Ενσωματωμένα Συστήματα

Ενότητα 9: Λειτουργικά Συστήματα Ενσωματωμένων Συστημάτων

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.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σκοπός ενότητας

- Η κατανόηση του χρονοπρογραμματισμού των ενσωματωμένων συστημάτων πολλαπλών διεργασιών.
- Η κατανόηση των προθεσμιών των ενσωματωμένων συστημάτων και οι τρόποι αντιμετώπισης τους από τα Λειτουργικά συστήματα.
- Η κατανόηση της ανάγκης των ΛΣ στα ενσωματωμένα συστήματα.

Πολλαπλές Εργασίες και πολλαπλές διεργασίες.

Αντιδραστικά Συστήματα

Πολλά ενσωματωμένα συστήματα κάνουν περισσότερα από μια λειτουργίες.

- Ανταποκρίνονται σε εξωτερικά συμβάντα.
  - Ελεγκτής κινητήρα.
  - Οθόνη ελέγχου ζώνης ασφαλείας.
- Απαιτούν ανταπόκριση σε πραγματικό χρόνο.
  - Αρχιτεκτονική συστήματος.
  - Υλοποίηση προγράμματος.
- Μπορεί να απαιτούν μια αλυσιδωτή αντίδραση μεταξύ πολλαπλών επεξεργαστών.

Multi-rate Συστήματα

- Οι εργασίες μπορεί να είναι σύγχρονες ή ασύγχρονες.
- Οι σύγχρονες εργασίες μπορεί να επαναλαμβάνονται σε διαφορετικούς ρυθμούς.
- Οι διεργασίες τρέχουν σε διαφορετικούς ρυθμούς βάσει των υπολογιστικών αναγκών των εργασιών.

Συστήματα πραγματικού χρόνου

- Εκτελούν έναν υπολογισμό έτσι ώστε να συνάδουν με εξωτερικούς χρονικούς περιορισμούς.
- Συχνότητα προθεσμίας:
  - **Περιοδική**.
  - **Απεριοδική** *(ασύγχρονα)*.
- Τύπος προθεσμίας:
  - **Hard**: μη τήρηση της προθεσμίας έχει ως αποτέλεσμα αποτυχία του συστήματος.
  - **Soft**: μη τήρηση της προθεσμίας έχει ως αποτέλεσμα υποβαθμισμένη ανταπόκριση.
  - **Firm**: η αργή ανταπόκριση γενικά είναι άχρηστη, αλλά κάποιες αργές ανταποκρίσεις μπορούν να γίνουν κατ' εξαίρεση ανεκτές.

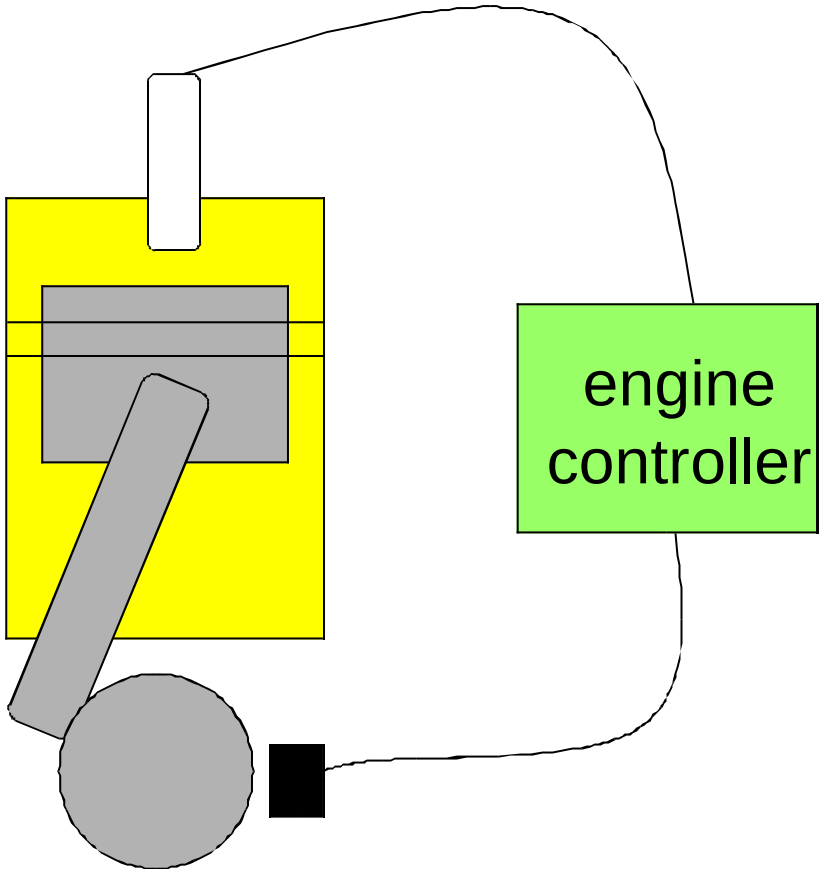
Αν οι περίοδοι των διεργασιών σχετίζονται μεταξύ τους (π.χ. 1/2,1/4...), τότε είναι απλός ο κώδικας χρονοδρομολόγησης όπως θα φανεί σε παραδείγματα (ίσως δε χρειάζεται OS).

Παράδειγμα: έλεγχος κινητήρα

- **Εργασίες:**
  - Έλεγχος σπινθηριστή *(μπουζί)*.
  - Έλεγχος στροφαλοφόρου άξονα.
  - Μείγμα καύσιμου/αέρα.
  - Αισθητήρας οξυγόνου.
  - Φίλτρο Kalman.

Η σχέση φάσης μεταξύ της μετακίνησης του εμβόλου και του σπινθήρα πρέπει να αλλάζει σαν συνάρτηση της ταχύτητας της μηχανής.

Πολλαπλοί αισθητήρες, πολλαπλές διακεκριμένες εργασίες.



Τυπικές τιμές ελεγκτών κινητήρα

Μεταβλητή	Ολικό χρονικό φάσμα (ms)	Περίοδος ενημέρωσης (ms)
Χρονισμός του μπουζί	300	2
Ρυθμιστική βαλβίδα *(πεταλούδα)*	40	2
Ροή αέρα	30	4
Τάση μπαταρίας	80	4
Ροή καυσίμου	250	10
Ανακυκλωμένο καυσαέριο	500	25
Κατάσταση διακοπτών	100	20
Θερμοκρασία αέρα	Δευτερόλεπτα	400
Βαρομετρική πίεση	Δευτερόλεπτα	1000
Σπίθα μπουζί *(παραμονή)*	10	1
Προσαρμογή καυσίμου	80	8
Καρμπυρατέρ	500	25
Ενεργοποιητές κατάστασης	100	100

Το πρόβλημα του χρονοπρογραμματισμού

- Μπορούμε να τηρήσουμε όλες τις προθεσμίες;
  - Πρέπει να μπορούμε να τηρήσουμε τις προθεσμίες σε όλες τις περιπτώσεις.
- Πόση CPU *“ιπποδύναμη”* χρειαζόμαστε για να τηρήσουμε τις προθεσμίες μας;

Χρονικές απαιτήσεις στις διεργασίες (1/2)

- **Περίοδος** *(Period)*: το διάστημα ανάμεσα στις ενεργοποιήσεις των διεργασιών.
- **Διάστημα έναρξης** *(Initiation interval)*: το αντίστροφο της περιόδου.
- **Χρόνος μύησης** *(Initiation time)*: ο χρόνος μέσα στον οποίο η διεργασία γίνεται έτοιμη.

Χρονικές απαιτήσεις στις διεργασίες (2/2)

- **Χρόνος έκδοσης** *(Release time)*: ο χρόνος μέσα στον οποίο η διεργασία γίνεται έτοιμη. *(Η απεριοδική διεργασία ξεκινάει από ένα γεγονός, όπως η άφιξη δεδομένων).*

- **Προθεσμία** *(Deadline)*: ο χρόνος μέσα στον οποίο πρέπει να ολοκληρωθεί η διεργασία (μετριέται από το χρόνο έναρξης).

Χρονικές παραβιάσεις

- Τι συμβαίνει αν μια διεργασία δεν ολοκληρωθεί εντός της προθεσμίας?
  - **Hard deadline:** το σύστημα αποτυγχάνει αν χαθεί η προθεσμία *(π.χ. Έλεγχος καυσίμου κινητήρα).*
    Μπορεί να ενεργοποιηθούν αντισταθμιστικά μέτρα όπως η προσέγγιση δεδομένων, ή εναλλαγή σε μια ειδική κατάσταση λειτουργίας.
  - **Soft deadline:** ο χρήστης ίσως το παρατηρήσει, αλλά το σύστημα δεν αποτυγχάνει απαραιτήτως *(π.χ. Mp3 player)*.
    Μπορεί να αγνοήσουν εντελώς την αποτυχία ή να ενεργήσουν για την αποφυγή διάδοσης κακών δεδομένων *(π.χ. Εισαγωγή σιγής σε τηλεφωνική κλήση VOIP).*

Παράδειγμα: σφάλμα λογισμικού σε πύραυλο (1/2)

- Η πρώτη εκτόξευση του διαστημικού πυραύλου καθυστέρησε από ένα χρονικό σφάλμα λογισμικού:
  - **Primary control system** *(Σύστημα ελέγχου ηλεκτρονικού λογισμικού 1ου βαθμού)* PASS και σύστημα υποστήριξης *(backup system)* BFS.
  - **PASS**, 4 υπολογιστές, λειτουργία πλειοψηφίας *(για ανοχή σε σφάλματα)*
  - *(Backup Flight Control System)* **BFS** απέτυχε να συγχρονιστεί με το PASS. Αυτόματη απεμπλοκή επειδή αποφασίστηκε ότι το PASS ήταν ελαττωματικό.

Παράδειγμα: σφάλμα λογισμικού σε πύραυλο (2/2)

- Η αλλαγή μιας ρουτίνας *(πριν 2 χρόνια)* πρόσθεσε μικρή καθυστέρηση η οποία κατέρριψε τους υπολογισμούς για το χρόνο έναρξης.

- 1 στις 67 η πιθανότητα προβλήματος χρονισμού.

Υλοποίηση του βρόχου loop

- Η απλούστερη υλοποίηση έχει ένα βρόχο.
  - Κανένας έλεγχος πάνω στο χρόνο εκτέλεσης.

```
while (TRUE) {
   p1();
   p2();
}
```

Υλοποίηση χρονομετρημένου βρόχου

- Ενσωματώνει το σύνολο όλων των διεργασιών σε μια ενιαία συνάρτηση που υλοποιεί τη συνολική εργασία.
- Χρησιμοποιεί χρονοδιακόπτη για να ελέγξει την εκτέλεση της εργασίας.
  - Κανένας έλεγχος στο χρονισμό των επιμέρους διεργασιών.

```
void pall(){
   p1();
   p2();
}
```

Υλοποίηση πολλαπλών χρονιστών

- Κάθε εργασία έχει τη δική της συνάρτηση.
- Κάθε εργασία έχει το δικό της χρονιστή *(timer)*.
  - Ίσως να μην υπάρχουν αρκετοί χρονιστές για να υλοποιήσουν όλους τους ρυθμούς.

```
void pA(){ /* rate A */
p1();
p3();
}
void B(){ /* rate B */
p2();
p4();
p5();
}
```

Υλοποίηση χρονοδιακόπτη + μετρητή

- Χρησιμοποιεί ένα μετρητή λογισμικού για να διαιρέσει το χρονιστή.
- Λειτουργεί μόνο για καθαρά πολλαπλάσια της περιόδου του χρονιστή.

```
int p2count = 0;
void pall(){
p1();
if (p2count >= 2) {
      p2();
      p2count = 0;
      }
else p2count++;
p3();
}
```

Υλοποιώντας διεργασίες (1/2)

- Όλες αυτές οι υλοποιήσεις είναι ανεπαρκείς.
- Χρειάζεται καλύτερος έλεγχος του χρονισμού.
- Χρειάζεται ένας καλύτερος μηχανισμός από τις υπορουτίνες.

Υλοποιώντας διεργασίες (2/2)

- Λόγω της αυξημένης πολυπλοκότητας διαχείρισης του κώδικα και το σεβασμό των διαφορετικών απαιτήσεων χρονισμού σε κάθε εργασία, απαιτείται η χρήση ενός λειτουργικού συστήματος.
- Ενσωματωμένα συστήματα, τα οποία δεν έχουν πολύπλοκες απαιτήσεις δε χρειάζεται να επιβαρύνονται με την εκτέλεση ενός ΛΣ.

Διεργασίες και Λειτουργικά Συστήματα

Λειτουργικά Συστήματα

Λειτουργικά συστήματα

- Το λειτουργικό σύστημα ελέγχει τους πόρους:
  - που πάνε στη CPU,
  - που σχετίζονται με είσοδο/έξοδο (I/O),
  - μνήμης που έχει διατεθεί.
- Ο πιο σημαντικός πόρος είναι η CPU.
  - Η πρόσβαση στη CPU ελέγχεται από το χρονοπρογραμματιστή.

Δομή λειτουργικού συστήματος

- Το λειτουργικό σύστημα χρειάζεται να παρακολουθεί:
  - τις προτεραιότητες των διεργασιών,
  - την κατάσταση χρονοπρογραμματισμου,
  - το block ενεργοποίησης διεργασίας.
- Οι διεργασίες μπορούν να δημιουργηθούν:
  - στατικά πριν ξεκινήσει το σύστημα,
  - δυναμικά κατά τη διάρκεια της εκτέλεσης.

Ενσωματωμένος vs. Γενικού σκοπού χρονοπρογραμματισμός

- Οι σταθμοί εργασίας προσπαθούν να αποφύγουν την ύπαρξη διεργασιών που δεν έχουν επαρκή πρόσβαση στη CPU.
  - Αμεροληψία = πρόσβαση στη CPU.
- Τα ενσωματωμένα συστήματα πρέπει να τηρούν τις προθεσμίες.
  - Οι διεργασίες χαμηλής προτεραιότητας ίσως να μην τρέχουν για πολύ χρόνο.

Διεργασίες και λειτουργικά συστήματα

- Πολλαπλές εργασίες και πολλαπλές διεργασίες.
  - Προδιαγραφές του χρονοδιαγράμματος της διεργασίας.
- Προτιμητέα τα λειτουργικά συστήματα πραγματικού χρόνου *(real-time OS).*

Η χρήση διεργασιών για την τμηματοποίηση των λειτουργικών και η ενσωμάτωση του ελέγχου γίνεται πολύ ευκολότερα στο λειτουργικό σύστημα. **RTOS**: ΛΣ που παρέχουν ευκολίες/υπηρεσίες για την ικανοποίηση απαιτήσεων πραγματικού-χρόνου	

POSIX

- Χρησιμοποιούμε το POSIX λειτουργικό σύστημα.
- Το POSIX έχει δημιουργηθεί από έναν οργανισμό προτύπων.
- Τα POSIX ΛΣ είναι συμβατά σε επίπεδο πηγαίου κώδικα *(η εφαρμογή μπορεί να μεταγλωττιστεί για άλλα ΛΣ).*
- Υπάρχουν επεκτάσεις πραγματικού χρόνου POSIX *(async/sync IO, διαμοιραζόμενη μνήμη, ρολόγια, σήματα πραγματικού χρόνου,...).*

Εργασίες και διεργασίες

- Μια εργασία είναι μια λειτουργική περιγραφή μιας συνδεδεμένης ομάδας λειτουργιών.
- *(Μια εργασία μπορεί επίσης να είναι μια συλλογή διεργασιών.)*

- Μια διεργασία είναι μια μοναδική εκτέλεση ενός προγράμματος.
  - Αρκετά αντίγραφα ενός προγράμματος μπορεί να τρέχουν ταυτόχρονα ή σε διαφορετικούς χρόνους.
- Μια διεργασία έχει τη δική της κατάσταση:
  - καταχωρητές,
  - μνήμη.
- Το λειτουργικό σύστημα διαχειρίζεται τις διεργασίες.

Η διεργασία καθορίζει την κατάσταση ενός εκτελούμενου προγράμματος.

Γιατί πολλαπλές διεργασίες;

- Πολλαπλές εργασίες σημαίνουν πολλαπλές διεργασίες.
- Οι διεργασίες βοηθούν στη χρονική πολυπλοκότητα:
  - πολλαπλοί ρυθμοί:
    - ✓ Πολυμέσα.
    - ✓ Αυτοκινητοβιομηχανία.
  - Ασύγχρονη είσοδος *(δε γνωρίζουμε πότε θα συμβεί):*
    - ✓ Διεπαφές χρήστη.
    - ✓ Συστήματα επικοινωνιών.

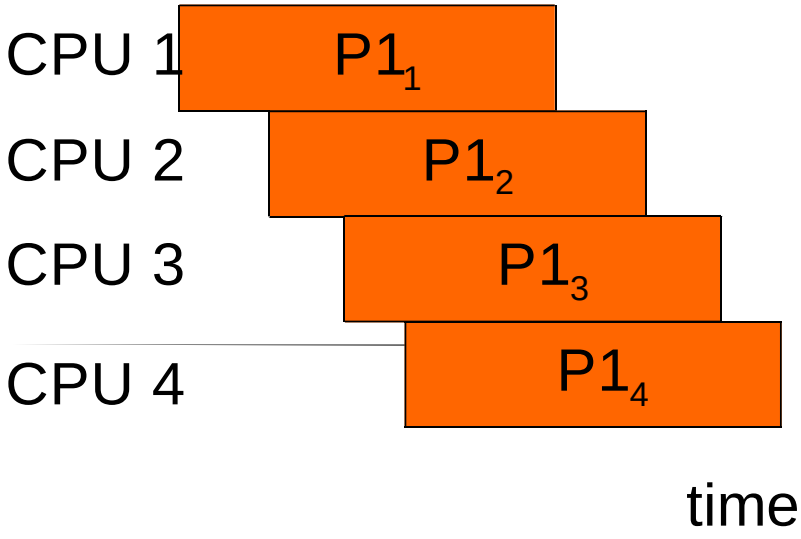
Όταν πολλαπλές λειτουργίες πρέπει να εκτελεσθούν σε πολύ διαφορετικούς χρόνους, ένα μοναδικό πρόγραμμα μπορεί εύκολα να γίνει πολύ πολύπλοκο και δύσκολα διαχειρίσιμο.

