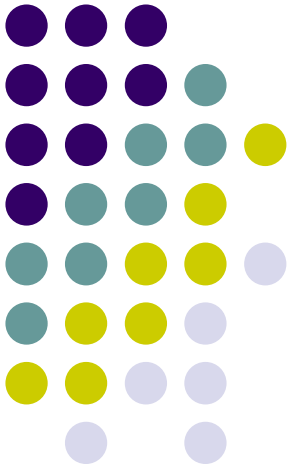
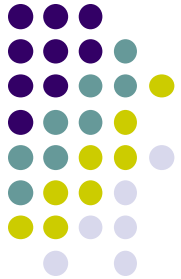
Κατανεμημένα Συστήματα

Μάθημα #6

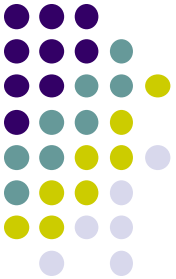


Περιεχόμενα



- Συγχρονισμός
- Φυσικά Ρολόγια
- Λογικά Ρολόγια
- Χρονοσφραγίδες Lamport
- Διανυσματικές χρονοσφραγίδες
- Διασφάλιση αιτιοκρατικής παράδοσης μηνυμάτων

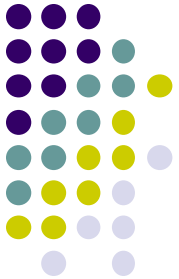
ΣΥΓΧΡΟΝΙΣΜΟΣ



Ανάγκη συγχρονισμού διεργασιών σε ένα κατανεμημένο σύστημα ώστε:

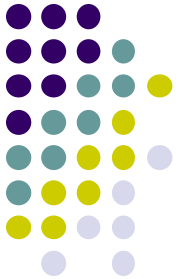
- **Να αποφεύγεται η ταυτόχρονη προσπέλαση σε κοινόχρηστους πόρους**
- **Να επιτυγχάνεται συμφωνία για τη χρονική σειρά των διαφόρων συμβάντων**

Συγχρονισμός



- Σε ένα συγκεντρωτικό σύστημα μια διεργασία Α για να μάθει την ώρα πραγματοποιεί μία κλήση συστήματος και ο πυρήνας μαθαίνει την ώρα.
- Αν η διεργασία Β ρωτήσει την ώρα μετά τη διεργασία Α θα πάρει τιμή μεγαλύτερη από αυτήν που πήρε η Α.
- Στα κατανεμημένα συστήματα η επίτευξη καθολικής συμφωνίας για την ώρα δεν είναι κάτι απλό.

Πραγματικά (Φυσικά) Ρολόγια

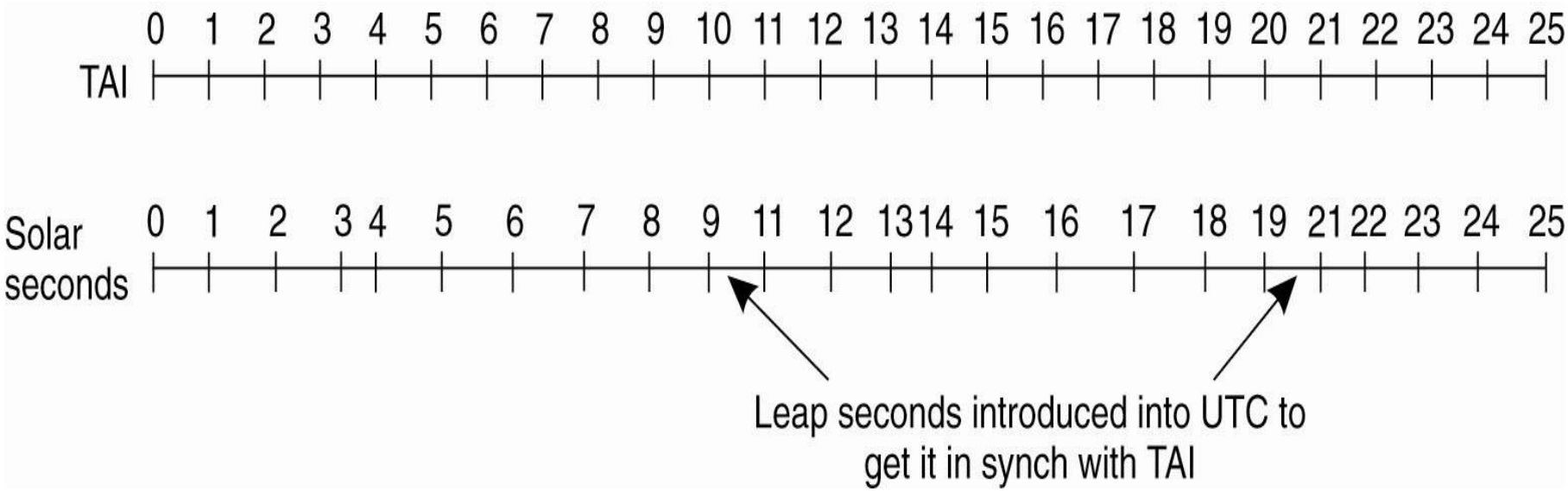


Χρονόμετρο υπολογιστή: Κρύσταλλος χαλαζία (high-frequency oscillator), μετρητής και καταχωρητής διατήρησης (holding register)

- **Κάθε ταλάντωση του κρυστάλλου μειώνει το μετρητή κατά ένα. Όταν ο μετρητής φθάσει στο 0, προκαλείται μία διακοπή και ο μετρητής επαναφορτώνεται από τον καταχωρητή διατήρησης. Κάθε διακοπή ονομάζεται χτύπος ρολογιού.**
- **Ένα χρονόμετρο μπορεί να προγραμματιστεί ώστε να προκαλεί διακοπή 60 (ή/και περισσότερες) φορές το δευτερόλεπτο.**
- **Σε ένα δίκτυο υπολογιστών, οι συχνότητες ταλάντωσης των κρυστάλλων των ρολογιών μπορεί να διαφέρουν (απόκλιση ρολογιού)**

- **Ατομικό ρολόι (1948): το δευτερόλεπτο ορίζεται ως το διάστημα που χρειάζεται το άτομο του καισίου 133 για να πραγματοποιήσει 9.192.631.770 μεταπτώσεις.**
- **TAI – International Atomic Time: 50 εργαστήρια στον κόσμο διαθέτουν ρολόγια καισίου και αναφέρουν το πλήθος των χτύπων των ρολογιών τους στο Διεθνές Γραφείο Ώρας το οποίο υπολογίζει το μέσο όρο χτύπων για να παράγει την TAI**
- **Η TAI δεν είναι απόλυτα συντονισμένη με την ηλιακή ώρα.**
- **Κάθε φορά που η διαφορά ανάμεσα στην TAI και την ηλιακή ώρα φτάνει τα 800 msec το Διεθνές Γραφείο Ώρας προσθέτει συμπληρωματικά δευτερόλεπτα (UTC – Universal Coordinated Time)**

Συμπληρωματικά δευτερόλεπτα (Leap Seconds)



TAI seconds are of constant length, unlike solar seconds. Leap seconds are introduced when necessary to keep in phase with the sun.

UTC (Universal Coordinated Time) is TAI with leap seconds.

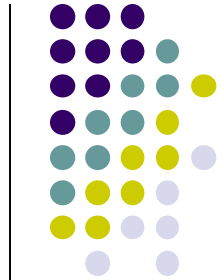
Υπηρεσίες UTC προσφέρουν:

- **Ραδιοσταθμοί βραχέων κυμάτων σε διάφορες χώρες (ΗΠΑ -WWV,Αγγλία -MSF, κ.λπ.) που εκπέμπουν ένα σύντομο παλμό στην αρχή κάθε δευτερολέπτου UTC**
- **Αρκετοί δορυφόροι της γης**

Και στις δύο περιπτώσεις απαιτείται ακριβής γνώση της σχετικής θέσης του πομπού και του δέκτη ώστε να συνεκτιμηθεί η καθυστέρηση της διάδοσης του σήματος

Απαιτείται η μηχανή να διαθέτει αντίστοιχους δέκτες (π.χ., δέκτη WWV)

Συγχρονισμός ρολογιών



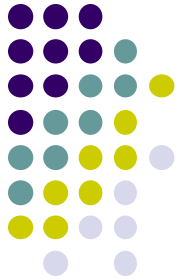
Αν μια μηχανή διαθέτει δέκτη WWV, ο στόχος είναι να παραμείνουν όλα τα άλλα μηχανήματα συγχρονισμένα με αυτήν.

Αν καμία μηχανή δεν διαθέτει δέκτη WWV, ο στόχος είναι να διατηρηθούν όλες οι μηχανές συγχρονισμένες όσο το δυνατόν καλύτερα

Έστω C η τιμή του ρολογιού μιας μηχανής m. Αν η UTC ώρα είναι t, η τιμή του ρολογιού της m είναι $C_m(t)$. Ιδανικά, για κάθε m,

$$C_m(t)=t, \text{ ήτοι, } dC/dt=1$$

Συγχρονισμός ρολογιών



Έστω μία **σταθερά ρ** (η οποία δίνεται συνήθως από τον κατασκευαστή – π.χ. 10^{-6}sec/sec) τέτοια ώστε:

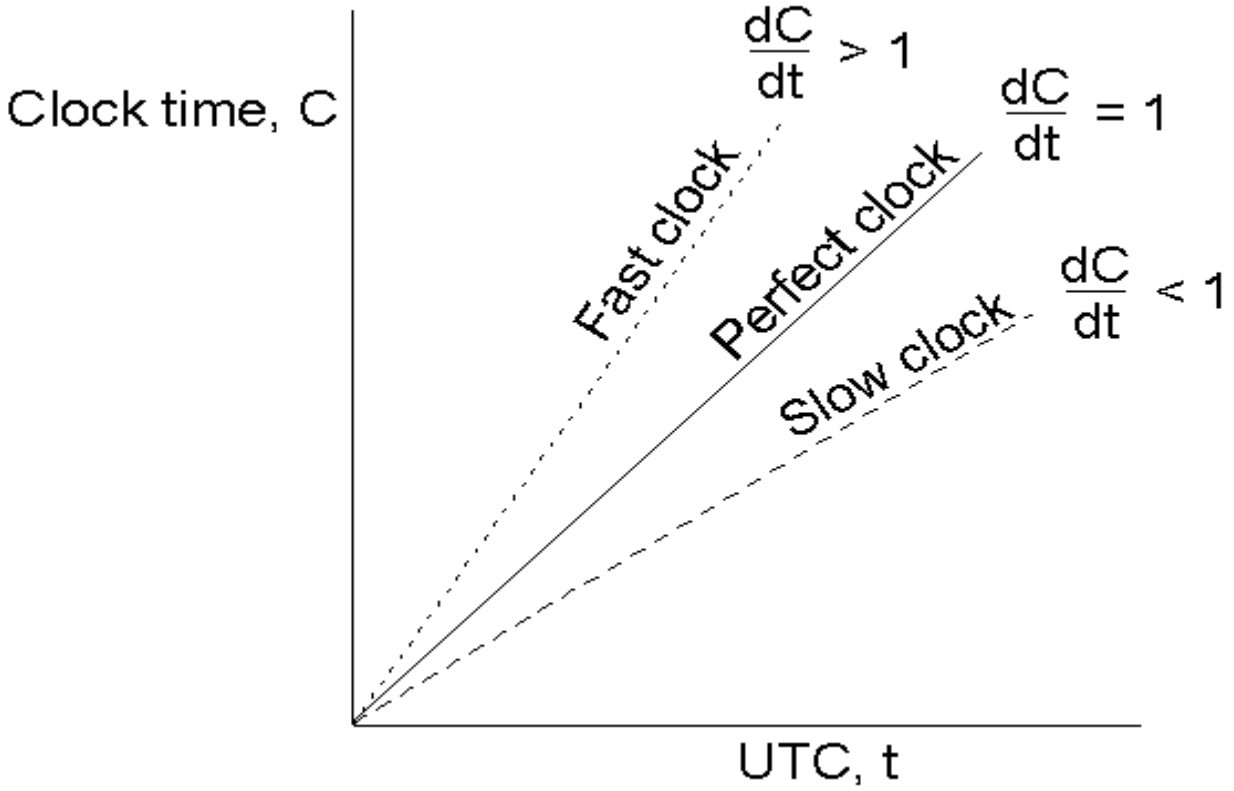
$$1-\rho <= dC/dt <= 1+\rho$$

λέμε ότι το χρονόμετρο δουλεύει ‘ορθά’ ή αλλιώς ‘μέσα στο πλαίσιο των προδιαγραφών του’.

Η ρ καλείται **μέγιστος ρυθμός απόκλισης**.

