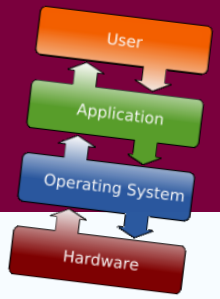


Δρομολόγηση Μονο-επεξεργαστικού Συστήματος

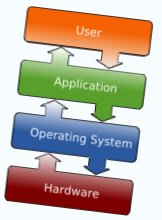
Ανδρέας Λ. Συμεωνίδης

Αν. Καθηγητής
Τμήμα Ηλεκτρολόγων Μηχ/κών
&
Μηχ/κών Υπολογιστών, Α.Π.Θ.

Email:asymeon@eng.auth.gr

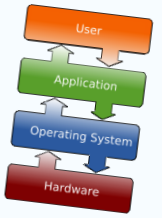
Στόχοι της Δ-9

- Να ορίσει τα κριτήρια αποτίμησης της απόδοσης και βελτιστοποίησης
- Να παρουσιάσει τους τύπους και τις πολιτικές δρομολόγησης του επεξεργαστή
- Να αναλύσει τους βασικότερους αλγορίθμους δρομολόγησης:
  - First Come First Served (FCFS)
  - Shortest Job First (SJF)
  - Shortest Remaining Time First (SRTF)
  - Round Robin (RR)
- Να ορίσει τον χρόνο καταιγισμού
- Να συζητήσει δυο ειδικότερες περιπτώσεις δρομολόγησης:
  - Με ουρές πολλαπλών επιπέδων
  - Με ανάδραση



Κριτήρια αποτίμησης της απόδοσης

- **Δικαιοσύνη (Fairness):** συχνή χρήση της CPU από κάθε διεργασία
- **Χρησιμοποίηση (Utilization)**: το ποσοστό του χρόνου κατά το οποίο μια συσκευή χρησιμοποιείται (χρόνος χρήσης / συνολικός χρόνος)
- **Ρυθμο-απόδοση (Throughput)**: ο αριθμός των εργασιών που ολοκληρώνονται σε μια χρονική περίοδο (εργασίες / sec)
- **Χρόνος επιστροφής (Turnaround time)**: ο χρόνος εκτέλεσης μιας διεργασίας – από την αρχή μέχρι την ολοκλήρωση.
- **Χρόνος εξυπηρέτησης (Service time)**: ο χρόνος που απαιτείται από μια συσκευή για να διαχειριστεί μια απαίτηση

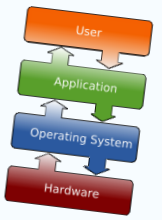


Κριτήρια αποτίμησης της απόδοσης (συν.)

- **Χρόνος αναμονής (Queuing time)**: Ο χρόνος σε μια ουρά αναμονής ΕΤΟΙΜΩΝ διεργασιών (seconds)
- **Χρόνος παραμονής (Residence time)**: ο χρόνος που ξοδεύεται από μια απαίτηση σε μια συσκευή

Residence time = Service time + Queuing time

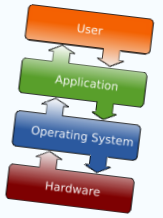
- **Χρόνος απόκρισης (Response time)**: ο χρόνος από τότε που μια απαίτηση υποβάλλεται μέχρι τη στιγμή που παράγεται η πρώτη απόκριση, όχι την έξοδο
- **Εναλλαγές πλαισίων (Context switches)**: χρόνος που ξοδεύεται και διατηρεί την CPU άεργη
- **Πολυπλοκότητα του αλγορίθμου δρομολόγησης**: χρόνος που απαιτείται για την επιλογή της επόμενης διεργασίας



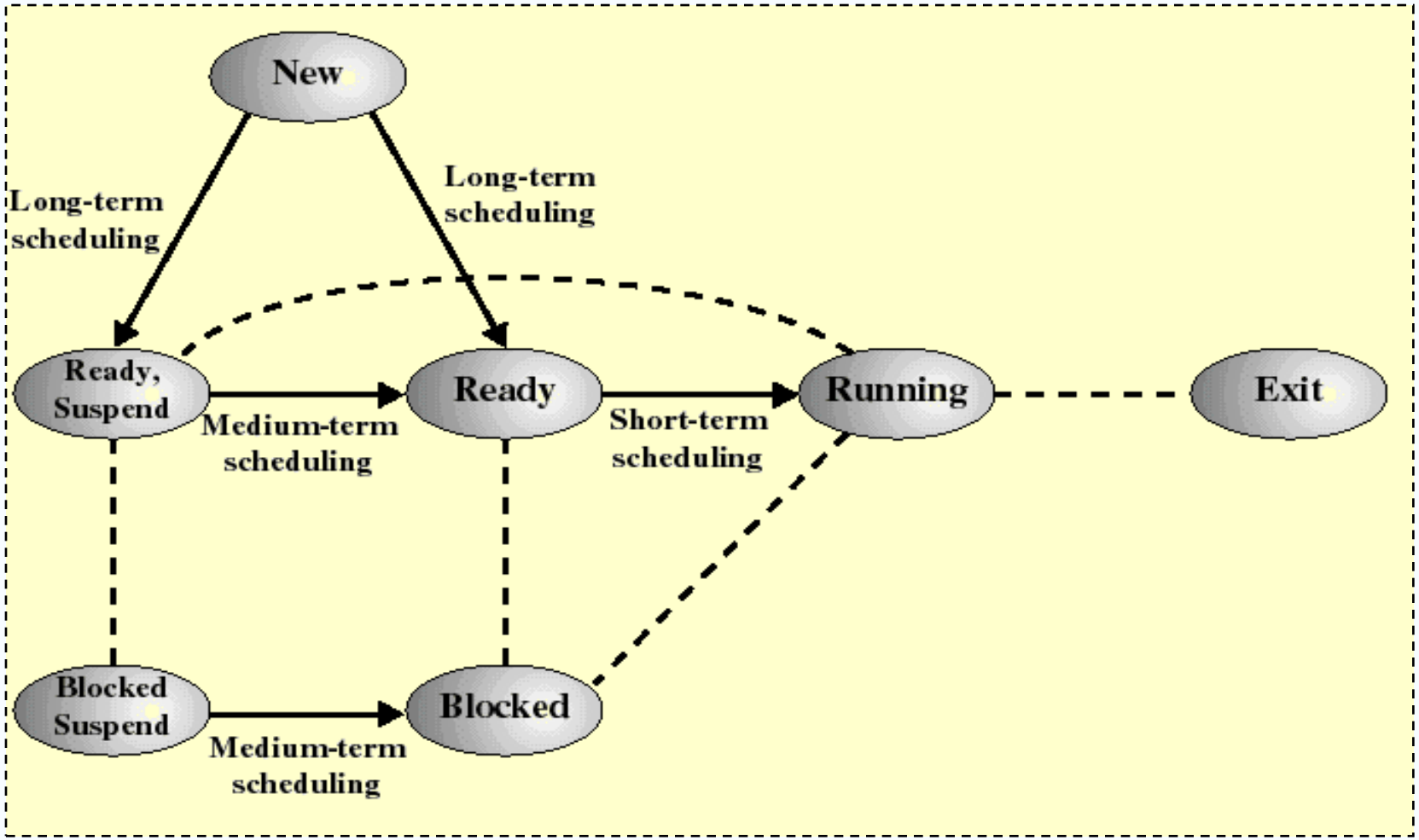
Κριτήρια βελτιστοποίησης

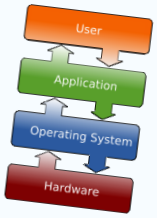
- **Μεγιστοποίηση:**
  - **Χρήσης της CPU**
  - **Ρυθμο-απόδοσης**
- **Ελαχιστοποίηση των χρόνων:**
  - **Επιστροφής**
  - **Αναμονής**
  - **Απόκρισης**

Τα κριτήρια αυτά είναι συχνά αλληλοσυγκρουόμενα!!!



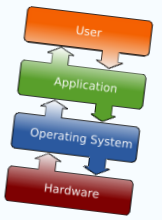
Τύποι δρομολόγησης του επεξεργαστή





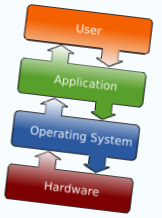
Μακροπρόθεσμη δρομολόγηση

- Καθορίζει ποια προγράμματα θα γίνουν αποδεκτά για επεξεργασία από το σύστημα
- Ελέγχει τον βαθμό πολυπρογραμματισμού του συστήματος
- Αν γίνουν αποδεκτές περισσότερες διεργασίες:
  - Λιγότερες διεργασίες θα ανασταλούν, θα υπάρξει καλύτερη χρήση της CPU
  - Κάθε διεργασία θα λάβει λιγότερο ποσοστό χρόνου της CPU
- Ο μακροπρόθεσμος δρομολογητής προσπαθεί να διατηρήσει μια ισορροπία ανάμεσα στις **προοριζόμενες για τον επεξεργαστή διεργασίες** (processor-bound) και στις **προοριζόμενες για Ε/Ε διεργασίες** (I/O-bound)



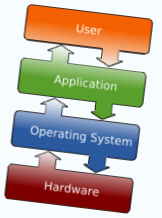
Μεσοπρόθεσμη δρομολόγηση

- Ποια διεργασία θα προστεθεί στις διεργασίες που βρίσκονται ολόκληρες ή εν μέρει στην κύρια μνήμη
- Οι αποφάσεις για εναλλαγή των διεργασιών βασίζονται στην ανάγκη της διαχείρισης του πολύ-προγραμματισμού
- Γίνεται από το λογισμικό της διαχείρισης μνήμης



Βραχυπρόθεσμη δρομολόγηση

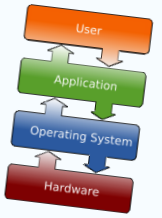
- Αποφασίζει ποια διεργασία πρόκειται να εκτελεστεί ως επόμενη
- Αναφέρεται συνήθως ως **CPU scheduling** ή **διεκπεραίωση** (ο βραχυπρόθεσμος δρομολογητής λέγεται διεκπεραιωτής – dispatcher).
- Ενεργοποιείται από ένα γεγονός που μπορεί να οδηγήσει στην επιλογή μιας άλλης διεργασίας για εκτέλεση:
  - διακοπές ρολογιού
  - διακοπές Ε/Ε
  - κλήσεις του ΛΣ
  - σήματα



Κριτήρια βραχυπρόθεσμης δρομολόγησης

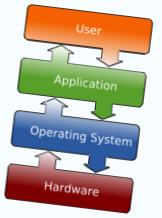
- Κριτήρια προσανατολισμένα στο χρήστη:
  - **Χρόνος απόκρισης**
  - **Χρόνος επιστροφής**

- Κριτήρια προσανατολισμένα στο σύστημα:
  - **Χρήση επεξεργαστή**
  - **Δικαιοσύνη**
  - **Ρυθμοαπόδοση**

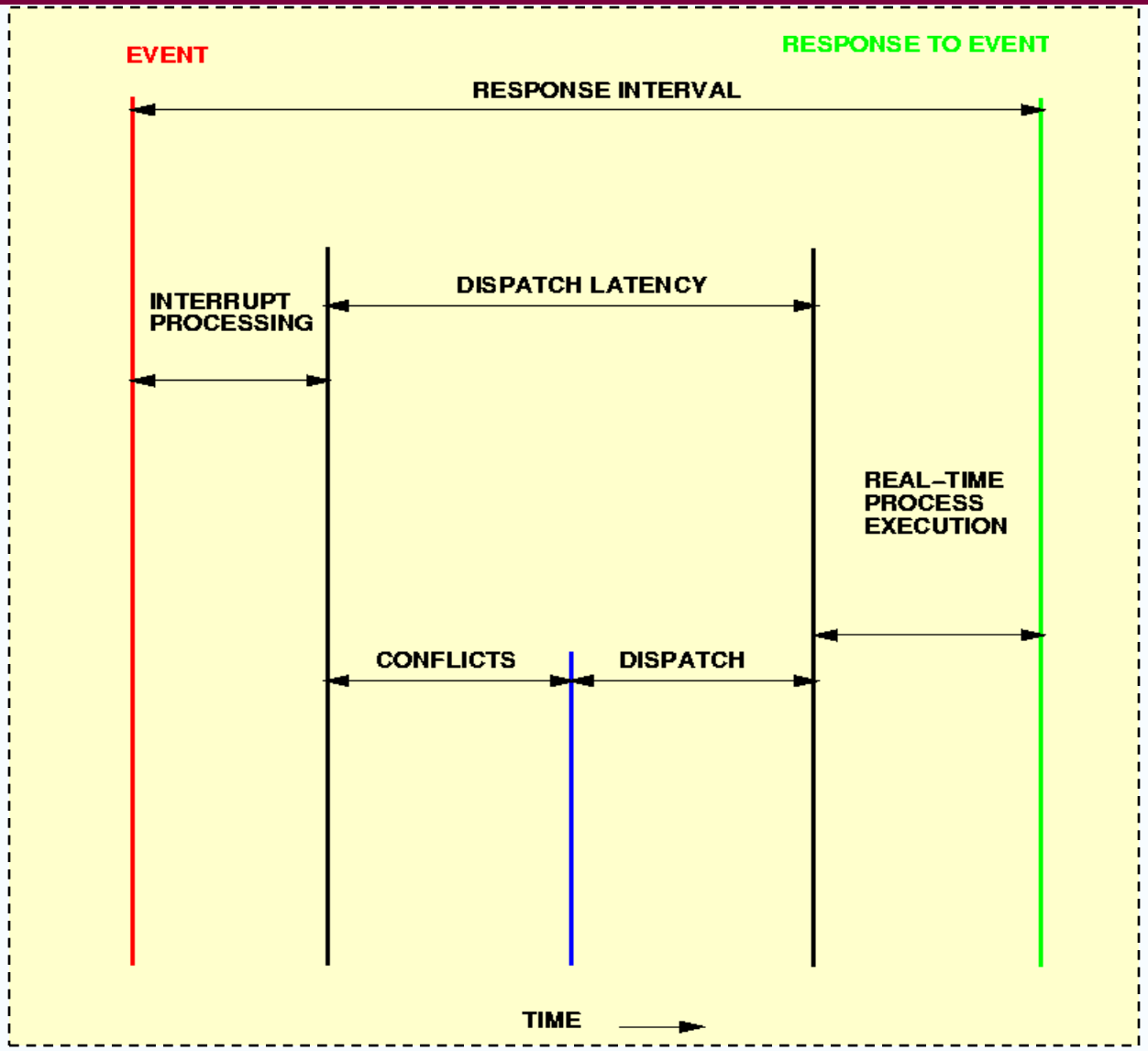


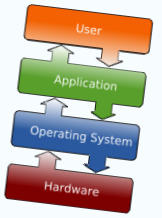
Διεκπεραιωτής (Dispatcher)

- Ο έλεγχος της CPU δίνεται στη διεργασία που έχει επιλεγεί από τον βραχυχρόνιο δρομολογητή.
- Η εκχώρηση αυτή περιλαμβάνει :
  - Εναλλαγή πλαισίου
  - Εναλλαγή σε κατάσταση χρήστη
  - Άλμα στην κατάλληλη διεύθυνση του προγράμματος του χρήστη για την εκκίνηση του προγράμματος
- **Λανθάνουσα κατάσταση διεκπεραιωτή (Dispatch latency):** χρόνος που χρειάζεται ο διεκπεραιωτής για να σταματήσει μια διεργασία και να αρχίσει την εκτέλεση μιας άλλης



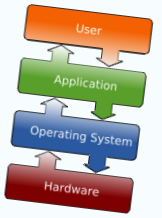
Λανθάνουσα κατάσταση διεκπεραιωτή



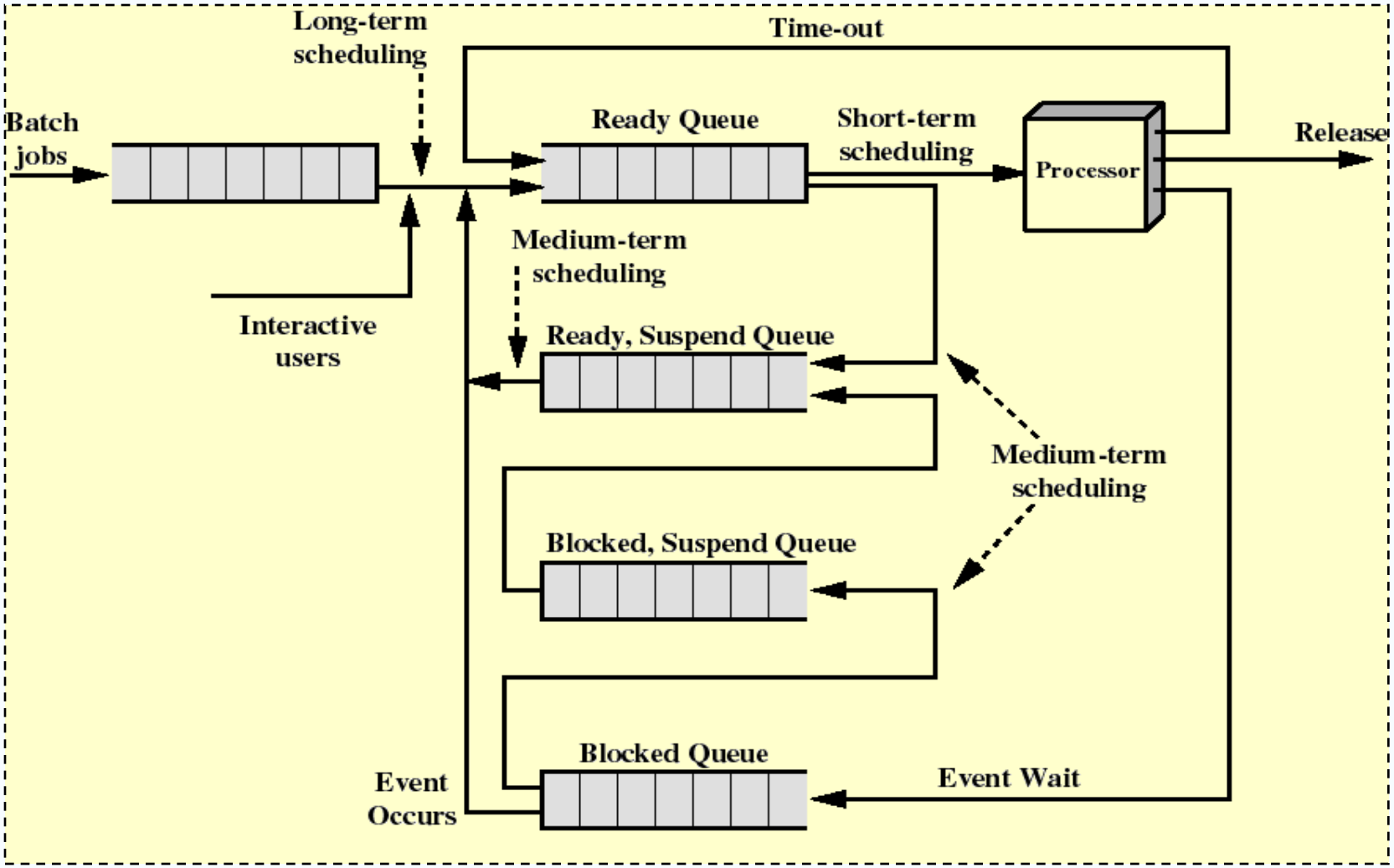


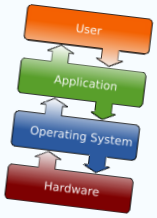
Προτεραιότητες

- Εφαρμόζονται με την ύπαρξη πολλαπλών ουρών έτοιμων διεργασιών, όπου κάθε ουρά αντιπροσωπεύει ένα επίπεδο προτεραιότητας
- Ο δρομολογητής θα επιλέγει πάντοτε μια διεργασία υψηλότερης προτεραιότητας έναντι μιας με μικρότερη προτεραιότητα
- Η χαμηλή προτεραιότητα μπορεί να υποφέρει από παρατεταμένη στέρηση
- Επιτρέπεται σε μια διεργασία να αλλάζει την προτεραιότητά της, ανάλογα με το διάστημα που βρίσκεται στο σύστημα ή με το ιστορικό εκτέλεσής της