Απαιτήσεις ρυθμού σε διεργασίες

- **Περίοδος**: το διάστημα μεταξύ των ενεργοποιήσεων της διεργασίας.
- **Ρυθμός**: το αντίστροφο της περιόδου.
- Ο ρυθμός έναρξης μπορεί να είναι υψηλότερος από την περίοδο--- πολλά αντίγραφα της διεργασίας τρέχουν ταυτόχρονα *(διοχετευμένες εκτελέσεις).*

Περίοδος διεργασίας: Ο χρόνος μεταξύ διαδοχικών εκτελέσεων.

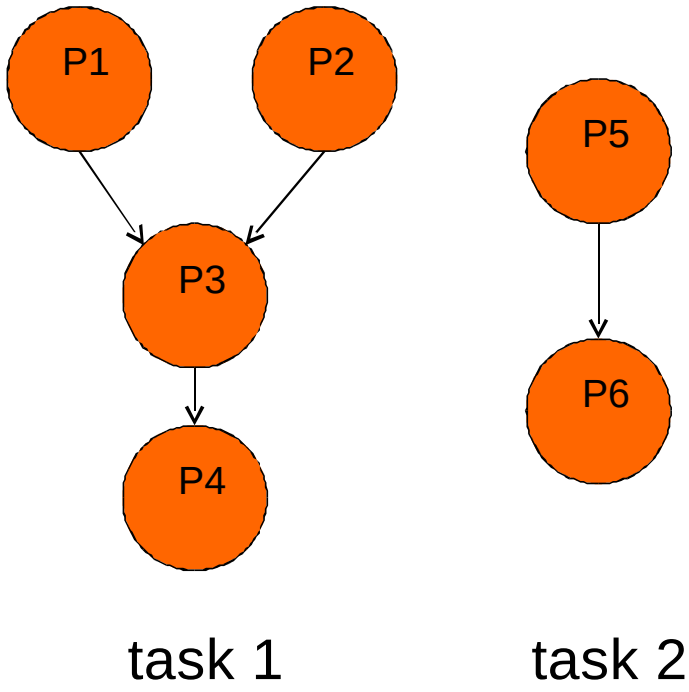
Ρυθμός της διεργασίας: Αντίστροφο της περιόδου.



Task graphs *(γράφημα εργασίας)*

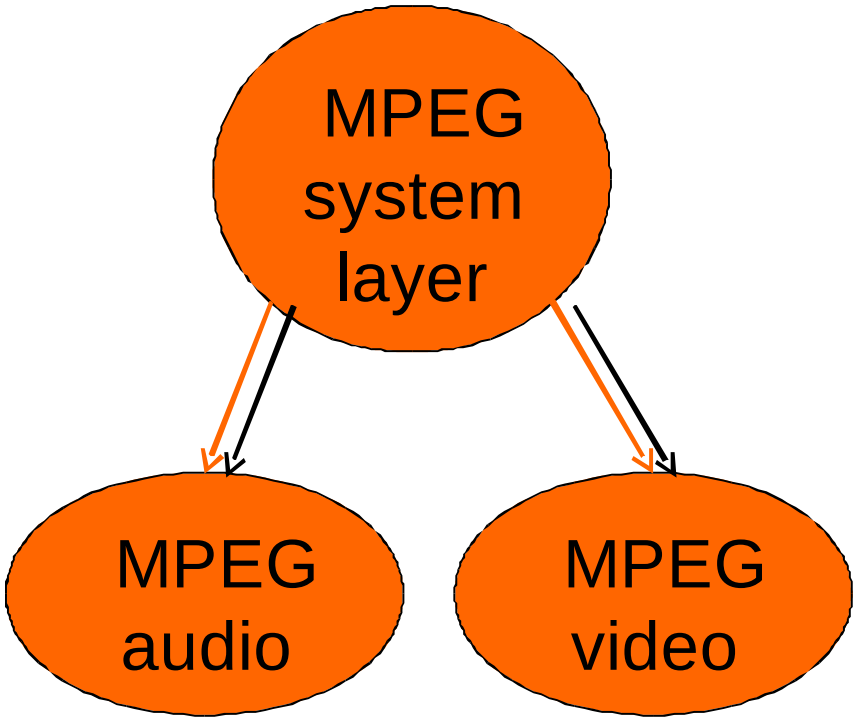
- Οι εργασίες μπορεί να έχουν εξαρτήσεις δεδομένων---πρέπει να εκτελούνται με συγκεκριμένη σειρά.
- Το γράφημα εργασίας δείχνει εξαρτήσεις δεδομένων/ελέγχου μεταξύ των διεργασιών.
- **Εργασία**: συνδεδεμένο σύνολο διεργασιών.
- **Σύνολο εργασιών**: Μια ή περισσότερες εργασίες *(ολόκληρο το γράφημα εργασίας).*

Ένα σύνολο διεργασιών με εξαρτήσεις δεδομένων. Δεν απεικονίζεται το είδος της επικοινωνίας.



Επικοινωνία μεταξύ εργασιών

- Το γράφημα εργασίας υποθέτει ότι όλες οι διεργασίες σε κάθε εργασία τρέχουν με τον ίδιο ρυθμό, οι εργασίες δεν επικοινωνούν.
- Στην πραγματικότητα, κάποιο ποσοστό της **επικοινωνίας μεταξύ των εργασιών** είναι απαραίτητο.
  - Είναι δύσκολο να απαιτηθεί άμεση ανταπόκριση για multi-rate επικοινωνία.



Χαρακτηριστικά εκτέλεσης διεργασίας

- Χρόνος εκτέλεσης διεργασίας T_i.
  - Χρόνος εκτέλεσης με απουσία προτίμησης.
  - Πιθανές χρονικές μονάδες: **δευτερόλεπτα, κύκλοι ρολογιού**.
  - Οι χρόνοι εκτέλεσης χειρότερης και καλύτερης περίπτωσης μπορεί να είναι χρήσιμοι σε μερικές περιπτώσεις.
- Πηγές μεταβλητότητας:
  - Εξαρτήσεις δεδομένων.
  - Σύστημα μνήμης.
  - Διασωλήνωση CPU.

Χρησιμοποίηση

- Χρησιμοποίηση CPU:
  - Ποσοστό της CPU που κάνει χρήσιμη δουλειά.
  - Συχνά υπολογίζεται υποθέτοντας μηδενική χρονοπρογραμματιστική επιβάρυνση.
- Χρησιμοποίηση:
  - U = *(CPU χρόνος για χρήσιμη εργασία)/(συνολικός CPU χρόνος)*

    $= [\Sigma_{t1 \le t \le t2} T(t)] / [t2 - t1]$

    $= T/t$

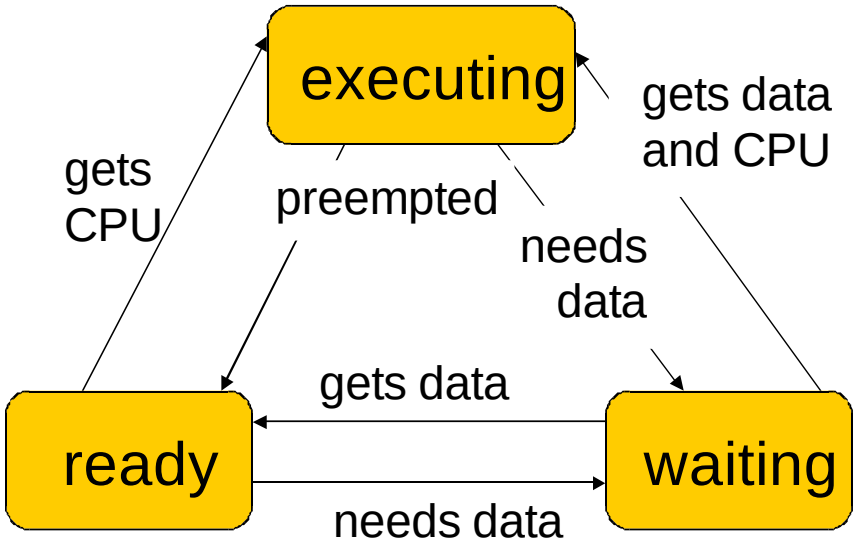
t = χρόνος περιόδου

T = χρόνος υπολογισμού

Κατάσταση μιας διεργασίας – μοντέλο 3 καταστάσεων

- Μια διεργασία μπορεί να είναι σε μια από τις τρεις καταστάσεις:
  - Να **εκτελείται** στη CPU,
  - **έτοιμη** να τρέξει,
  - Να **περιμένει** για δεδομένα.

Η εργασία της επιλογής της σειράς εκτέλεσης των διεργασιών είναι γνωστή σαν χρονοπρογραμματισμός.



Το πρόβλημα χρονοπρογραμματισμού

- Μπορούμε να τηρήσουμε όλες τις προθεσμίες;
  - Πρέπει να μπορούμε να τηρήσουμε τις προθεσμίες σε όλες τις περιπτώσεις.
- Πόση CPU *"ιπποδύναμη"* χρειαζόμαστε για να τηρήσουμε τις προθεσμίες μας;

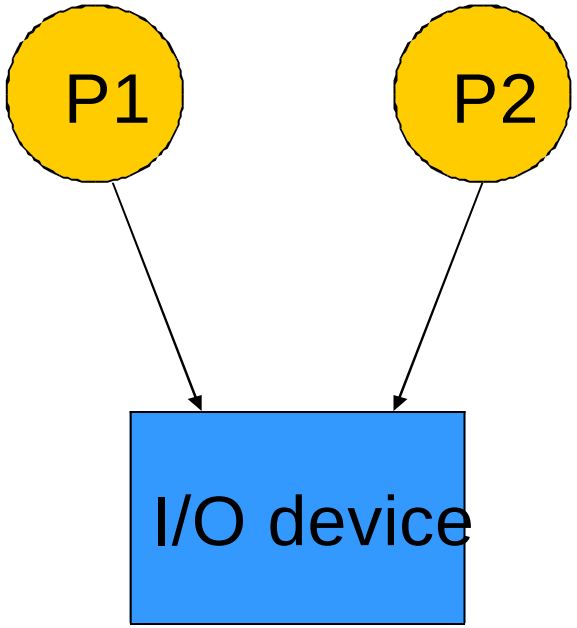
Προτεραιότητα διεργασίας:

Ένας ακέραιος αριθμός που ανατίθεται σε κάθε διεργασία. Κάθε φορά επιλέγεται η διεργασία με την υψηλότερη προτεραιότητα.

Υπάρχει πολυπλοκότητα ως προς την υλοποίηση. Είναι οι προτεραιότητες σταθερές ή μπορούν να αλλάζουν; Πως καθορίζονται;

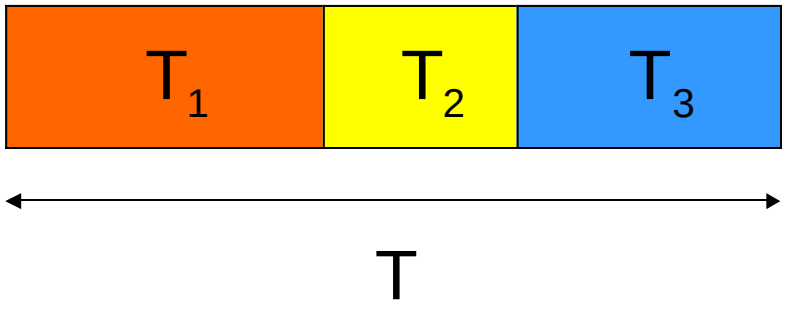
Χρονοπρογραμματιστική εφικτότητα

- Οι περιορισμοί πόρων κάνουν την ανάλυση χρονοπρογραμματισμού NP-hard.
- Πρέπει να φαίνεται ότι οι προθεσμίες τηρούνται για όλους τους χρόνους των απαιτήσεων των πόρων.



Εφικτότητα απλού επεξεργαστή

- Υποθέτουμε:
  - Δεν υπάρχουν διενέξεις πόρων.
  - Συνεχείς χρόνοι εκτέλεσης διεργασιών.
- Απαιτούμε:
  - $T \geq \Sigma_i \ T_i$
  - Δεν μπορεί να χρησιμοποιηθεί περισσότερο από το 100% της CPU.



Χρονοπρογραμματισμός προτεραιότητας

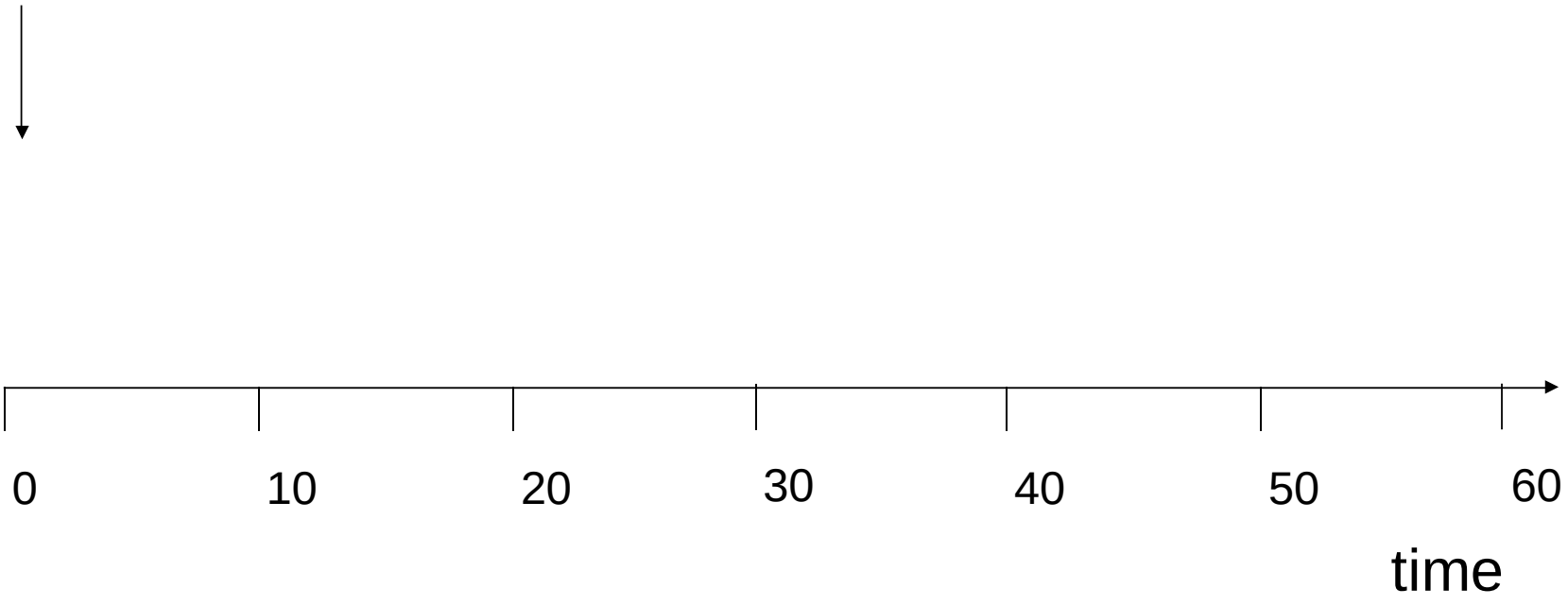
- Κάθε διεργασία έχει μια προτεραιότητα.
- Η CPU πηγαίνει στη διεργασία με την υψηλότερη προτεραιότητα που είναι έτοιμη.
- Οι προτεραιότητες καθορίζουν την πολιτική χρονοπρογραμματισμού:
  - Καθορισμένη προτεραιότητα ,
  - Προτεραιότητες που ποικίλουν στο χρόνο.

Παράδειγμα χρονοπρογραμματισμού προτεραιότητας (1/6)

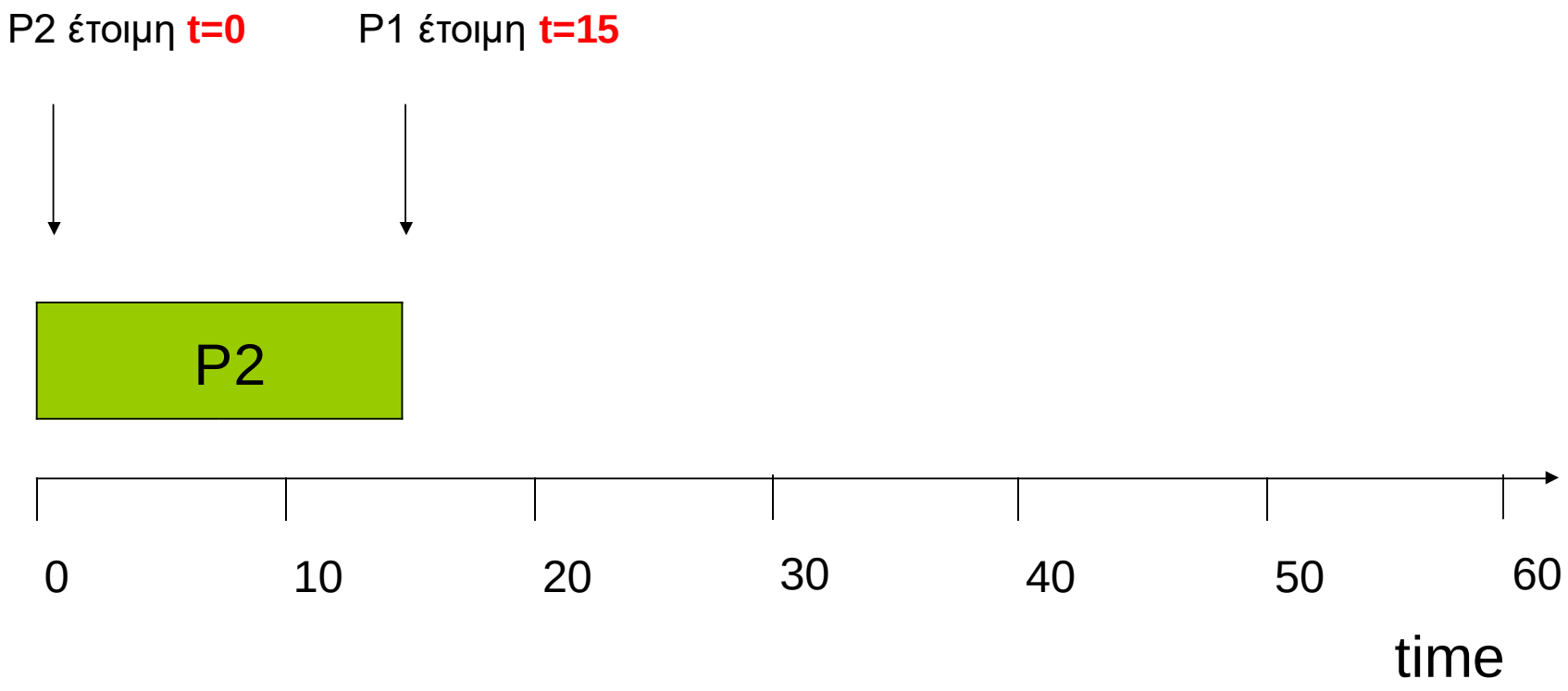
- Κανόνες:
  - Κάθε διεργασία έχει μια καθορισμένη προτεραιότητα *(1 υψηλότερη),*
  - Η διεργασία με την υψηλότερη προτεραιότητα που είναι έτοιμη παίρνει τη CPU,
  - Η διεργασία συνεχίζει μέχρι να ολοκληρωθεί.
- Διεργασίες
  - P1: προτεραιότητα 1, χρόνος εκτέλεσης 10.
  - P2: προτεραιότητα 2, χρόνος εκτέλεσης 30.
  - P3: προτεραιότητα 3, χρόνος εκτέλεσης 20.