- Αν δύο ρολόγια αποκλίνουν προς την αντίθετη κατεύθυνση από την ώρα UTC η διαφορά τους τη στιγμή Δt μπορεί να είναι μέχρι $2\rho\,\Delta t$

- Για να μη διαφέρουν περισσότερο από δ δευτερόλεπτα πρέπει να επανασυγχρονίζονται τουλάχιστον κάθε $\delta/2\rho$ δευτερόλεπτα



Σχέση μεταξύ της ώρας ρολογιού και της ώρας UTC

Ο Αλγόριθμος συγχρονισμού ρολογιών του Christian

- Είναι κατάλληλος όταν μία μηχανή διαθέτει δέκτη WWV (διακομιστής ώρας) και θέλουμε όλες οι άλλες μηχανές να είναι συγχρονισμένες με αυτήν (παρόμοια για δέκτη GPS)

Αλγόριθμος Christian:

Περιοδικά (κάθε $\delta/2\rho$ δευτερόλεπτα)

1. Κάθε μηχανή (αποστολέας) ρωτάει τον διακομιστή ώρας για την τρέχουσα ώρα
2. Ο διακομιστής στέλνει την τρέχουσα ώρα C_{UTC}
3. Ο αποστολέας *ρυθμίζει το ρολόι του σύμφωνα με την* C_{UTC}

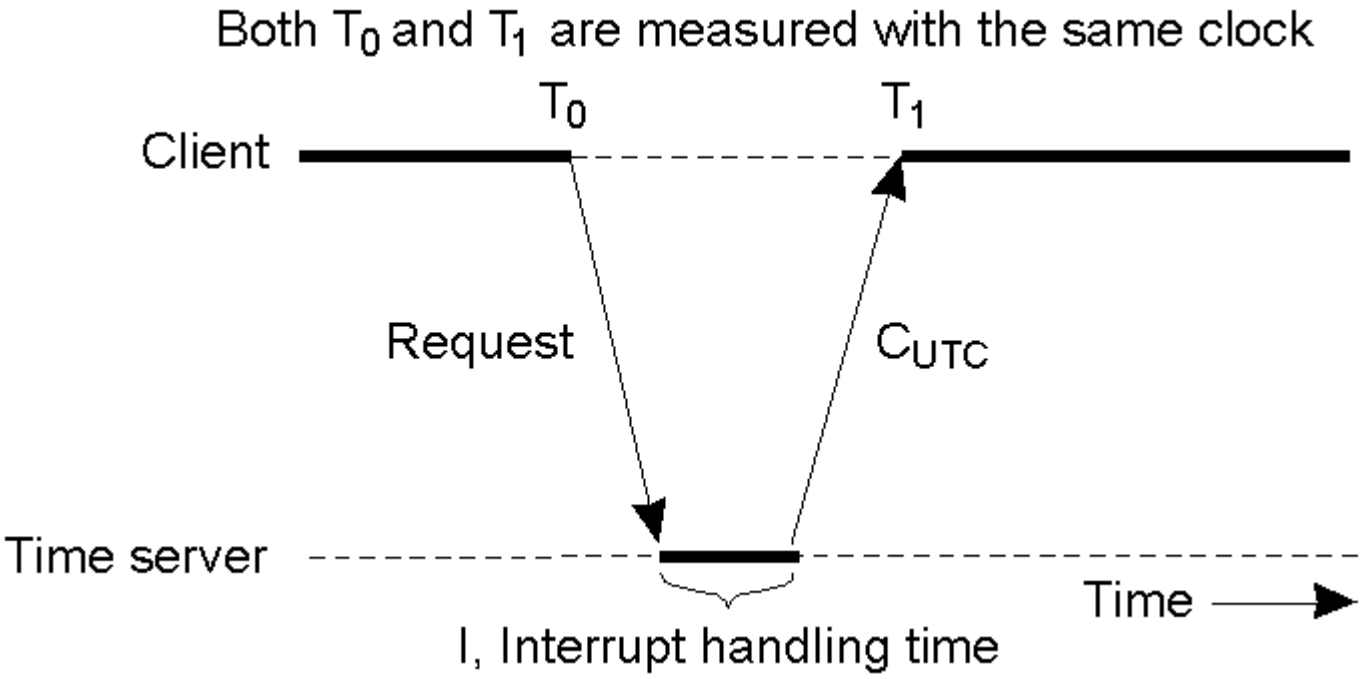
Πρόβλημα 1: **Αν το ρολόι του αποστολέα είναι γρήγορο, η C_{UTC} θα είναι μικρότερη από την τρέχουσα ώρα του αποστολέα.**

Γίνεται βαθμιαία ρύθμιση της ώρας του ρολογιού του αποστολέα.

Πρόβλημα 2: **Πρέπει να ληφθεί υπόψη ο χρόνος μετάδοσης του μηνύματος του διακομιστή.**

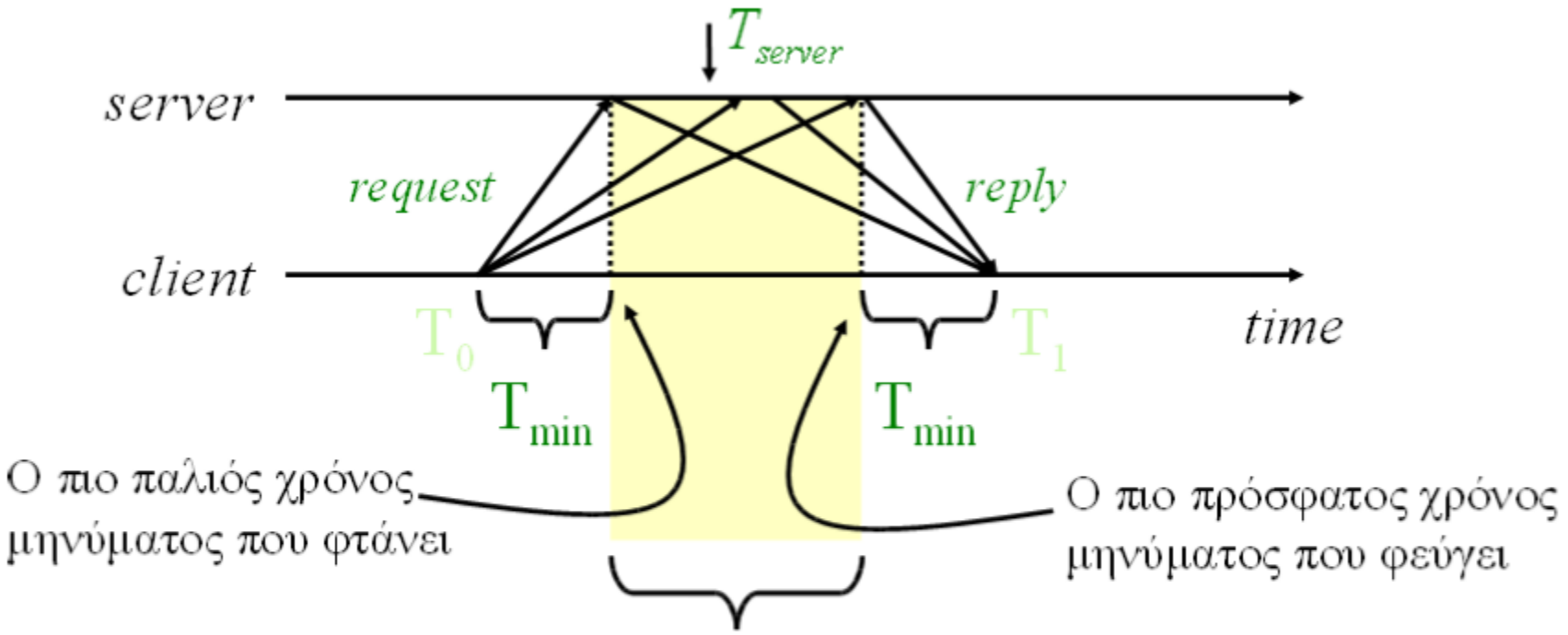
Γίνεται εκτίμηση του χρόνου μετάδοσης (π.χ., (T1-T0)/2) και η εκτίμηση αυτή προστίθεται στην C_{UTC}

Ο Αλγόριθμος συγχρονισμού ρολογιών του Christian



Λήψη της τρέχουσας ώρας από έναν διακομιστή ώρας

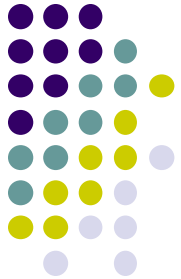
Ο Αλγόριθμος συγχρονισμού ρολογιών του Christian



$$\text{εύρος} = T_1 - T_0 - 2T_{min}$$

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

Ο Αλγόριθμος συγχρονισμού ρολογιών του Christian

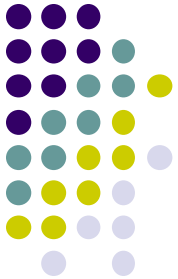


- Χρησιμοποιεί time server για τον συγχρονισμό – είναι σχεδιασμένος κυρίως για LAN γιατί βασίζεται αυστηρά στην καλή εκτίμηση του round trip time
- Ο χρόνος μετάδοσης μετ' επιστροφής (Round Trip Time – RTT) εισάγει καθυστέρηση που πρέπει να εκτιμηθεί
- Εκτίμηση για την καθυστέρηση μιας διαδρομής - ο client μπορεί να θέσει το ρολόι του σε *T+ RTT/2*
- *Αν μπορεί να εκτιμηθεί ο χρόνος επεξεργασίας της διακοπής στον server (I) το ρολόι μπορεί υπό προϋποθέσεις να τεθεί σε T+(RTT-I)/2*
- Στατιστική επεξεργασία (επαναλαμβανόμενες μετρήσεις, διόρθωση εκτιμήσεων)

Ο Αλγόριθμος συγχρονισμού ρολογιών του Christian

- Αποστολή αιτήματος στις 5:08:15.100 (T0)
- Λήψη απάντησης στις 5:08:15.900 (T1)
- Η απάντηση είναι 5:09:25.300 (T)
- RTT = T1 -T0 = 5:08:15.900 - 5:08:15.100 = 800 ms
- Βασική Εκτίμηση: T+ RTT/2 = 5:09:25.300 + 400 = 5:09.25.700
- Αν έχουμε εκτίμηση για I = 50ms μπορούμε υπό προϋποθέσεις να θέσουμε την ώρα σε T+ (RTT-I)/2
- Εναλλακτική Εκτίμηση: 5:09:25.300 + 375 = 5:09.25.675

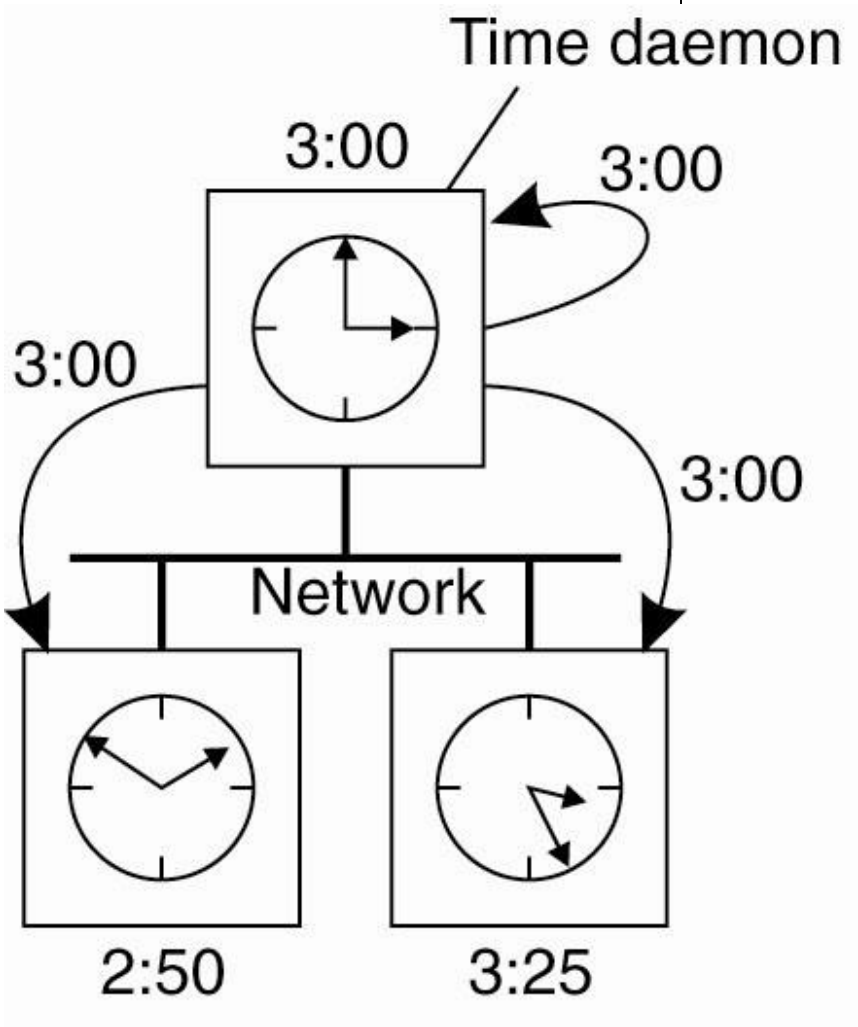
Ο αλγόριθμος Berkeley



- **Είναι κατάλληλος όταν καμία μηχανή δε διαθέτει δέκτη WWV/GPS**
- Ο διακομιστής (δαίμονας ώρας) διενεργεί περιοδικούς ελέγχους σε κάθε μηχανή για να πληροφορηθεί την ώρα της.
- Βάσει των απαντήσεων, υπολογίζει **μια μέση ώρα** και δίνει εντολή σε όλες τις μηχανές να ρυθμίσουν τα ρολόγια τους σύμφωνα με την καινούργια ώρα