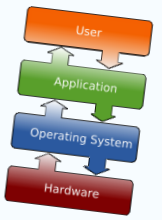
Διάγραμμα ουρών κατά τη δρομολόγηση



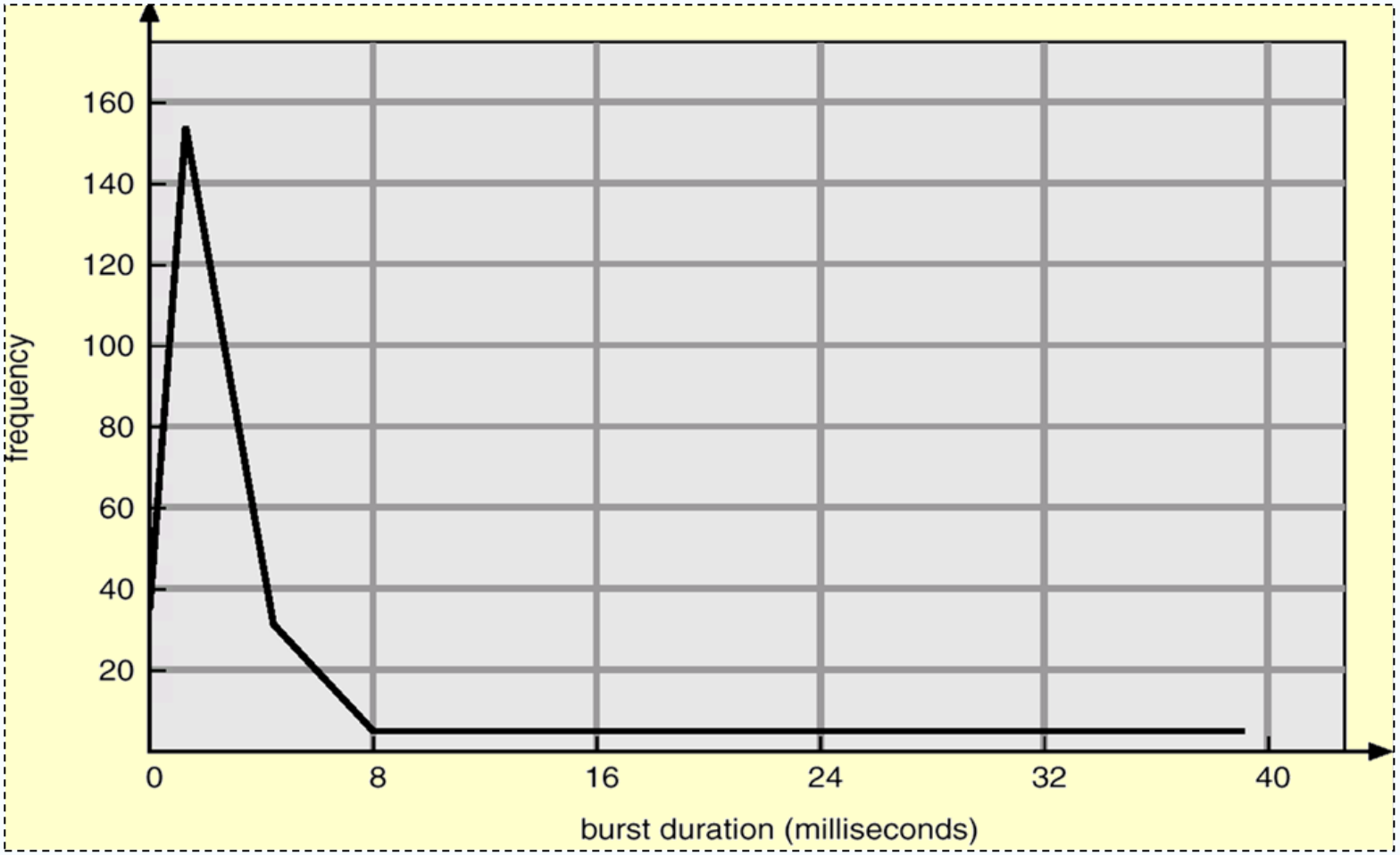


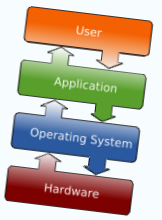
Ο κύκλος καταιγισμού ΚΜΕ – Ε/Ε

- **Κύκλος καταιγισμού (burst cycle) ΚΜΕ – Ε/Ε**: χαρακτηρίζει την εκτέλεση της διεργασίας που εναλλάσσεται μεταξύ των δραστηριοτήτων της ΚΜΕ και των Ε/Ε.
  - Οι χρόνοι της ΚΜΕ είναι γενικά πολύ μικρότεροι των χρόνων για Ε/Ε
- Κάθε κύκλος αποτελείται από ένα CPU burst διάρκειας συνήθως **μερικών *msecs***, ακολουθούμενο από ένα (συνήθως μεγαλύτερο) Ε/Ε burst
- Μια διεργασία τερματίζεται κατά τη διάρκεια ενός CPU burst
- Οι προοριζόμενες για τον επεξεργαστή διεργασίες έχουν μεγαλύτερα CPU bursts από εκείνα όσων προορίζονται για Ε/Ε



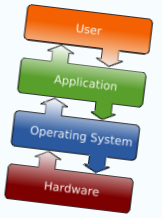
Ιστόγραμμα συχνοτήτων των χρόνων καταιγισμού της CPU





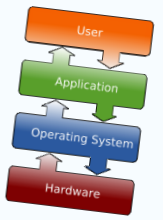
Ο δρομολογητής της CPU

- Ο δρομολογητής της CPU επιλέγει από τις έτοιμες διεργασίες που βρίσκονται στην κύρια μνήμη
- Είναι συστατικό του ΛΣ
  - Είναι από μόνος του μια διεργασία
  - Χρησιμοποιεί τους πόρους του συστήματος
    - CPU
    - Μνήμη
  - Δεν θα πρέπει να καταναλώνει σημαντικό χρόνο της CPU, διαφορετικά η επιβράδυνση θα είναι μεγάλη



Παράγοντες που επιδρούν στη δρομολόγηση

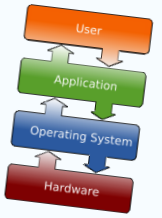
- Αν η διεργασία είναι CPU-bound ή I/O-bound
- Αν η διεργασία είναι αλληλεπιδραστική ή batch
- Προτεραιότητα διεργασιών
- Χρόνος που έχει εκτελεστεί
- Χρόνος που απαιτείται για να ολοκληρωθεί
- Συχνότητα προεκχώρησης
- Συχνότητα εμφάνισης σφαλμάτων σελίδας



Ο νόμος της διατήρησης του Kleinrock

- Ανεξάρτητα από τον αλγόριθμο δρομολόγησης που χρησιμοποιείται, δεν είναι δυνατόν να ωφεληθεί μια κατηγορία διεργασιών χωρίς να υπάρξει επίπτωση στις υπόλοιπες.

- **Παράδειγμα:** μια μικρή βελτίωση για τις μικρές διεργασίες (π.χ. στον χρόνο αναμονής) προκαλεί δυσανάλογη επιβράδυνση στις μεγάλες διεργασίες.



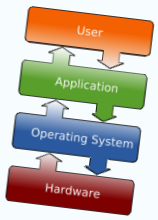
Πολυπλοκότητα

- Σε έναν επεξεργαστή δρομολογούνται *n* διεργασίες. Πόσοι διαφορετικοί συνδυασμοί δρομολόγησης υπάρχουν (το αποτέλεσμα να εκφραστεί σαν συνάρτηση του *n*);

- **Απάντηση**

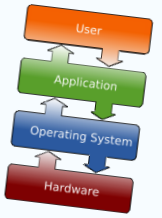
$$n!=n*(n-1)*(n-2)*(n-3)*\ldots*3*2*1$$

Οι αλγόριθμοι με πολυπλοκότητα *O(n!)* είναι μη αποτελεσματικοί!!!



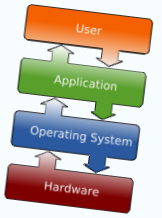
Αλγόριθμοι Δρομολόγησης

- Καθορίζουν τον τρόπο επιλογής της επόμενης προς εκτέλεση διεργασίας, καθώς και τη σειρά με την οποία εκτελούνται οι διεργασίες
- Ορίζουν τις ενέργειες εκ μέρους του δρομολογητή
- Τύποι αλγορίθμων δρομολόγησης
  - Προεκχωρούμενοι
  - Μη προεκχωρούμενοι
- Παραδείγματα αλγορίθμων δρομολόγησης
  - FCFS, SJF, SRTF, priority-based, round robin
  - multilevel, multilevel feedback



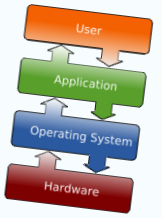
Μη προεκχωρούμενη δρομολόγηση

- Μια διεργασία παραμένει στη CPU μέχρι να απελευθερώσει από μόνη της τη CPU
  - Μεγάλοι χρόνοι αναμονής και απόκρισης
  - Ενδεχόμενο παρατεταμένης στέρησης
- Απλή και εύκολη στην υλοποίηση
- Δεν είναι κατάλληλη για συστήματα πολλών χρηστών



Προεκχωρούμενη δρομολόγηση

- Η εκτέλεση μιας διεργασίας μπορεί να διακόπτεται από το ΛΣ οποιαδήποτε στιγμή. Πιθανές αιτίες:
  - Η άφιξη μιας νέας διεργασίας με υψηλότερη προτεραιότητα
  - Η πρόκληση μιας διακοπής
  - Η αλλαγή της κατάστασης μιας διεργασίας
  - Η υπέρβαση ενός χρονικού ορίου
- Εμποδίζεται η μονοπώληση της χρήσης της CPU από μία διεργασία
- Μπορεί να οδηγήσει σε συνθήκες ανταγωνισμού
  - Επιλύονται χρησιμοποιώντας συγχρονισμό μεταξύ των διεργασιών



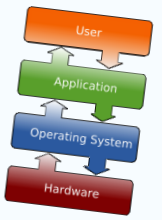
Παραδείγματα αλγορίθμων δρομολόγησης

I. **FCFS** (First Come First Served) – πρώτη ήλθε πρώτη εξυπηρετήθηκε

II. **SJF** ή **SPN** (Shortest Job First ή Shortest Process (Job) Next) – η συντομότερη διεργασία πρώτη

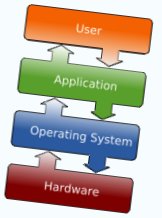
III. **Shortest-Remaining-Time-First** (SRTF) - η διεργασία με το μικρότερο εναπομείναντα χρόνο πρώτη

IV. **RR** (Round Robin) – εξυπηρέτηση εκ περιτροπής



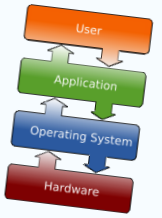
I. First-Come, First-Served (FCFS)

- Βασικές αρχές
  - Οι διεργασίες εξυπηρετούνται με τη σειρά που φθάνουν
  - Ακόμη και αν φθάνουν την ίδια χρονική στιγμή, η σειρά άφιξης είναι διακριτή, αλλιώς η διεργασία που θα εξυπηρετηθεί κατά προτεραιότητα επιλέγεται τυχαία
  - Κατάσταση απόφασης: χωρίς προεκχώρηση
  - Η διεργασία εκτελείται μέχρι να ανασταλεί από μόνη της
- Είναι πολύ απλός αλγόριθμος και υλοποιείται εύκολα αλλά είναι ακατάλληλος για συστήματα πολλών χρηστών.



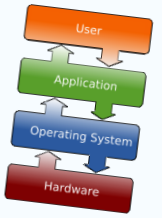
FCFS - ιδιότητες

- Χρησιμοποιεί μια ουρά FIFO
- Η επιλογή της επόμενης διεργασίας είναι ταχύτατη και ανεξάρτητη από το πλήθος των διεργασιών στην ουρά των έτοιμων διεργασιών
- Συχνά προκύπτουν μεγάλοι χρόνοι αναμονής και απόκρισης



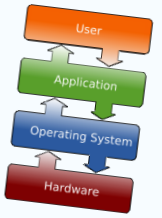
FCFS - παράδειγμα

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1



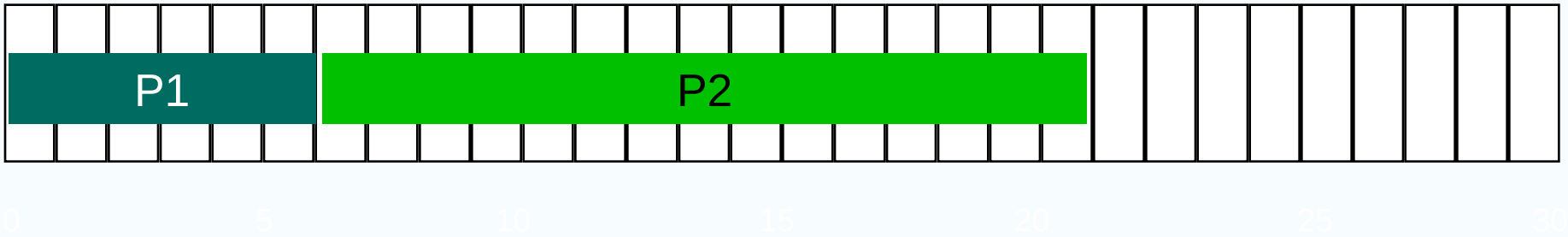
FCFS - παράδειγμα

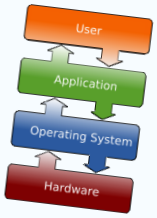
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1



FCFS - παράδειγμα

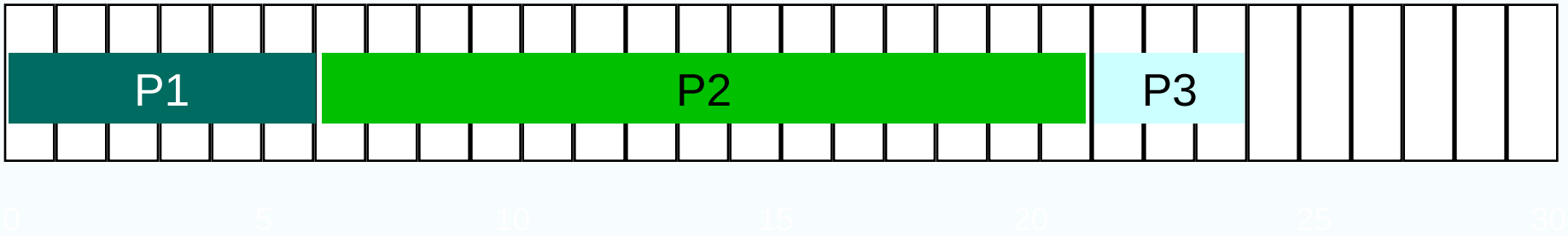
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1

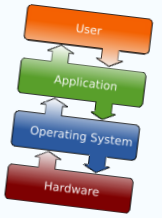




FCFS - παράδειγμα

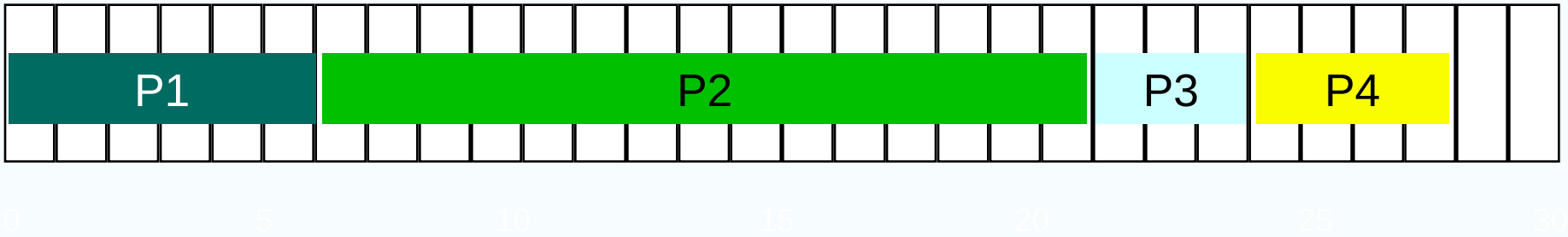
Process #	*Arrival Time*	*Burst Length*	*Priority*
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1

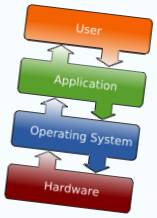




FCFS - παράδειγμα

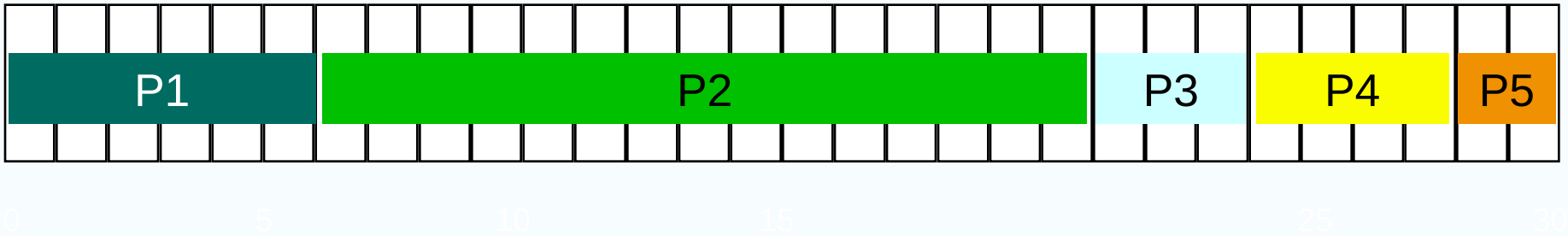
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1