Παράδειγμα χρονοπρογραμματισμού προτεραιότητας (2/6)

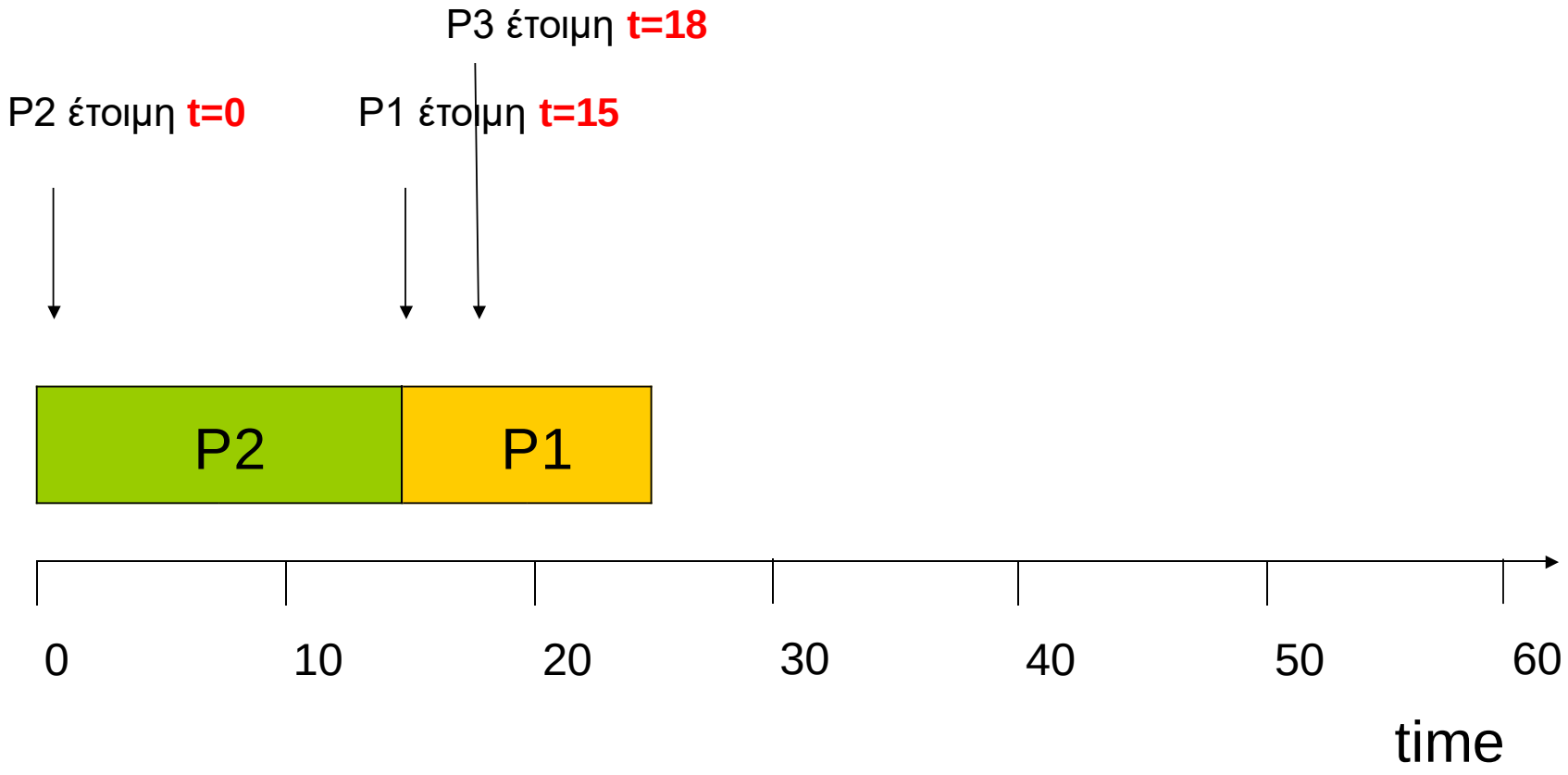
P2 έτοιμη **t=0**



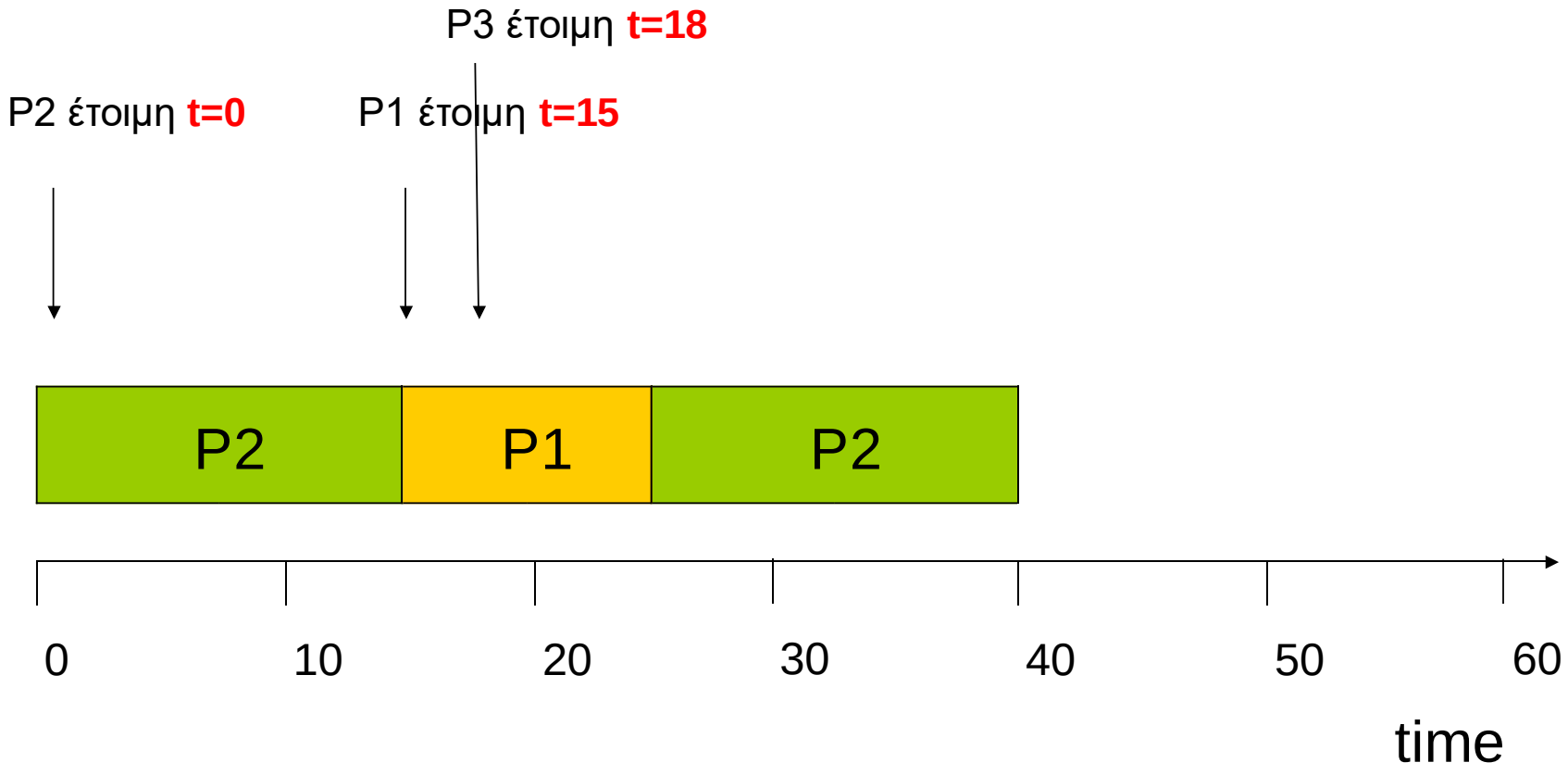
Παράδειγμα χρονοπρογραμματισμού προτεραιότητας (3/6)



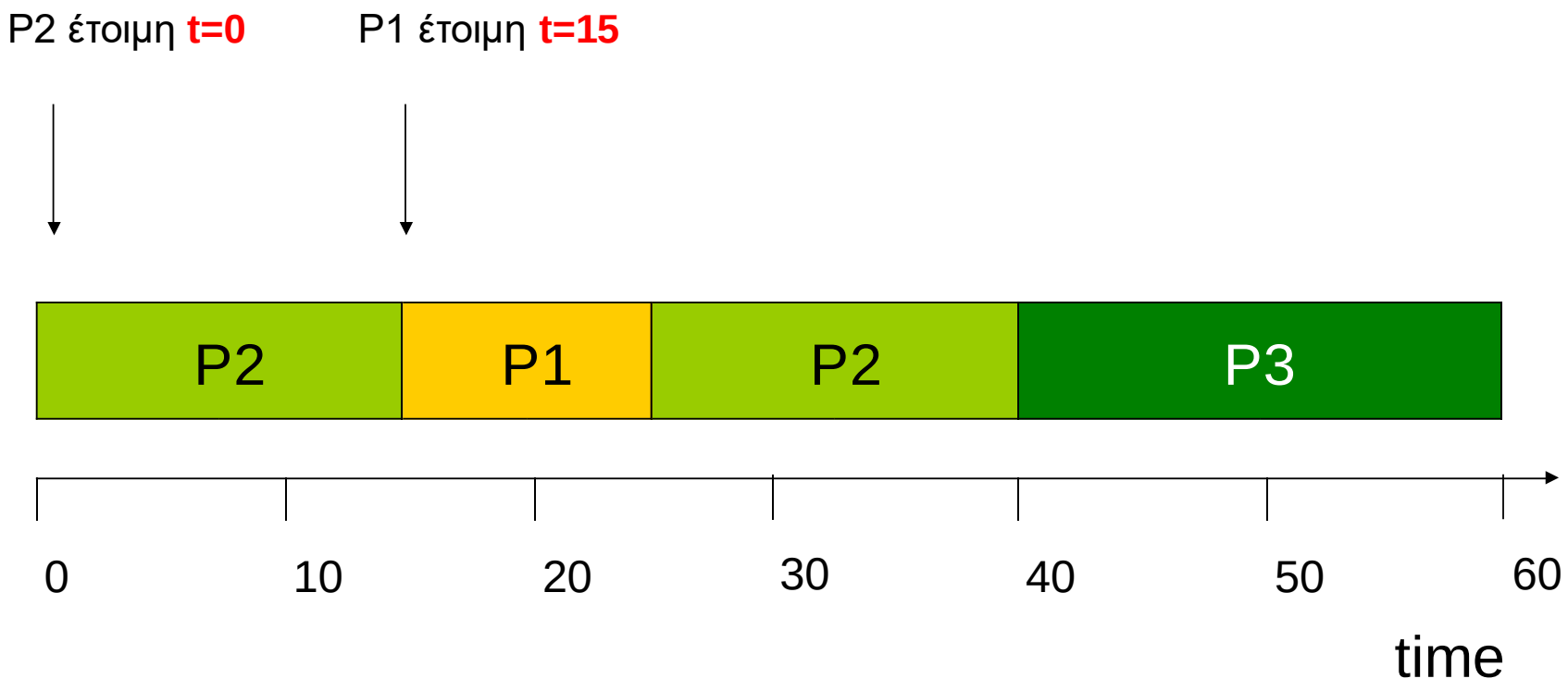
Παράδειγμα χρονοπρογραμματισμού προτεραιότητας (4/6)



Παράδειγμα χρονοπρογραμματισμού προτεραιότητας (5/6)



Παράδειγμα χρονοπρογραμματισμού προτεραιότητας (6/6)



Διαδιεργασιακή επικοινωνία

- **Διαδιεργασιακή επικοινωνία** *(IPC)*:
  Το ΛΣ παρέχει μηχανισμούς τέτοιους,
  ώστε οι διεργασίες να έχουν πρόσβαση
  στα δεδομένα.
- Δύο ειδών σημασιολογίες:
  - **blocking**: η διεργασία προς αποστολή περιμένει για απάντηση,
  - **non-blocking**: η διεργασία προς αποστολή συνεχίζει.

IPC μορφές

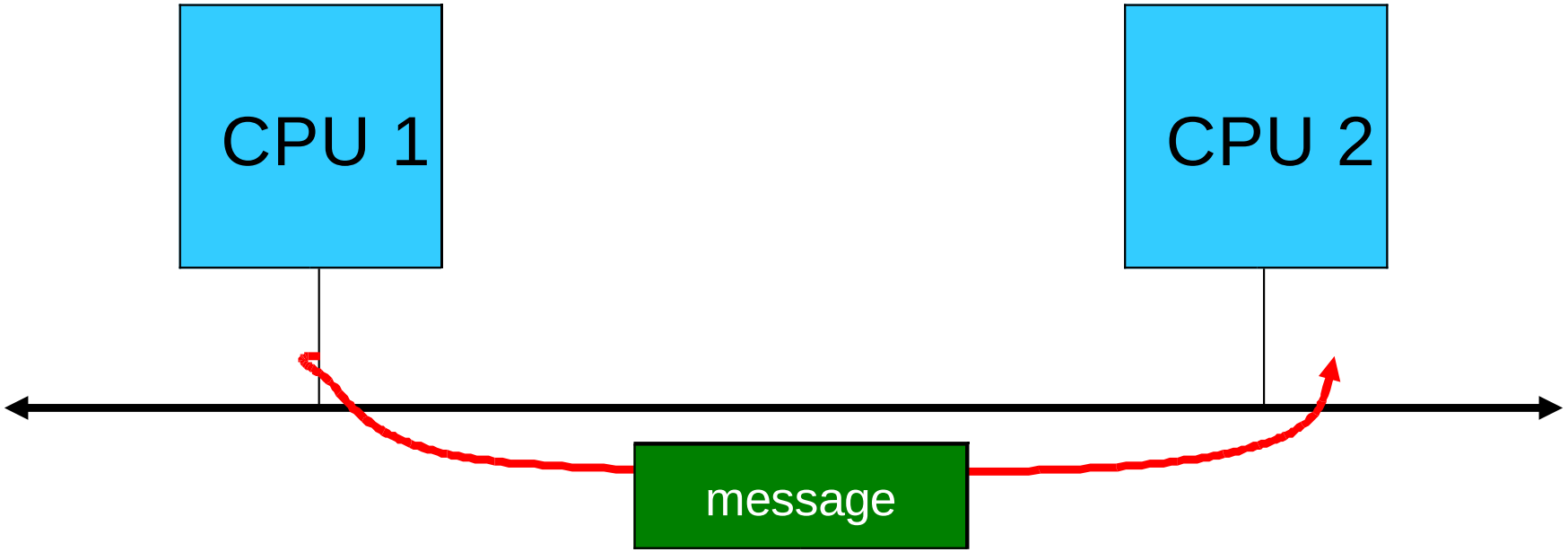
- Διαμοιραζόμενη μνήμη:
  - Οι διεργασίες έχουν κάποια κοινή μνήμη,
  - Πρέπει να συνεργάζονται για να αποφεύγουν κατεστραμμένα/χαμένα μηνύματα.
- Διέλευση μηνύματος:
  - Οι διεργασίες στέλνουν μηνύματα μέσω ενός καναλιού επικοινωνίας---δεν υπάρχει χώρος κοινών διευθύνσεων.
- POSIX σήματα.

