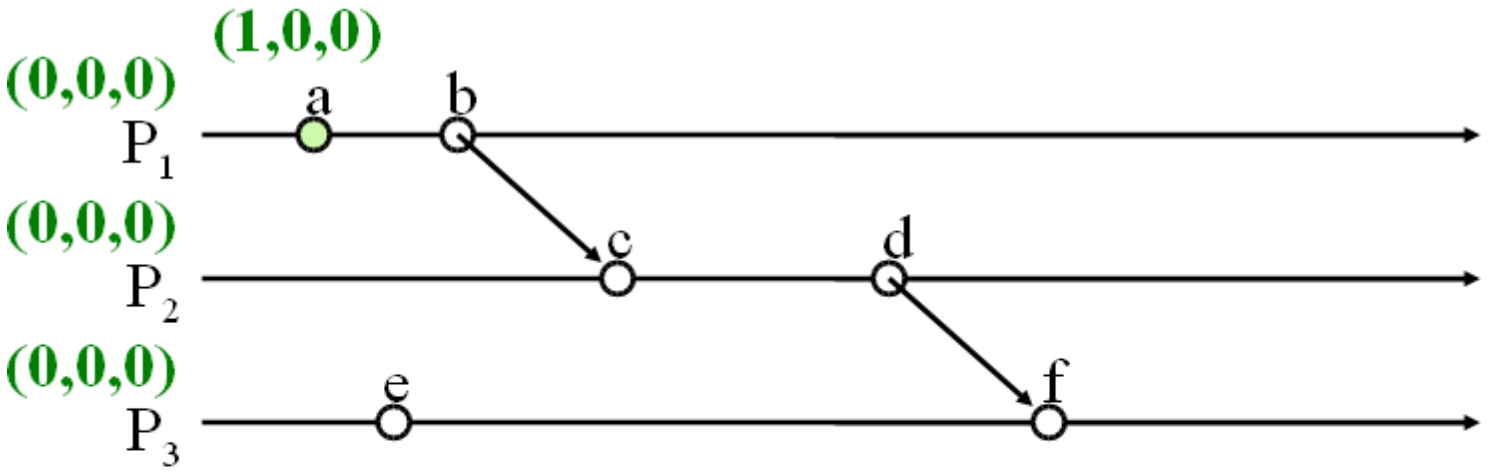
Συγκρίνοντας διανυσματικές χρονοσφραγίδες

- **Θέτω:**
- V = V’ εάν V [*i*] = V’[*i*] για κάθε *i* = 1, … , *N*
  V -> V’ εάν V [*i*] -> V’[*i*] για κάθε *i* = 1, … , *N*
  - Για δύο οποιαδήποτε γεγονότα e, e’.
    - Αν e -> e’ τότε V(e) < V(e’).
  - Ακριβώς όπως και στον αλγόριθμο του Lamport .
    - Αν V(e) < V(e’) τότε και e -> e’.
  - Δύο γεγονότα είναι **συντρέχοντα** αν δεν ισχύει.
    - V(e) -> V(e’) αλλά ούτε και V(e’) -> V(e).

Διανυσματικές Χρονοσφραγίδες (1/11)