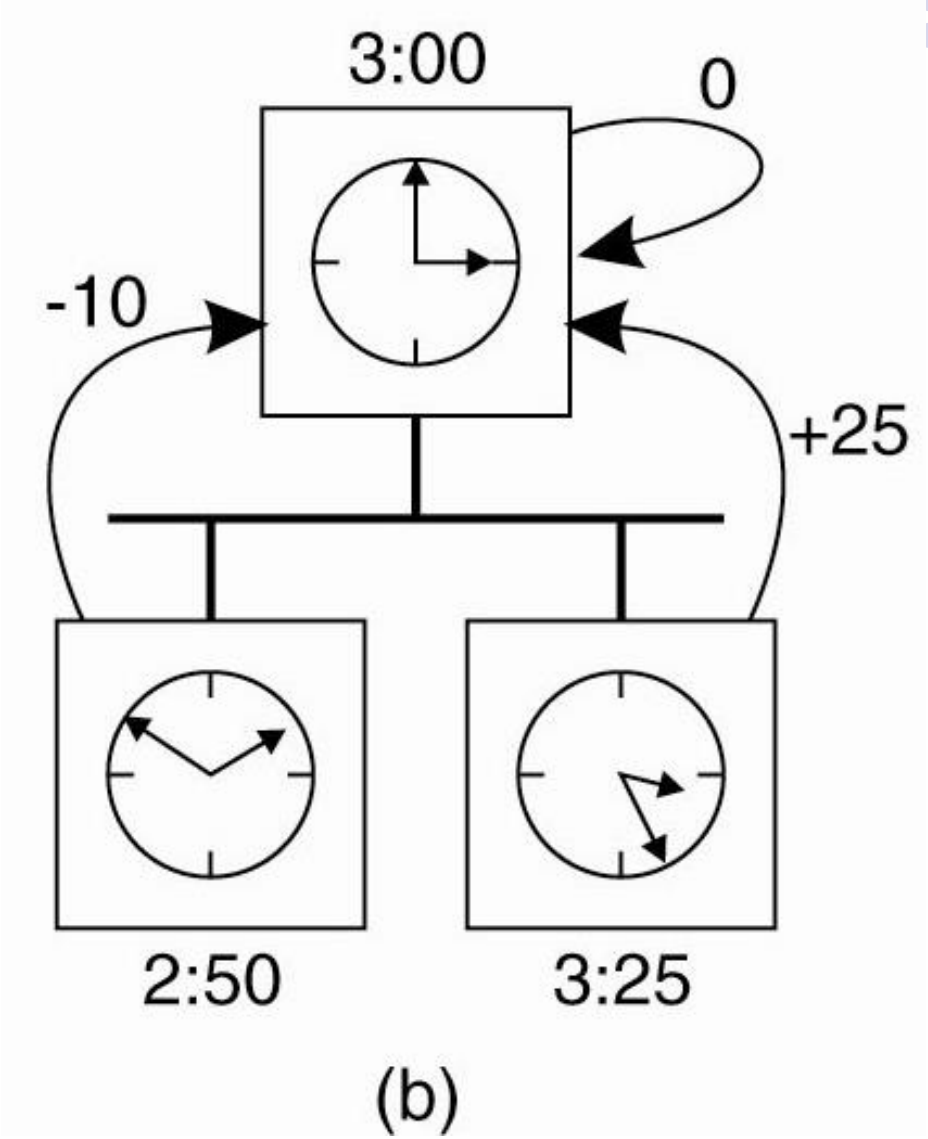
Ο αλγόριθμος Berkeley

- Αρχικά, ο time daemon ρωτά όλες τις άλλες μηχανές σχετικά με τις τιμές των ρολογιών τους



Ο αλγόριθμος Berkeley

- Οι μηχανές απαντούν (offsets)

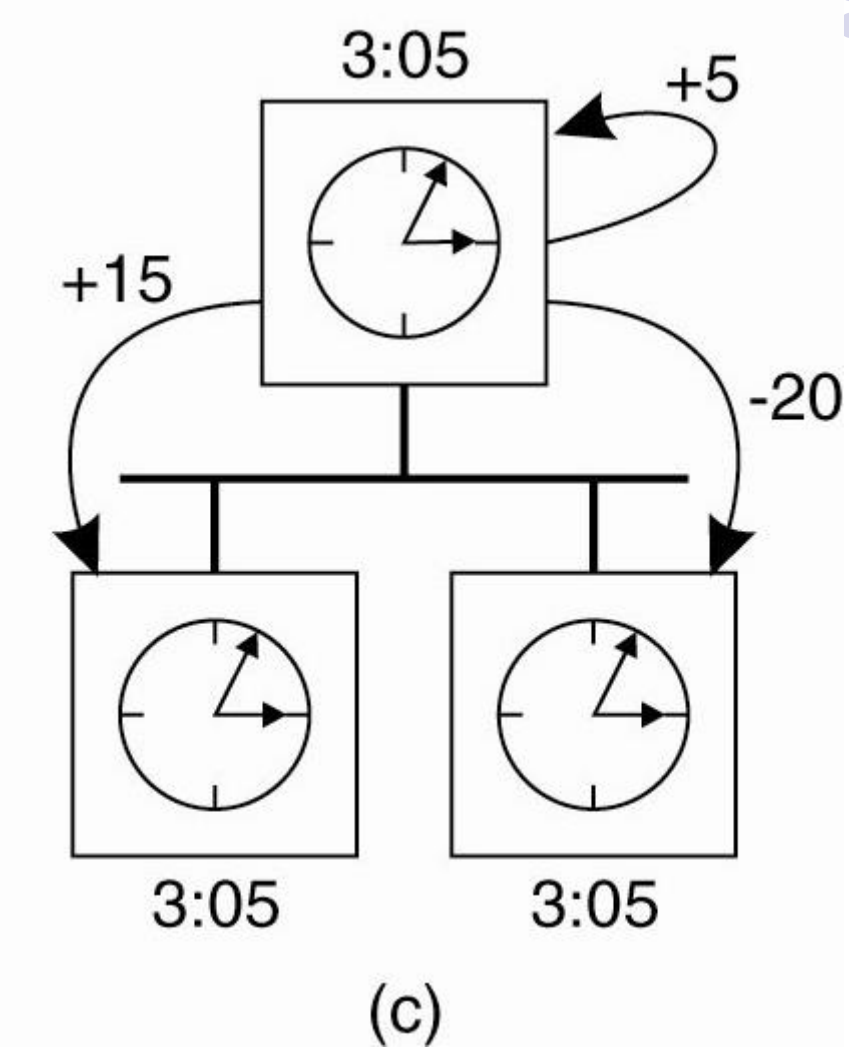


Ο αλγόριθμος Berkeley

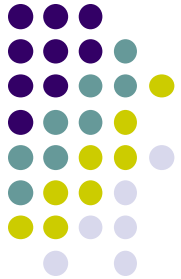
Ο time daemon υπαγορεύει σε όλους πώς να προσαρμόσουν το ρολόι τους, με βάση τη μέση τιμή που υπολόγισε

(Average = 3:05)

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

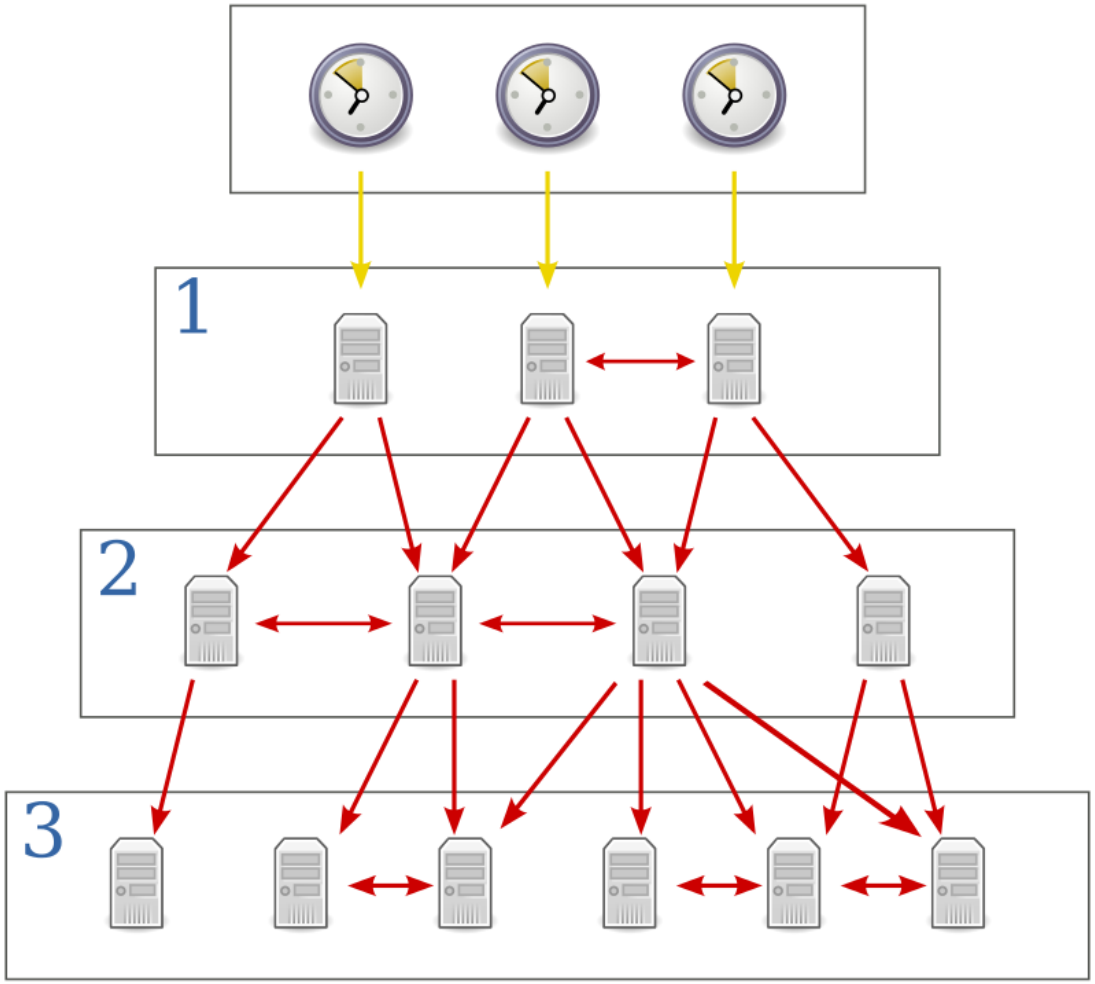


Το πρωτόκολλο NTP

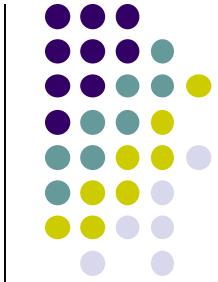


- Από τα πιο παλιά πρωτόκολλα του Internet. Όλοι οι κόμβοι συμμετέχουν στο συγχρονισμό του δικτύου.
- Ακρίβεια σε μερικά microseconds. Εκτός από την εκτίμηση του RTT, λαμβάνεται επίσης υπόψη **η διαφοροποίηση απόκρισης λόγω δικτύου**
- Το NTP χρησιμοποιεί UTC. Χρησιμοποιεί τη θύρα UDP 123. Internet Standard, version 3: RFC 1305.
- Χρησιμοποιεί δίκτυο από time servers για τον συγχρονισμό
- ‘Εχει σχεδιαστεί για το διαδίκτυο. Οι time servers είναι συνδεδεμένοι σε ένα **δέντρο συγχρονισμού**
- Η ρίζα είναι σε επαφή με το UTC
- Κάθε κόμβος συγχρονίζει τα παιδιά του

Το πρωτόκολλο NTP



ΛΟΓΙΚΑ ΡΟΛΟΓΙΑ



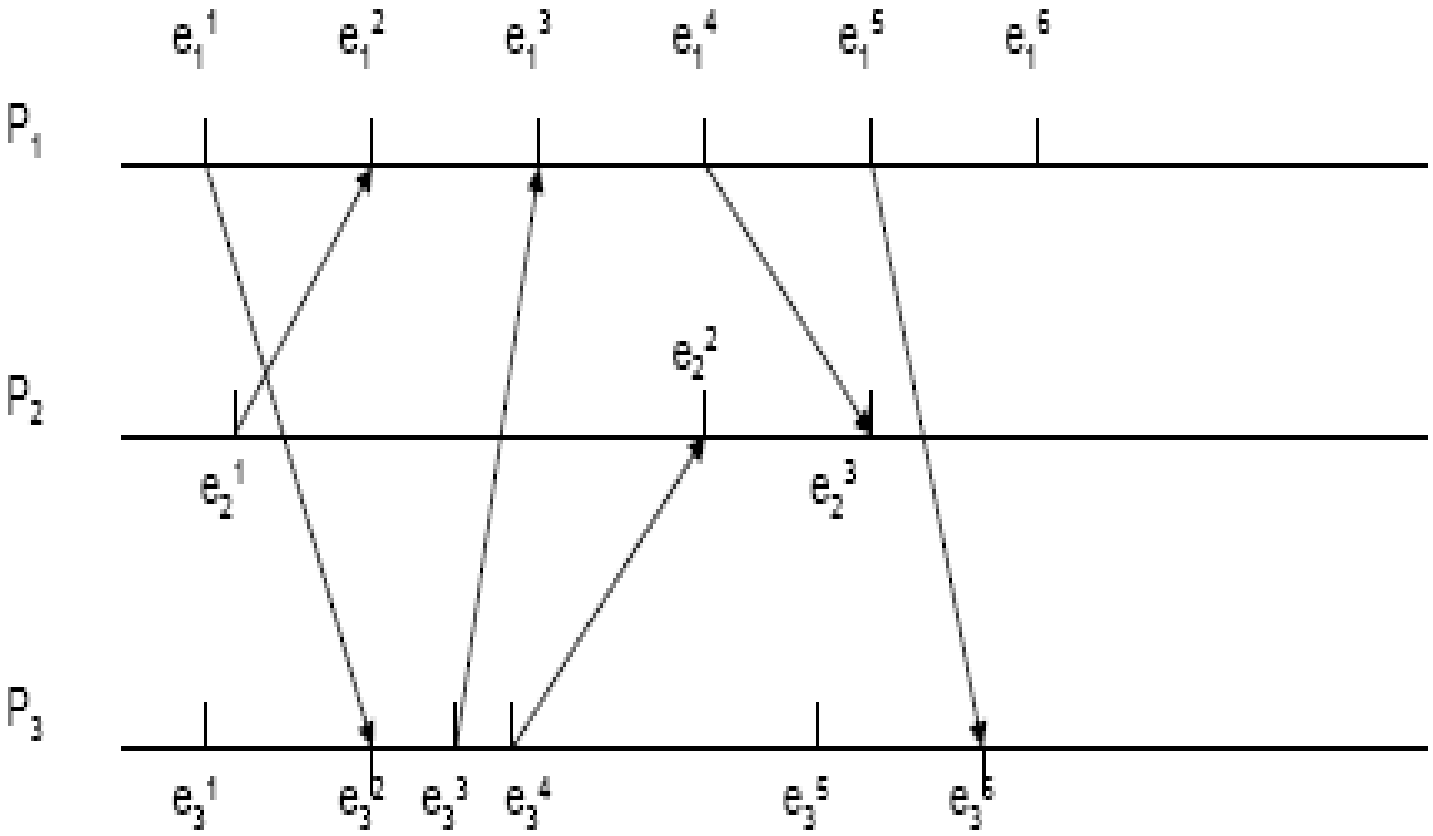
Ένας κατανεμημένος υπολογισμός περιγράφει την εκτέλεση ενός συνόλου ακολουθιακών διεργασιών