Διέλευση μηνύματος (1/3)

- Διέλευση μηνύματος σε ένα δίκτυο:

Διέλευση μηνύματος (2/3)

- Διέλευση μηνύματος σε ένα δίκτυο:

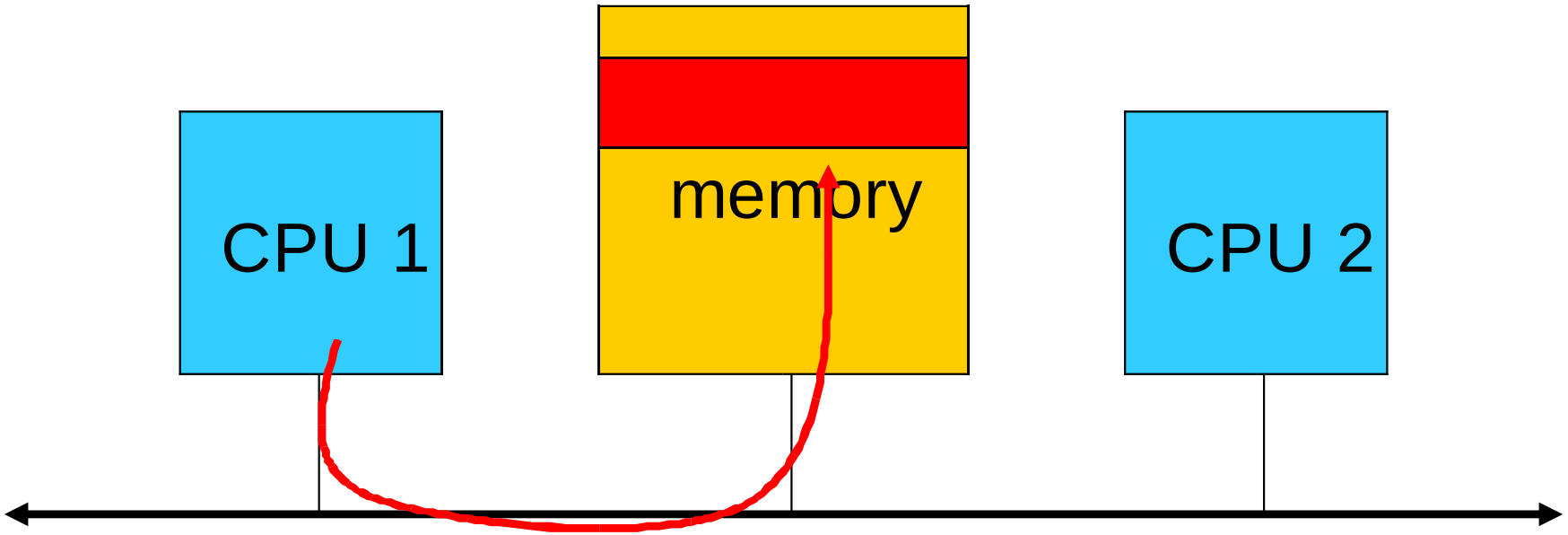


Διέλευση μηνύματος (3/3)

- Διέλευση μηνύματος σε ένα δίκτυο:

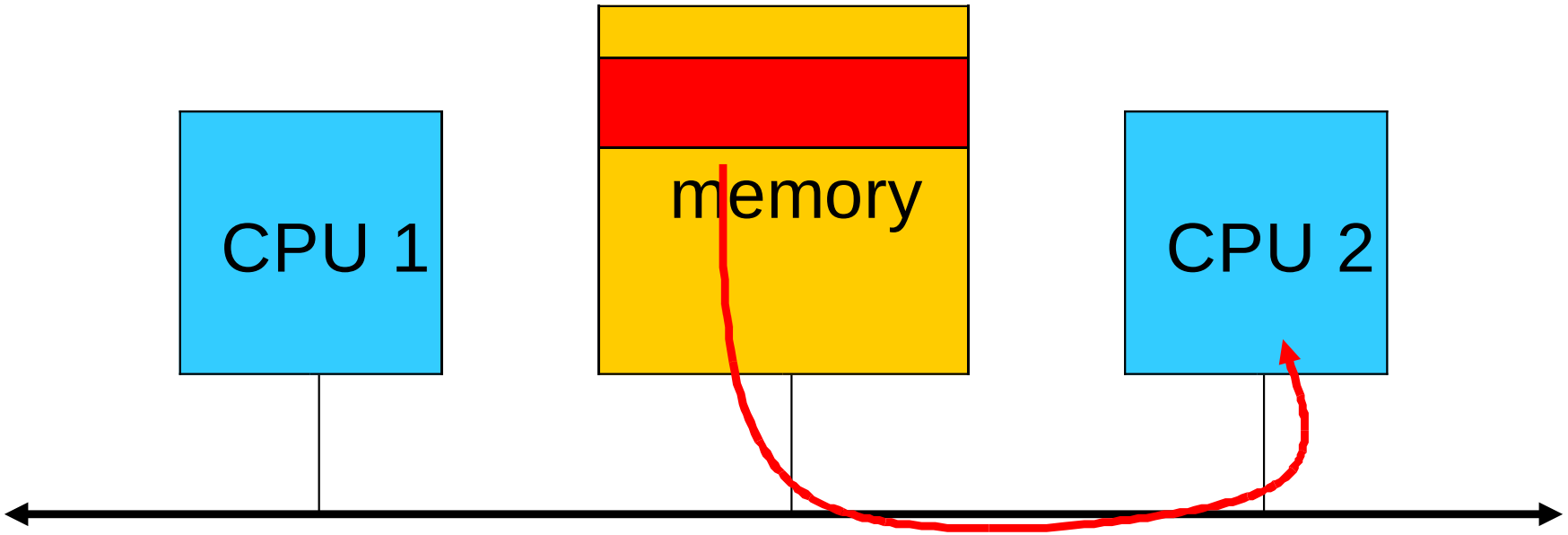
Διαμοιραζόμενη μνήμη (1/2)

- Διαμοιραζόμενη μνήμη σε ένα δίαυλο:



Διαμοιραζόμενη μνήμη (2/2)

- Η κοινόχρηστη μνήμη απεικονίζεται στο χώρο μνήμης της διεργασίας με τη χρήση της nmap()



Κατάσταση ανταγωνισμού
σε διαμοιραζόμενη μνήμη

- Πρόβλημα όταν 2 CPUs προσπαθούν να γράψουν στην ίδια τοποθεσία:
  - Η CPU 1 διαβάζει τη σημαία και βλέπει 0.
  - Η CPU 2 διαβάζει τη σημαία και βλέπει 0.
  - Η CPU 1 θέτει τη σημαία σε ένα και γράφει στην τοποθεσία.
  - Η CPU 2 θέτει τη σημαία σε ένα και ξαναγράφει την τοποθεσία.

Απαιτείται κατάλληλη προστασία με αμοιβαίο αποκλεισμό.

Ατομικό test-and-set

- Το πρόβλημα μπορεί να λυθεί με ένα ατομικό test-and-set:
  - Μια απλή λειτουργία διαύλου διαβάζει την τοποθεσία στη μνήμη, την ελέγχει, γράφει σε αυτή.
- ARM test-and-set παρεχόμενο από SWP:

```
            ADR r0,SEMAPHORE
            LDR r1,#1
GETFLAG SWP r1,r1,[r0]
            BNZ GETFLAG
```

Κρίσιμες Περιοχές

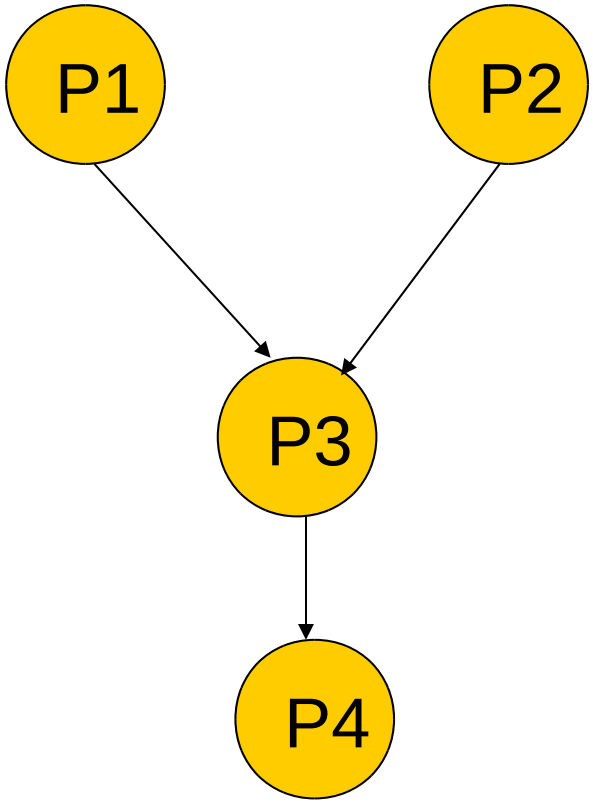
- **Κρίσιμη περιοχή**: κομμάτι του κώδικα που δεν μπορεί να διακοπεί από άλλη διεργασία.
- *Παραδείγματα:*
  - Γράψιμο στην ίδια τοποθεσία μνήμης,
  - Πρόσβαση σε I/O συσκευή.

Σημαφόροι

- **Σημαφόρος:** δομή δεδομένων του ΛΣ για τον έλεγχο της πρόσβασης στις κρίσιμες περιοχές.
- Πρωτόκολλο:
  - Πρόσβαση στο σημαφόρο με **P()**.
  - Εκτέλεση των λειτουργιών της κρίσιμης περιοχής.
  - Απελευθέρωση του σημαφόρου με **V().**

Εξαρτήσεις δεδομένων διεργασίας

- Μια διεργασία μπορεί να μην είναι ικανή να ξεκινήσει μέχρι να τελειώσει μια άλλη.
- Οι εξαρτήσεις δεδομένων ορίζονται σε ένα **γράφημα εργασιών**.
- Όλες οι διεργασίες σε μια εργασία τρέχουν με τον ίδιο ρυθμό.



Άλλες λειτουργίες του ΛΣ

- Ημερομηνία/ώρα.
- Σύστημα αρχείων.
- Δικτύωση.
- Ασφάλεια.

ΠΑΡΑΡΤΗΜΑ
Πολιτικές χρονοπρογραμματισμού

Διεργασίες και Λ.Σ.

- Πολιτικές χρονοπρογραμματισμού:
  - RMS,
  - EDF.
- Παραδοχές χρονοπρογραμματιστικής μοντελοποίησης.

Η πολιτική χρονοπρογραμματισμού καθορίζει τον τρόπο με τον οποίο επιλέγονται οι διεργασίες για να προωθηθούν στην κατάσταση εκτέλεσης.

Ποσοστιαίος μονοτονικός χρονοπρογραμματισμός

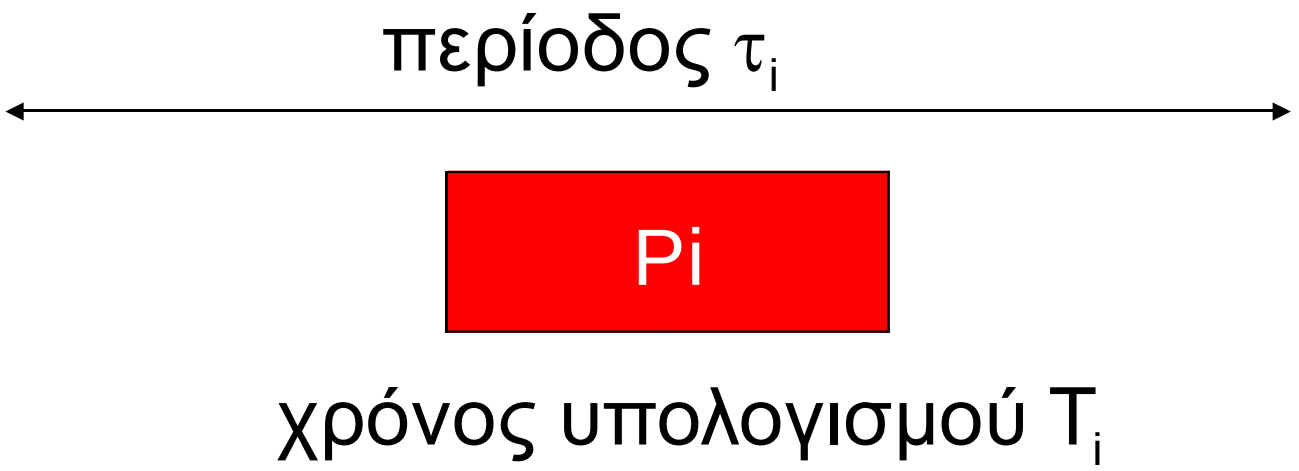
- **RMS** *(Liu and Layland,1973)*: ευρέως χρησιμοποιούμενη, αναλύσιμη πολιτική χρονοπρογραμματισμού RTOS.
- Ανάλυση γνωστή ως **Ποσοστιαία μονοτονική ανάλυση** *(RMA).*
- Στατική πολιτική χρονοπρογραμματισμού.

RMA μοντέλο (θεωρητικό υπόβαθρο)

- Στατική πολιτική χρονοπρογραμματισμού.
- Όλη η διεργασία τρέχει σε μια CPU.
- Μηδενικός χρόνος μεταγωγής πλαισίου.
- Καμία εξάρτηση δεδομένων μεταξύ διεργασιών.
- Ο χρόνος εκτέλεσης διεργασίας είναι συνεχής.
- Η προθεσμία είναι στο τέλος της περιόδου.
- Η διεργασία **υψηλότερης προτεραιότητας** που είναι έτοιμη/τρέχει.
  - Οι προτεραιότητες ανατίθενται με σειρά κατάταξης περιόδου *(η συντομότερη περίοδος παίρνει την υψηλότερη προτεραιότητα).*

Παράμετροι διεργασίας

- Τι είναι ο χρόνος υπολογισμού της διεργασίας i, τ_i είναι η περίοδος της διεργασίας i.



Παράδειγμα RMS (1/2)

Διεργασία	Χρόνος Εκτέλεσης	Περίοδος
P1	1	4
P2	2	6
P3	3	12

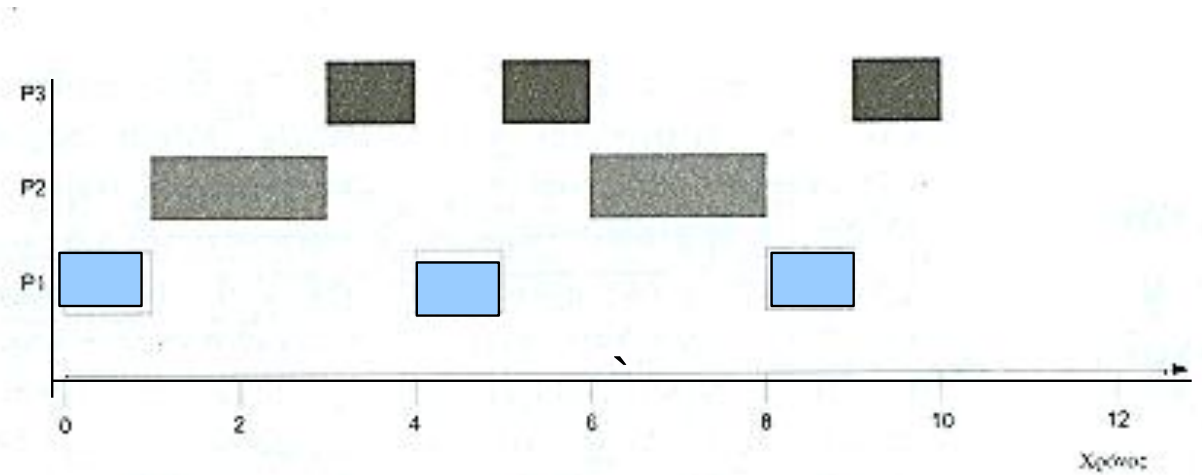
✓ **Ως εκ τούτου:**

Η P1 υψηλότερη προτεραιότητα κ.ο.κ.

Παράδειγμα RMS (2/2)

Για να καταλάβουμε τις αλληλεπιδράσεις χρειάζεται να δημιουργήσουμε ένα ξετυλιγμένο χρονοδιάγραμμα *(unrolled schedule)* για χρόνο ίσο με το ελάχιστο κοινό πολλαπλάσιο (12).

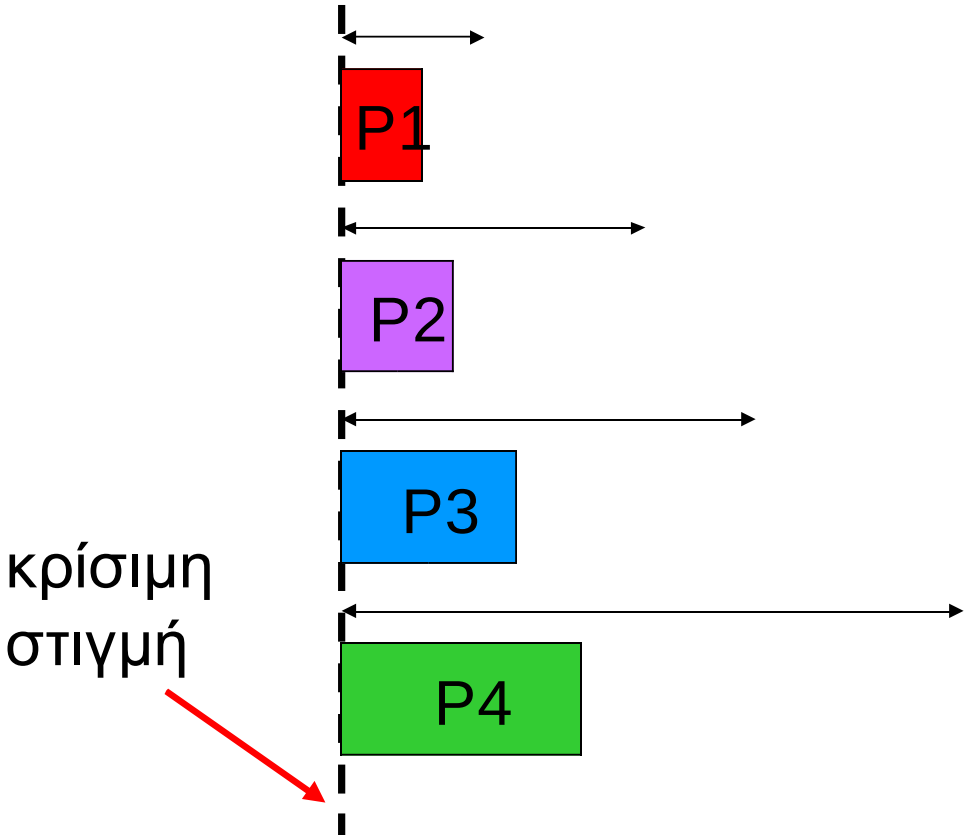
- ✓ η P3 διακόπτεται 3 φορές
- ✓ Αν P1,P2 είχαν χρόνους εκτέλεσης 2,3 τότε μη χρονοδρομολογήσιμο σύστημα