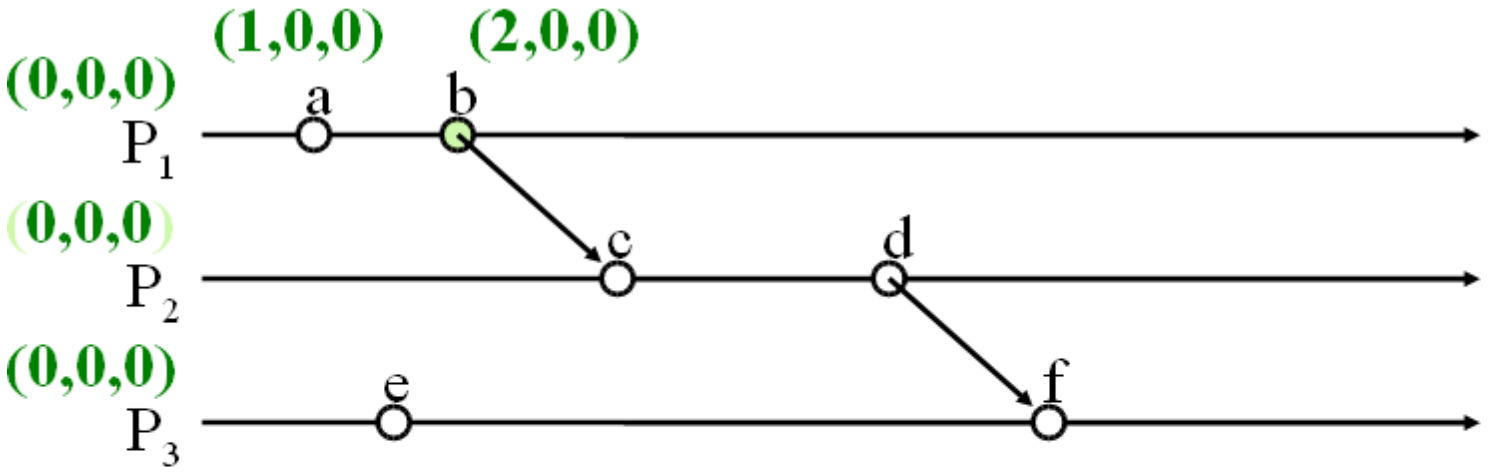


Διανυσματικές Χρονοσφραγίδες (2/11)



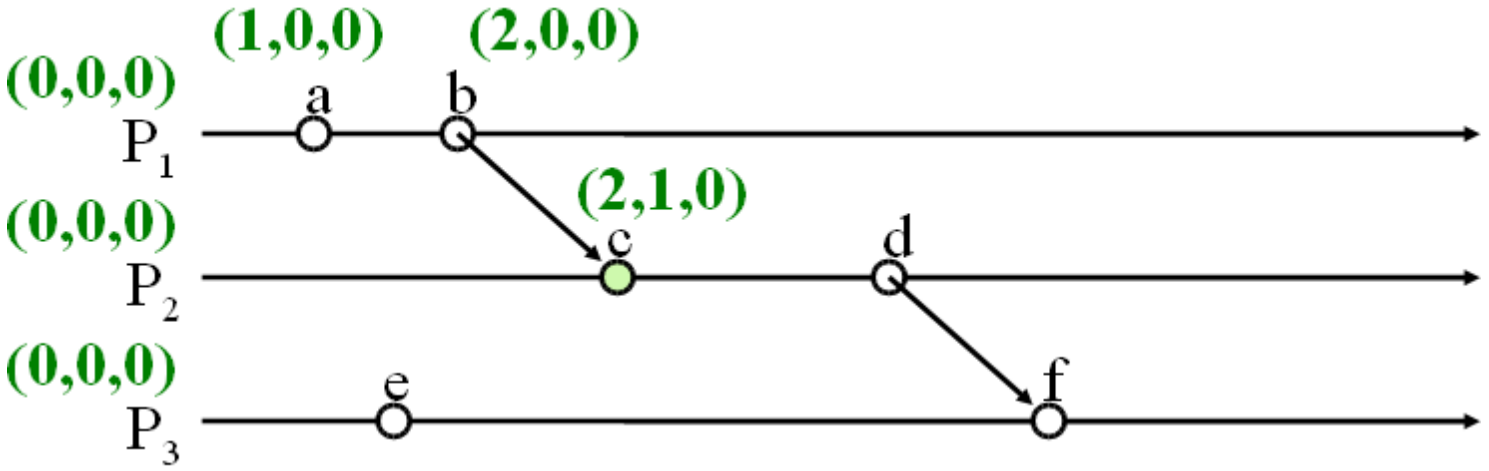
Γεγονός	Χρονοσφραγίδα
a	(1,0,0)

Διανυσματικές Χρονοσφραγίδες (3/11)