Κάθε διεργασία συνίσταται στην εκτέλεση μιας ακολουθίας γεγονότων. Δύο είδη γεγονότων:

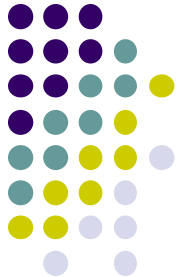
- ***Εσωτερικό γεγονός ή γεγονός υπολογισμού:*** προκαλεί αλλαγή της κατάστασης της διεργασίας (αλλαγή τιμών τοπικών μεταβλητών)
- ***Γεγονός επικοινωνίας*** *με κάποια άλλη διεργασία:* προκαλεί μετακίνηση ενός μηνύματος *m* από τον outbuf του αποστολέα στον inbuf του παραλήπτη μέσω των ***send(m)*** **και** ***receive(m)***

Διάγραμμα χώρου-χρόνου: απεικόνιση ενός κατανεμημένου υπολογισμού μέσα στο χρόνο

Διάγραμμα χώρου-χρόνου (state-time diagram)



Πρότυπο Κατανεμημένου Υπολογισμού



- Η **τοπική ιστορία** μιας διεργασίας P_i συμβολίζεται με
  $$h_i = e_i^1 e_i^2 \cdots e_i^k$$
  και αποτελεί την ακολουθία γεγονότων που έχουν εκτελεστεί στην P_i
- Η τοπική ιστορία είναι **χρονικά πλήρως διατεταγμένη**.

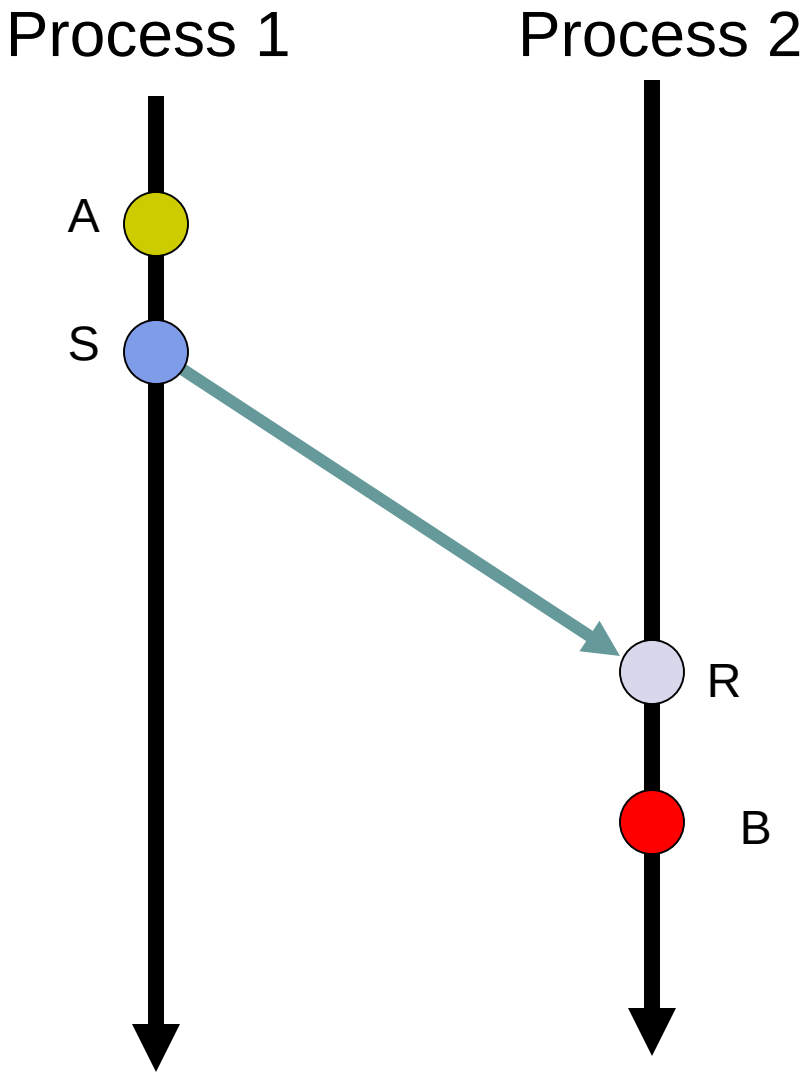
- Η **καθολική ιστορία** Η ενός κατανεμημένου υπολογισμού ορίζεται ως η ένωση των τοπικών ιστοριών όλων των διεργασιών που συμμετέχουν σε αυτόν, δηλ.,
  $$H = h_1 U\ h_2 U\ \ldots U\ h_n$$
- Η καθολική ιστορία είναι μόνο **μερικά διατεταγμένη**

Πως μπορούν τα στοιχεία της Η να διαταχθούν?

Αιτιότητα

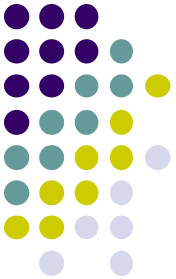
Αν οι διεργασίες 1 και 2 δεν επικοινωνούν ποτέ δεν μπορούμε να πούμε αν το γεγονός A συμβαίνει πριν ή μετά το B. Δηλ., δεν μπορούμε να ορίσουμε διάταξη μεταξύ τους.

Η επικοινωνία προσδιορίζει διάταξη



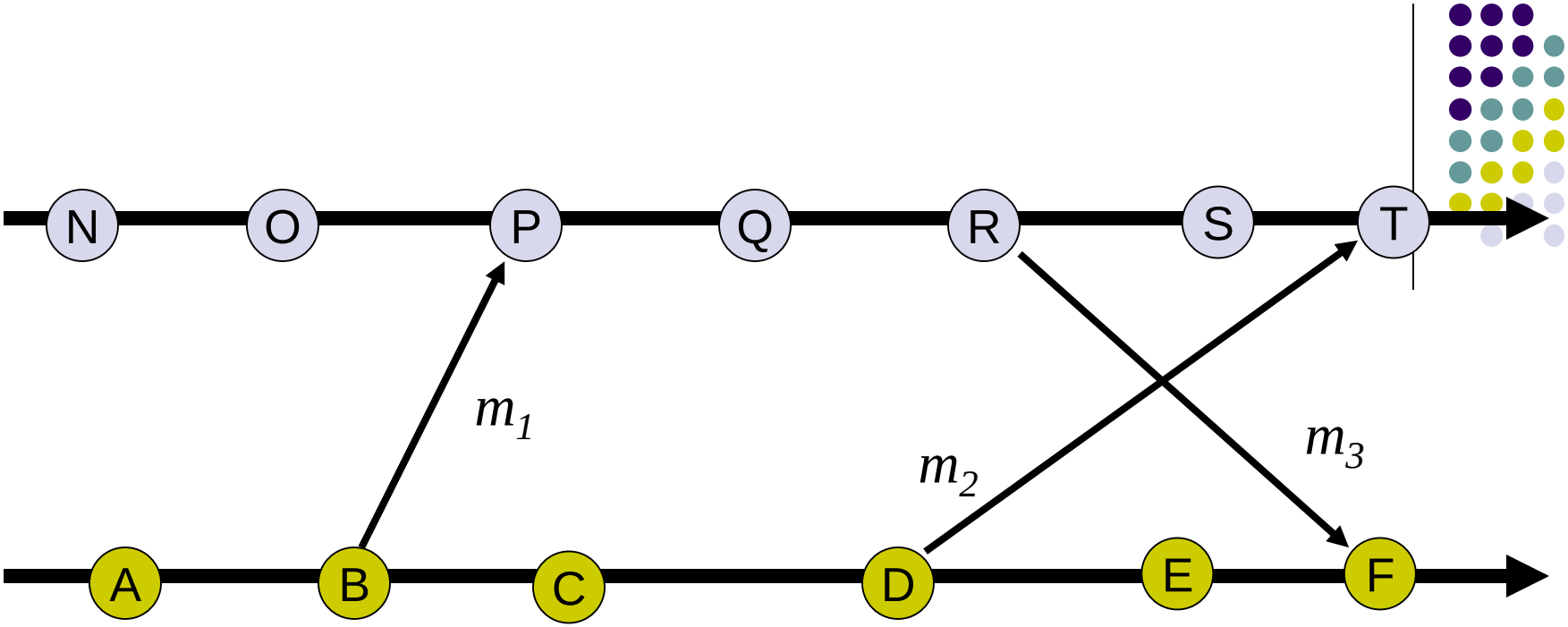
- Αν η Διεργασία 1 στείλει ένα μήνυμα (event S) το οποίο παραλαμβάνεται από τη Διεργασία 2 (event R), τι μπορούμε να θεωρήσουμε σχετικά με τη διάταξη των γεγονότων A, B, S, και R?
- Βασική παρατήρηση: Αν η Διεργασία 1 στέλνει ένα μήνυμα στη Διεργασία 2, ξέρουμε ότι από φυσικής άποψης η αποστολή σίγουρα προηγείται της παραλαβής.

Η σχέση συμβαίνει-πριν



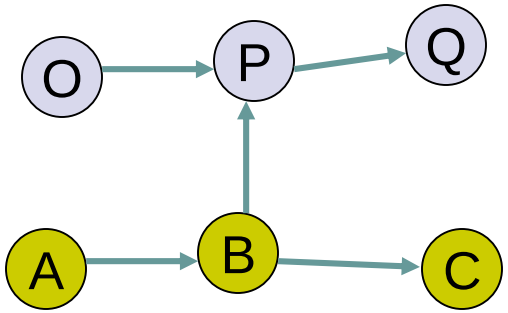
- Αν τα α και β είναι γεγονότα στην ίδια διεργασία και το α **συμβαίνει πριν** από το β τότε η σχέση **α *συμβαίνει πριν* το β** ήτοι **α → β είναι αληθής** και λέμε ότι ***«το α αιτιωδώς προηγείται του β»***
- Αν α είναι η αποστολή μηνύματος από μια διεργασία και β είναι η λήψη του μηνύματος από μια άλλη διεργασία τότε η σχέση α → β είναι αληθής.
- Αν τα α και β λάβουν χώρα σε δύο διεργασίες που δεν επικοινωνούν ποτέ τότε οι σχέσεις α → β και β → α δεν είναι αληθείς. Τα συμβάντα αυτά λέμε ότι είναι ***ταυτόχρονα ή σύνδρομα (concurrent)*** και γράφουμε **α || β**.

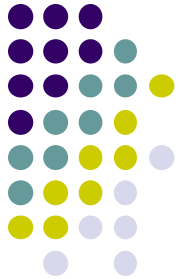
Η σχέση είναι **μεταβατική**, ήτοι αν α → β και β → γ τότε α → γ



- B ? P
- A ? N
- A ? C
- N ? O

- A ? Q
- Q ? C
- O ? C
- S ? E

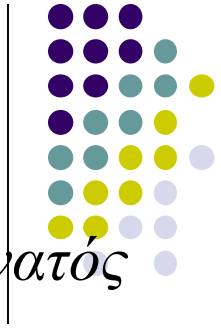




Αν το κατανεμημένο σύστημα διαθέτει ένα καθολικό πραγματικό ρολόι (*συγχρονισμός των ρολογιών* όλων των διεργασιών) τότε ισχύει πάντα ότι για οποιαδήποτε δύο γεγονότα α, β

Αν α → β τότε Time(α) < Time(β)

Λογικά Ρολόγια



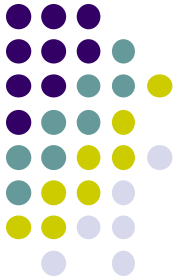
Lamport (1978): *Παρόλο που ο συγχρονισμός ρολογιών είναι δυνατός δεν είναι απαραίτητο να είναι απόλυτος*

Συνήθως, έχει **σημασία η συμφωνία όλων των διεργασιών ως προς τη σειρά με την οποία λαμβάνουν χώρα τα διαφορετικά γεγονότα** και όχι η συμφωνία τους για την ακριβή ώρα.