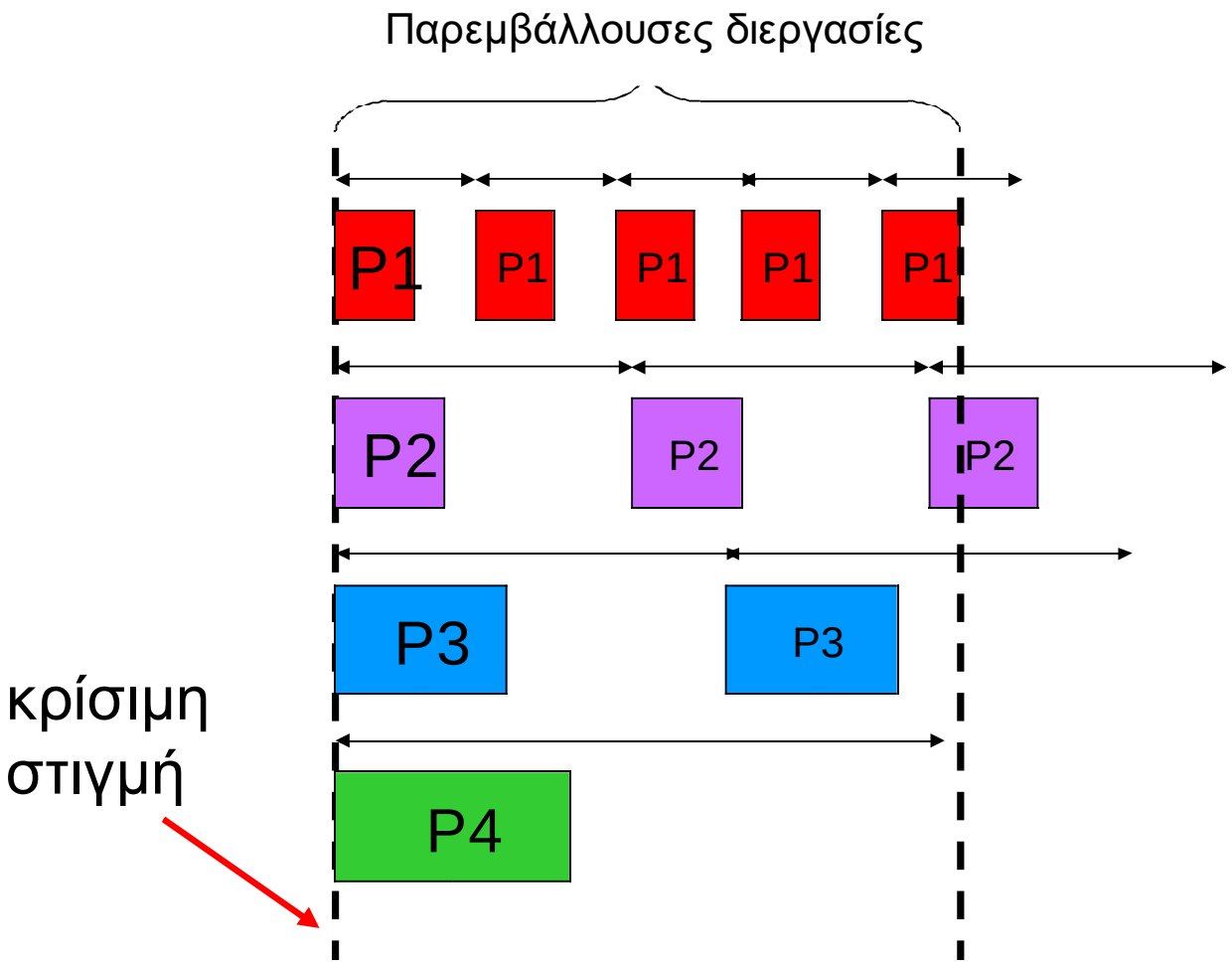
Πορροστιαία μονοτονική ανάλυση

- **Χρόνος απόκρισης**: ο απαιτούμενος χρόνος για την ολοκλήρωση της διεργασίας.
- [**Critical instant** *(κρίσιμη στιγμή)*: χρονοπρογραμματιστική κατάσταση που δίνει το χειρότερο χρόνο απόκρισης.]
- Η κρίσιμη στιγμή συμβαίνει όταν όλες οι διεργασίες υψηλότερης προτεραιότητας είναι έτοιμες να εκτελεστούν.
- Η ανάλυση κρίσιμης στιγμής υπονοεί ότι οι προτεραιότητες πρέπει να ανατεθούν με τη σειρά των περιόδων.
- Το μοντέλο στηρίζεται από μαθηματικές σχέσεις.

Κρίσιμη στιγμή (1/2)



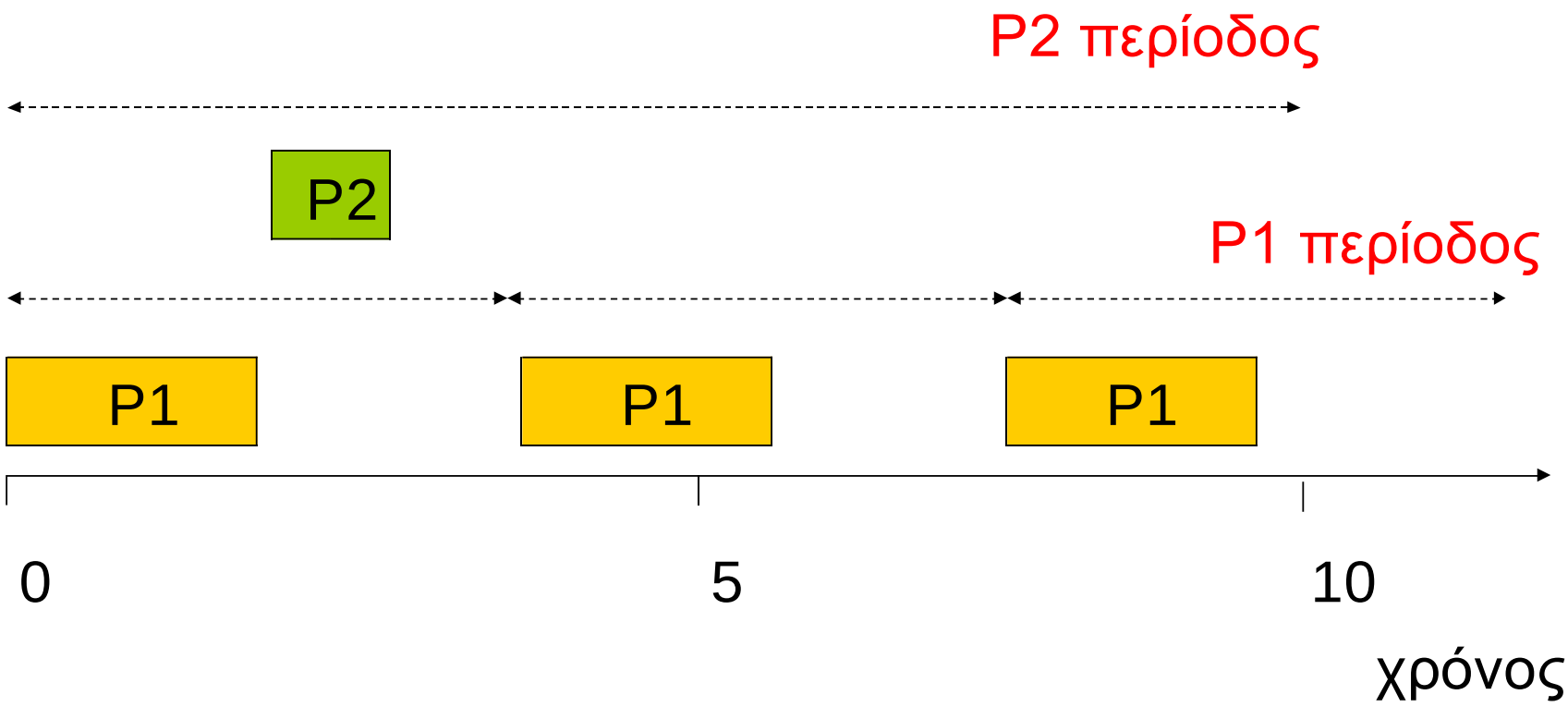
Κρίσιμη στιγμή (2/2)



Προτεραιότητες Ποσοστιαίου μονοτονικού χρονοπρογραμματισμού (RMS)

- Βέλτιστη *(καθορισμένη)* ανάθεση προτεραιότητας:
  - Η διεργασία με τη μικρότερη περίοδο παίρνει την υψηλότερη προτεραιότητα,
  - Η προτεραιότητα αντιστρόφως ανάλογη με την περίοδο,
  - Αυθαίρετη απομάκρυνση εξαρτήσεων.
- Κανένα σχήμα καθορισμένης προτεραιότητας δεν το κάνει καλύτερα.

RMS παράδειγμα



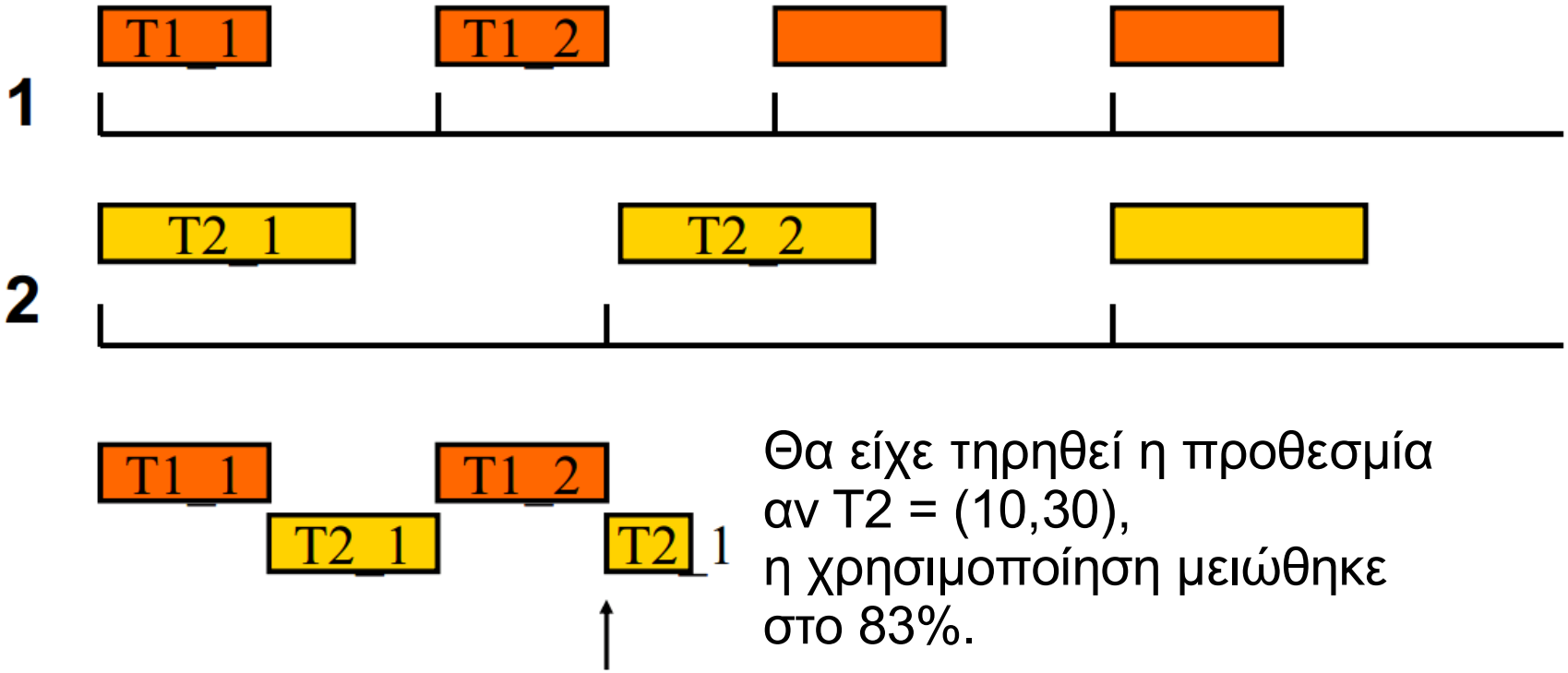
RMS CPU χρησιμοποίηση

- Το μοντέλο RMS δεν επιτρέπει στο σύστημα να αξιοποιήσει το 100% των διαθέσιμων κύκλων, γιατί οι προτεραιότητες ανατίθενται στατικά.
- Η χρησιμοποίηση για n διεργασίες είναι

$$\Sigma_i = T_i / \tau_i$$

- Καθώς ο αριθμός των εργασιών προσεγγίζει το άπειρο, η μέγιστη χρησιμοποίηση προσεγγίζει το 69%.
- Δύο διεργασίες προσεγγίζουν το 83%
- **Max_cpu_utilization = $m(2^{1/m}-1)$**

RMS χαμένη προθεσμία



Θα είχε τηρηθεί η προθεσμία αν T2 = (10,30), η χρησιμοποίηση μειώθηκε στο 83%.

Η εργασία 1 της T2 χάνει την προθεσμία της

RMS χρησιμοποίηση CPU

- Ο RMS δεν μπορεί να χρησιμοποιήσει το 100% της CPU, ακόμα και με μηδενική επιβάρυνση μεταγωγής πλαισίου.
- Πρέπει να κρατήσει τους κύκλους αδράνειας διαθέσιμους για να χειριστεί το σενάριο χειρότερης περίπτωσης.
- Ωστόσο, ο RMS εγγυάται ότι όλες οι διεργασίες θα τηρούν πάντα τις προθεσμίες τους.

RMS υλοποίηση

- Αποτελεσματική υλοποίηση:
  - σάρωση διεργασιών,
  - επιλογή της ενεργής διεργασίας με την υψηλότερη προτεραιότητα.

Earliest-deadline-first χρονοπρογραμματισμός

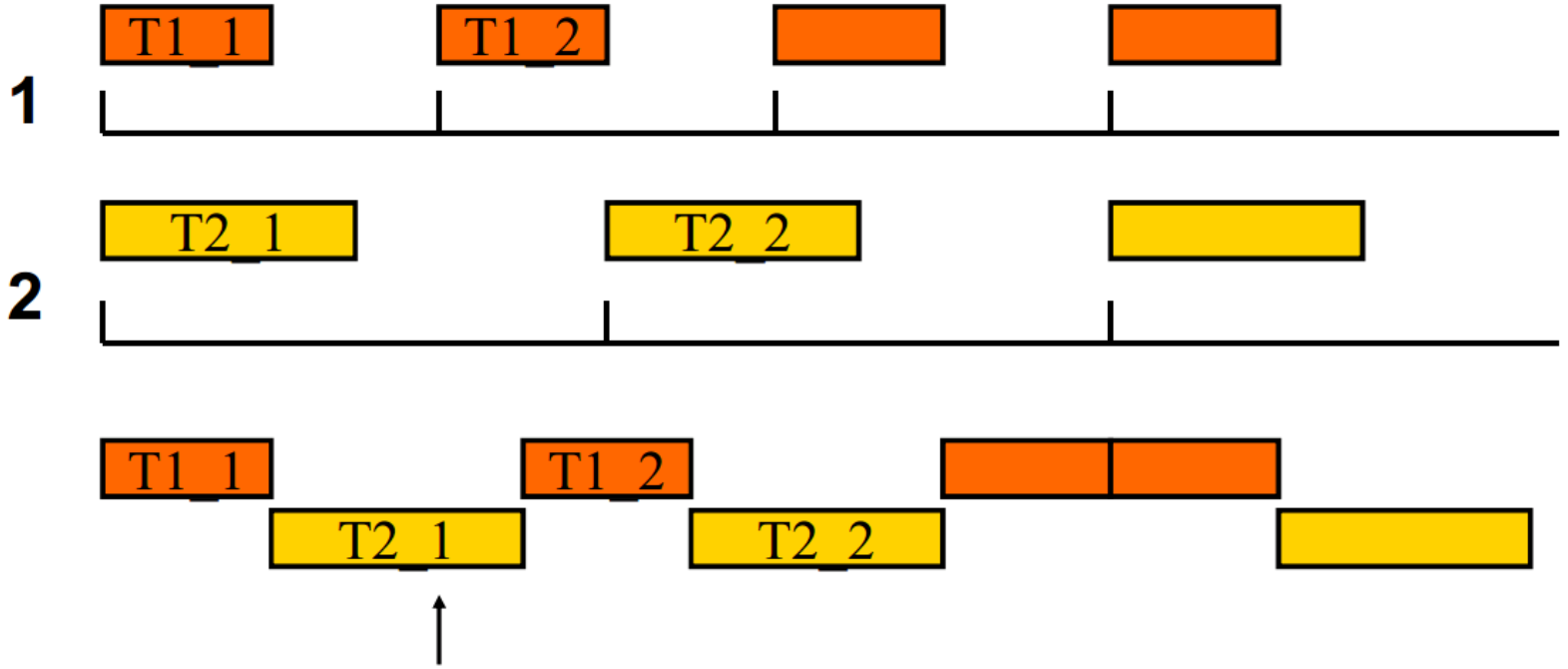
- **EDF**: χρονοπρογραμματιστικό σχήμα δυναμικής προτεραιότητας.
- Η διεργασία πλησιέστερα στην προθεσμία της έχει την υψηλότερη προτεραιότητα.
- Απαιτεί επανυπολογισμό των διεργασιών σε κάθε διακοπή του χρονιστή.

EDF ανάλυση

- Η EDF μπορεί να χρησιμοποιήσει το 100% της CPU.
- Αλλά η EDF μπορεί να αποτύχει στο να τηρήσει μια προθεσμία.

EDF τήρηση προθεσμίας

T1 = (10,20), T2 = (15,30), η χρησιμοποίηση είναι 100%



Η T2 έχει προτεραιότητα επειδή η προθεσμία της λήγει πιο σύντομα

EDF υλοποίηση

- Σε κάθε διακοπή του χρονιστή:
  - Υπολογίζει το χρόνο για την προθεσμία,
  - Διαλέγει τη διεργασία που είναι πιο κοντά στην προθεσμία.
- Γενικά θεωρείται πολύ ακριβό για χρήση στην πράξη.
- Το σημαντικότερο πρόβλημα είναι η διατήρηση των διεργασιών ταξινομημένων κατά το χρόνο ως την προθεσμία.