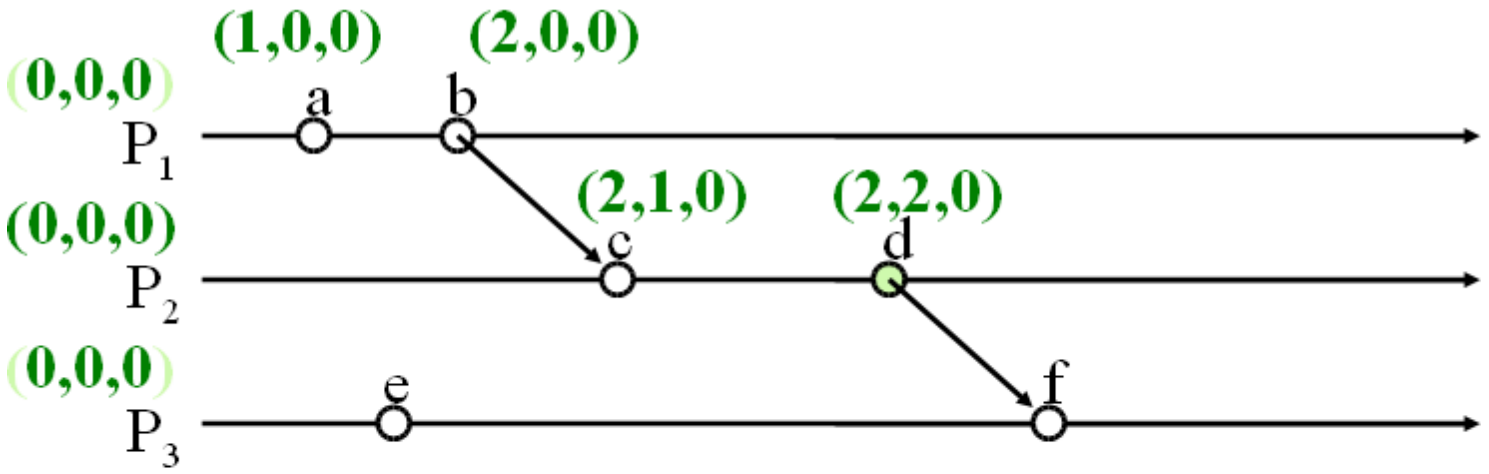
Γεγονός	Χρονοσφραγίδα
a	(1,0,0)
b	(2,0,0)

Διανυσματικές Χρονοσφραγίδες (4/11)



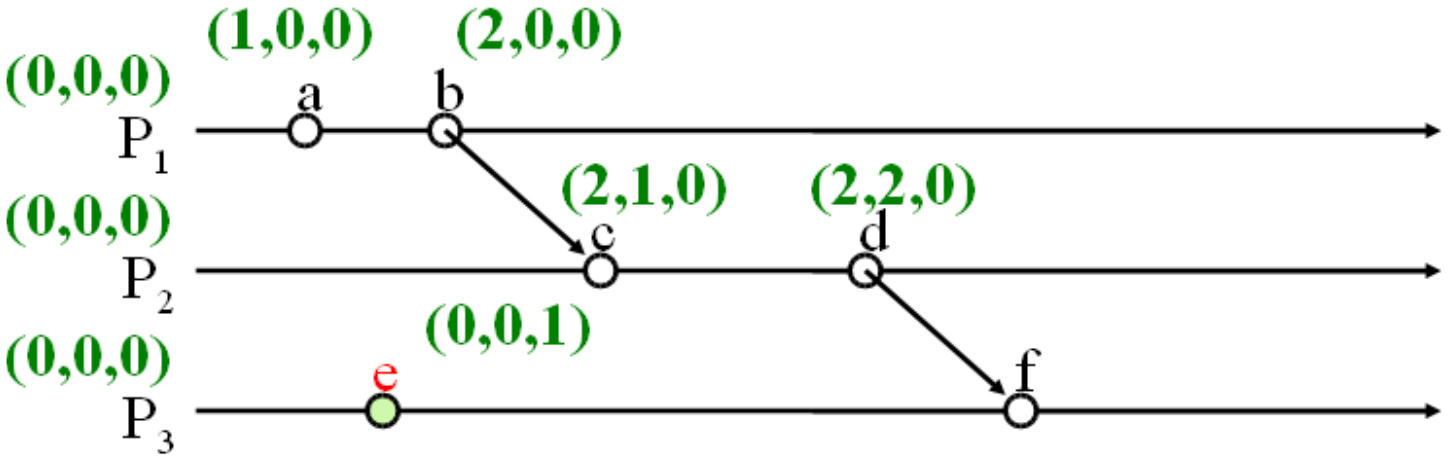
Γεγονός	Χρονοσφραγίδα
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)

Διανυσματικές Χρονοσφραγίδες (5/11)