Λογικά ρολόγια: τα ρολόγια που σχετίζονται με αλγορίθμους για τους οποίους σημασία έχει η εσωτερική συνέπεια των ρολογιών και όχι αν αυτά είναι κοντά στην πραγματική ώρα.

Βασική Ιδέα: **Αν το γεγονός A συμβαίνει πριν το B, τότε θα πρέπει: ο χρόνος LT (Logical Time) του A να είναι μικρότερος από τον χρόνο LT (Logical Time) του B, ήτοι LT(A) < LT(B).**

Η σχέση συμβαίνει-πριν

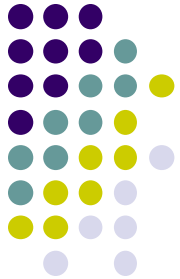


- Αν α και β είναι δύο γεγονότα μέσα στην ίδια διεργασία και το α λάβει χώρα πριν από το β τότε LT(α) < LT(β).

- Αν α είναι η αποστολή μηνύματος από μια διεργασία και β είναι η λήψη του μηνύματος από μια άλλη διεργασία τότε τα LT(α), LT(β) πρέπει να οριστούν έτσι ώστε κάθε διεργασία να συμφωνεί ότι LT(α) < LT(β).

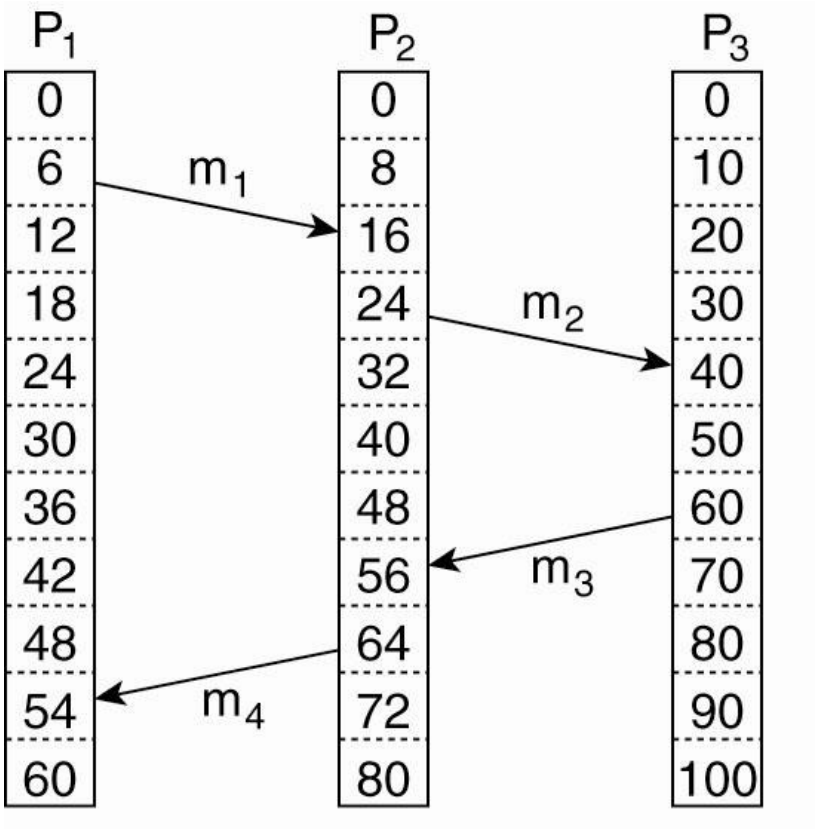
- Αν τα α και β είναι ταυτόχρονα τότε δεν ισχύει αναγκαστικά κάποια σχέση μεταξύ των LT(α), LT(β) .

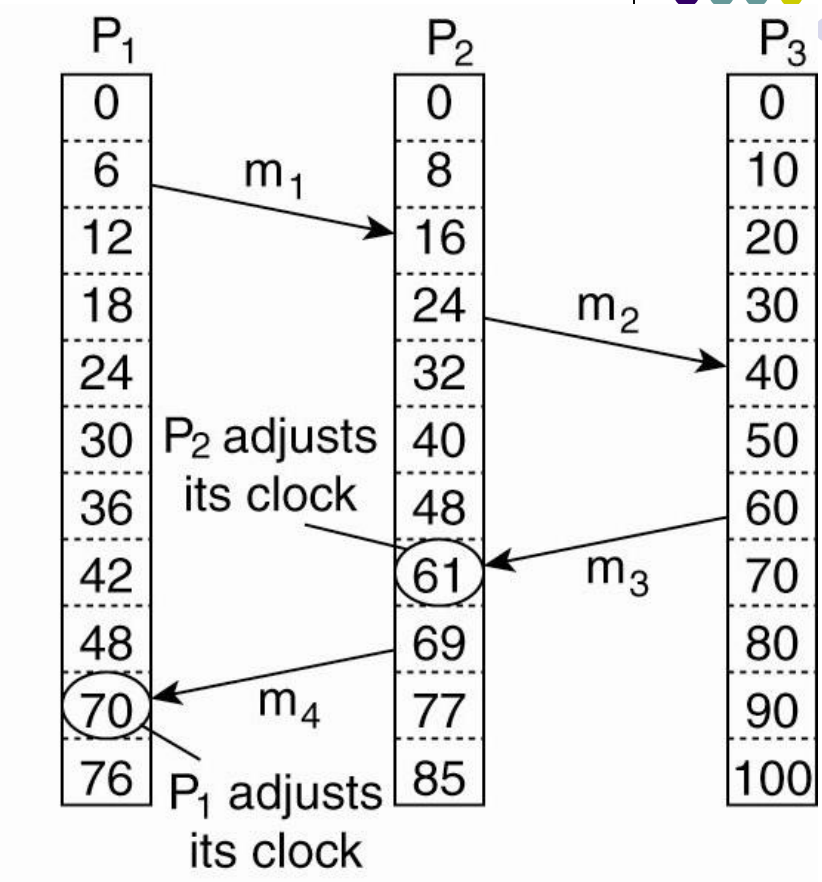
Χρονοσφραγίδες Lamport



- Πως μπορούμε να διατηρούμε ένα ρολόι, έτσι ώστε αν το γεγονός A **συμβαίνει πριν** από το γεγονός B, να ισχύει και LT(A) < LT(B)?
  - Αν A και B ανήκουν στην ίδια διεργασία, τότε απλά χρησιμοποίησε το ρολόι της μηχανής
  - Αν ανήκουν σε διαφορετικές διεργασίες, δεν μας ενδιαφέρει αν δεν ανταλλάσσουν κάποιο μήνυμα μεταξύ τους
  - Αν ανταλλάσσουν κάποιο μήνυμα χρησιμοποίησέ το για να προσαρμόσεις το ρολόι κατάλληλα

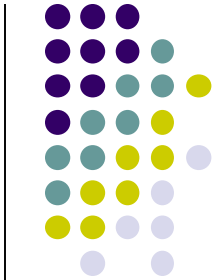
Ο Αλγόριθμος του Lamport





- Three processes, each with its own clock. The clocks run at different rates.
- After receiving a message, adjust clock if necessary.

Ο Αλγόριθμος του Lamport

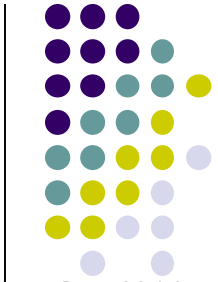


Κάθε διεργασία P_i διατηρεί μια τοπική μεταβλητή LT_i η οποία καλείται λογικό ρολόι και αποθηκεύει μη αρνητικούς ακέραιους και έχει αρχική τιμή 0.

1. Πριν από την εκτέλεση ενός γεγονότος η P_i εκτελεί $LT_i \leftarrow LT_i + 1$.
2. Όταν η διεργασία P_i στέλνει μήνυμα m στην P_j, θέτει την χρονοσφραγίδα $ts(m)$ (timestamp) του m ίση με LT_i .
3. Όταν η διεργασία P_j λάβει το m προσαρμόζει το διάνυσμά της θέτοντας

   $LT_j \leftarrow \max\{LT_j, ts(m)\}$

   Στη συνέχεια η P_j εκτελεί το βήμα 1. (μαρκάροντας έτσι με LT_j+1 το γεγονός παραλαβής)

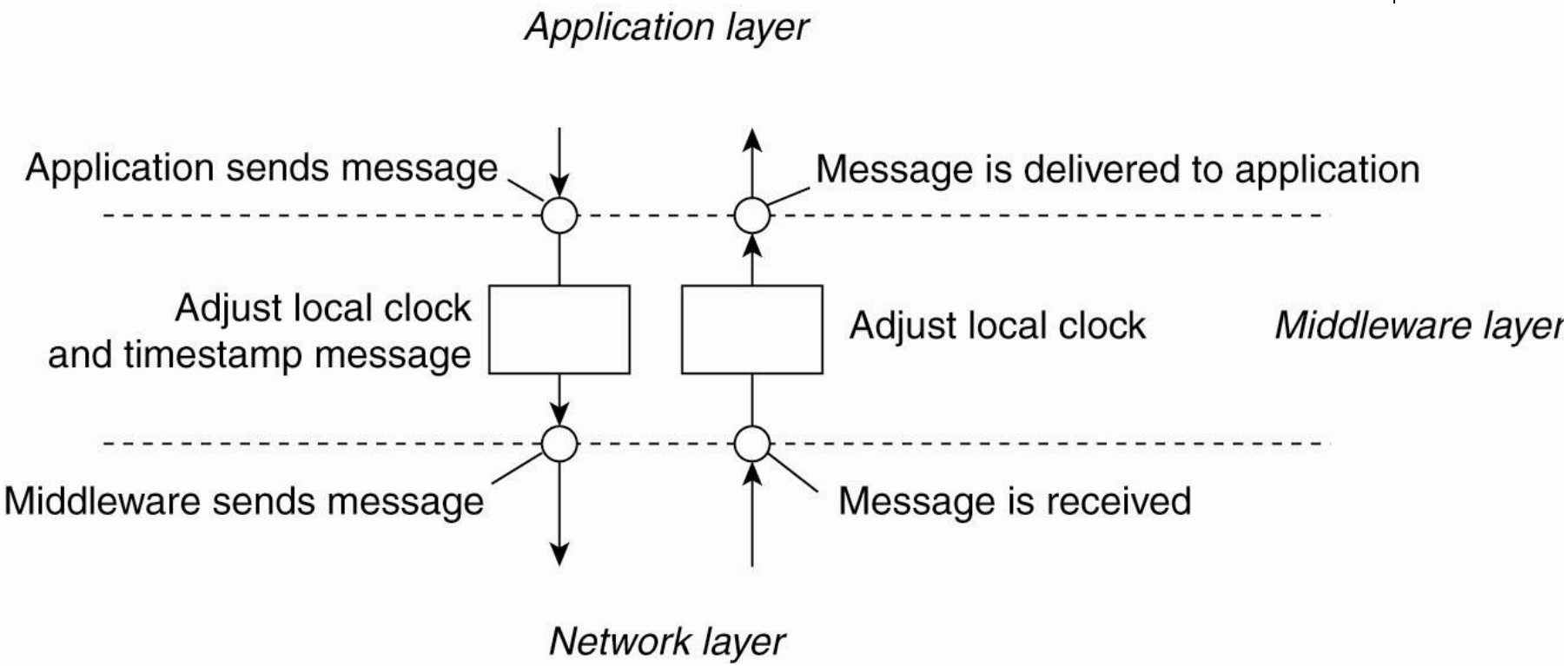


Ο Αλγόριθμος του Lamport μας δίνει ένα τρόπο να αποδίδουμε την ώρα σε όλα τα συμβάντα ενός κατανεμημένου συστήματος έτσι ώστε:

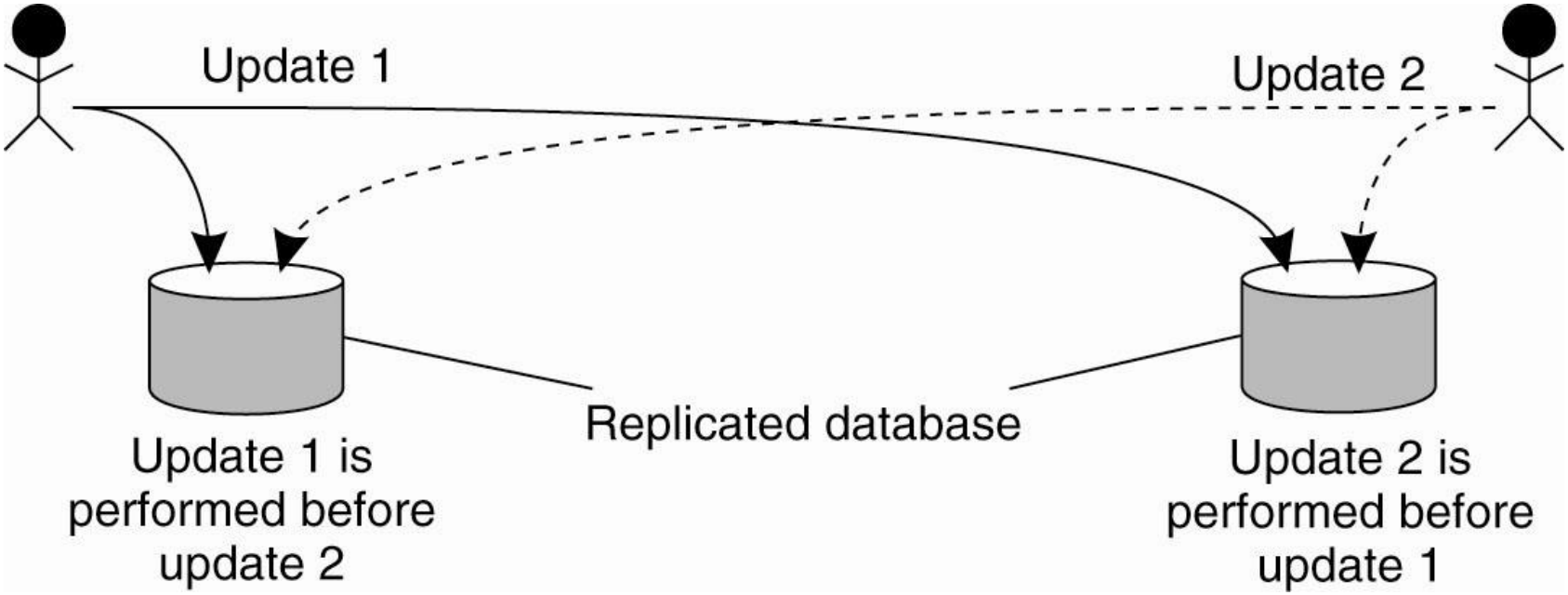
- Αν το α συμβεί πριν από το β στην ίδια διεργασία τότε $LT(\alpha) < LT(\beta)$.
- Αν τα α και β αντιπροσωπεύουν την αποστολή και τη λήψη ενός μηνύματος αντίστοιχα, τότε $LT(\alpha) < LT(\beta)$.
- Για όλα τα διαφορετικά συμβάντα $LT(\alpha) \neq LT(\beta)$