Υπερπερίοδος

- **Υπερπερίοδος**: ελάχιστο κοινό πολλαπλάσιο (LCM) των περιόδων εργασιών.
- Πρέπει να δούμε το χρονοδιάγραμμα της υπερπεριόδου για να βρούμε όλες τις αλληλεπιδράσεις των εργασιών.
- Η περίοδος μπορεί να είναι πολύ μεγάλη αν οι περίοδοι των εργασιών δεν επιλεχθούν προσεκτικά.

Παράδειγμα υπερπεριόδου

- Μεγάλη υπερπερίοδος:
  - P1 7 ms.
  - P2 11 ms.
  - P3 15 ms.
  - LCM = 1155 ms.
- Μικρότερη υπερπερίοδος:
  - P1 8 ms.
  - P2 12 ms.
  - P3 16 ms.
  - LCM = 96 ms.

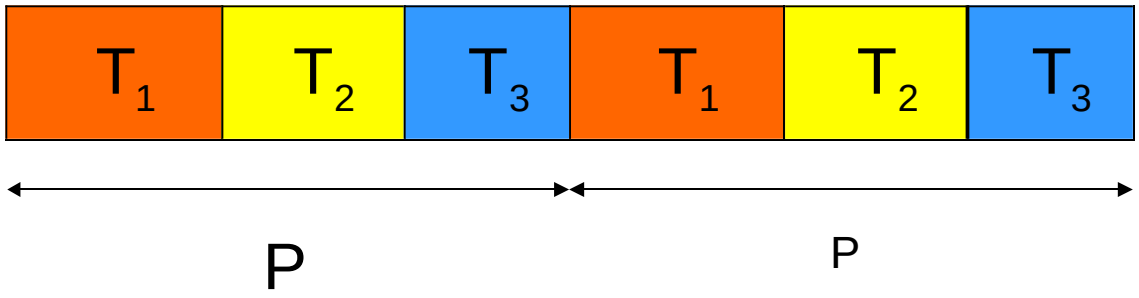
Παράδειγμα δυνατότητας απλού επεξεργαστή

- P1 περίοδος 1 ms, CPU χρόνος 0.1 ms.
- P2 περίοδος 1 ms, CPU χρόνος 0.2 ms.
- P3 περίοδος 5 ms, CPU χρόνος 0.3 ms.

LCM		**5,00E-03**	
	period	**CPU time**	**CPU time/LCM**
P1	1,00E-03	1,00E-04	5,00E-04
P2	1,00E-03	2,00E-04	1,00E-03
P3	5,00E-03	3,00E-04	3,00E-04
	total CPU/LCM		1,80E-03
	utilization		3,60E-01

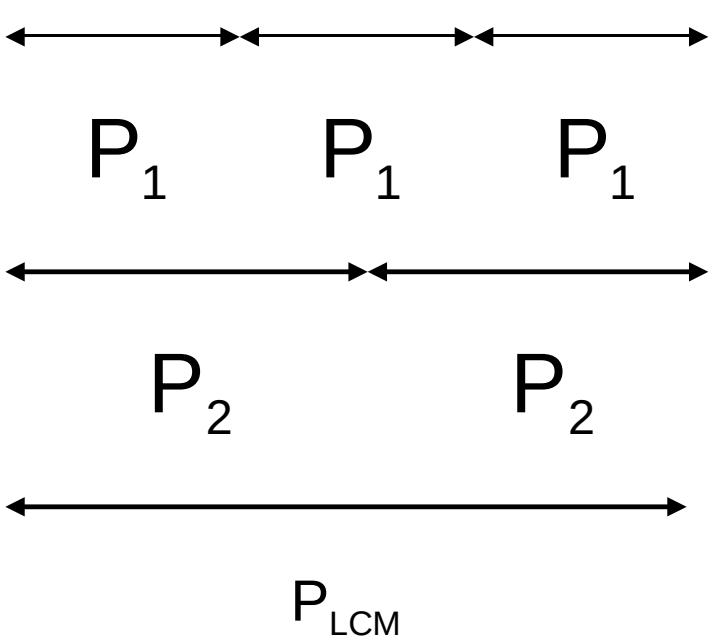
Cyclostatic/TDMA

- Χρονοδιάγραμμα σε χρονοθυρίδες.
  - *Ίδια ενεργοποίηση διεργασίας ανεξαρτήτως του φόρτου εργασίας.*
- Οι χρονοθυρίδες μπορεί να είναι του ίδιου μεγέθους ή όχι.



TDMA παραδοχές

- Το χρονοδιάγραμμα είναι βασισμένο στο ελάχιστο κοινό πολλαπλάσιο *(LCM)* των περιόδων της διεργασίας.
- Ασήμαντος χρονοπρογραμματιστής ➔πολύ μικρή χρονοπρογραμματιστική επιβάρυνση.



TDMA χρονοπρογραμματιστική δυνατότητα

- Πάντα ίδια CPU χρησιμοποίηση *(υποθέτοντας συνεχείς χρόνους εκτέλεσης διεργασίας).*

- Δεν μπορεί να χειριστεί απρόσμενους φόρτους.
  - Πρέπει να χρονοπρογραμματίσει μια χρονοθυρίδα για απεριοδικά γεγονότα.

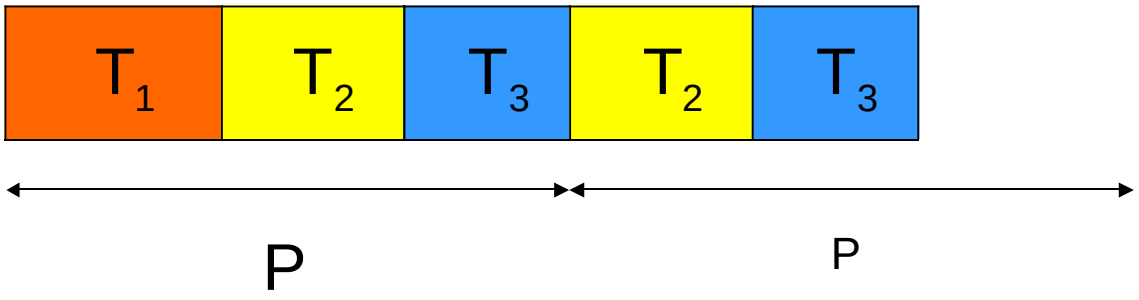
TDMA χρον/κή δυνατότητα *(παράδειγμα)*

- TDMA περίοδος = 10 ms.
- P1 CPU χρόνος 1 ms.
- P2 CPU χρόνος 3 ms.
- P3 CPU χρόνος 2 ms.
- P4 CPU χρόνος 2 ms.

TDMA period	**1,00E-02**
	CPU time
P1	1,00E-03
P2	3,00E-03
P3	2,00E-03
P4	2,00E-03
Total	8,00E-03
utilization	8,00E-01

Round-robin

- Χρονοπρογραμματίζει τη διεργασία μόνο αν είναι έτοιμη.
  - Πάντα ελέγχει τις διεργασίες με την ίδια σειρά.
- Παραλλαγές:
  - Συνεχής περίοδος συστήματος.
  - Έναρξη round-robin ξανά αφού τελειώσει ένας γύρος.



Round-robin παραδοχές

- Το χρονοδιάγραμμα είναι βασισμένο στο ελάχιστο κοινό πολλαπλάσιο *(LCM)* των περιόδων διεργασιών.
- Καλύτερη υλοποίηση με ισότιμες χρονοθυρίδες για τις διεργασίες.
- Απλός χρονοπρογραμματιστής → χαμηλή χρονοπρογραμματιστική επιβάρυνση.
- Μπορεί να υλοποιηθεί σε hardware.

Round-robin χρονοπρογραμματιστική δυνατότητα

- Μπορεί να δεσμεύσει το μέγιστο φόρτο στη CPU.
  - Ίσως αφήσει αχρησιμοποίητους κύκλους της CPU.
- Μπορεί να προσαρμοστεί ώστε να χειρίζεται απροσδόκητο φόρτο.
  - Χρησιμοποιεί χρονοθυρίδες στο τέλος της περιόδου.

Χρονοπρογραμματιστική δυνατότητα και επιβάρυνση

- Η χρονοπρογραμματιστική διεργασία καταναλώνει χρόνο στη CPU.
  - Δεν είναι όλος ο χρόνος στη CPU διαθέσιμος για τις διεργασίες.
- Η χρονοπρογραμματιστική επιβάρυνση πρέπει να ληφθεί υπόψη για το ακριβές χρονοδιάγραμμα.
  - Ίσως να αγνοηθεί αν είναι ένα μικρό τμήμα του συνολικού χρόνου εκτέλεσης.

Περιοδικές διεργασίες

- Χρειάζεται κώδικας για τον έλεγχο της εκτέλεσης των διεργασιών.

- Απλούστερη υλοποίηση:

  διεργασία = υπορουτίνα.

Επιδιορθώση χρονοπρογραμματιστικών προβλημάτων

- Τι γίνεται αν το σετ των διεργασιών δεν μπορεί να χρονοπρογραμματιστεί;
  - Αλλάξτε τις προθεσμίες στις απαιτήσεις.
  - Μειώστε τους χρόνους εκτέλεσης των διεργασιών.
  - Αγοράστε μια πιο γρήγορη CPU.

Αντιστροφή προτεραιότητας

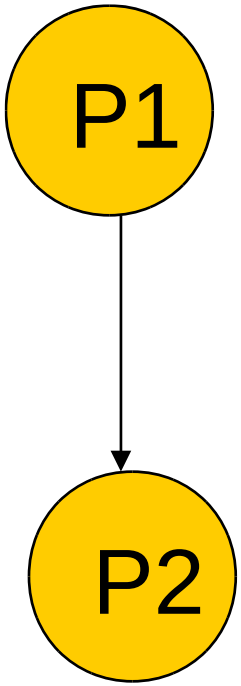
- **Αντιστροφή προτεραιότητας:**
  η διεργασία χαμηλής προτεραιότητας εμποδίζει τη διεργασία υψηλής προτεραιότητας να τρέξει.
- Μη σωστή χρήση των πόρων του συστήματος μπορεί να προκαλέσει χρονοπρογραμματιστικά προβλήματα:
  - η διεργασία χαμηλής προτεραιότητας δεσμεύει την I/O συσκευή.
  - Η διεργασία υψηλής προτεραιότητας χρειάζεται την I/O συσκευή, αλλά δεν μπορεί να την έχει μέχρι να τελειώσει η διεργασία χαμηλής προτεραιότητας.
- Μπορεί να οδηγήσει σε αδιέξοδο.

Επίλυση αντιστροφής προτεραιότητας

- Ανάθεση προτεραιοτήτων στους πόρους του συστήματος.
- Ορίζουμε τη διεργασία να κληρονομεί την προτεραιότητα ενός πόρου που ζητά.
  - Μια διεργασία χαμηλής προτεραιότητας κληρονομεί την προτεραιότητα μιας συσκευής εάν εκείνη τη στιγμή είναι η διεργασία με την υψηλότερη προτεραιότητα.

Εξαρτήσεις δεδομένων

- Οι εξαρτήσεις δεδομένων μας επιτρέπουν να βελτιώνουμε τη χρήση
  - Περιορισμένος συνδυασμός των διεργασιών που μπορούν να τρέχουν συγχρόνως.
- Οι P1 και P2 δεν μπορούν να τρέχουν συγχρόνως.



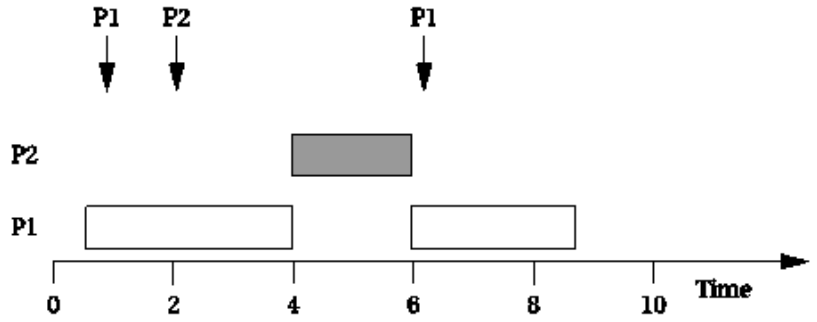
Χρόνος εναλλαγής πλαισίου

- Μη μηδενικός χρόνος εναλλαγής πλαισίου μπορεί να πιέσει τα όρια ενός αυστηρού χρονοδιαγράμματος.
- Δύσκολο να υπολογιστούν οι επιπτώσεις--- εξαρτάται από τη σειρά των μεταγωγών των πλαισίων.
- Στην πράξη, η επιβάρυνση εναλλαγής πλαισίου του ΛΣ είναι μικρή *(εκατοντάδες κύκλοι ρολογιού)* σε σχέση με τις πολλές περιόδους κοινών εργασιών *(ms – μs).*

Αξιολόγηση της Απόδοσης Λειτουργικών Συστημάτων

Χρονοπρογραμματιστική και εναλλαγής πλαισίου επιβάρυνση

Διεργασία	Χρόνος εκτέλεσης	Προθεσμία
P1	3	5
P2	3	10



Με την επιβάρυνση μεταγωγής πλαισίου της 1, δεν υπάρχει εφικτό χρονοδιάγραμμα.

$$2TP1 + TP2 = 2*(1+3)+(1\_3)=11$$

Χρόνος εκτέλεσης διεργασίας

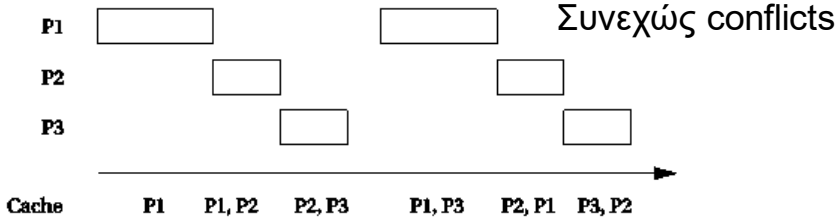
- Ο χρόνος εκτέλεσης διεργασίας δεν είναι συνεχής.
- Παραπάνω χρόνος στη CPU μπορεί να είναι καλός.
- Παραπάνω χρόνος στη CPU μπορεί επίσης να είναι κακός:
  - Η επόμενη διεργασία τρέχει νωρίτερα, προκαλώντας νέα προτίμηση.

Διεργασίες και caches

- Οι διεργασίες μπορούν να επιφέρουν πρόσθετα προβλήματα στην cache .
  - Ακόμα και αν οι ανεξάρτητες διεργασίες συμπεριφέρονται σωστά, οι διεργασίες ίσως παρεμποδίζουν η μια την άλλη.
- Ο χρόνος εκτέλεσης χειρότερης περίπτωσης με κακή συμπεριφορά είναι συνήθως πολύ χειρότερος από το χρόνο εκτέλεσης με καλή συμπεριφορά της cache.

Επιπτώσεις του χρονοπρογραμματισμού στην cache

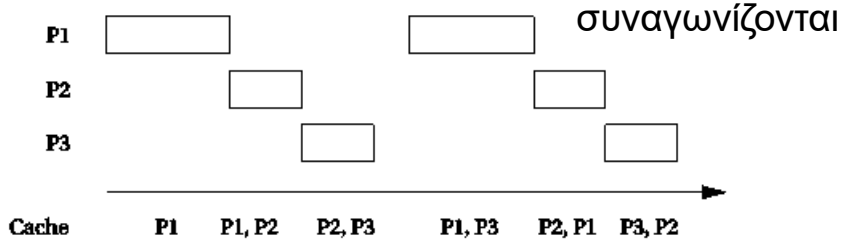
Χρονοδιάγραμμα 1
(LRU cache):



Συνεχώς conflicts

Χρονοδιάγραμμα 2
(η μισή cache προορίζεται για την P1):

Η P1 έχει κλειδώσει σελίδες στη cache, ενώ οι P2, P3 συναγωνίζονται



Στρατηγικές βελτιστοποίησης ισχύος για διεργασίες

Βελτιστοποίηση ισχύος

- **Διαχείριση ισχύος**: καθορίζεται το πώς οι πόροι του συστήματος χρονοπρογραμματίζονται χρησιμοποιούνται για τον έλεγχο της κατανάλωσης ισχύος.
- Το ΛΣ μπορεί να χειριστεί την ισχύ ακριβώς όπως διαχειρίζεται το χρόνο .
- Το ΛΣ μειώνει την ισχύ απενεργοποιώντας κάποιες μονάδες.
  - Ίσως έχει λειτουργίες μερικής απενεργοποίησης.
- Οι καταστάσεις λειτουργίας πρέπει να αλλάζουν προσεκτικά.
- Ο προσδιορισμός του χρόνου εναλλαγής μέσα και έξω από μια λειτουργία ανοίγματος της ισχύος απαιτεί μια ανάλυση της συνολικής δραστηριότητας του συστήματος .

Διαχείριση ισχύος και απόδοση

- Η διαχείριση ισχύος και η απόδοση είναι συχνά αντίθετες.
- Η εισαγωγή της λειτουργίας απενεργοποίησης ισχύος καταναλώνει
  - ενέργεια,
  - χρόνο.
- Η μη εισαγωγή της λειτουργίας απενεργοποίησης ισχύος καταναλώνει
  - ενέργεια,
  - χρόνο.

Πολιτικές απλής διαχείρισης ισχύος

- **Request-driven:** ενεργοποίηση μετά από λήψη αιτήματος. Προσθέτει καθυστέρηση στην απόκριση.
- **Predictive shutdown:** προσπαθεί να μαντέψει πόσος χρόνος απομένει μέχρι το επόμενο αίτημα.
  - Ίσως ξεκινήσει από μόνο του εν αναμονή ενός νέου αιτήματος.
  - Αν μαντέψει λάθος, θα επιβαρύνει με πρόσθετη καθυστέρηση κατά την έναρξη.