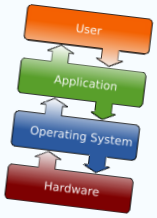




FCFS - παράδειγμα

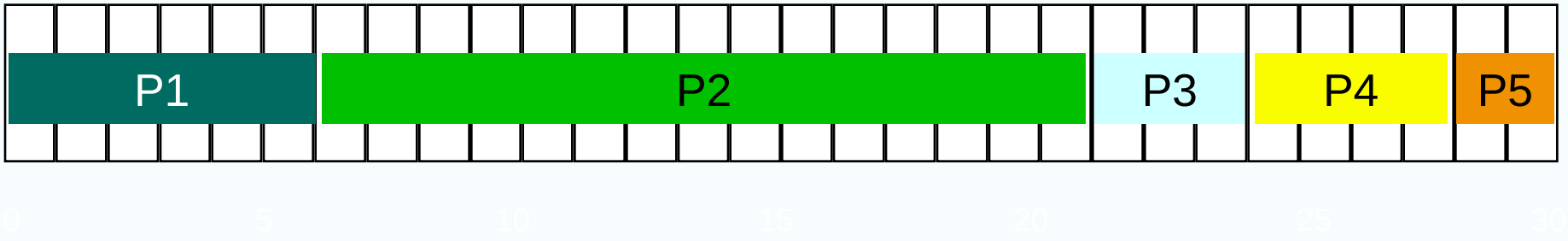
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1



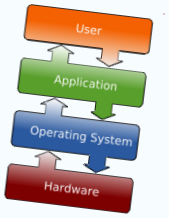


FCFS - παράδειγμα

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1

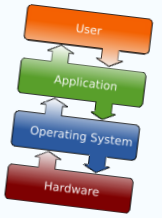


Process #	Waiting Time	Response Time	Turnaround Time	#of Context Switches
P1	0	0	6	1
P2	6	6	21	1
P3	21	21	24	1
P4	24	24	28	1
P5	28	28	30	1
Average	79/5 = 15.8	79/5 = 15.8	21.8	1



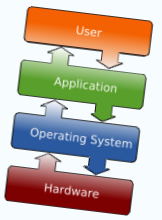
FCFS - μειονεκτήματα

- Μια διεργασία που δεν χρησιμοποιεί Ε/Ε θα μονοπωλεί τη χρήση του επεξεργαστή
- Ευνοούνται οι προοριζόμενες για τη CPU διεργασίες.
- Οι προοριζόμενες για Ε/Ε διεργασίες θα περιμένουν μέχρι να ολοκληρωθούν οι προοριζόμενες για τη CPU διεργασίες. Θα περιμένουν ακόμη και αν οι Ε/Ε λειτουργίες τους έχουν ολοκληρωθεί
- Υπάρχουν μεγάλες διακυμάνσεις στον μέσο χρόνο επιστροφής
- Είναι ακατάλληλος αλγόριθμος για αλληλεπιδραστικά συστήματα



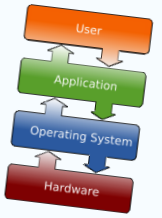
II. Shortest-Job-First (SJF)

- Η διεργασία δηλώνει το χρόνο καταιγισμού της στην CPU
- Δύο σχήματα:
  - **Non-Preemptive** – αν εκχωρηθεί η CPU, η διεργασία δεν προεκχωρείται μέχρι να ολοκληρωθεί ο καταιγισμός της.
  - **Preemptive** – αν υπάρξει μια νέα διεργασία με χρόνο καταιγισμού της CPU μικρότερο από τον εναπομένοντα χρόνο της τρέχουσας διεργασίας, τότε την αντικαθιστά. Η περίπτωση με προεκχώρηση είναι γνωστή και ως ο αλγόριθμος: Η διεργασία με το μικρότερο εναπομείναντα χρόνο πρώτη (Shortest-Remaining-Time-First (SRTF).
- Είναι η βέλτιστη λύση: δίνει τον ελάχιστο μέσο χρόνο αναμονής για ένα δεδομένο σύνολο διεργασιών.



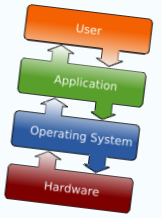
SJF

- Βασικές αρχές:
  - Επιλέγεται η διεργασία με το μικρότερο χρόνο καταιγισμού στη CPU
  - Ενσωματώνει αναμφίβολα προτεραιότητες : οι συντομότερες διεργασίες έχουν δεδομένη προτεραιότητα
  - Αποτελεί μια προφανή βελτίωση του αλγορίθμου FCFS
  - Απαιτείται ο υπολογισμός του χρόνου καταιγισμού (CPU burst time) για κάθε διεργασία



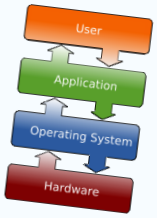
SJF – ιδιότητες

- Δίνει πολύ καλύτερο μέσο χρόνο αναμονής σε σχέση με τον αλγόριθμο FCFS
- Χωρίς προεκχώρηση, ωστόσο, δεν είναι κατάλληλος σε ένα περιβάλλον καταμερισμού χρόνου
- Απαιτείται η γνώση των χρόνων καταιγισμού στη CPU που γενικά είναι δύσκολο και συνήθως ακατόρθωτο
- Η επιλογή είναι περισσότερο σύνθετη από αυτήν του FCFS
- Είναι πιθανή η παρατεταμένη στέρηση
  - Αν νέες μικρής διάρκειας διεργασίας φθάνουν στο σύστημα οι προγενέστερες, μεγάλης διάρκειας διεργασίες, δεν θα εξυπηρετηθούν ποτέ



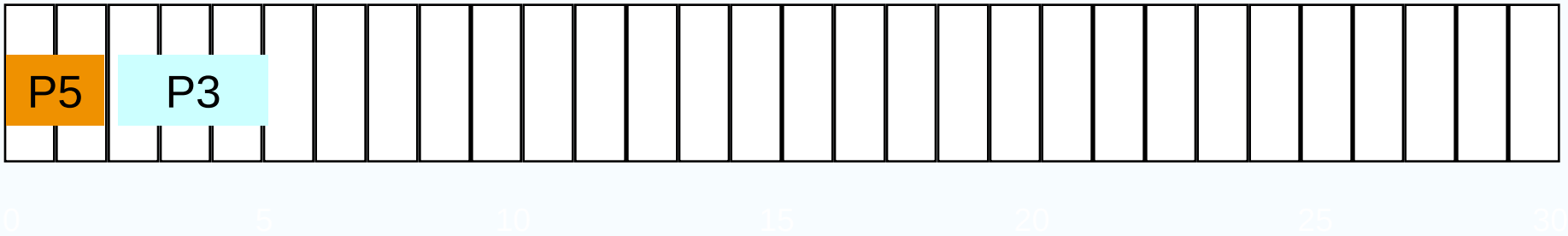
SJF – παράδειγμα-1

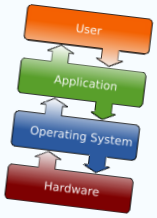
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1



SJF – παράδειγμα-1 (συν.)

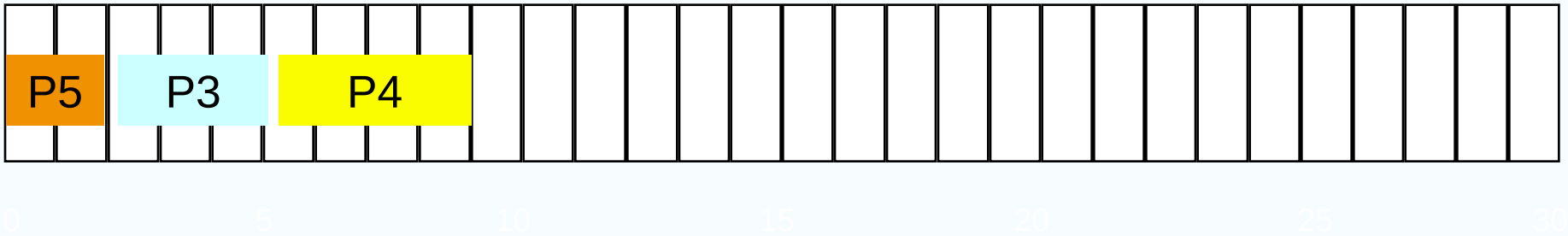
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1

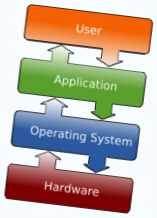




SJF – παράδειγμα-1 (συν.)

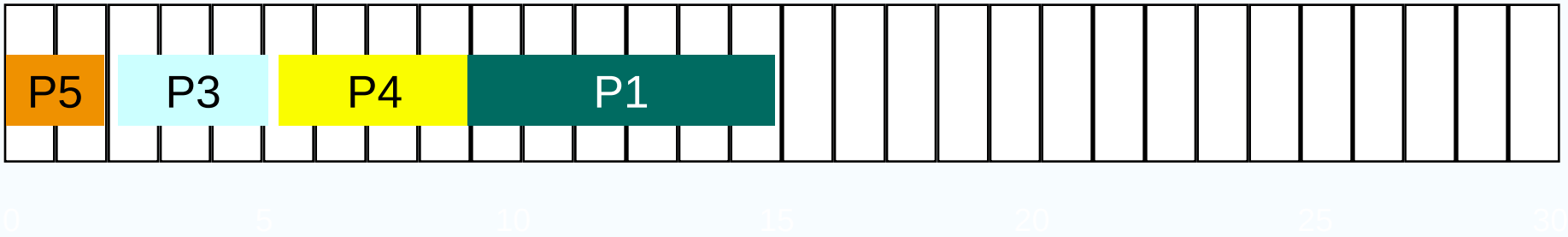
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1

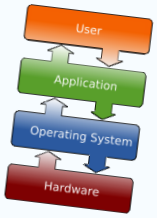




SJF – παράδειγμα-1 (συν.)

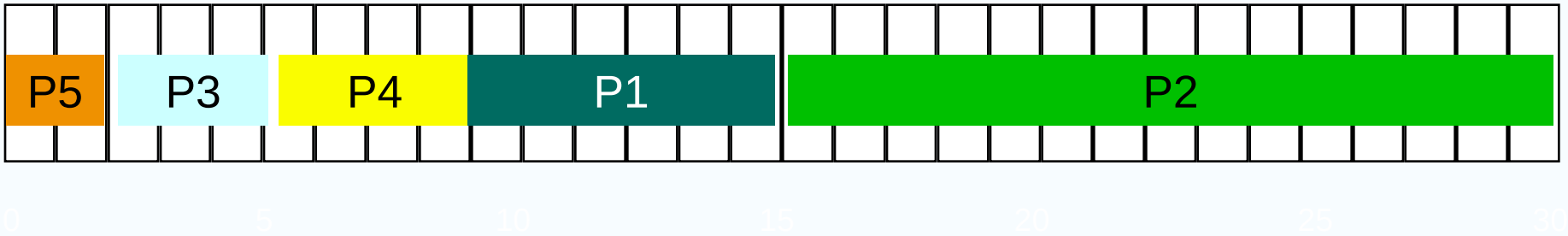
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1

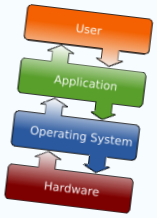




SJF – παράδειγμα-1 (συν.)

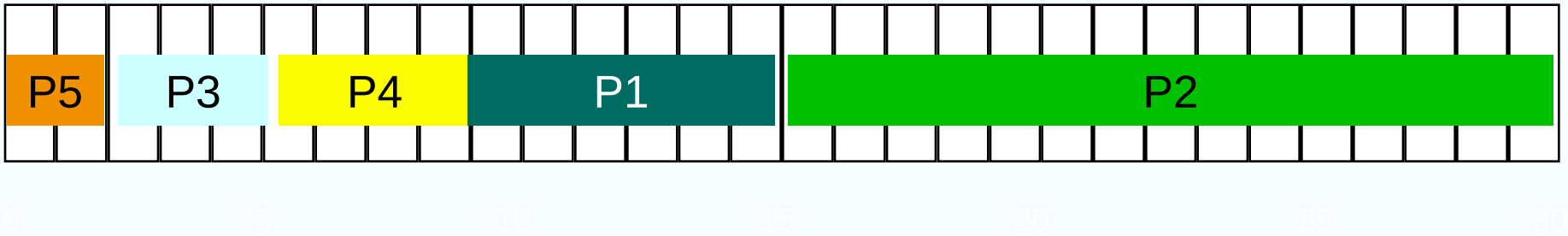
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1



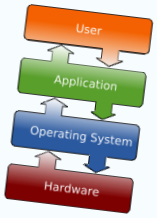


SJF – παράδειγμα-1 (συν.)

Process #	*Arrival Time*	*Burst Length*	*Priority*
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1



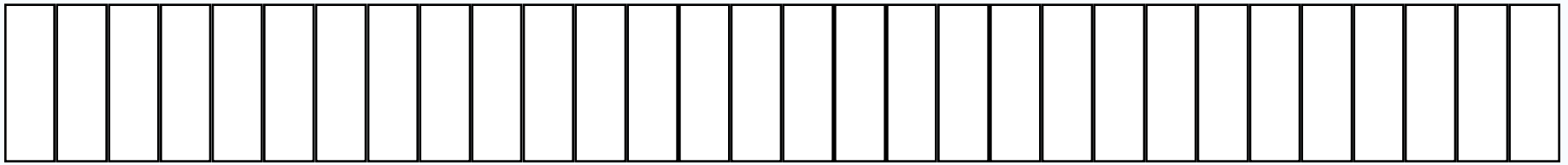
Process #	*Waiting Time*	*Response Time*	*Turnaround Time*	*#of Context Switches*
P1	9	9	15	1
P2	15	15	30	1
P3	2	2	6	1
P4	5	5	9	1
P5	0	0	2	1
Average	31/5 = 6.2	31/5 = 6.2	62/5 = 12.4	1

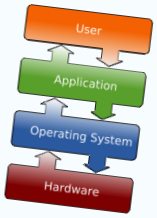


SJF – παράδειγμα - 2

Process #	*Arrival Time*	*Burst Length*	*Priority*
P1	0	6	1
P2	3	15	1
P3	5	3	1
P4	8	4	1
P5	14	2	1

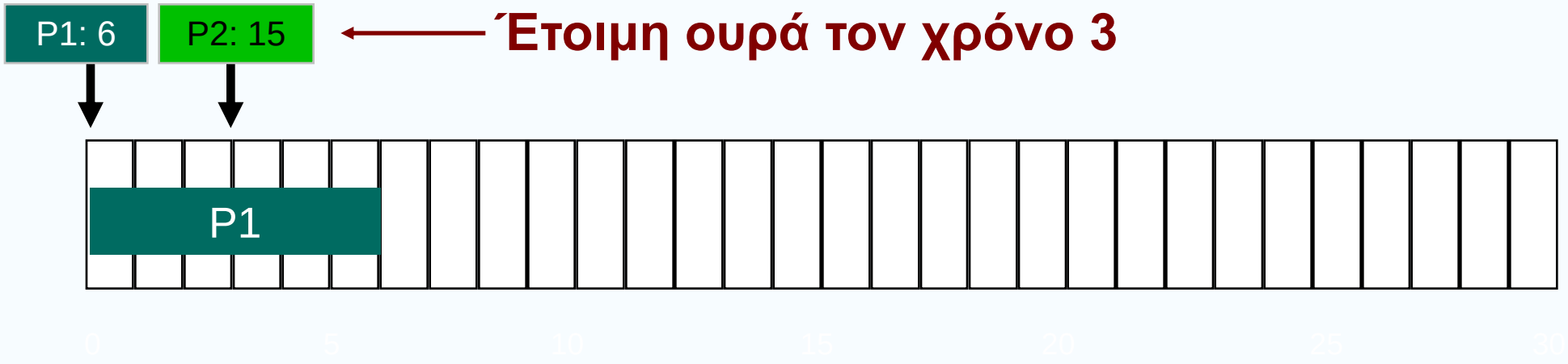
P1: 6 ← **Έτοιμη ουρά τον χρόνο 3**

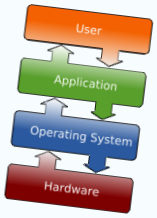




SJF – παράδειγμα - 2 (συν.)

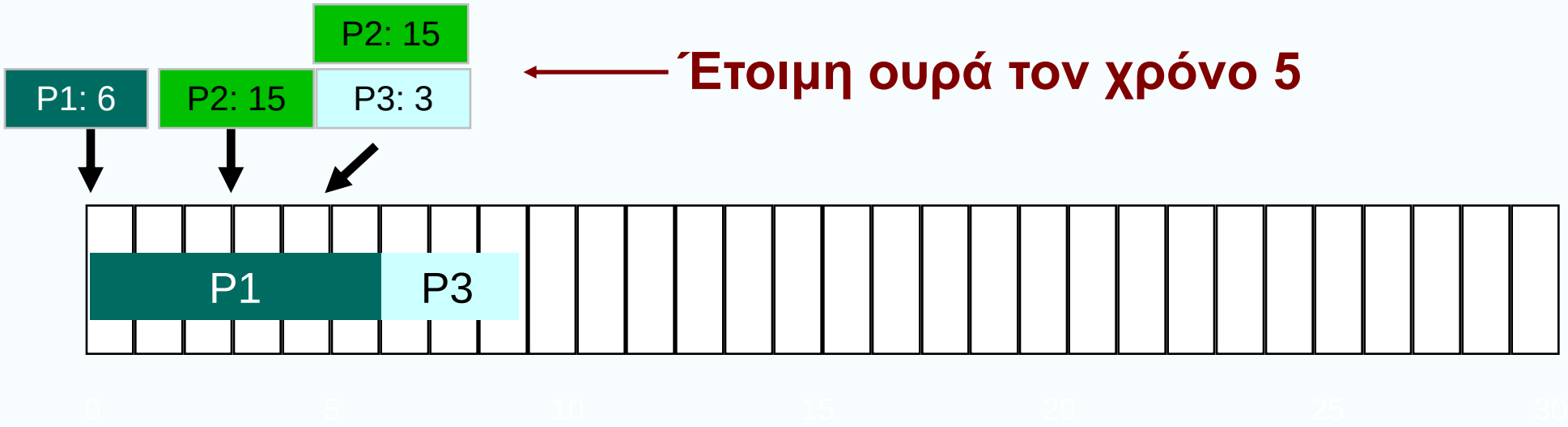
Process #	*Arrival Time*	*Burst Length*	*Priority*
P1	0	6	1
P2	3	15	1
P3	5	3	1
P4	8	4	1
P5	14	2	1

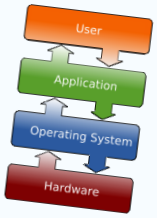




SJF – παράδειγμα - 2 (συν.)