  (αυτό μπορούμε να το πετύχουμε και για ταυτόχρονα συμβάντα) (πως; προσθέτοντας ως επιπλέον ένδειξη ένα μοναδικό αναγνωριστικό της διεργασίας)

Υλοποίηση μηχανισμού Λογικών Ρολογιών

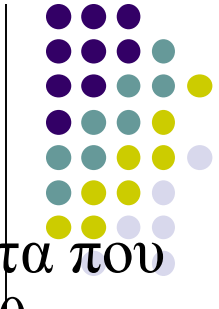


Ολικά διατεταγμένη πολυεκπομπή



Updating a replicated database and leaving it in an inconsistent state.

Ολικά διατεταγμένη πολυεκπομπή

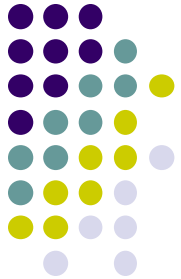


Λειτουργία πολυεκπομπής σύμφωνα με την οποία τα μηνύματα που στέλνονται ταυτόχρονα μεταδίδονται με την ίδια σειρά σε κάθε αποδέκτη, ώστε να εξασφαλίζεται η συνέπεια.

Χρήση των χρονοσφραγίδων του Lamport: Έστω μια ομάδα διεργασιών που στέλνουν μηνύματα (πολυεκπομπή) η μία στην άλλη.

- Κάθε μήνυμα φέρει μια χρονοσφραγίδα (λογική ώρα του αποστολέα).
- Κάθε μήνυμα στέλνεται και στον αποστολέα του.
- Τα μηνύματα από τον ίδιο αποστολέα λαμβάνονται με τη σειρά που στάλθηκαν.
- Όταν μία διεργασία λάβει ένα μήνυμα το τοποθετεί σε μία τοπική ουρά με βάση τη χρονοσφραγίδα του.
- Ο παραλήπτης στέλνει μέσω πολυεκπομπής ένα μήνυμα επιβεβαίωσης στις άλλες διεργασίες.

Ολικά διατεταγμένη πολυεκπομπή



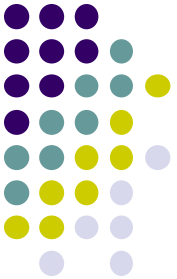
- Ένα μήνυμα ουράς παραδίδεται στην εκτελούμενη εφαρμογή μόνον όταν το μήνυμα βρίσκεται στην αρχή της ουράς και έχει αναγνωρισθεί από κάθε άλλη διεργασία.
- Στη συνέχεια το μήνυμα και οι σχετικές επιβεβαιώσεις διαγράφονται από την ουρά

Επειδή

- δύο μηνύματα δεν έχουν την ίδια χρονοσφραγίδα
- κάθε διεργασία έχει το ίδιο αντίγραφο της ουράς

όλα τα μηνύματα παραδίδονται στις εφαρμογές με την ίδια σειρά!

Διανυσματικές χρονοσφραγίδες (vector timestamps)



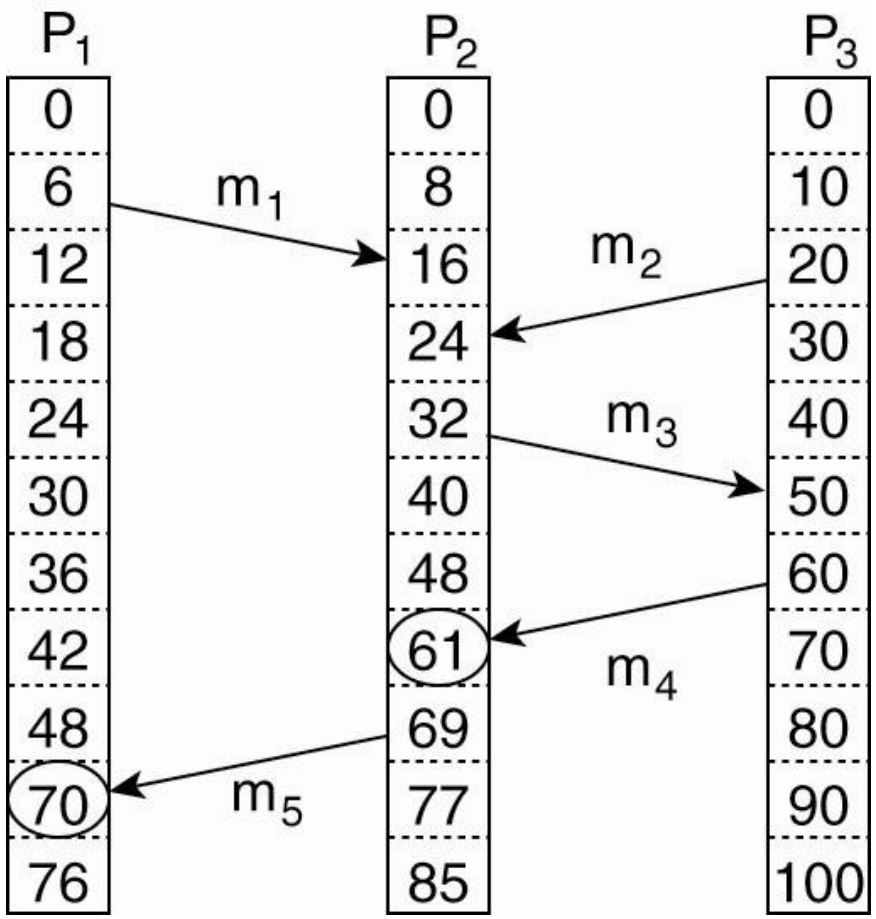
- Οι χρονοσφραγίδες του Lamport οδηγούν σε μια κατάσταση διάταξης του κατανεμημένου συστήματος βάσει της ιδιότητας ότι αν $\alpha \rightarrow \beta$ τότε $LT(\alpha) < LT(\beta)$,
- **Δεν ισχύει όμως το αντίστροφο**. Δηλαδή,

  αν $LT(\alpha) < LT(\beta)$, τότε δεν ισχύει απαραίτητα ότι $\alpha \rightarrow \beta$

- Το πρόβλημα είναι ότι η σχέση «συμβαίνει-πριν» είναι μερική διάταξη ενώ οι χρονοσφραγίδες είναι ακέραιοι και ορίζουν ολική διάταξη

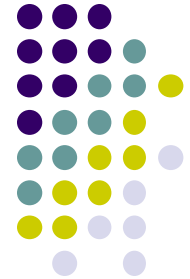
- Οι χρονοσφραγίδες του Lamport **δεν αποτυπώνουν την αιτιότητα**. Το ζήτημα αυτό αντιμετωπίζουν τα **διανυσματικά ρολόγια (vector clocks)** ή **διανυσματικές χρονοσφραγίδες (vector timestamps)**.

Ταυτόχρονα Συμβάντα (Concurrent Transmissions) με Λογικά Ρολόγια



- Το m1 ελήφθη πριν σταλεί το m3, και το λογικό ρολόι το δείχνει.
- Πότε όμως ελήφθη το m1 σε σχέση με την αποστολή του m2?
- Το λογικό ρολόι δείχνει ότι προηγείται η λήψη του m1 αλλά δεν είναι έτσι αναγκαστικά!

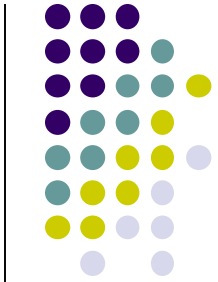
Παράδειγμα: Ηλεκτρονικός πίνακας ανακοινώσεων του Internet



- Οι χρήστες-διεργασίες εντάσσονται σε ειδικές ομάδες συζήτησης.
- Οι δημοσιεύσεις στο πλαίσιο μιας ομάδας (άρθρα ή αποκρίσεις σε άρθρα) στέλνονται μέσω πολυεκπομπής σε όλα τα μέλη μιας ομάδας.
- Θα πρέπει **να διασφαλιστεί ότι οι αποκρίσεις σε άρθρα παραδίδονται μετά τα σχετικά άρθρα.**

Η χρήση ολικά διατεταγμένης εκπομπής αν και λύνει το πρόβλημα δεν ενδείκνυται διότι είναι υπερβολικά ισχυρή για τη συγκεκριμένη εφαρμογή.

Διανυσματικές χρονοσφραγίδες



Οι **διανυσματικές χρονοσφραγίδες έχουν την ιδιότητα ότι**

αν $vt(\alpha) < vt(\beta)$ **τότε** $\alpha \rightarrow \beta$ **(**α **προηγείται αιτιακά του** β**)**

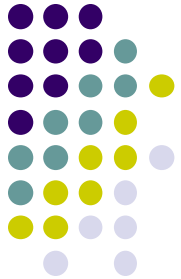
Υλοποίηση διανυσματικών χρονοσφραγίδων:

Κάθε διεργασία P_i διατηρεί διάνυσμα $\boldsymbol{VC_i}$ με τις εξής ιδιότητες:

1. $VC_i[i]$ είναι ο αριθμός των συμβάντων που έχουν λάβει χώρα μέχρι τώρα στην P_i. Δηλ., $VC_i[i]$ ισούται με την τιμή του τοπικού λογικού ρολογιού της P_i .

2. Αν $VC_i[j] = k$ τότε η P_i γνωρίζει ότι k συμβάντα έχουν λάβει χώρα μέχρι τώρα στην P_j. Δηλ., η P_i γνωρίζει την τιμή του τοπικού λογικού ρολογιού της P_j.

Διανυσματικές χρονοσφραγίδες

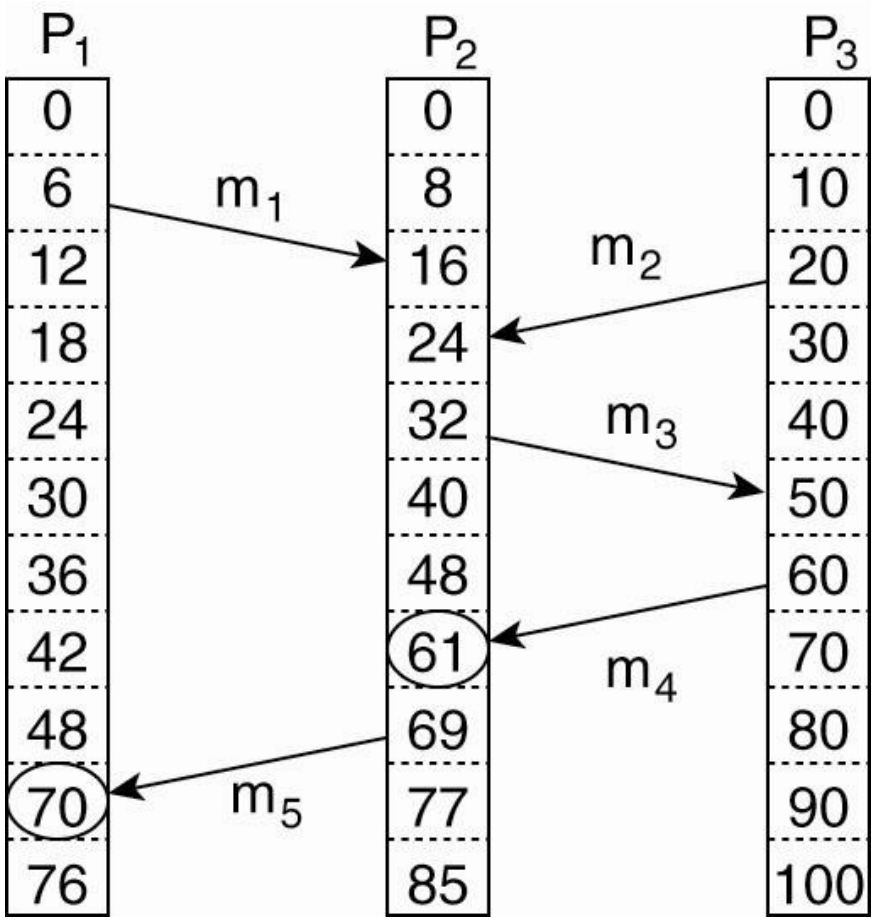


Πώς τηρούνται οι δύο προηγούμενες ιδιότητες?