Προγνωστικός τερματισμός

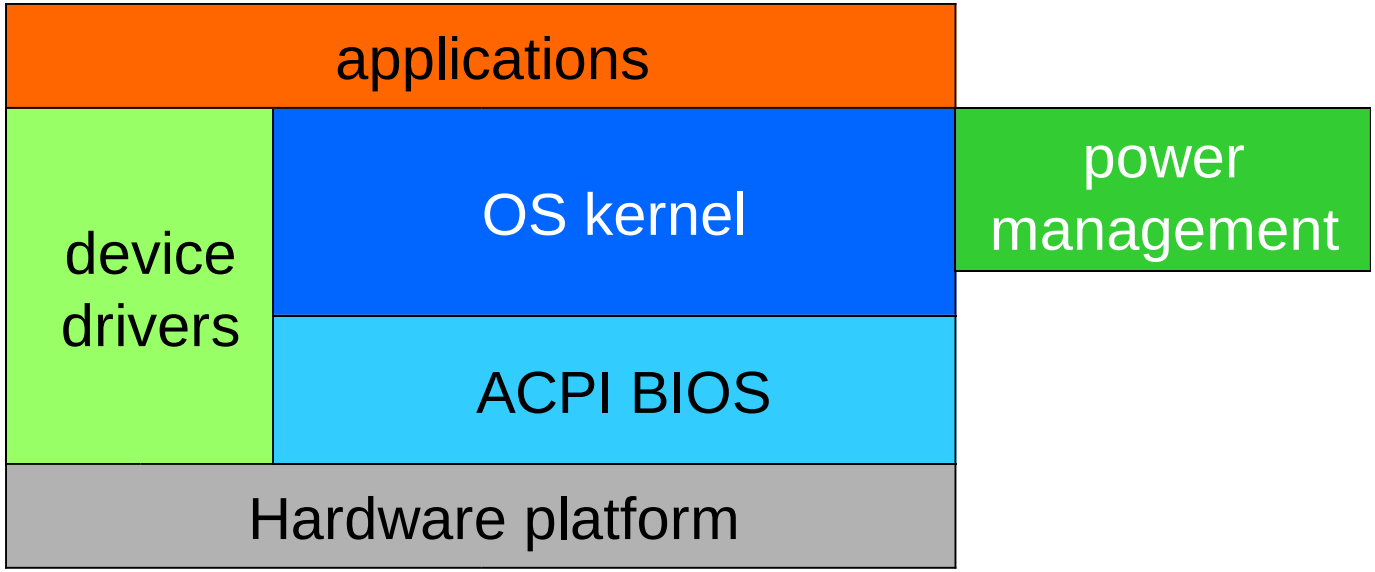
- Οι τεχνικές προγνωστικού τερματισμού είναι πιθανοτικές.
- Υποθέτουμε ότι τα αιτήματα υπηρεσιών είναι πιθανοτικά.
- Βελτιστοποίηση αναμενόμενων αξιών:
  - Κατανάλωση ισχύος,
  - Χρόνος απόκρισης.
- 2 ειδών προβλήματα
  - Ο αιτών περιμένει για να ξεκινήσει το σύστημα.
  - Το σύστημα ξεκινάει χωρίς να υπάρχει λόγος.

➔ **Απλός** προγνωστικός: τερματισμός μετά από χρόνο Ton, έναρξη ξανά μετά από χρόνο Toff.

Προχωρημένη διαμόρφωση και διεπαφή ισχύος (ACPI)

- **ACPI**

  ... ανοιχτό πρότυπο για υπηρεσίες διαχείρισης ισχύος...



ACPI καθολικές καταστάσεις ισχύος

Πέντε βασικές καθολικές καταστάσεις ισχύος:

- G3: mechanical off.
- G2: soft off *(απαιτεί πλήρη επανεκκίνηση ΛΣ):*
  - S1: χαμηλή καθυστέρηση έναρξης χωρίς απώλειες πλαισίων
  - S2: χαμηλή καθυστέρηση με απώλειες στις καταστάσεις των CPU/cache
  - S3: χαμηλή καθυστέρηση με απώλειες όλων των καταστάσεων εκτός της μνήμης
  - S4: κατάσταση χαμηλότερης ισχύος με όλες τις συσκευές απενεργοποιημένες
- G1: κατάσταση ύπνου.
- G0: κατάσταση λειτουργίας.
- Συμβατική κατάσταση παλαιού τύπου.

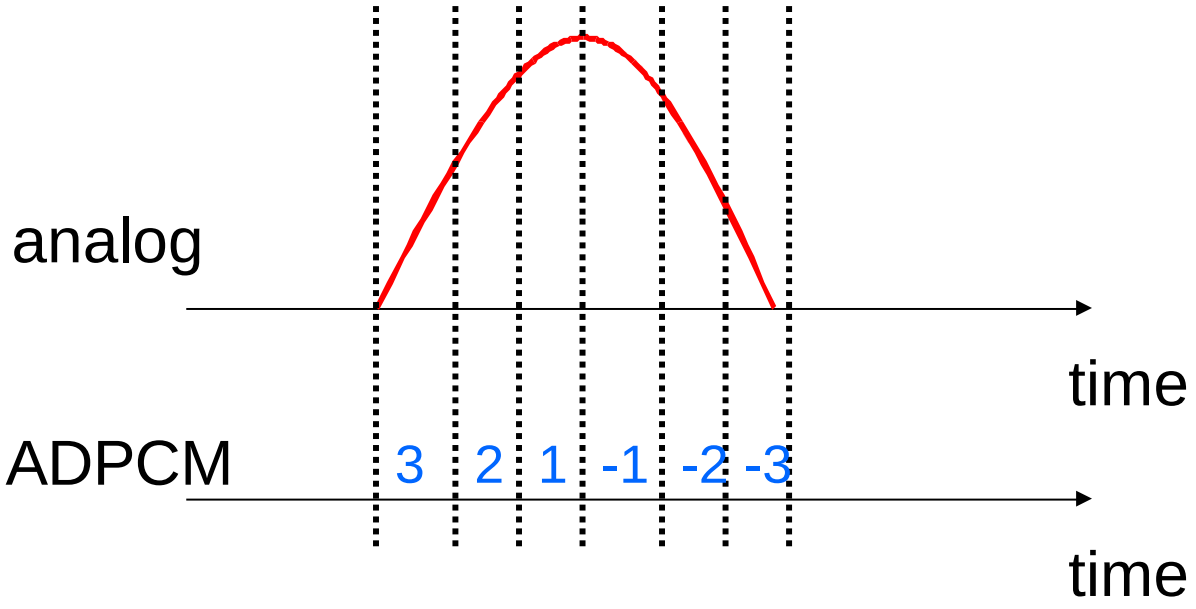
Διεργασίες - Λειτουργικά Συστήματα

Παράρτημα:

- Παράδειγμα Σχεδίασης.
- Αυτόματος Τηλεφωνητής.

Θεωρία της λειτουργίας

- Συμπίεση ήχου με τη χρήση **διαμόρφωσης κώδικα προσαρμοστικού διαφορικού παλμού** (ADPCM).

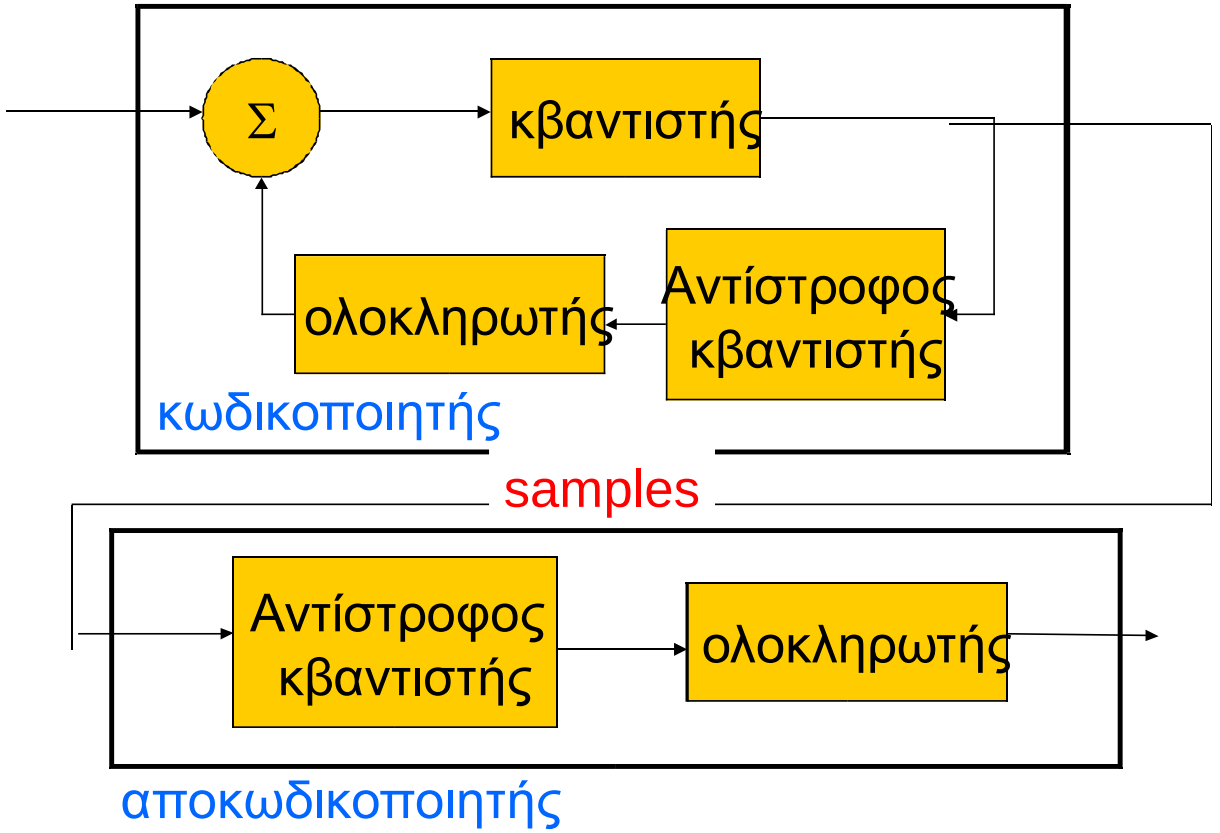


- Μπορεί να παράγει 2X βαθμούς συμπίεσης στα δεδομένα φωνής.
- Εύρος τιμών [-3,..,3]

ADPCM κωδικοποίηση

- Κωδικοποιημένα σε μια μικρή αλφάβητο με θετικές και αρνητικές τιμές.
  - {-3,-2,-1,1,2,3}
- Ελαχιστοποιεί το σφάλμα ανάμεσα στην προβλεπόμενη τιμή και την πραγματική τιμή του σήματος.

ADPCM σύστημα συμπίεσης



Υπάρχει feedback για ελαχιστοποίηση του σφάλματος ανάμεσα στην προβλεφθείσα τιμή και την πραγματική τιμή.

Όροι τηλεφωνικού συστήματος

- **Γραμμή συνδρομητή**: γραμμή τηλεφώνου.
- **Κεντρικό γραφείο**: σύστημα τηλεφωνικών μεταγωγών.
- **Off-hook**: τηλέφωνο ενεργό.
- **On-hook**: τηλέφωνο ανενεργό.

Πραγματική και προσομοιωμένη γραμμή συνδρομητή

- Πραγματική γραμμή συνδρομητή:
  - 90V RMS ηχητικό σήμα,
  - συζυγή αναλογικά σήματα,
  - Προστασία από αστραπές, κτλ.
- Προσομοιωμένη γραμμή συνδρομητή:
  - Είσοδος μικροφώνου,
  - Έξοδος ηχείου,
  - διακόπτες για κουδούνισμα, off-hook, κτλ.

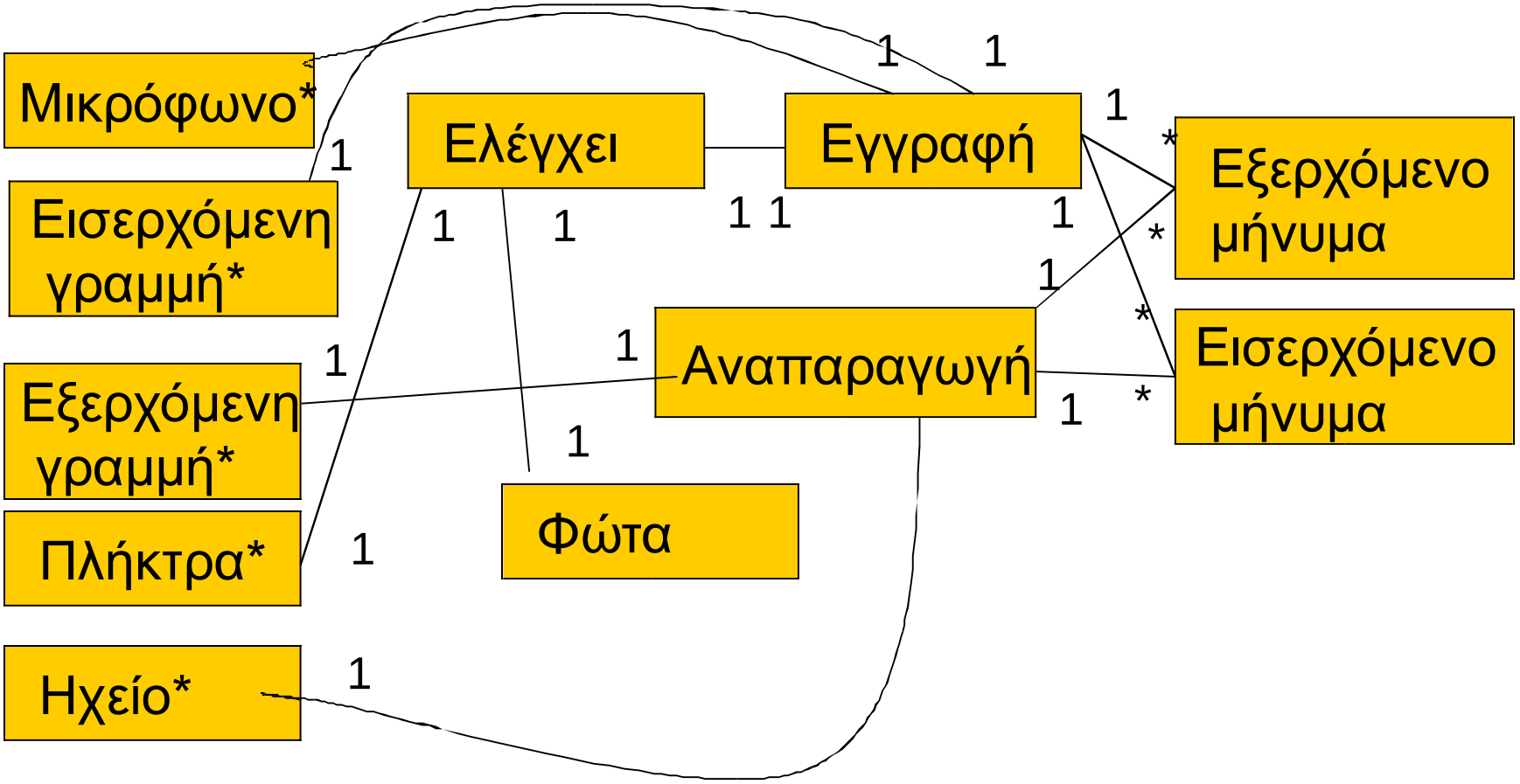
Απαιτήσεις

Inputs	Telephone: voice samples, ring. User interface: microphone, play messages button, record OGM button.
Outputs	Telephone: voice samples, on-hook/off-hook command. User interface: speaker, # messages indicator, message light.
Functions	Default mode: detects ring, signals off-hook, pays OGM, records ICM Playback: play all messages, wait 5 seconds for new playback. OGM editing: OGM up to 10 sec.
Performance	About 30 minutes voice (@ 8kHz).
Manufacturing cost	Consumer product range ($50)
Power	AC plug
Physical size/weight	Comparable to desk phone.