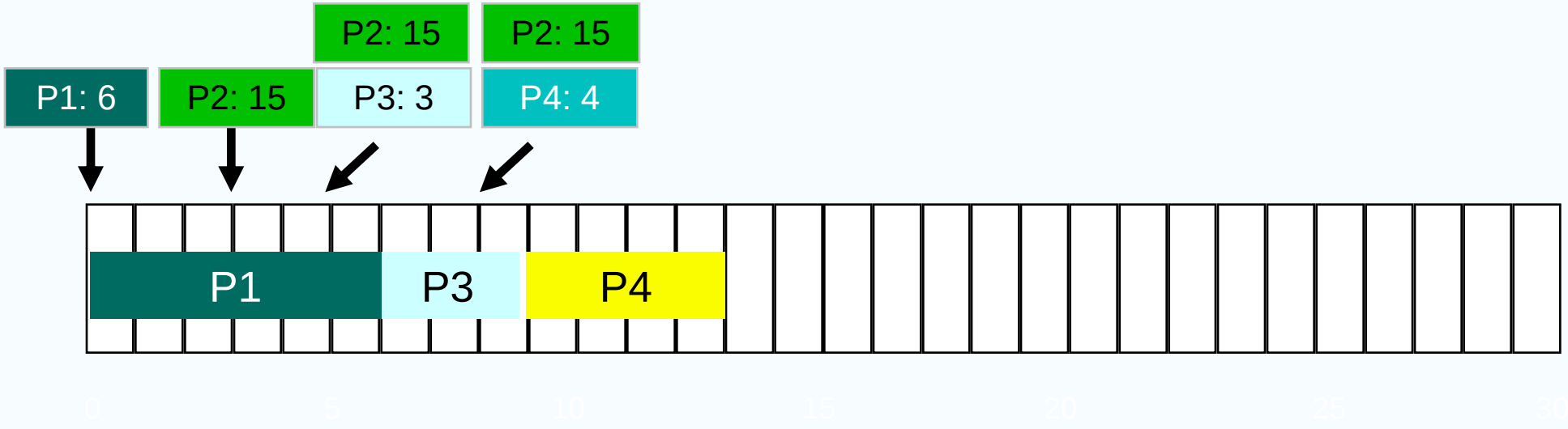
Process #	*Arrival Time*	*Burst Length*	*Priority*
P1	0	6	1
P2	3	15	1
P3	5	3	1
P4	8	4	1
P5	14	2	1

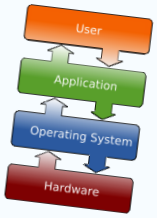




SJF – παράδειγμα - 2 (συν.)

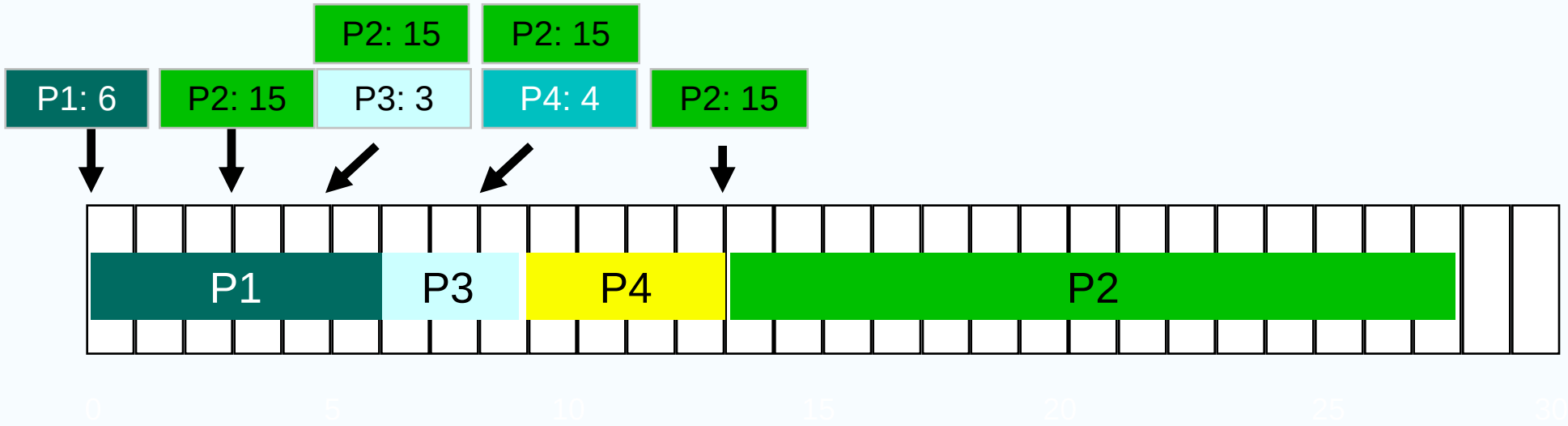
Process #	*Arrival Time*	*Burst Length*	*Priority*
P1	0	6	1
P2	3	15	1
P3	5	3	1
P4	8	4	1
P5	14	2	1

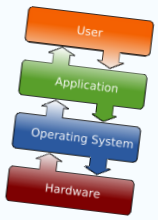




SJF – παράδειγμα - 2 (συν.)

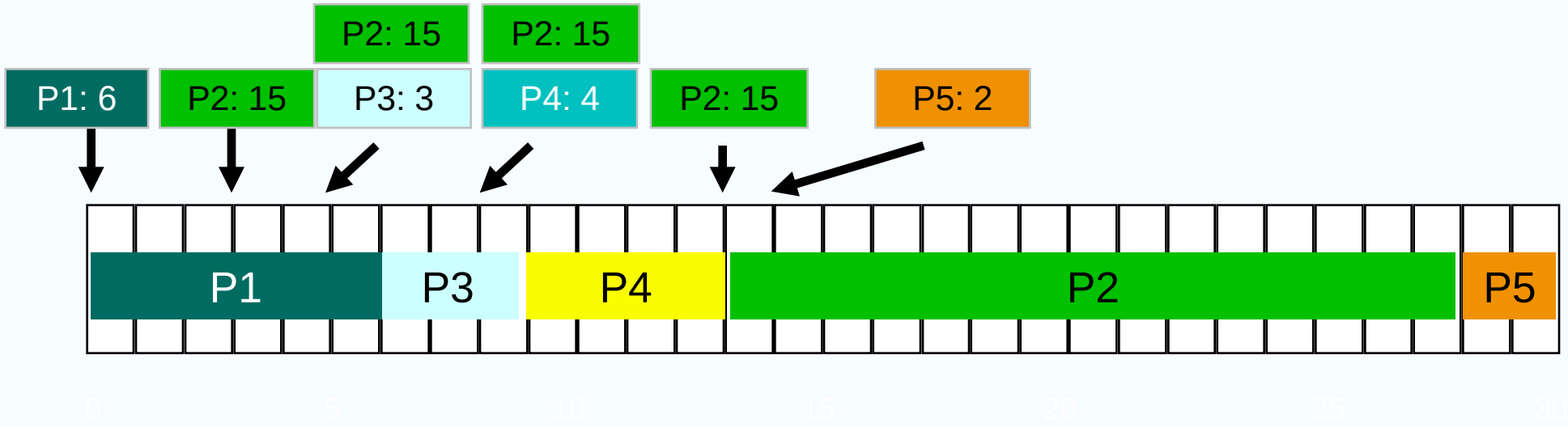
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	5	3	1
P4	8	4	1
P5	14	2	1

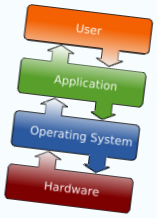




SJF – παράδειγμα - 2 (συν.)

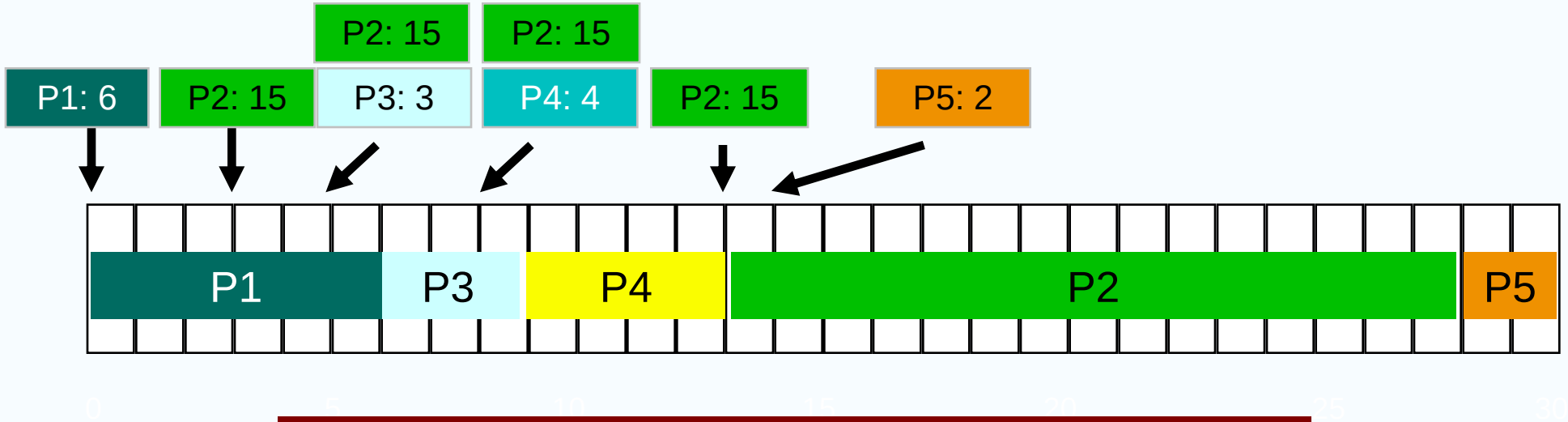
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	5	3	1
P4	8	4	1
P5	14	2	1



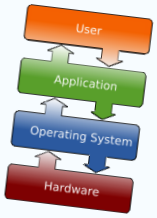


SJF – παράδειγμα - 2 (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	5	3	1
P4	8	4	1
P5	14	2	1

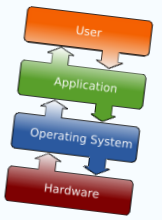


Process #	Waiting Time	Response Time	Turnaround Time	#of Context Switches
P1	0	0	6	1
P2	13-3 = 10	13-3 = 10	28-3 = 25	1
P3	6-5 = 1	6-5 = 1	9-5 = 4	1
P4	9-8 = 1	9-8 = 1	13-8 = 5	1
P5	28-14 = 14	28-14 = 14	30-14 = 16	1
Average	26/5 = 5.2	26/5 = 5.2	50/5 = 10	1



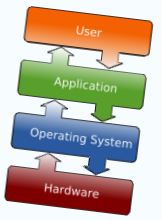
III. Shortest Remaining Time First (SRTF)

- Βασικές αρχές:
  - επιλέγεται η διεργασία με το μικρότερο εναπομείναντα χρόνο
    - Ο υπολειπόμενος χρόνος είναι ο συνολικός χρόνος καταιγισμού μείον τον χρόνο που παρέμεινε προς εξυπηρέτηση η διεργασία στη CPU
  - Αν φθάσει μια διεργασία με μικρότερο χρόνο καταιγισμού από τον υπολειπόμενο χρόνο καταιγισμού της τρέχουσας διεργασίας, η τρέχουσα διεργασία προεκχωρείται.
- Δεν είναι πρακτική εξ αιτίας της αναγκαστικής πρόβλεψης που απαιτείται για τους χρόνους καταιγισμού



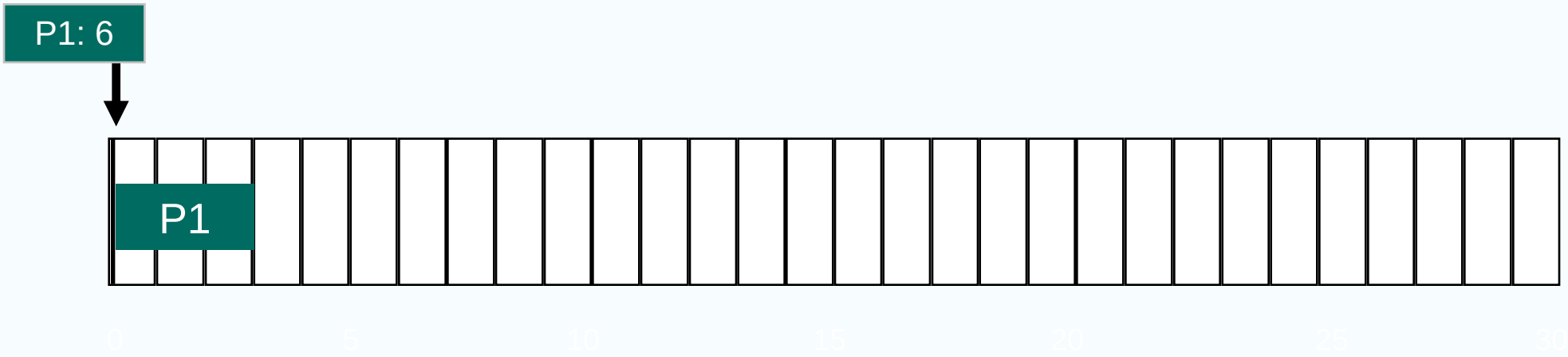
SRTF

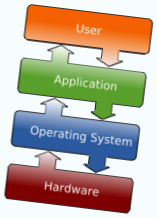
- Ο χρόνος άφιξης μιας νέας διεργασίας είναι σημαντικός
- Είναι χρήσιμη η καταγραφή των διεργασιών που βρίσκονται στην ουρά των έτοιμων διεργασιών
  - Η πολιτική που συνήθως ακολουθείται: οι προεκχωρούμενες διεργασίες οδηγούνται στην ουρά των έτοιμων διεργασιών
- Η απόφαση για δρομολόγηση λαμβάνεται όταν
  - Μια διεργασία έχει ολοκληρώσει το χρόνο καταιγισμού της στη CPU
  - Μια νέα διεργασία φθάνει στην ουρά των έτοιμων διεργασιών
- Δίνει καλούς χρόνους απόκρισης στις διεργασίες μικρής διάρκειας, γι' αυτό **προτιμάται στα πολυχρηστικά συστήματα**
- Αν ληφθούν υπόψη οι εναλλαγές πλαισίων η χρονική επιβάρυνση είναι σημαντική
- Η παρατεταμένη στέρηση είναι πιθανή



SRTF - παράδειγμα

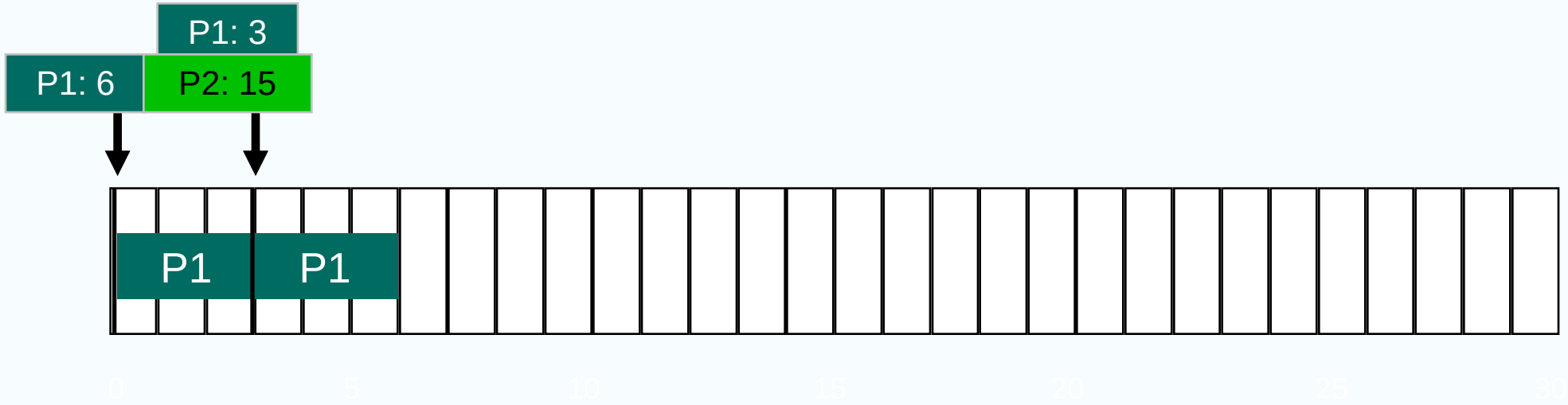
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

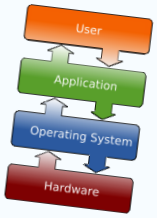




SRTF – παράδειγμα (συν.)

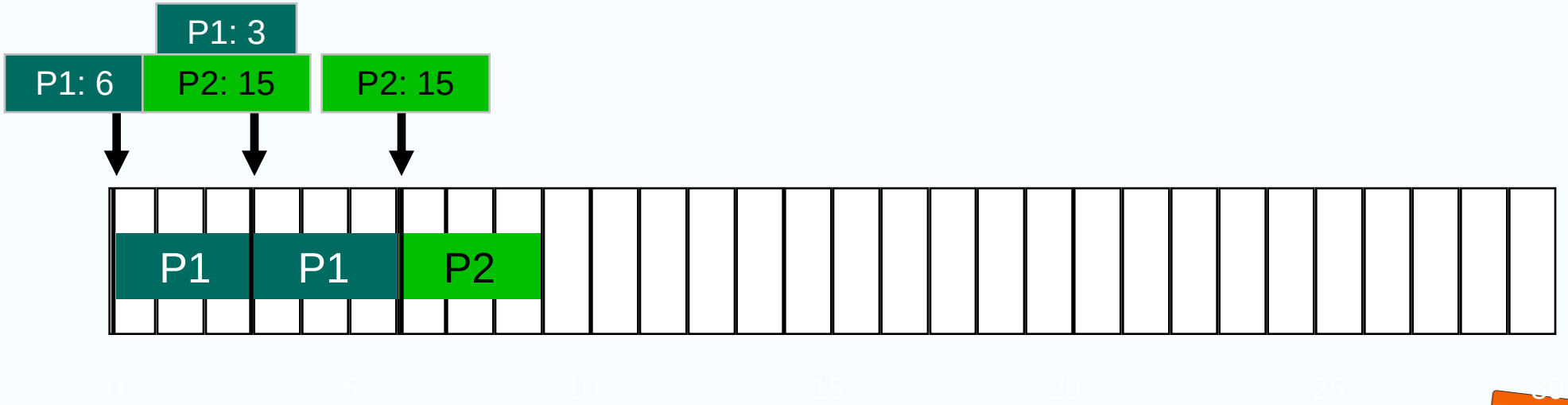
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

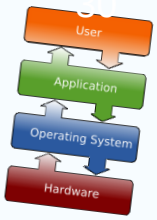




SRTF – παράδειγμα (συν.)