1. **Η πρώτη τηρείται με την προσαύξηση της $VC_i[i]$ κάθε φορά που λαμβάνει χώρα ένα καινούργιο συμβάν στην P_i.**

2. **Η δεύτερη ιδιότητα τηρείται με την προσκόλληση διανυσμάτων *(χρονοσφραγίδων)* στα μηνύματα που στέλνονται. Ήτοι, όταν η P_i στέλνει το μήνυμα *m* στέλνει παράλληλα και το τρέχον διάνυσμά της ως *χρονοσφραγίδα*.**

Ταυτόχρονα Συμβάντα (Concurrent Transmissions) με Διανυσματικές Χρονοσφραγίδες

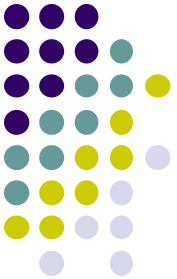


- Το m1 ελήφθη πριν σταλεί το m3, και το λογικό ρολόι το δείχνει.
- Πότε όμως ελήφθη το m1 σε σχέση με την αποστολή του m2?
- Το λογικό ρολόι δείχνει ότι προηγείται η λήψη του m1 αλλά δεν είναι έτσι αναγκαστικά!

Διανυσματικές Χρονοσφραγίδες

- $T_{snd}(m_1) = (6,0,0)$
- $T_{rcv}(m_1) = (6,16,0)$
- $T_{snd}(m_2) = (0, 0, 20)$
- $T_{rcv}(m_2) = (6,24,20)$

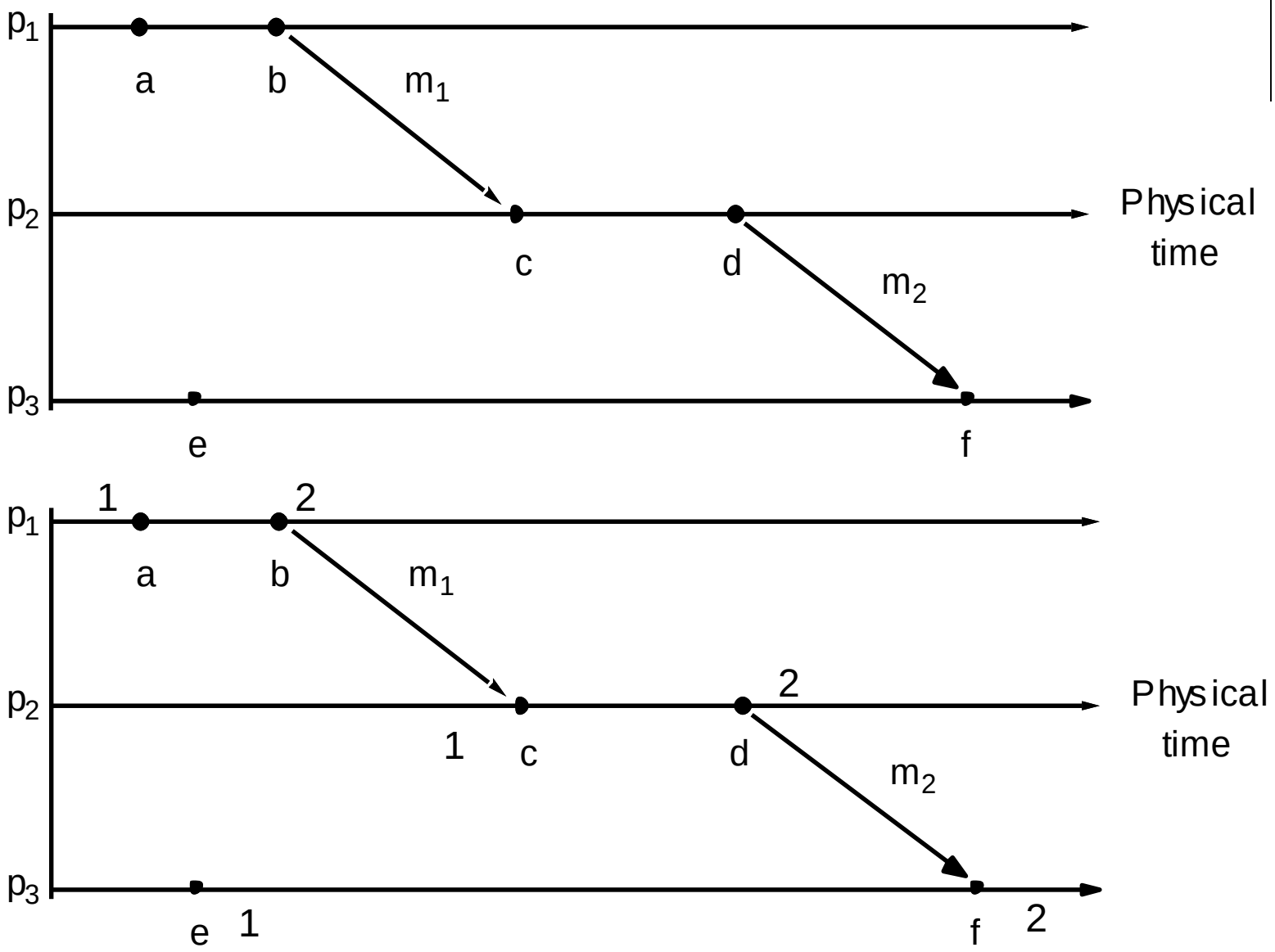
Διανυσματικές χρονοσφραγίδες - Αλγόριθμος



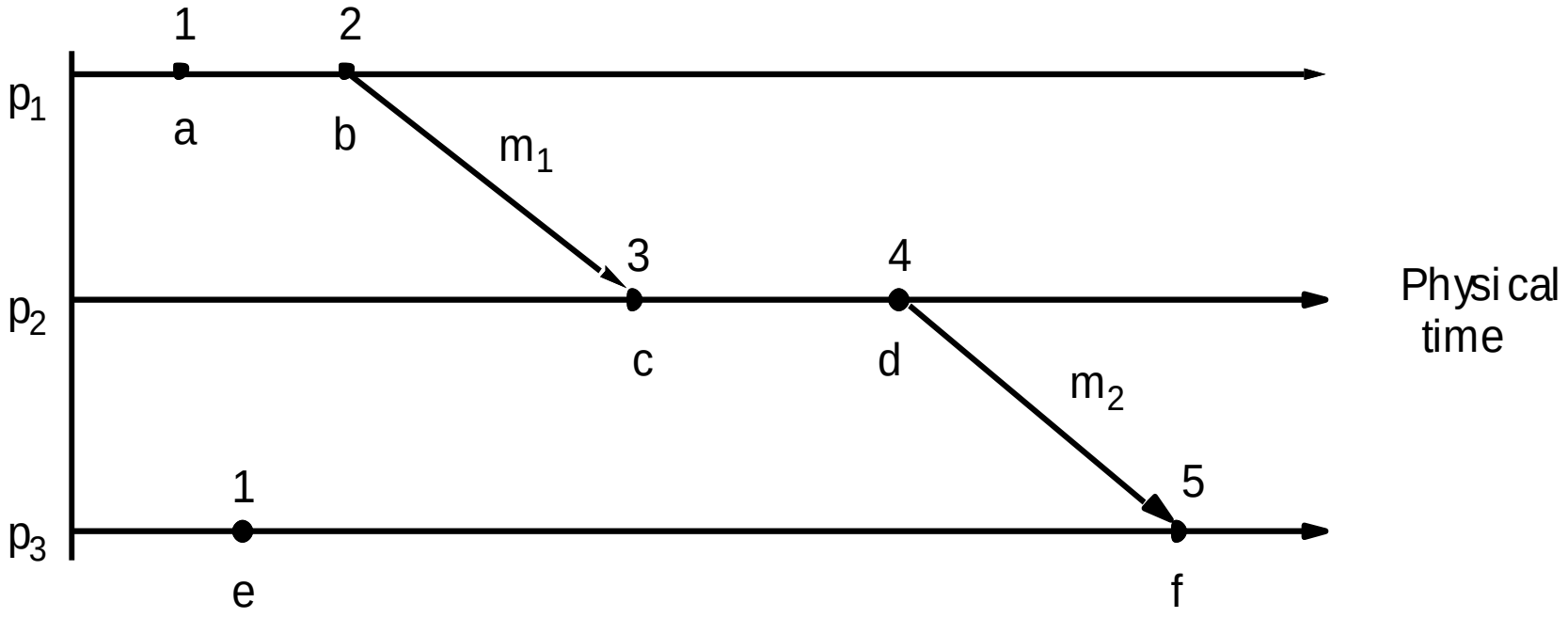
1. Πριν από την εκτέλεση ενός συμβάντος η P_i εκτελεί $VC_i[i] \leftarrow VC_i[i] + 1$.
2. Όταν η διεργασία P_i στέλνει ένα μήνυμα m στην P_j, θέτει την διανυσματική χρονοσφραγίδα $vt(m)$ ίση με VC_i (αφού έχει εκτελέσει το προηγούμενο βήμα)
3. Όταν η διεργασία P_j λάβει το m προσαρμόζει το διάνυσμά της θέτοντας $VC_j[k] \leftarrow \max\{VC_j[k], vt(m)[k]\}$ για κάθε k.

- Key point: $vt(\mathrm{m})[i]$ -1 είναι ο αριθμός των γεγονότων τα οποία γνωρίζει η διεργασία Pi και τα οποία αιτιακά προηγούνται του m.