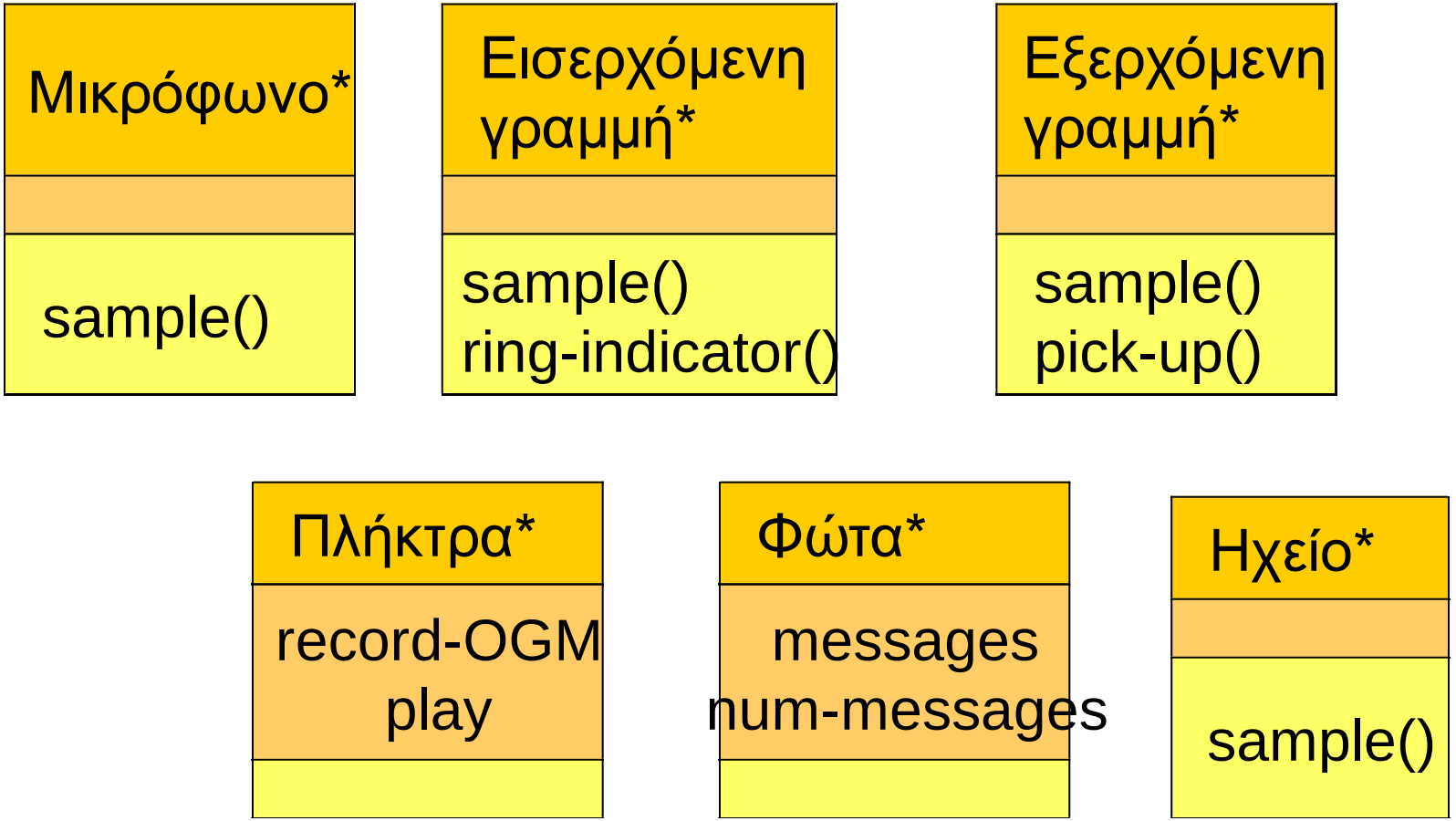
Σχόλια πάνω στην ανάλυση

- Οι DRAM απαιτήσεις επηρεάζονται από την τιμή της DRAM
  - (α) την τιμή κατά τη χρονική στιγμή στην οποία η συσκευή πηγαίνει για κατασκευή,
  - (β) την εκτιμώμενη λιανική τιμή.
- Οι λεπτομέρειες του πρωτοκόλλου διεπαφής χρήστη θα μπορούσαν να δοκιμαστούν σε ένα πρωτότυπο βασισμένο σε προσωπικό υπολογιστή.

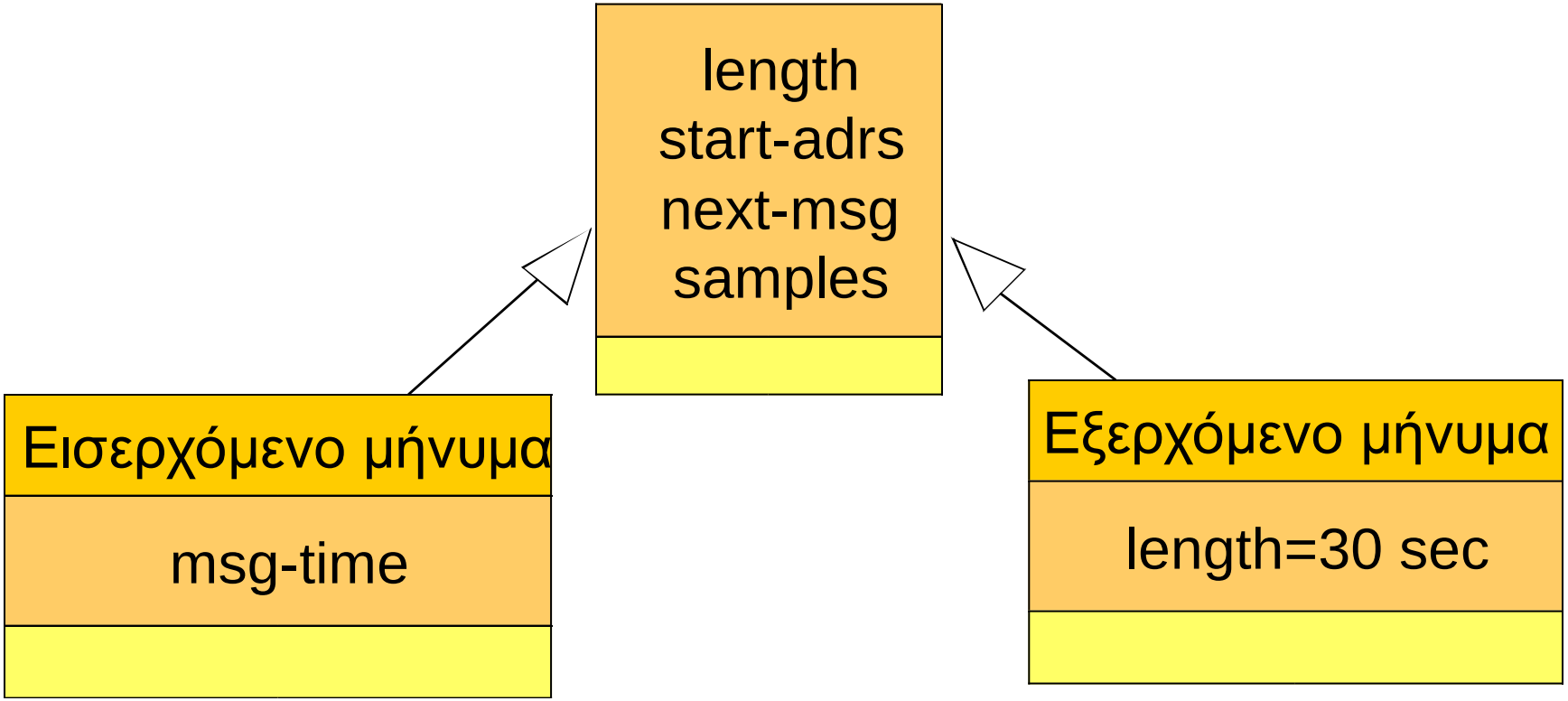
Διάγραμμα κλάσεων μηχανής απαντήσεων



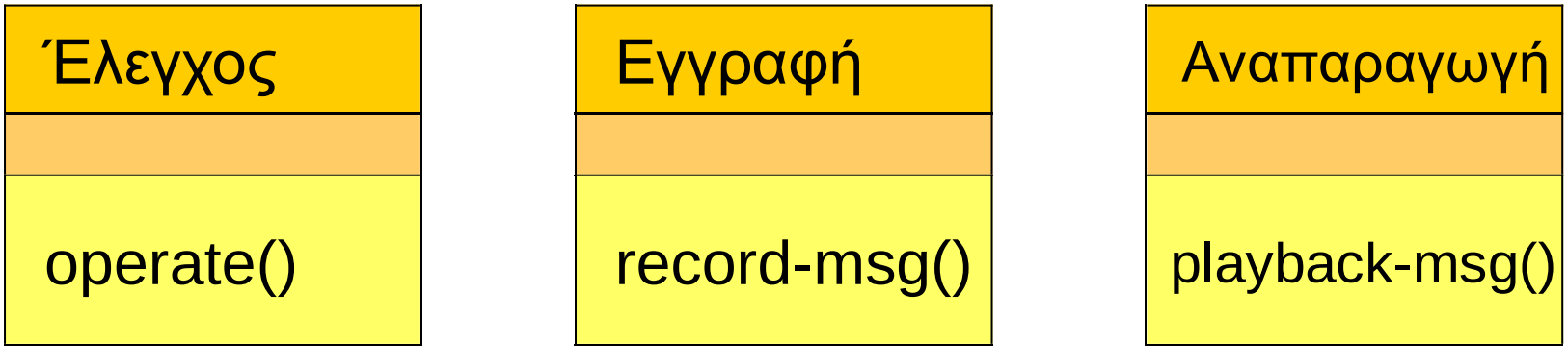
Κλάσεις της φυσικής διεπαφής



Κλάσεις μηνυμάτων



Λειτουργικές κλάσεις



Αντιδραστικα συστήματα

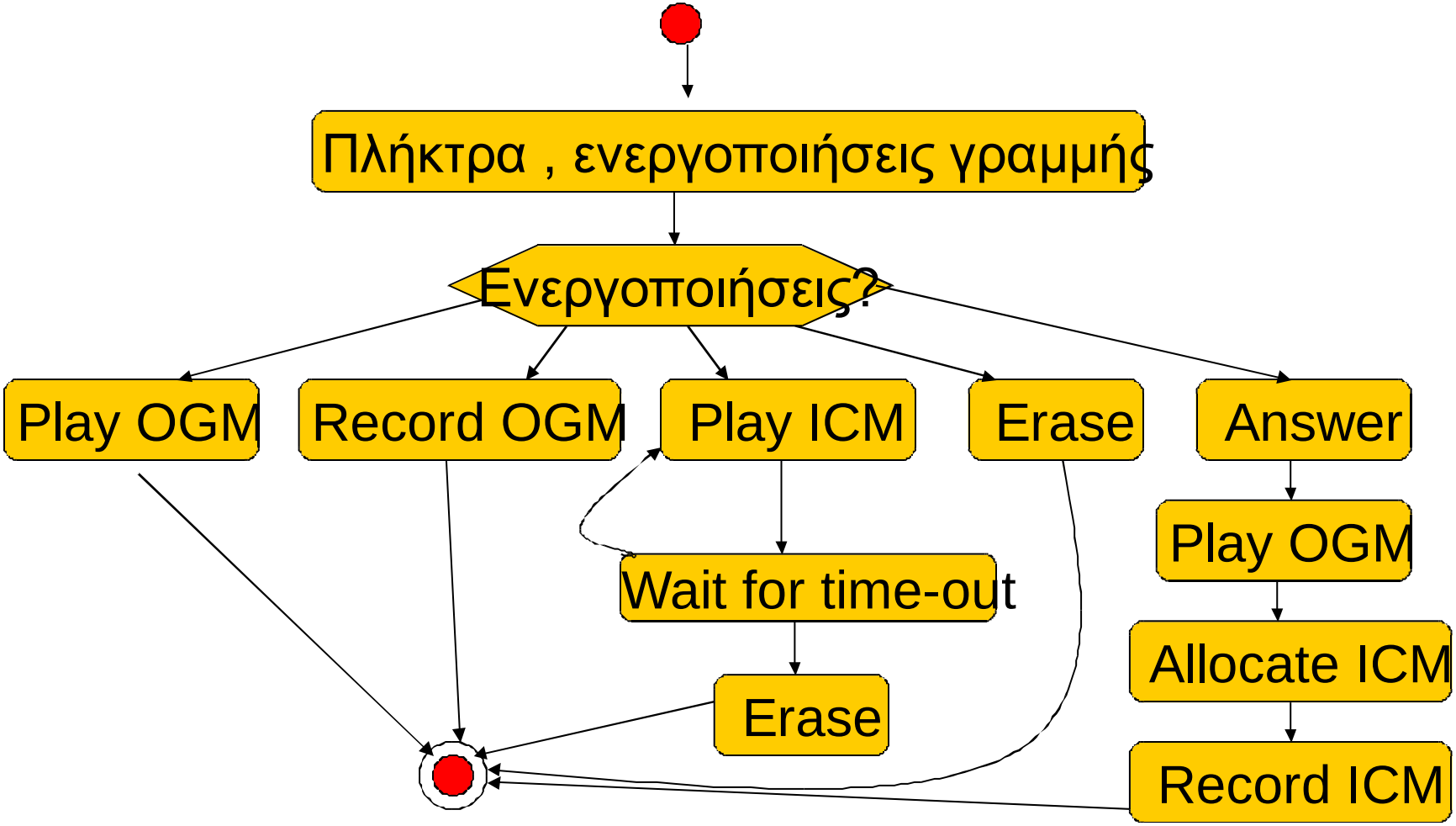
Πολλά ενσωματωμένα συστήματα κάνουν περισσότερα από μια λειτουργίες.

- Αποκρίνονται σε εξωτερικά συμβάντα.
  - Ελεγκτής κινητήρα.
  - Παρακολουθητής ζώνης ασφαλείας.
- Απαιτούν απόκριση σε πραγματικό χρόνο.
  - Αρχιτεκτονική συστήματος.
  - Υλοποίηση προγράμματος.
- Ίσως απαιτούν μια αλυσιδωτή αντίδραση μεταξύ πολλαπλών επεξεργαστών.

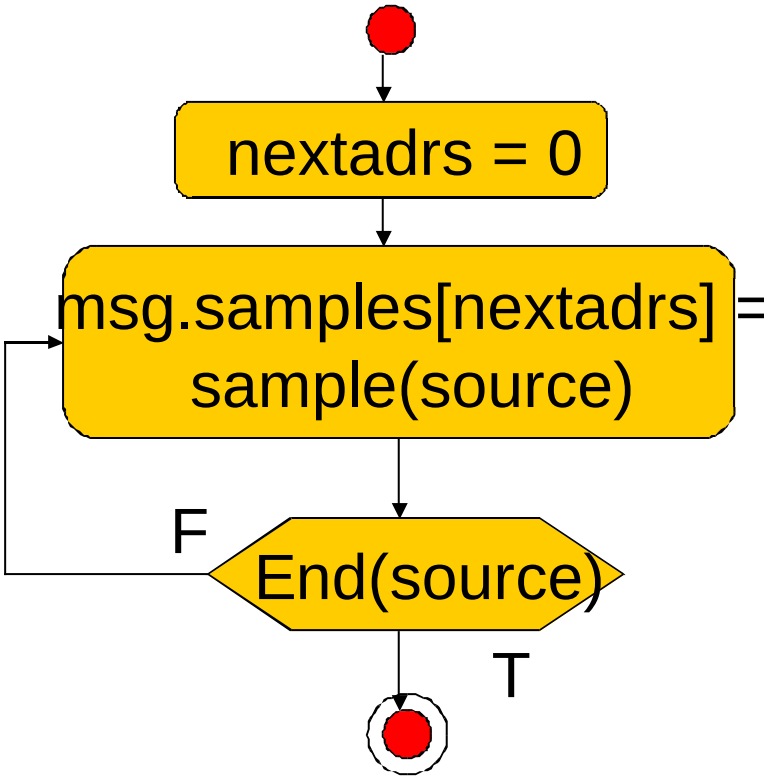
Συνιστώσες λογισμικού

- Μονάδα μπροστινού πλαισίου.
- Μονάδα ηχείου.
- Μονάδα τηλεφωνικής γραμμής.
- Μονάδες εισόδου και εξόδου τηλεφώνου.
- Μονάδα συμπίεσης.
- Μονάδα αποσυμπίεσης.

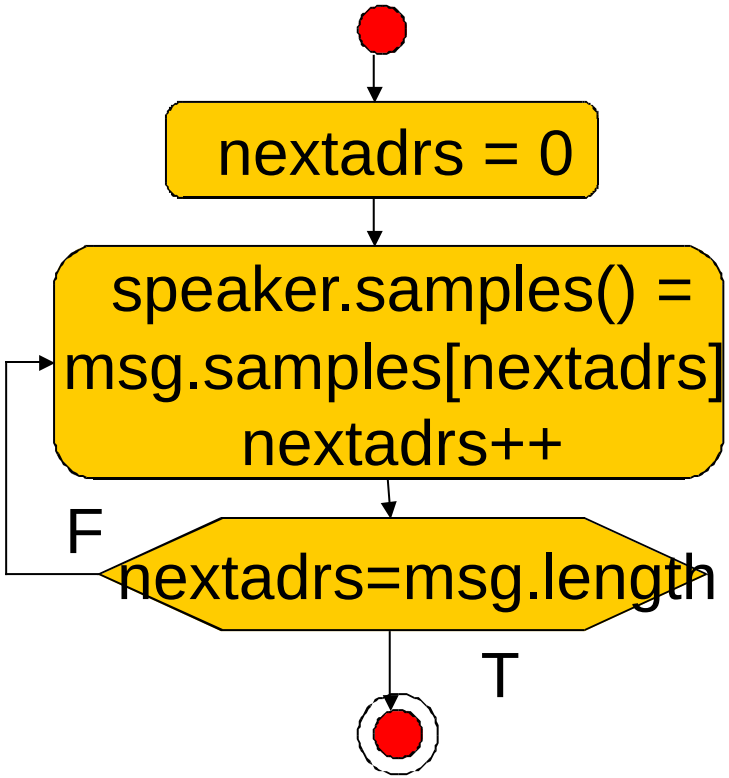
Οι έλεγχοι ενεργοποιούν συμπεριφορές



Record-msg/playback-msg συμπεριφορές



record-msg



playback-msg

Πλατφόρμα hardware

- CPU.
- Μνήμη.
- Μπροστινό πλαίσιο.
- 2 A/Ds:
  - *Γραμμή συνδρομητή, μικρόφωνο.*
- 2 D/A:
  - *Γραμμή συνδρομητή, ηχείο.*

Σχεδιασμός συνιστωσών και δοκιμή

- Πρέπει να δοκιμάζεται τόσο η επίδοση όσο και η δοκιμή.
  - Ο χρόνος συμπίεσης δε θα πρέπει να κυριαρχεί σε άλλες εργασίες.
- Έλεγχος για καταστάσεις σφαλμάτων:
  - **Υπερχείλιση μνήμης**,
  - **Προσπάθεια διαγραφής άδειων πακέτων μηνυμάτων, κτλ.**

Ολοκλήρωση συστήματος και δοκιμή

- Μπορεί να δοκιμαστεί μερική ολοκλήρωση στην πλατφόρμα του host, ολική δοκιμή απαιτεί ολοκλήρωση στην τελική πλατφόρμα *(πλατφόρμα στόχο).*
- Προσομοιωμένη τηλεφωνική γραμμή για δοκιμές:
  - Είναι νόμιμο *(αλλιώς μπορεί να προκαλέσει πρόβλημα στον πάροχο),*
  - Γίνεται ευκολότερη η δημιουργία καταστάσεων δοκιμής.

Βιβλιογραφία

Χρησιμοποιήθηκε υλικό από παρουσιάσεις των:

- *Wayne Wolf, Overheads for Computers as Components 2nd ed. ,2008 (6.2 – 6.10)*

Τέλος Ενότητας