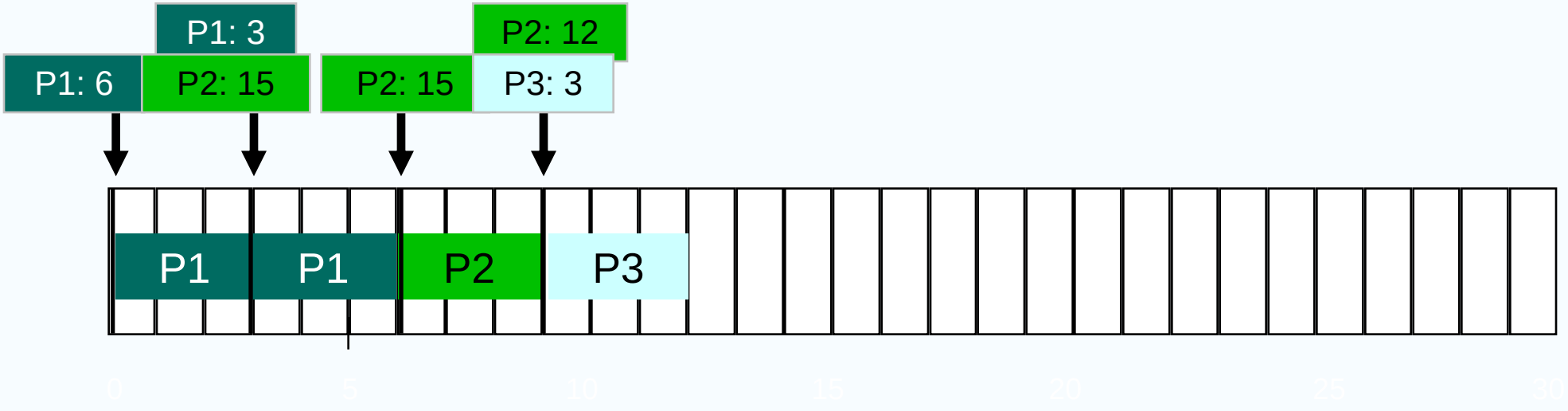
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

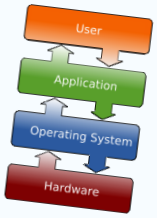




SRTF – παράδειγμα (συν.)

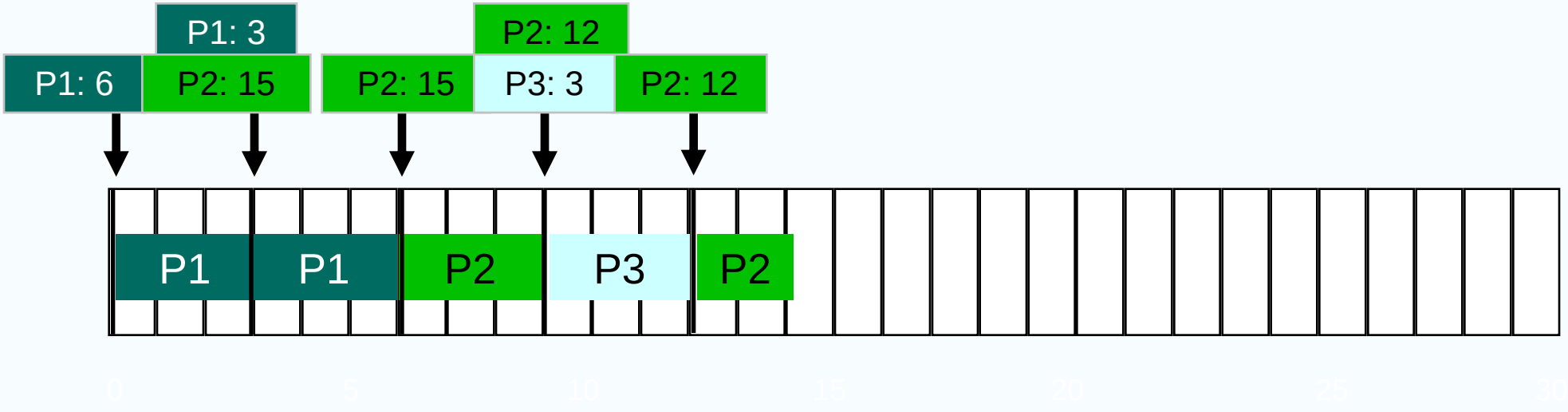
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

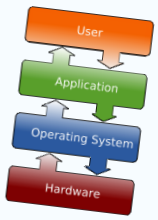




SRTF – παράδειγμα (συν.)

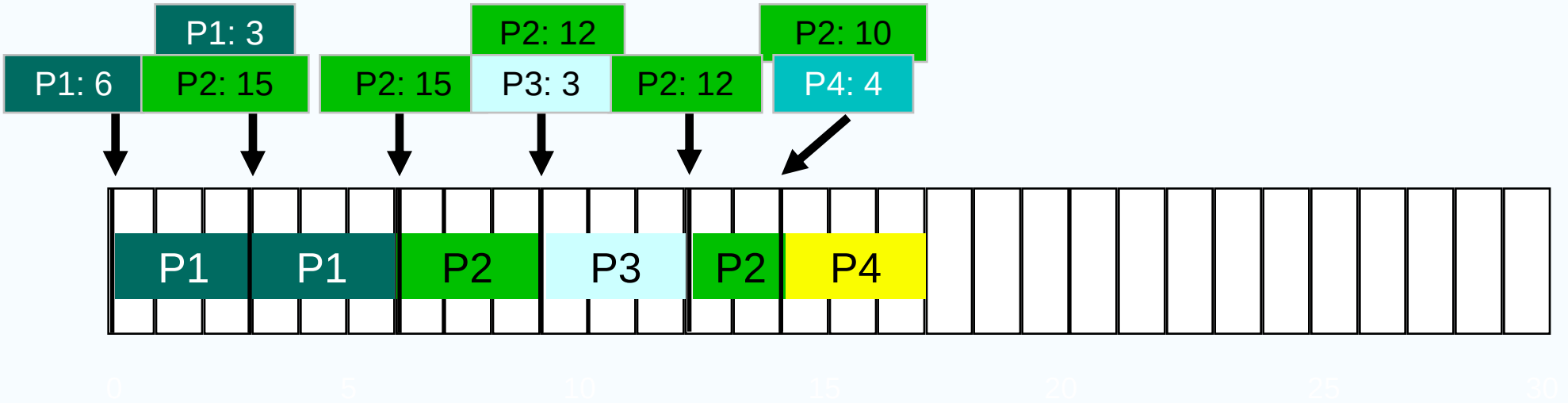
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

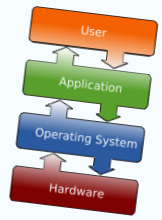




SRTF – παράδειγμα (συν.)

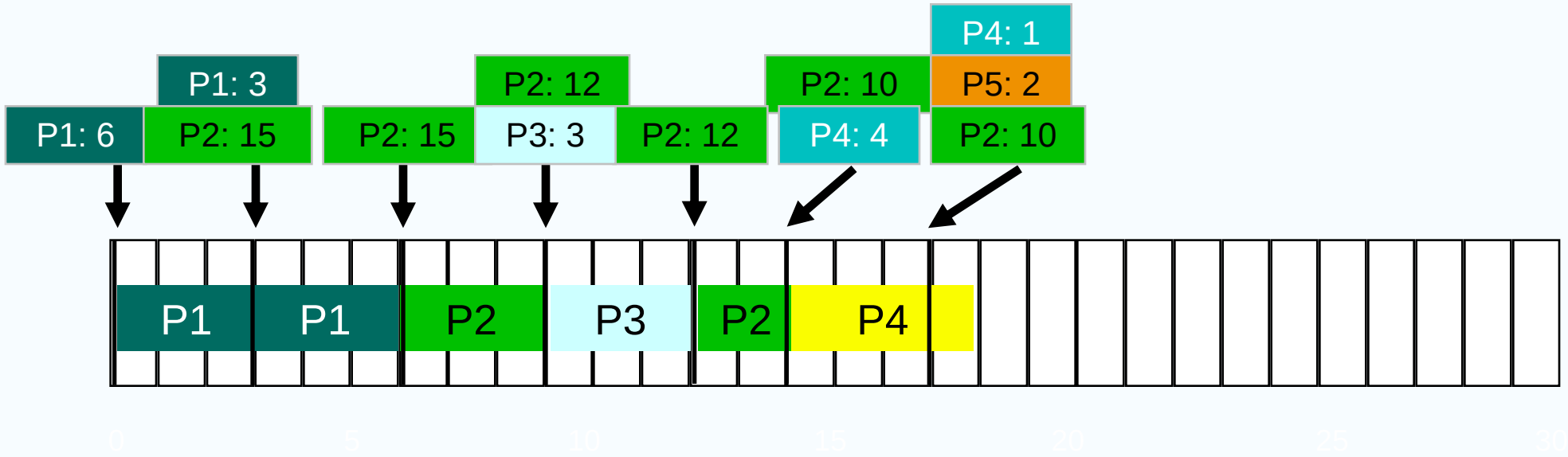
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

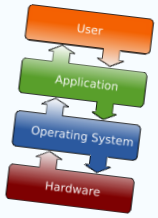




SRTF – παράδειγμα (συν.)

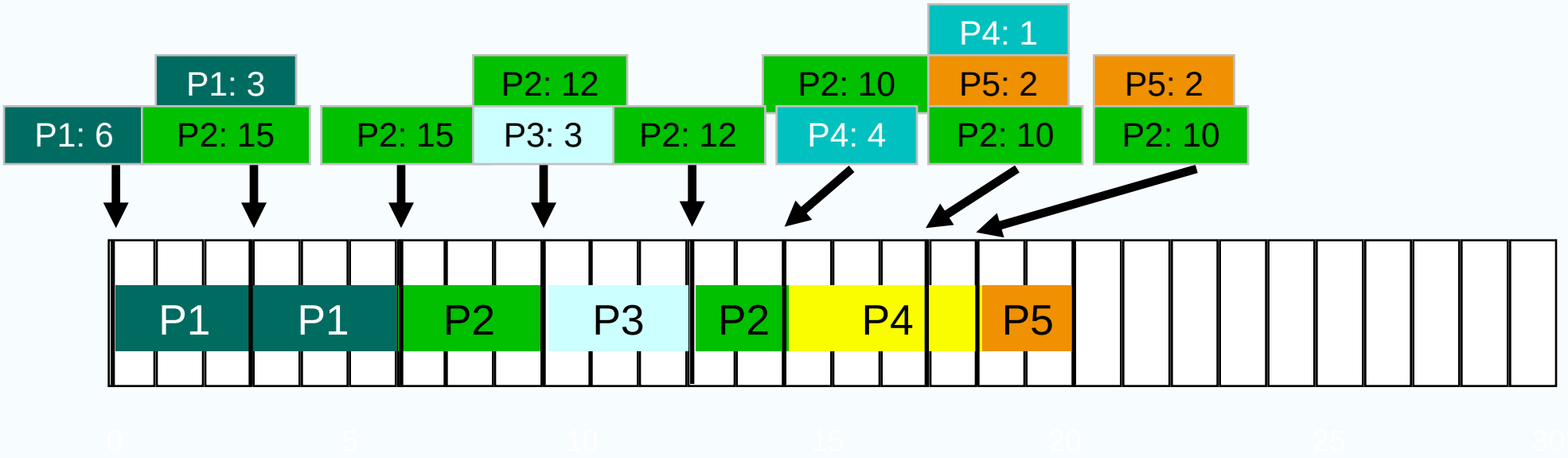
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

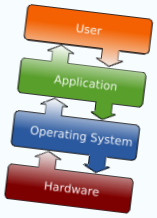




SRTF – παράδειγμα (συν.)

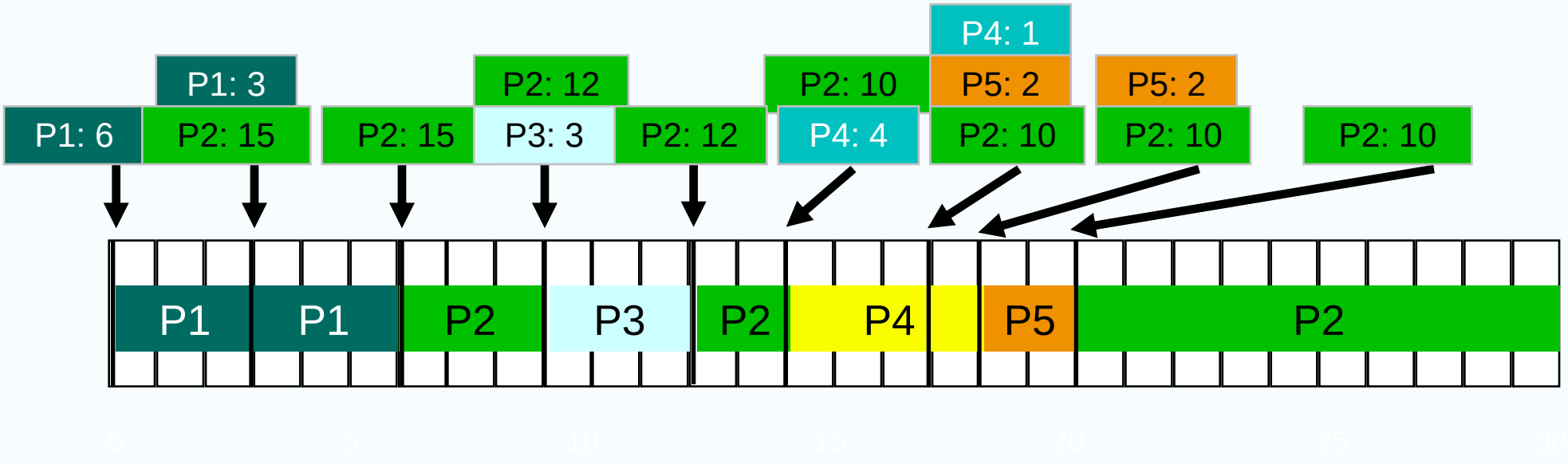
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

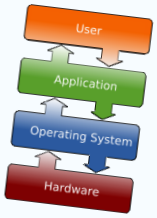




SRTF – παράδειγμα (συν.)

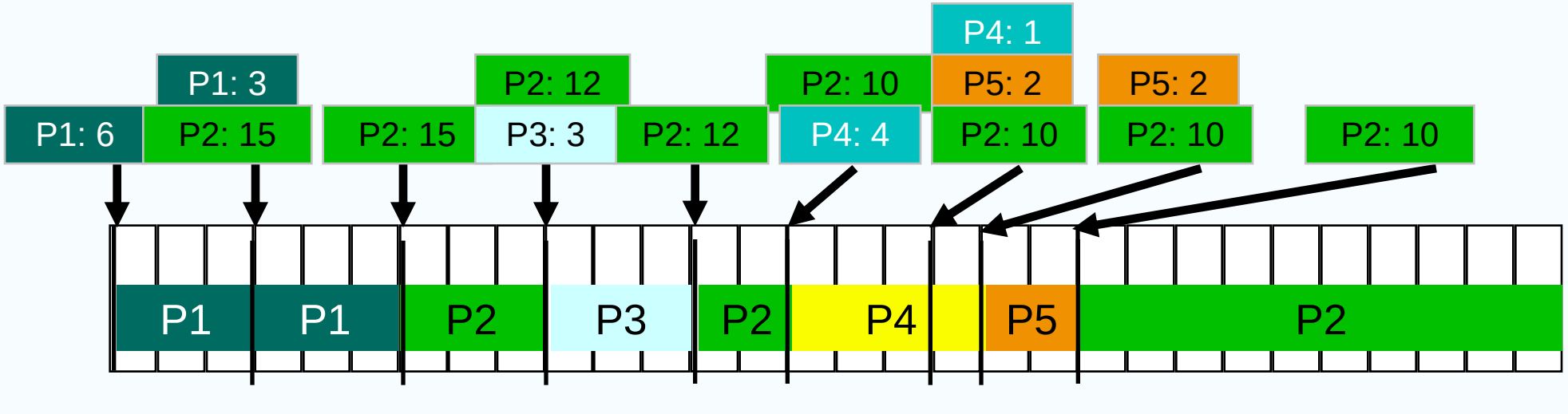
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1



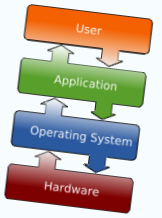


SRTF – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1



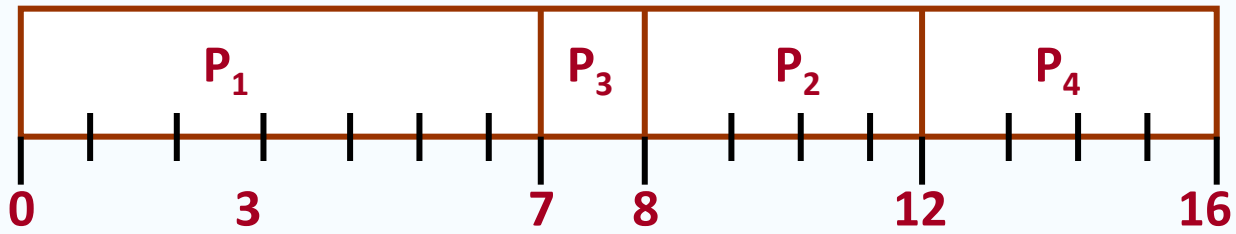
Process #	Waiting Time	Response Time	Turnaround Time	#of Context Switches
P1	0	0	6	1 (2)
P2	(6-3) + (12-9) + (20-14) = 12	(6-3) = 12	(30 - 3) = 27	3
P3	0	0	(12-9) = 3	1
P4	0	0	(18-14) = 3	1 (2)
P5	(18-17) = 1	(18-17) = 1	(20-17) = 3	1
Average	13/5 = 2.6	13/5 = 2.6	36/5 = 7.2	7



Παράδειγμα σύγκρισης SJF και SRTF

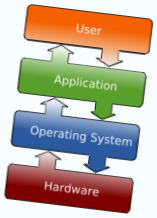
Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- **SJF**



- Average waiting time

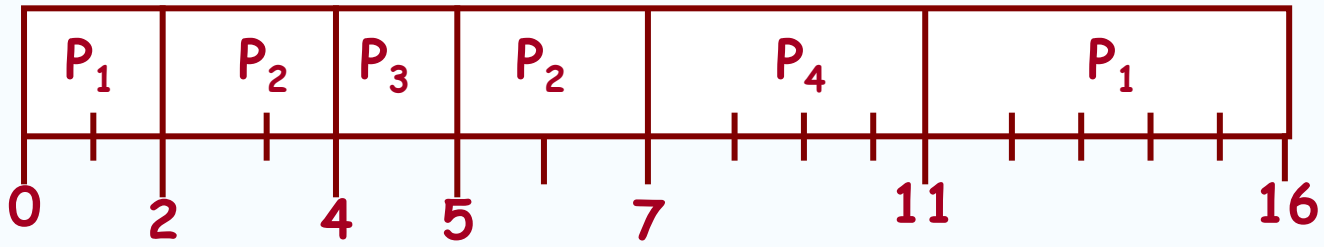
  = (0 + 6 + 3 + 7)/4 = 4



Παράδειγμα σύγκρισης SJF και SRTF

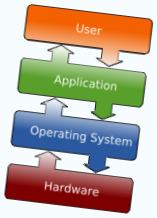
Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- **SRTF**



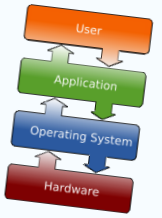
- Average waiting time

  = (9 + 1 + 0 +2)/4 = 3



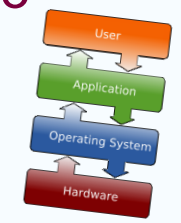
IV. Round Robin

- Βασικές αρχές
  - Στις διεργασίες δίνεται ένα σταθερό ποσό χρόνου της CPU και αναφέρεται ως: time quantum, time slice, slot
  - Η σειρά των διεργασιών είναι συνήθως FIFO
- Συνάρτηση επιλογής: ίδια με τον FCFS
- Κατάσταση απόφασης: προεκχώρηση
  - Μια διεργασία επιτρέπεται να εκτελείται μέχρι να συμπληρωθεί το κβάντο χρόνου
  - Στη συνέχεια δημιουργείται μια διακοπή ρολογιού και η εκτελούμενη διεργασία τίθεται στην ουρά των έτοιμων διεργασιών
- Χρησιμοποιείται συχνά σε περιβάλλοντα καταμερισμού χρόνου