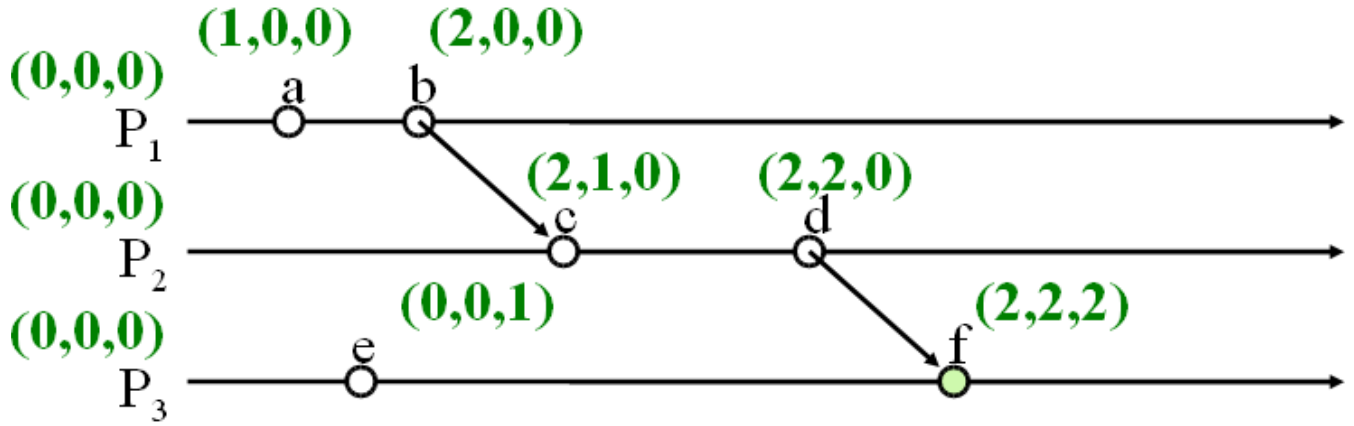
Γεγονός	Χρονοσφραγίδα
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)

Διανυσματικές Χρονοσφραγίδες (6/11)



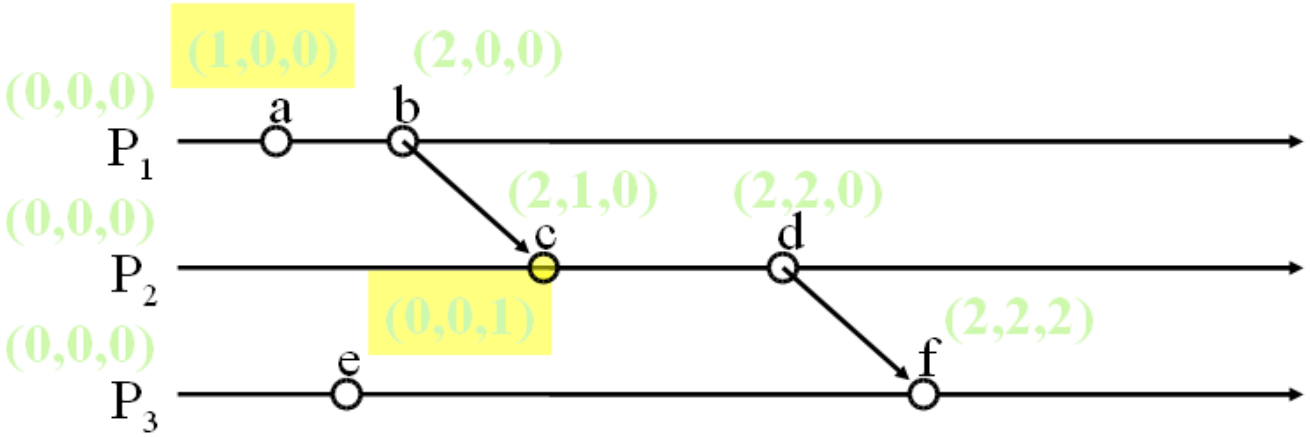
Γεγονός	Χρονοσφραγίδα
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)

Διανυσματικές Χρονοσφραγίδες (7/11)