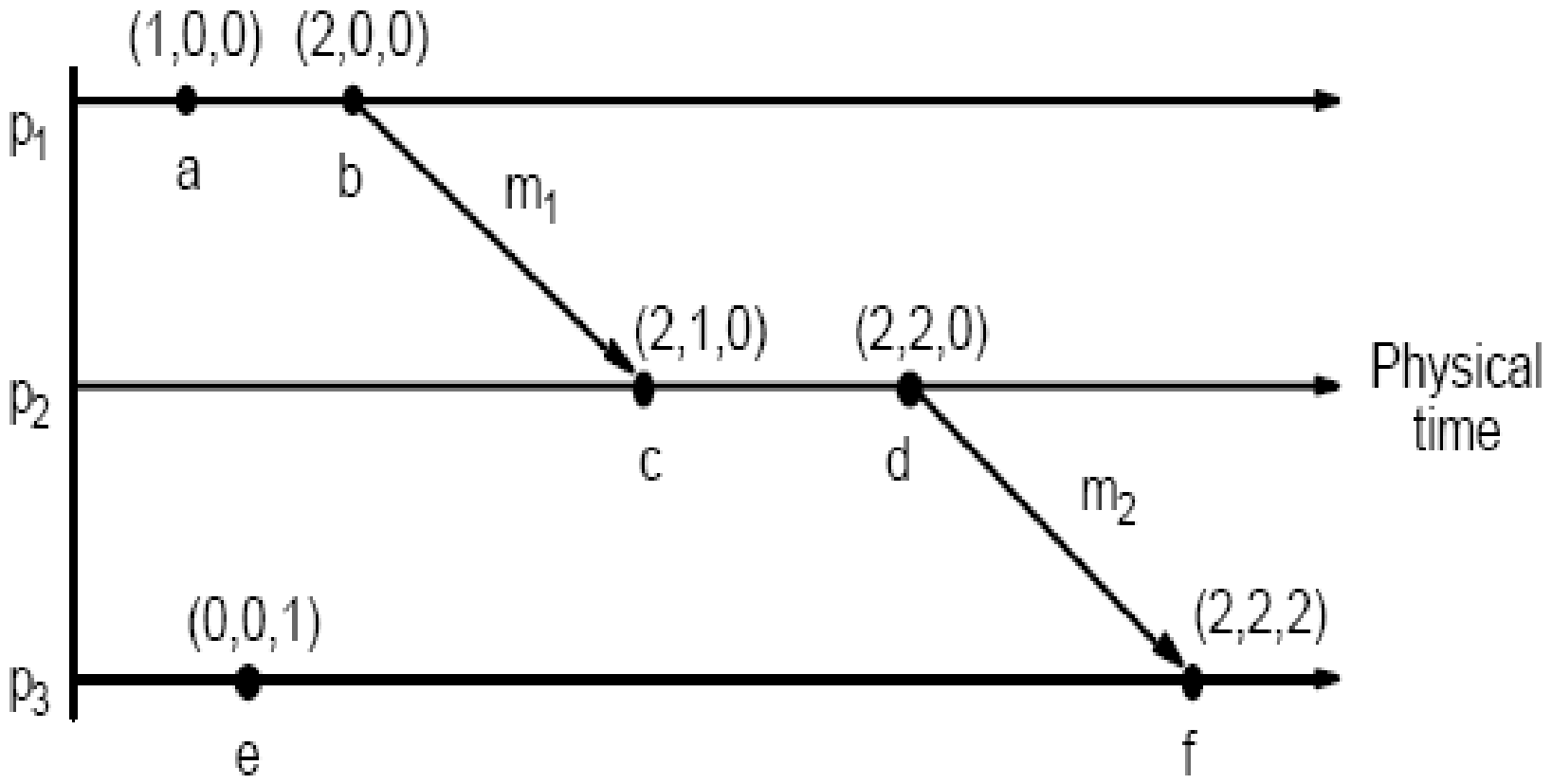
Χωρίς χρονοσφραγίδες



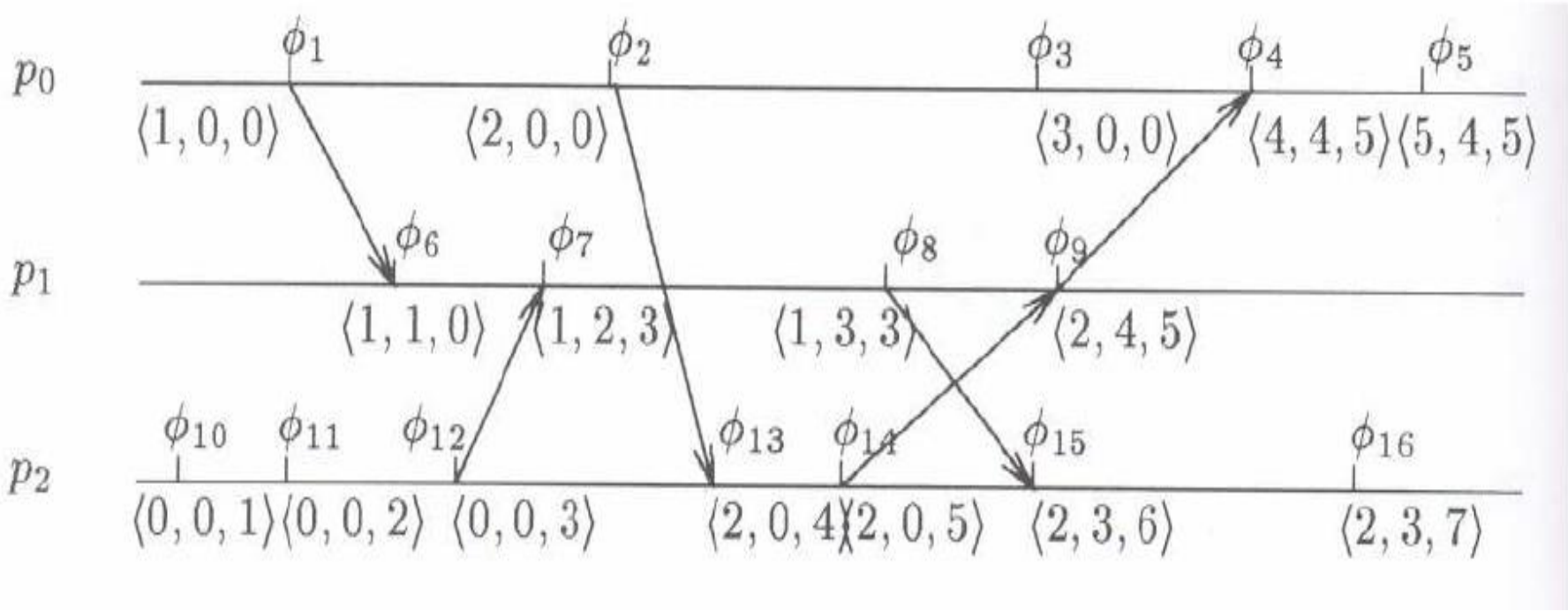
Χρονοσφραγίδες Lamport



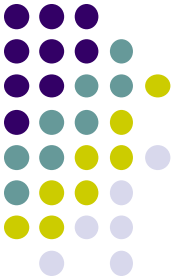
Διανυσματικές χρονοσφραγίδες



Διανυσματικές χρονοσφραγίδες (άλλο παράδειγμα)



Μερική Διάταξη Διανυσματικών Ρολογιών



- $VC_1 = VC_2$,

  Αν και μόνο αν: $VC_1[i] = VC_2[i]$, for all i = 1, … , n

- $VC_1 <= VC_2$,

  Αν και μόνο αν: $VC_1[i] <= VC_2[i]$, for all i = 1, … , n

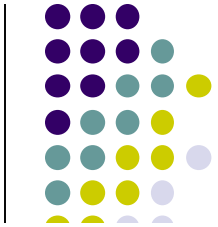
- $VC_1 < VC_2$,

  Αν και μόνο αν: $VC_1 <= VC_2$ & $VC_1 \neq VC_2$

- $VC_1 \parallel VC_2$ *(ταυτόχρονα συμβάντα)*

  Αν και μόνο αν: δεν ισχύει ούτε $VC_1<=VC_2$ ούτε $VC_2<=VC_1$

Διανυσματικές χρονοσφραγίδες



Θεώρημα

Έστω ότι a είναι μια αυθαίρετη εκτέλεση και έστω ότι φ_1 και φ_2 είναι δύο οποιαδήποτε γεγονότα της a. Αν $\varphi_1 \rightarrow \varphi_2$ τότε $VC(\varphi_1) < VC(\varphi_2)$.

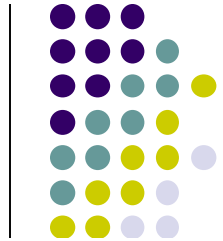
Απόδειξη

Έστω ότι τα φ_1, φ_2 είναι γεγονότα του ίδιου επεξεργαστή και ας υποθέσουμε ότι το φ_1 προηγείται του φ_2. *Γιατί ισχύει ο ισχυρισμός για τα φ_1 και φ_2 σε αυτή την περίπτωση;*

Έστω ότι το φ_1 είναι γεγονός στ οποίο αποστέλλεται κάποιο μήνυμα m και φ_2 είναι το γεγονός παραλαβής του μηνύματος. *Γιατί ισχύει ο ισχυρισμός για τα φ_1 και φ_2 σε αυτή την περίπτωση;*

Ισχύει η μεταβατική ιδιότητα για τη σχέση $\leq$ σε διανύσματα.

Διανυσματικές χρονοσφραγίδες



Θεώρημα

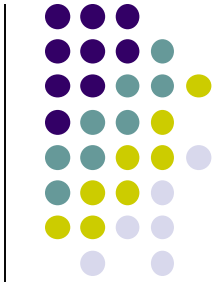
Έστω ότι a είναι μια αυθαίρετη εκτέλεση και έστω ότι φ_1 και φ_2 είναι δύο οποιαδήποτε γεγονότα της a. Αν $VC(\varphi_1) < VC(\varphi_2)$ τότε $\varphi 1 \rightarrow \varphi_2$.

Απόδειξη

- Έστω ότι φ_1 και φ_2 είναι δύο γεγονότα που συμβαίνουν ταυτόχρονα, το φ_1 από τη διεργασία p_i και το φ_2 από τη p_j, $p_i \neq p_j$.
- Ας υποθέσουμε ότι $VC_i[i](\varphi_1) = m$. Ο μόνος τρόπος το i-οστό στοιχείο του διανυσματικού ρολογιού μιας διεργασίας p_j να αποκτήσει τιμή τουλάχιστον m είναι μέσω μιας αλυσίδας μηνυμάτων που ξεκινούν από την p_i (είτε στο γεγονός φ_1 ή σε επόμενο γεγονός).
- Μια τέτοια αλυσίδα θα συνεπαγόταν ότι τα φ_1, φ_2 δεν συμβαίνουν ταυτόχρονα. Άτοπο!

Άρα, $VC_j[i](\varphi_2) < m \Rightarrow VC_i(\varphi_1)$ δεν μπορεί να είναι μικρότερο από $VC_j(\varphi_2)$.

Διασφάλιση αιτιοκρατικής παράδοσης μηνυμάτων



Παράδειγμα: Ηλεκτρονικός πίνακας ανακοινώσεων του Internet

Έστω ότι η $VC_i[i]$ προσαυξάνεται μόνον όταν η $\boldsymbol{P_i}$ στέλνει κάποιο μήνυμα.

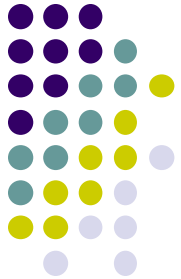
Όταν μία διεργασία $\boldsymbol{P_i}$ δημοσιεύει ένα άρθρο, το στέλνει μέσω πολυεκπομπής ως μήνυμα *a* με χρονοσφραγίδα $vt(a) = VC_i$

Έστω ότι η $\boldsymbol{P_j}$ στέλνει μια απόκριση στο άρθρο μέσω πολυεκπομπής ως μήνυμα *r* με χρονοσφραγίδα $vt(r) = VC_j$

Ισχύει ότι $vt(r)[j] > vt(a)[j]$

Τα μηνύματα *a* και *r* θα φθάσουν σε μια τρίτη διεργασία $\boldsymbol{P_k}$ αλλά δεν γνωρίζουμε με ποια σειρά.

Διασφάλιση αιτιοκρατικής παράδοσης μηνυμάτων



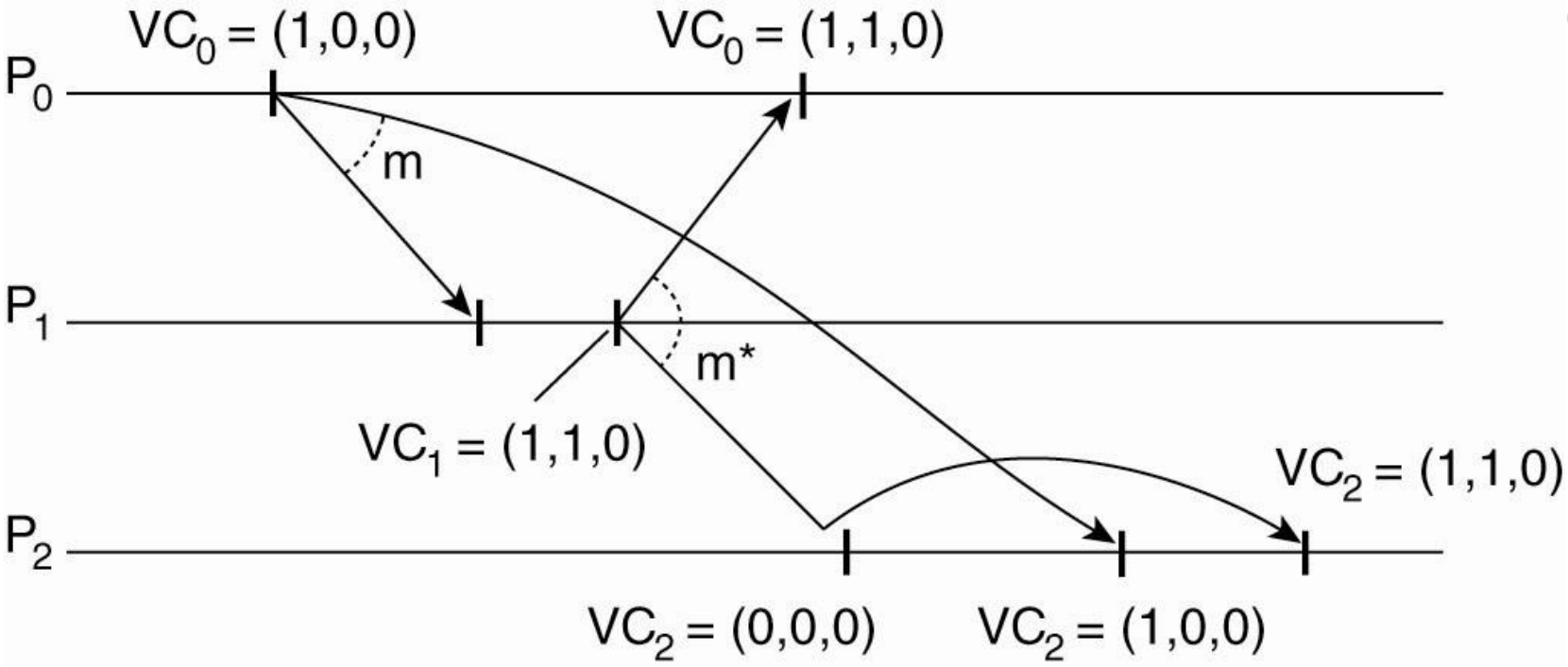
Όταν η P_κ λάβει το r από την $\boldsymbol{P_j}$ θα εξετάσει την $vt(r)$ και θα **αναβάλλει την παράδοση έως ότου** ληφθούν όλα τα μηνύματα που προηγούνται αιτιακά του r, ήτοι έως ότου

1. $\boldsymbol{vt(r)[j] = VC_k[j] + 1}$
2. $\boldsymbol{vt(r)[i] <= VC_k[i]}$ **για κάθε** $\boldsymbol{i \neq j}$

- Η πρώτη συνθήκη δηλώνει ότι το r είναι το επόμενο μήνυμα που παίρνει η P_κ από την P_j .

- Η δεύτερη συνθήκη δηλώνει ότι η P_κ έχει δει όλα τα μηνύματα που είχε δει η P_j κατά την αποστολή του r (επομένως έχει δει και το α)

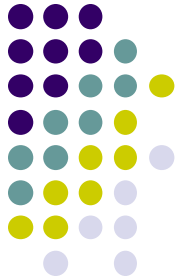
Παράδειγμα Διασφάλισης αιτιοκρατικής παράδοσης μηνυμάτων



Όταν η P_2 λάβει το *m (*vt(m*)=(1,1,0)*) από την P_1 θα αναβάλει την παράδοση έως ότου**

1. ***vt(m*)[1] = VC_2[1] + 1 (το οποίο ισχύει στο παράδειγμα)***
2. ***vt(m*)[i] <= VC_2[i] για κάθε $i \neq 1$ (δηλ. i=0, i=2)***

Επιχείρημα από άκρο σε άκρο (End-to-End Argument) στο σχεδιασμό συστημάτων



- Ordering could be completely handled by middleware.

  Advantages, disadvantages?
  - Advantage is convenience.
  - Disadvantage is causality where there is none.

- Or, could have application play a role.

Επιχείρημα από άκρο σε άκρο: Η εφαρμογή αναλαμβάνει τα ζητήματα διάταξης