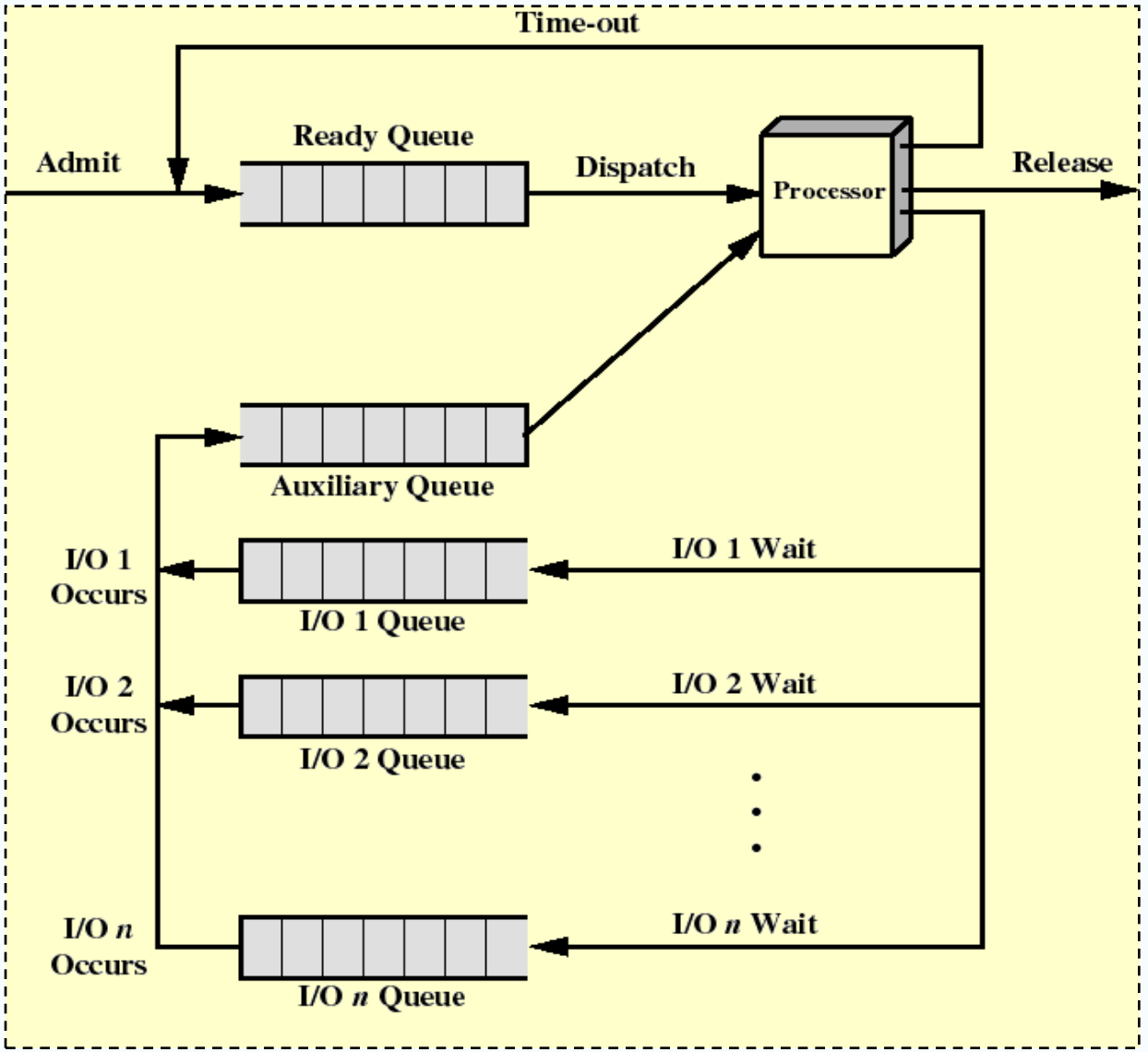


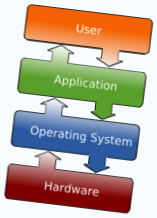
Round Robin: κριτική

- Ευνοούνται οι προοριζόμενες στην CPU διεργασίες
  - Μια προοριζόμενη για διεργασία Ε/Ε χρησιμοποιεί την CPU για χρονικό διάστημα μικρότερο του quantum χρόνου και στη συνέχεια αναστέλλεται περιμένοντας για Ε/Ε
  - Μια προοριζόμενη για την CPU διεργασία εκτελείται για όλο το quantum χρόνου και τίθεται μετά στην ουρά των έτοιμων διεργασιών (μπροστά από τις ανασταλμένες διεργασίες)
- **Λύση**: Ιδεατό round robin
  - Όταν μια Ε/Ε ολοκληρώνεται, η ανασταλμένη διεργασία μετακινείται σε μια βοηθητική ουρά που προτιμάται έναντι της βασικής ουράς των έτοιμων διεργασιών
  - Μια διεργασία που αποστέλλεται από την βοηθητική ουρά εκτελείται όχι περισσότερο από το βασικό quantum χρόνου μείον τον συνολικό χρόνο που δαπανήθηκε για εκτέλεση



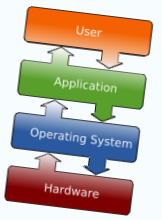
Διάγραμμα ουράς για ιδεατό Round Robin





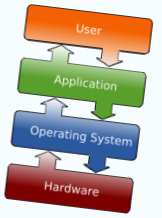
Round Robin

- Πλεονεκτήματα
  - Απλότητα
  - Χαμηλό overhead
  - Λειτουργεί αποτελεσματικά για αλληλεπιδραστικά συστήματα
- Μειονεκτήματα
  - Αν το quantum είναι πολύ μικρό, δαπανάται πολύς χρόνος για εναλλαγή πλαισίων.
  - Αν είναι πολύ μεγάλο προσεγγίζει τον FCFS.
  - Τυπικές τιμές για το quantum: 10 - 100 msec
- **Γενικός κανόνας:** επιλογή quantum έτσι ώστε η μεγάλη πλειοψηφία (80-90%) των διεργασιών να ολοκληρώνουν τον καταιγισμό τους σε ένα quantum



Μέγεθος του quantum χρόνου

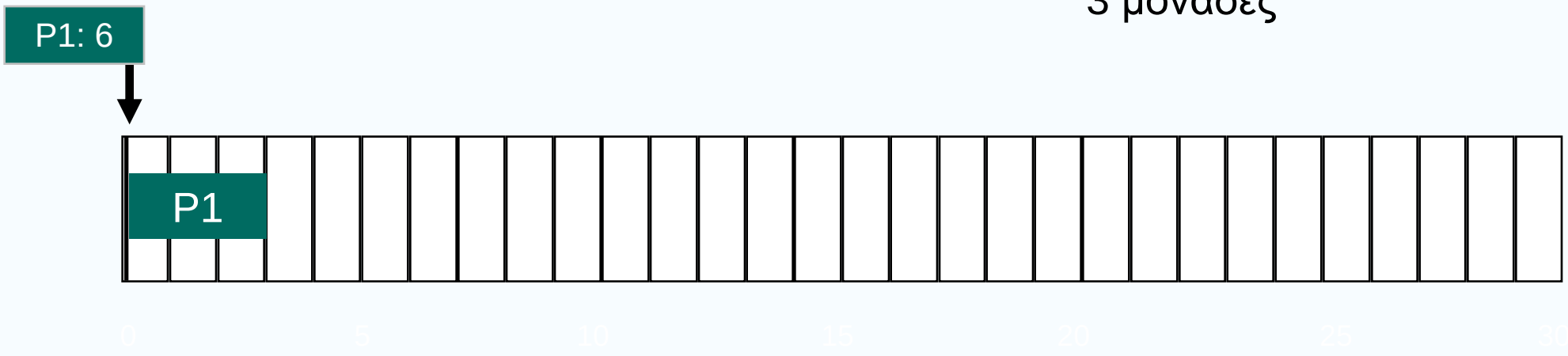
- Επιλογές
  - Μικρό ή μεγάλο quantum
  - Σταθερό ή μεταβλητό quantum
  - Το ίδιο για όλες τις διεργασίες ή διαφορετικό
- Αν το quantum είναι πολύ μεγάλο ο αλγόριθμος *RR* εκφυλίζεται σε *FCFS*
- Αν το quantum είναι πολύ μικρό υπάρχουν πολλές εναλλαγές πλαισίων
- **Βασική αρχή:** το quantum πρέπει να είναι ελαφρά μεγαλύτερο από το χρόνο που απαιτεί μια τυπική αλληλεπίδραση

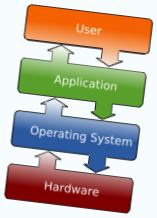


RR - παράδειγμα

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

Quantum χρόνου:
3 μονάδες

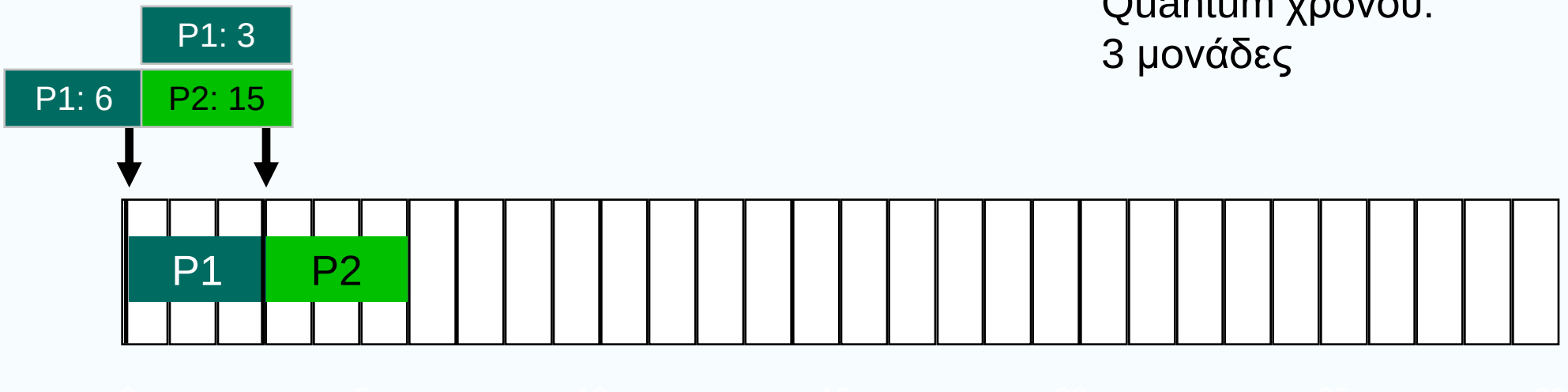


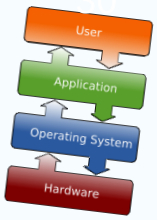


RR – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

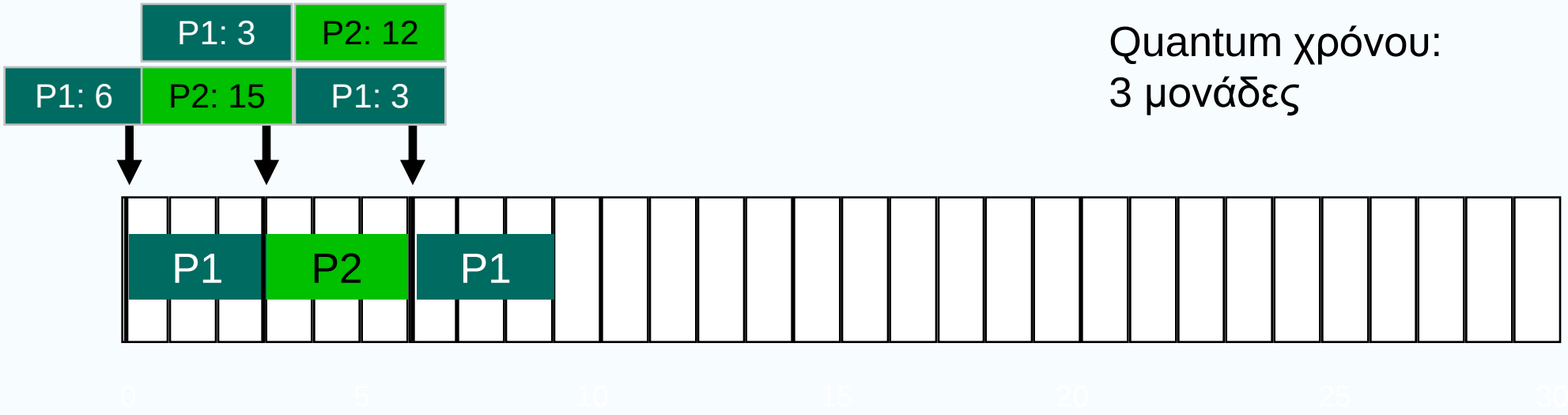
Quantum χρόνου:
3 μονάδες



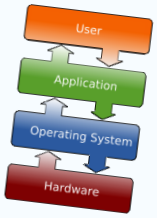


RR – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

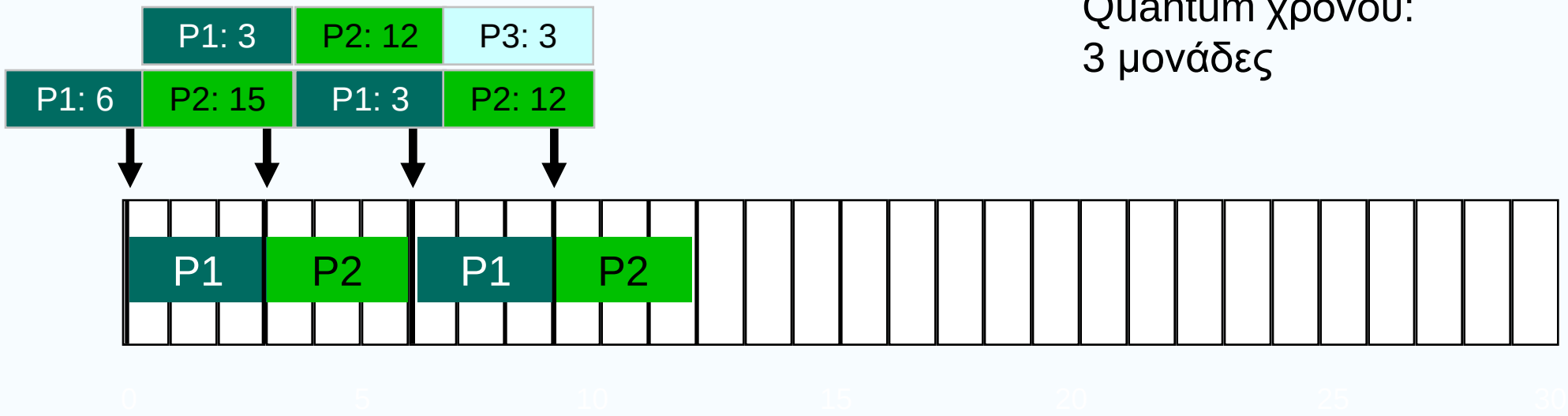


Quantum χρόνου:
3 μονάδες

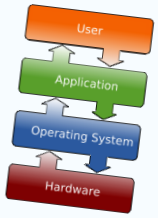


RR – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

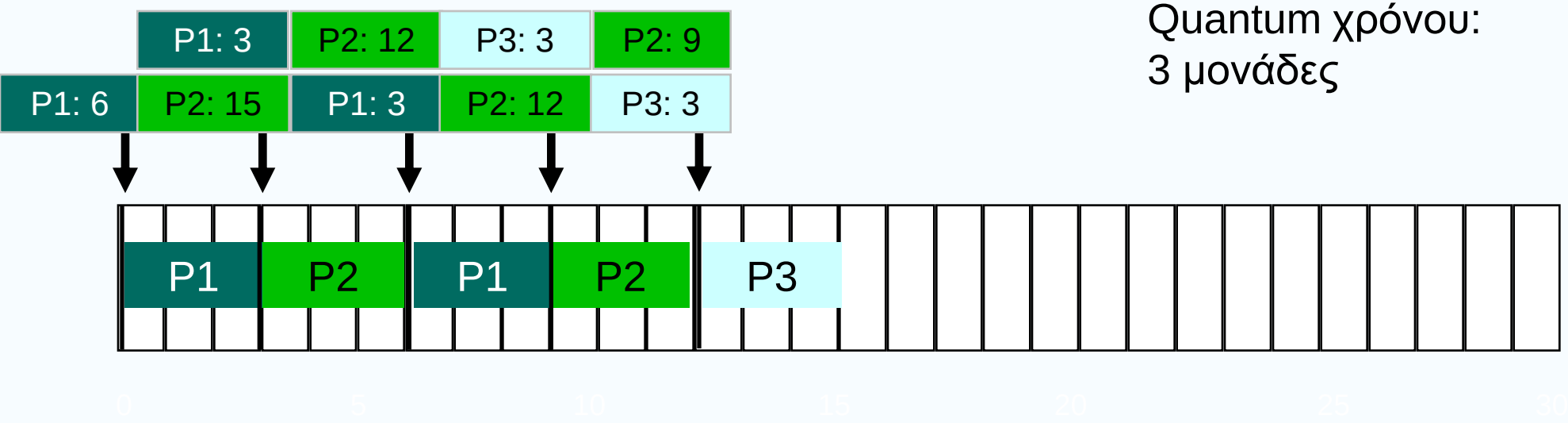


Quantum χρόνου:
3 μονάδες

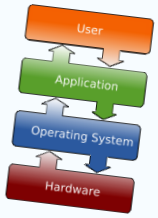


RR – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1



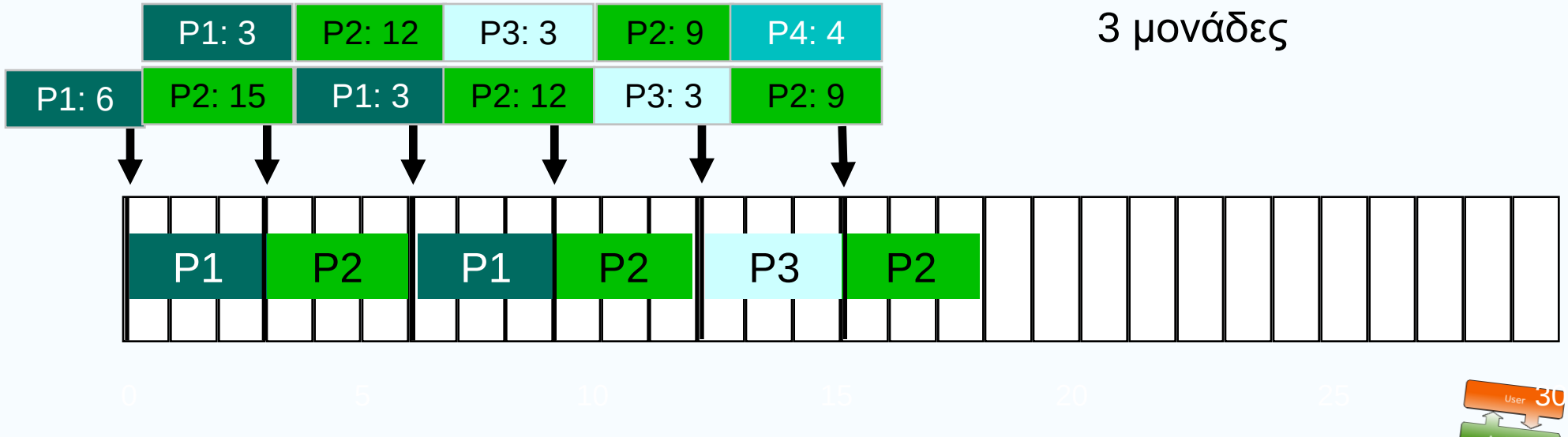
Quantum χρόνου:
3 μονάδες



RR – παράδειγμα (συν.)