Γεγονός	Χρονοσφραγίδα
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

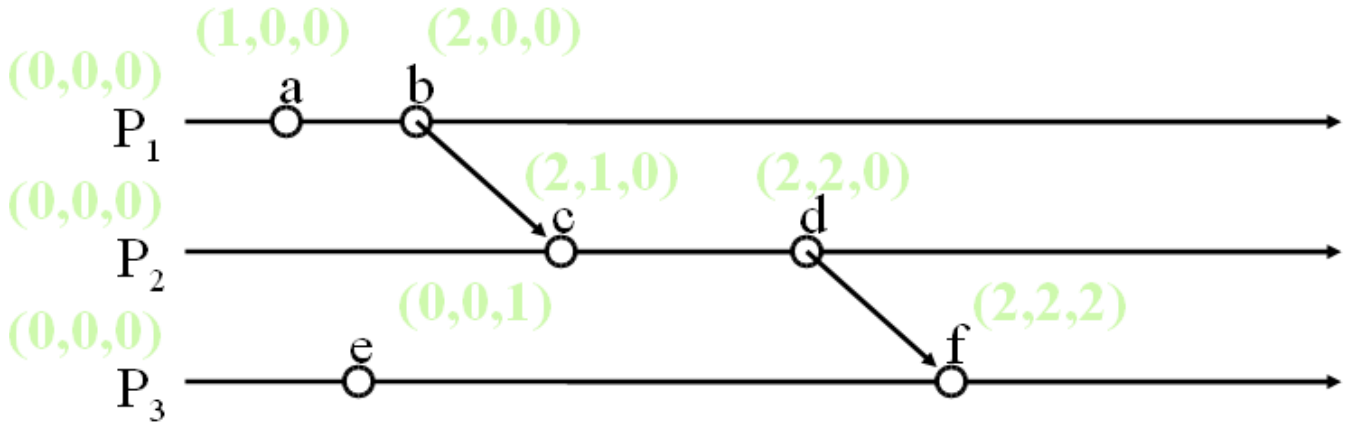
Διανυσματικές Χρονοσφραγίδες (8/11)



Γεγονός	Χρονοσφραγίδα
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

ταυτόχρονα γεγονότα

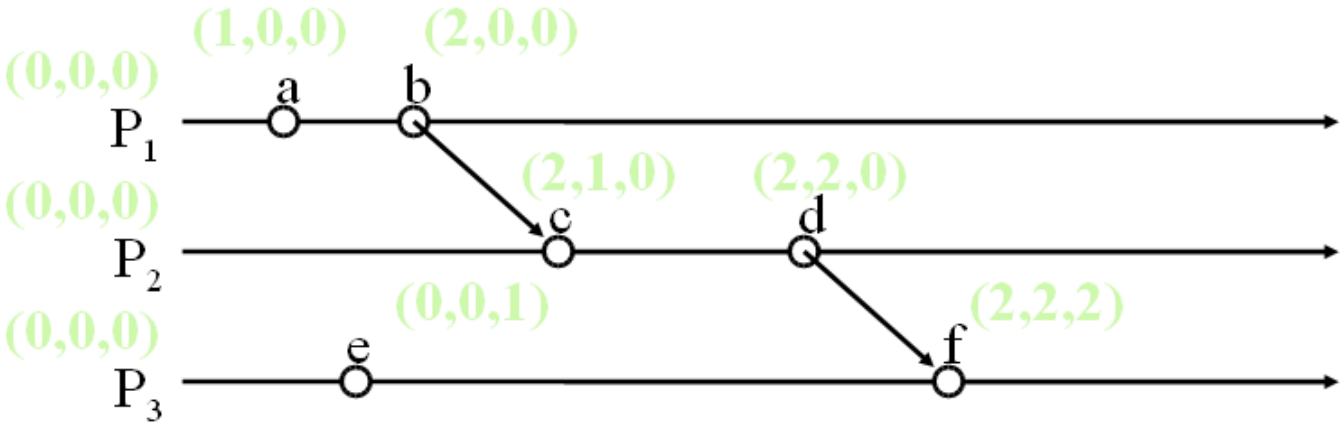
Διανυσματικές Χρονοσφραγίδες (9/11)