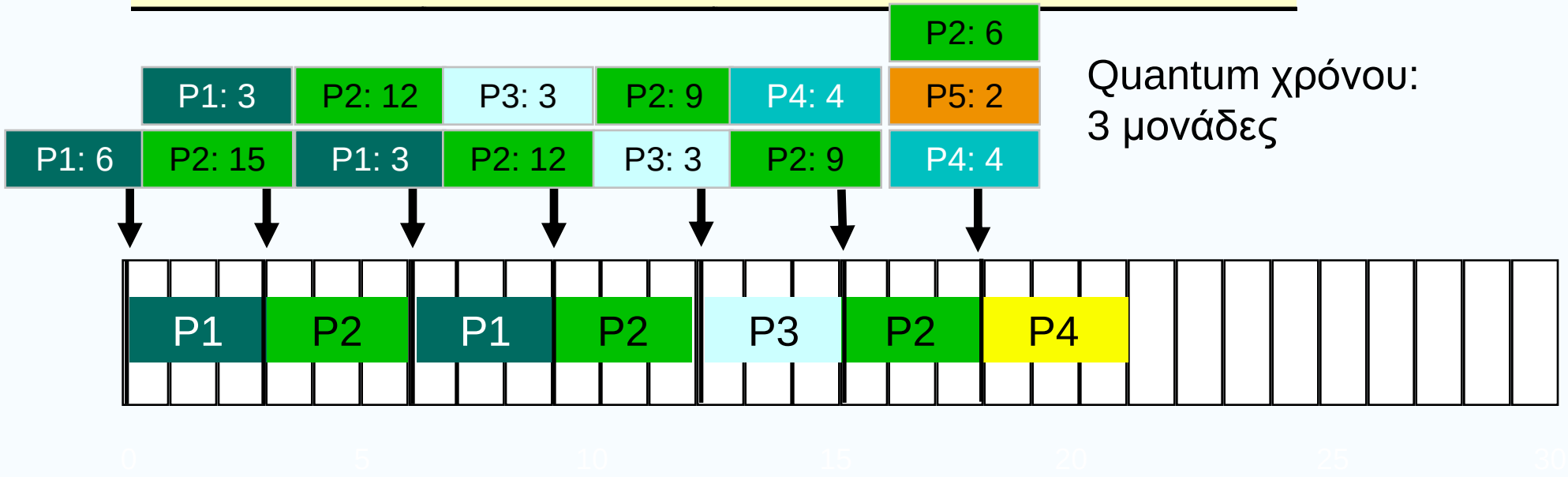
Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

Quantum χρόνου:
3 μονάδες

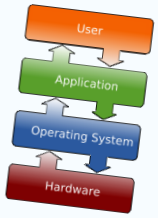


RR – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

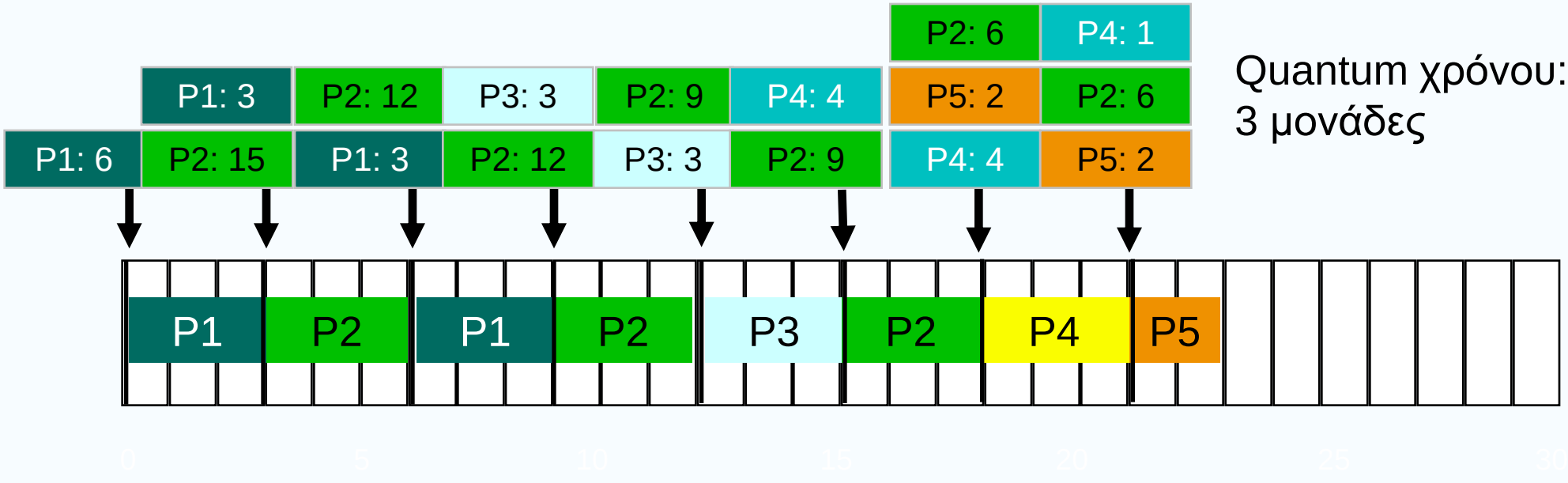


Quantum χρόνου:
3 μονάδες

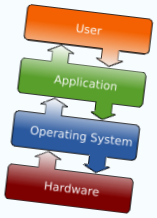


RR – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1



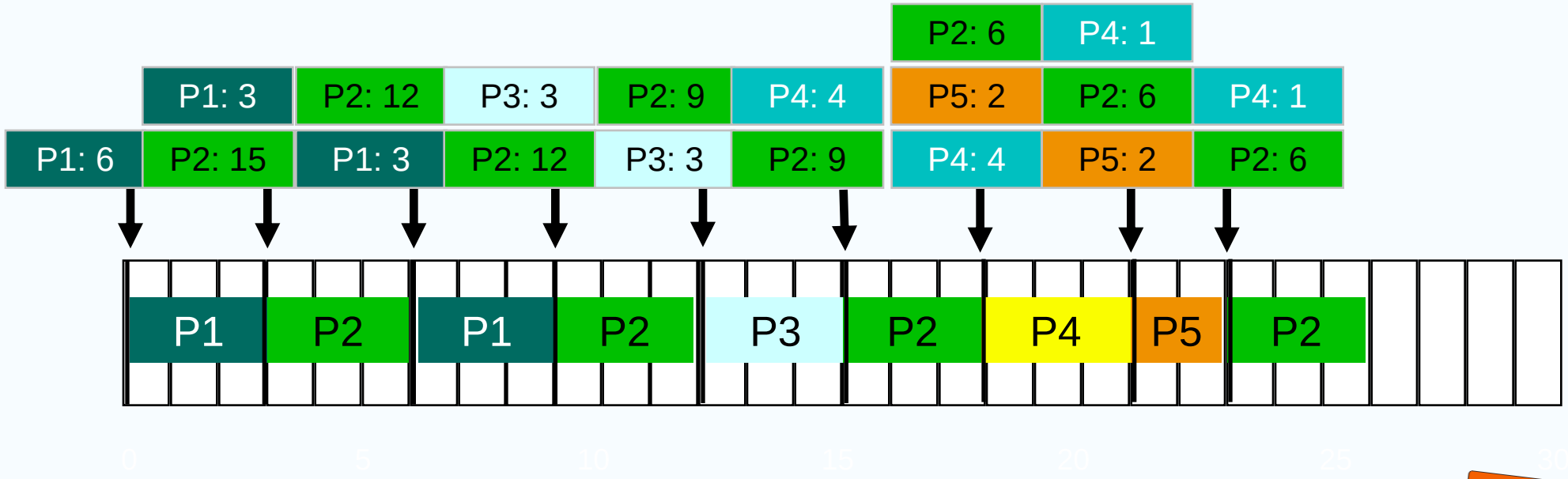
Quantum χρόνου:
3 μονάδες

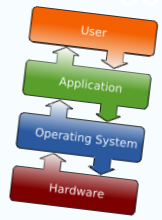


RR – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

Quantum χρόνου:
3 μονάδες

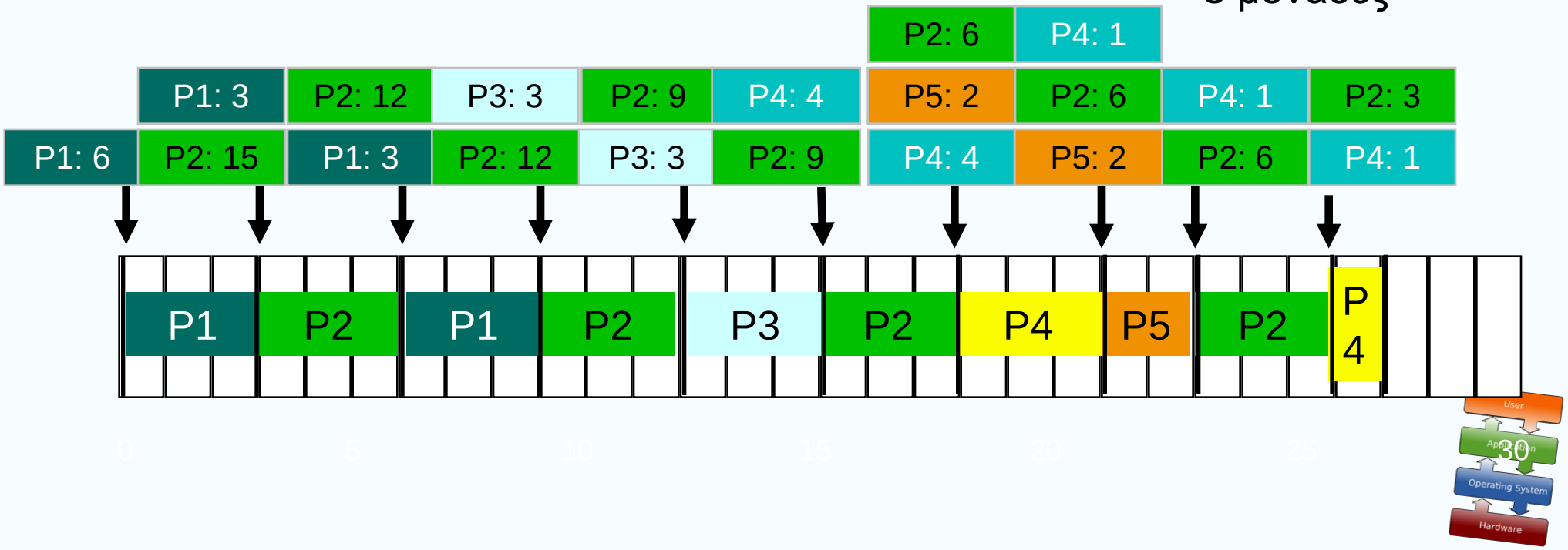




RR – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

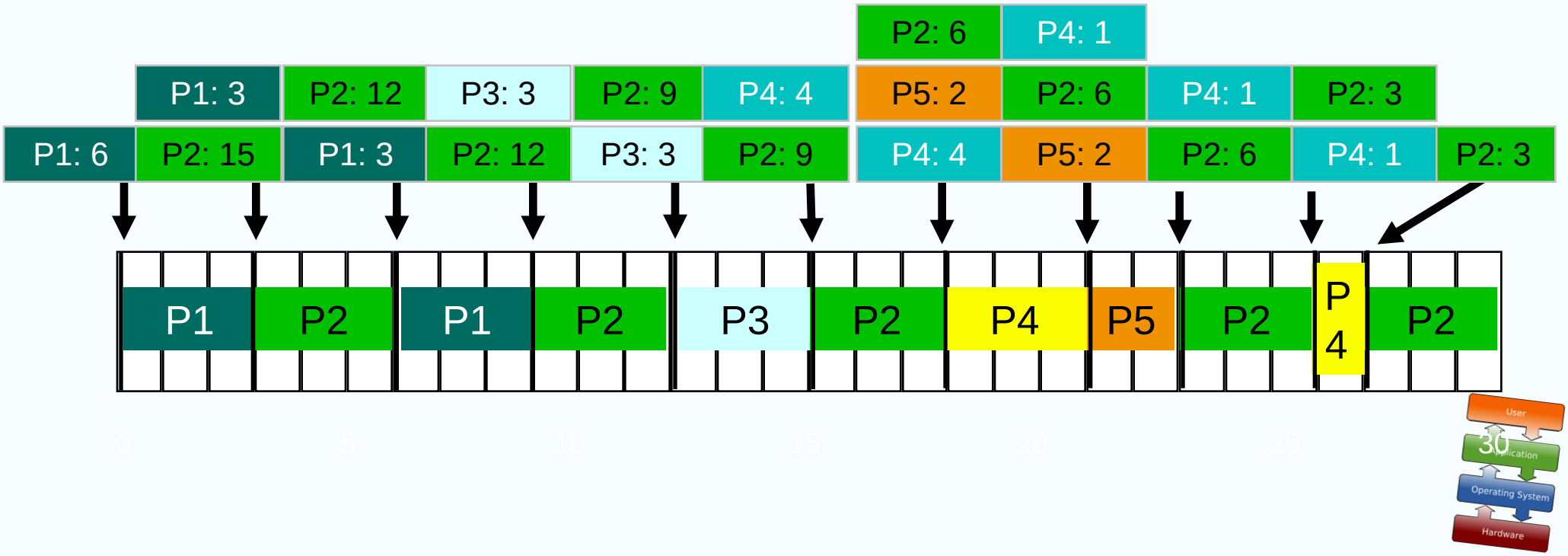
Quantum χρόνου:
3 μονάδες



RR – παράδειγμα (συν.)

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

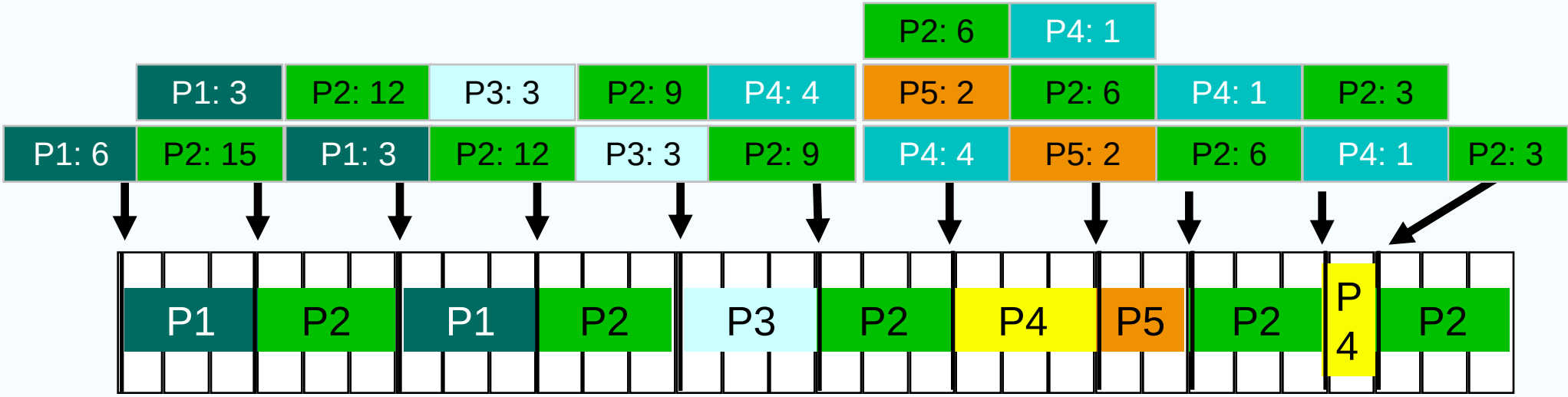
Quantum χρόνου:
3 μονάδες



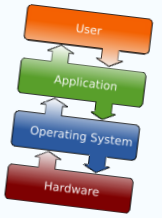
RR – παράδειγμα (συν.)