Γεγονός	Χρονοσφραγίδα
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

ταυτόχρονα γεγονότα

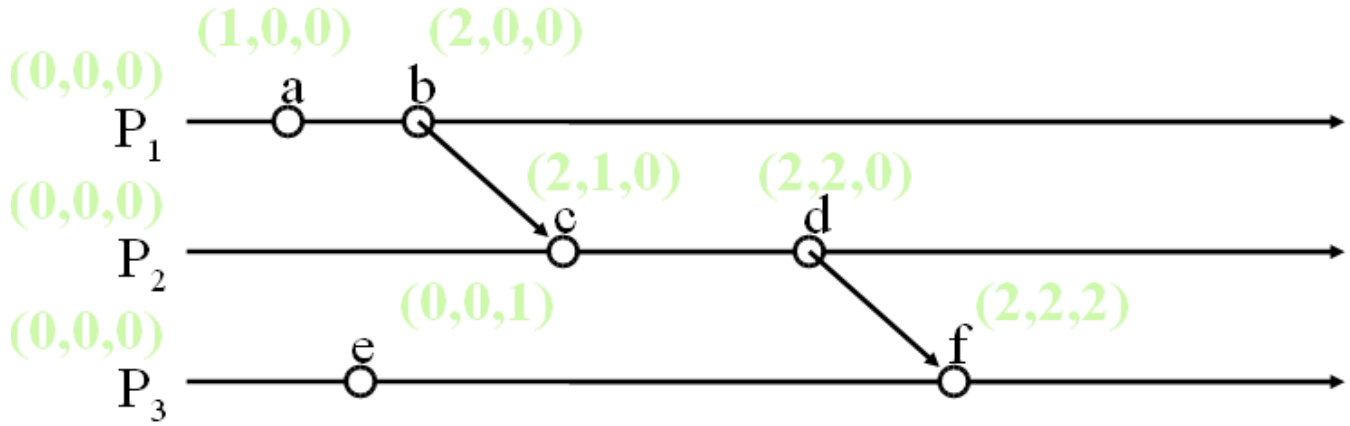
Διανυσματικές Χρονοσφραγίδες (10/11)



Γεγονός	Χρονοσφραγίδα
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

ταυτόχρονα γεγονότα

Διανυσματικές Χρονοσφραγίδες (11/11)



Γεγονός	Χρονοσφραγίδα
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

ταυτόχρονα γεγονότα

Σύνοψη: Λογικά Ρολόγια & Μερική Διάταξη

- **Αιτιότητα.**
  - Αν a → *b* τότε το γεγονός a μπορεί να επηρεάσει το γεγονός *b.*
- **Συγχρονισμός.**
  - Αν ούτε a → *b* αλλά ούτε και *b* → a τότε το ένα γεγονός δεν μπορεί να επηρεάσει το άλλο.
- **Μερική Διάταξη.**
  - Περιστασιακά γεγονότα βρίσκονται σε αλληλουχία.
- **Καθολική Διάταξη.**
  - Όλα τα γεγονότα βρίσκονται σε αλληλουχία.

Η ώρα epoch:

```
mdasyg@parsys:> date +%s
1621227336
```

```
mdasyg@parsys:> date
Mon May 17 07:55:41 EEST 2021
```

Η ώρα στο UNIX

- Μετριέται ως secs από την ημερομηνία epoch, (δηλαδή απο 1/1/1970).
- Θα εμφανιστεί σημαντικό πρόβλημα το 2038 { στις **03:14:07 UTC της Τρίτης, 19 Ιανουαρίου το 2038** } (θα κάνει overflow ο 32bit timer counter).
- Δεν γίνεται απλά να αλλάξει από 32 σε 64 bit ο timer => θα προκαλέσει σημαντική δυαδική ασυμβατότητα (incopatibility->incompatibility?)
- Προσπάθειες πάντως για να αλλάξει σε 64bit βρίσκονται ήδη σε εξέλιξη.
- Με την υιοθέτηση 64bit μετρητή δευτερολέπτων στους τελευταίους πυρήνες, το πρόβλημα αναβλήθηκε μέχρι: December 4, 292,277,026,596 AD.

Τέλος Ενότητας

Σημείωμα Αναφοράς

- Copyright Πανεπιστήμιο Δυτικής Μακεδονίας, Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών, Μηνάς Δασυγένης. «**Συστήματα Παράλληλης & Κατανεμημένης Επεξεργασίας**». Έκδοση: 1.0. Κοζάνη 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:
- https://eclass.uowm.gr/courses/ICTE268/

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
  - το Σημείωμα Αναφοράς
  - το Σημείωμα Αδειοδότησης
  - τη δήλωση Διατήρησης Σημειωμάτων
  - το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)
- μαζί με τους συνοδευόμενους υπερσυνδέσμους.