Quantum χρόνου:
3 μονάδες

Process #	Arrival Time	Burst Length	Priority
P1	0	6	1
P2	3	15	1
P3	9	3	1
P4	14	4	1
P5	17	2	1

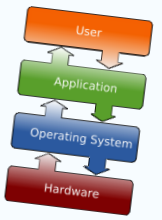


Process #	Waiting Time	Response Time	Turnaround Time	#of Context Switches
P1	0	0	9	2
P2	(9-6) + (15-12) + (23-18) + (27-26) = 12	0	(30 - 3) = 27	5
P3	(12-9) = 3	(12-9) = 3	(15-9) = 6	1
P4	(18-14) + (26-21) = 9	(18-14) = 4	(27-14) = 13	2
P5	(21-17) = 4	(21-17) = 4	(23-17) = 6	1
Average	28/5 = 5.6	11/5 = 2.2	52/5 = 10.4	11/5 =2.2



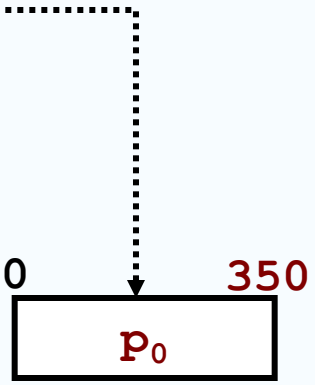
Παραδείγματα - Συμβολισμοί

- $P = \{p_i \mid 0 \leq i < n\}$ = Σύνολο διεργασιών
- S(pi) ∈ {Running, Ready, Blocked}
- t(pi) = Service Time
- W(pi) = Wait Time
- TTRnd(pi) = Turnaround Time



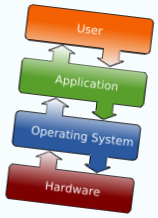
First-Come-First-Served

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



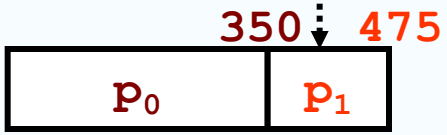
$T_{TRnd}(p_0) = t(p_0) = 350$

$W(p_0) = 0$



First-Come-First-Served

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

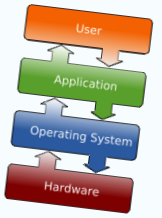


$T_{TRnd}(p_0) = t(p_0) = 350$

$T_{TRnd}(p_1) = (t(p_1) + T_{TRnd}(p_0)) = 125+350 = 475$

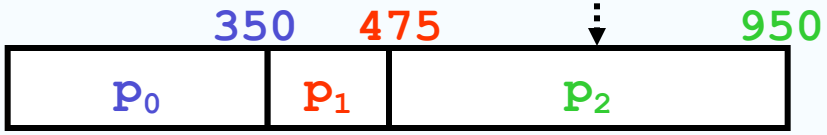
$W(p_0) = 0$

$W(p_1) = T_{TRnd}(p_0) = 350$



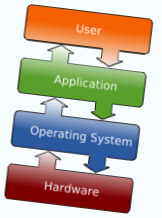
First-Come-First-Served

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



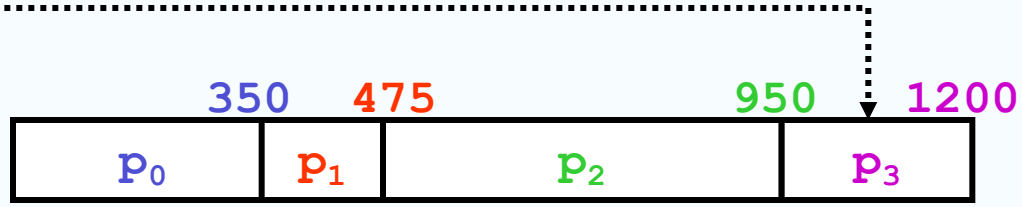
$T_{TRnd}(p_0) = t(p_0) = 350$
$T_{TRnd}(p_1) = (t(p_1)+T_{TRnd}(p_0)) = 125+350 = 475$
$T_{TRnd}(p_2) = (t(p_2)+T_{TRnd}(p_1)) = 475+475 = 950$

$W(p_0) = 0$
$W(p_1) = T_{TRnd}(p_0) = 350$
$W(p_2) = T_{TRnd}(p_1) = 475$



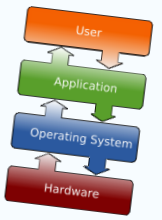
First-Come-First-Served

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



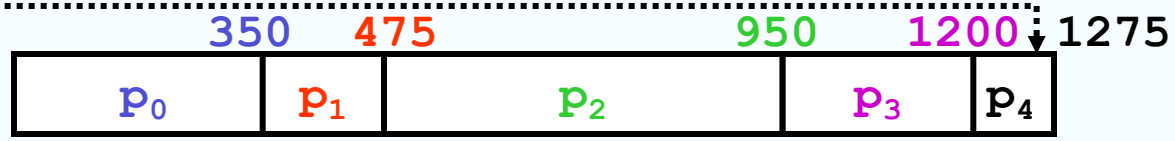
$T_{TRnd}(p_0) = t(p_0) = 350$
$T_{TRnd}(p_1) = (t(p_1) + T_{TRnd}(p_0)) = 125+350 = 475$
$T_{TRnd}(p_2) = (t(p_2) + T_{TRnd}(p_1)) = 475+475 = 950$
$T_{TRnd}(p_3) = (t(p_3) + T_{TRnd}(p_2)) = 250+950 = 1200$

$W(p_0) = 0$
$W(p_1) = T_{TRnd}(p_0) = 350$
$W(p_2) = T_{TRnd}(p_1) = 475$
$W(p_3) = T_{TRnd}(p_2) = 950$



First-Come-First-Served

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



$T_{TRnd}(p_0) = t(p_0) = 350$

$T_{TRnd}(p_1) = (t(p_1) + T_{TRnd}(p_0)) = 125+350 = 475$

$T_{TRnd}(p_2) = (t(p_2) + T_{TRnd}(p_1)) = 475+475 = 950$

$T_{TRnd}(p_3) = (t(p_3) + T_{TRnd}(p_2)) = 250+950 = 1200$

$T_{TRnd}(p_4) = (t(p_4) + T_{TRnd}(p_3)) = 75+1200 = 1275$

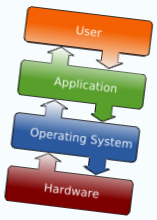
$W(p_0) = 0$

$W(p_1) = T_{TRnd}(p_0) = 350$

$W(p_2) = T_{TRnd}(p_1) = 475$

$W(p_3) = T_{TRnd}(p_2) = 950$

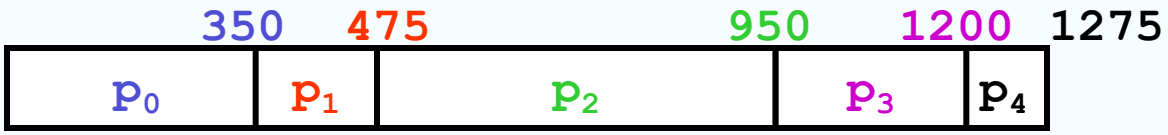
$W(p_4) = T_{TRnd}(p_3) = 1200$



FCFS Average Wait Time

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

- Ευκολία στην εφαρμογή
- Δεν έχει μεγάλη απόδοση



$T_{TRnd}(p_0) = t(p_0) = 350$
$T_{TRnd}(p_1) = (t(p_1) + T_{TRnd}(p_0)) = 125+350 = 475$
$T_{TRnd}(p_2) = (t(p_2) + T_{TRnd}(p_1)) = 475+475 = 950$
$T_{TRnd}(p_3) = (t(p_3) + T_{TRnd}(p_2)) = 250+950 = 1200$
$T_{TRnd}(p_4) = (t(p_4) + T_{TRnd}(p_3)) = 75+1200 = 1275$

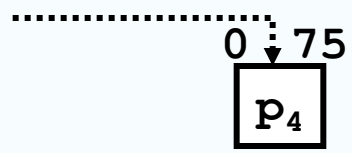
$W(p_0) = 0$
$W(p_1) = T_{TRnd}(p_0) = 350$
$W(p_2) = T_{TRnd}(p_1) = 475$
$W(p_3) = T_{TRnd}(p_2) = 950$
$W(p_4) = T_{TRnd}(p_3) = 1200$

$T_{TRnd} = (350+475+950+1200+1275)/5 = 4250/5 = 850$

$W_{avg} = (0+350+475+950+1200)/5 = 2975/5 = 595$

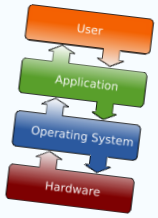
Shortest Job First

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



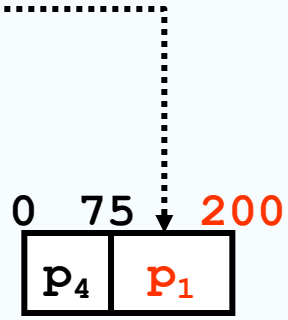
$T_{TRnd}(p_4) = t(p_4) = 75$

$W(p_4) = 0$



Shortest Job First

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

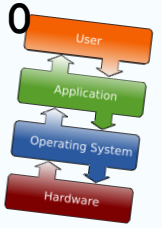


$T_{TRnd}(p_1) = t(p_1)+t(p_4) = 125+75 = 200$

$W(p_1) = 75$

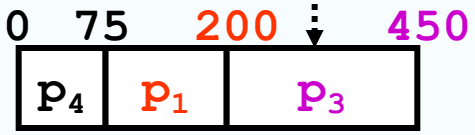
$T_{TRnd}(p_4) = t(p_4) = 75$

$W(p_4) = 0$



Shortest Job First

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



$T_{TRnd}(p_1) = t(p_1)+t(p_4) = 125+75 = 200$

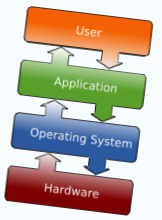
$W(p_1) = 75$

$T_{TRnd}(p_3) = t(p_3)+t(p_1)+t(p_4) = 250+125+75 = 450$

$W(p_3) = 200$

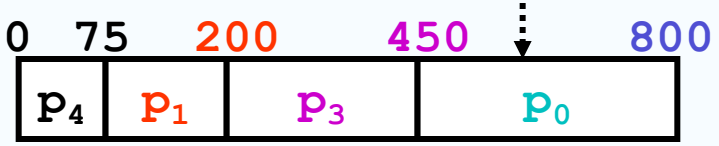
$T_{TRnd}(p_4) = t(p_4) = 75$

$W(p_4) = 0$



Shortest Job First

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

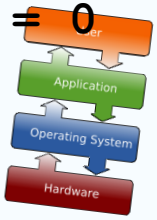


$T_{TRnd}(p_0) = t(p_0)+t(p_3)+t(p_1)+t(p_4) = 350+250+125+75 = 800$ $W(p_0) = 450$

$T_{TRnd}(p_1) = t(p_1)+t(p_4) = 125+75 = 200$ $W(p_1) = 75$

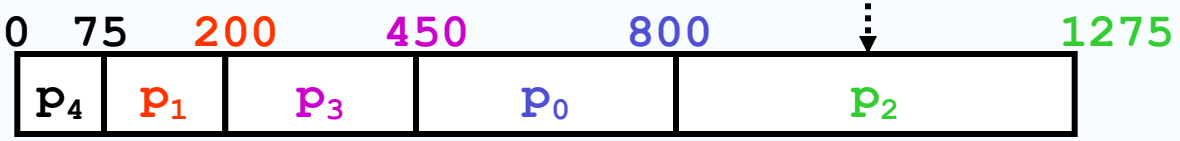
$T_{TRnd}(p_3) = t(p_3)+t(p_1)+t(p_4) = 250+125+75 = 450$ $W(p_3) = 200$

$T_{TRnd}(p_4) = t(p_4) = 75$ $W(p_4) = 0$



Shortest Job First

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



$T_{TRnd}(p_0) = t(p_0)+t(p_3)+t(p_1)+t(p_4) = 350+250+125+75 = 800$

$T_{TRnd}(p_1) = t(p_1)+t(p_4) = 125+75 = 200$

$T_{TRnd}(p_2) = t(p_2)+t(p_0)+t(p_3)+t(p_1)+t(p_4)$

$= 475+350+250+125+75 = 1275$

$T_{TRnd}(p_3) = t(p_3)+t(p_1)+t(p_4) = 250+125+75 = 450$

$T_{TRnd}(p_4) = t(p_4) = 75$

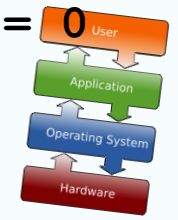
$W(p_0) = 450$

$W(p_1) = 75$

$W(p_2) = 800$

$W(p_3) = 200$

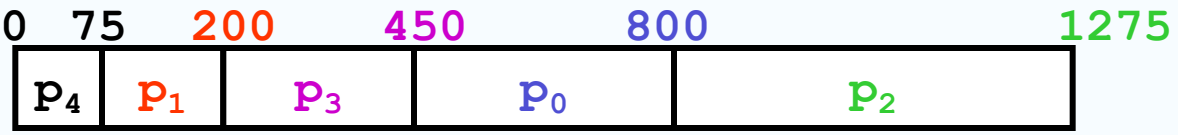
$W(p_4) = 0$



Shortest Job First

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

- Ελαχιστοποιεί τον χρόνο αναμονής
- Γνώση εκ των προτέρων των χρόνων εξυπηρέτησης



$T_{TRnd}(p_0) = t(p_0)+t(p_3)+t(p_1)+t(p_4) = 350+250+125+75 = 800$
$T_{TRnd}(p_1) = t(p_1)+t(p_4) = 125+75 = 200$
$T_{TRnd}(p_2) = t(p_2)+t(p_0)+t(p_3)+t(p_1)+t(p_4)$
$= 475+350+250+125+75 = 1275$
$T_{TRnd}(p_3) = t(p_3)+t(p_1)+t(p_4) = 250+125+75 = 450$
$T_{TRnd}(p_4) = t(p_4) = 75$

$W(p_0) = 450$
$W(p_1) = 75$
$W(p_2) = 800$
$W(p_3) = 200$
$W(p_4) = 0$

$T_{TRnd} = (800+200+1275+450+75)/5$
$= 2800/5 = 560$

$W_{avg} = (450+75+800+200+0)/5$
$= 1525/5 = 305$

Σύγκριση δρομολογήσεων χωρίς προεκχώρηση

- *First-Come-First-Serve*

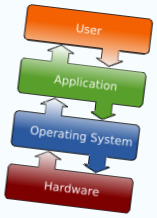
$T_{TRnd} = (350+475+950+1200+1275)/5 = 4250/5 = 850$

$W_{avg} = (0+350+475+950+1200)/5 = 2975/5 = 595$

- *Shortest Job First*

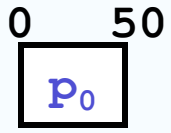
$T_{TRnd} = (800+200+1275+450+75)/5 = 2800/5 = 560$

$W_{avg} = (450+75+800+200+0)/5 = 1525/5 = 305$

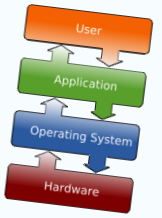


Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

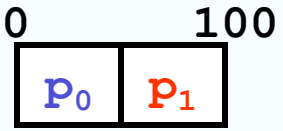


$W(p_0) = 0$



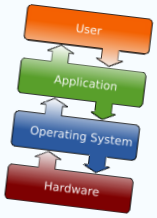
Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



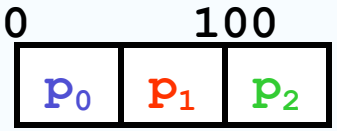
$W(p_0) = 0$

$W(p_1) = 50$

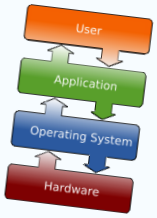


Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

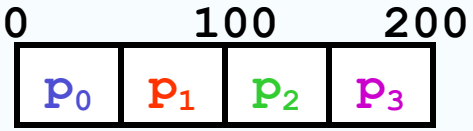


$W(p_0) = 0$
$W(p_1) = 50$
$W(p_2) = 100$

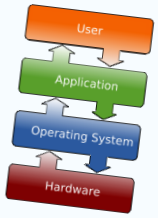


Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

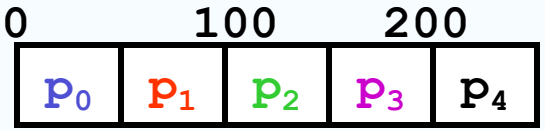


$W(p_0) = 0$
$W(p_1) = 50$
$W(p_2) = 100$
$W(p_3) = 150$

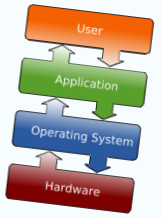


Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

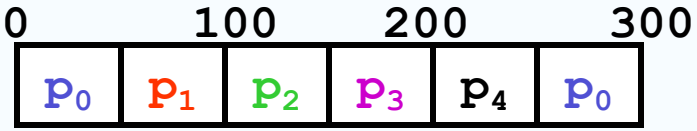


$W(p_0) = 0$
$W(p_1) = 50$
$W(p_2) = 100$
$W(p_3) = 150$
$W(p_4) = 200$

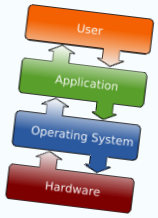


Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

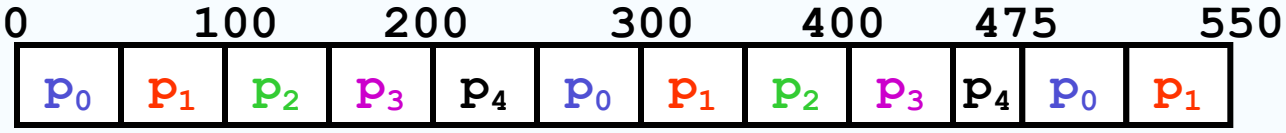


$W(p_0) = 0$
$W(p_1) = 50$
$W(p_2) = 100$
$W(p_3) = 150$
$W(p_4) = 200$



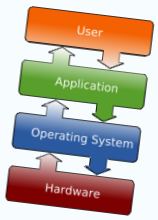
Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



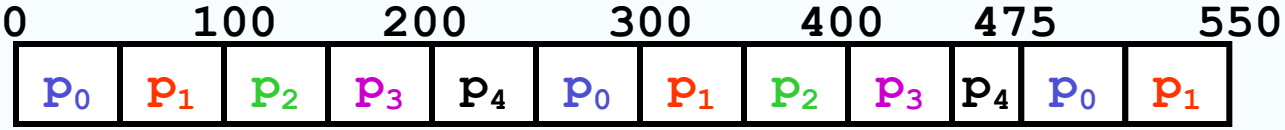
$W(p_0) = 0$
$W(p_1) = 50$
$W(p_2) = 100$
$W(p_3) = 150$
$W(p_4) = 200$

$T_{TRnd}(p_4) = 475$



Round Robin (TQ=50)

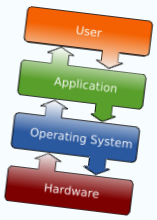
i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



$T_{TRnd}(p_1) = 550$

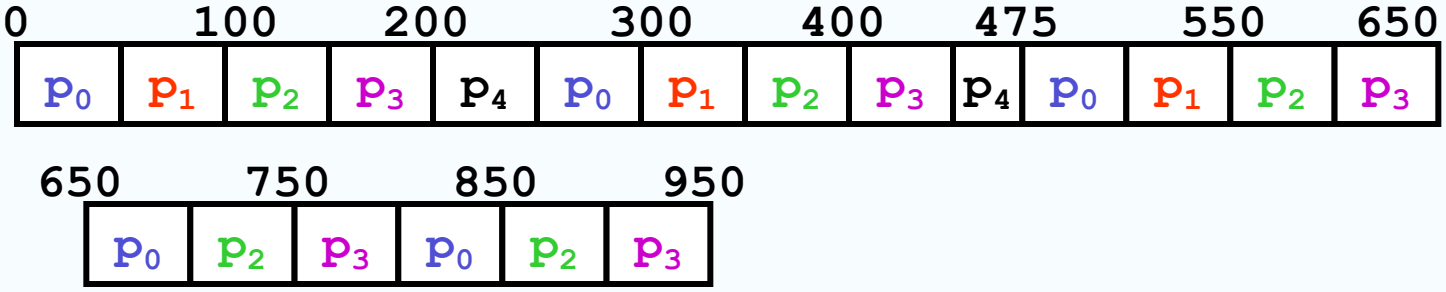
$T_{TRnd}(p_4) = 475$

$W(p_0) = 0$
$W(p_1) = 50$
$W(p_2) = 100$
$W(p_3) = 150$
$W(p_4) = 200$



Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



$T_{TRnd}(p_1) = 550$

$T_{TRnd}(p_3) = 950$

$T_{TRnd}(p_4) = 475$

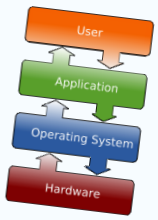
$W(p_0) = 0$

$W(p_1) = 50$

$W(p_2) = 100$

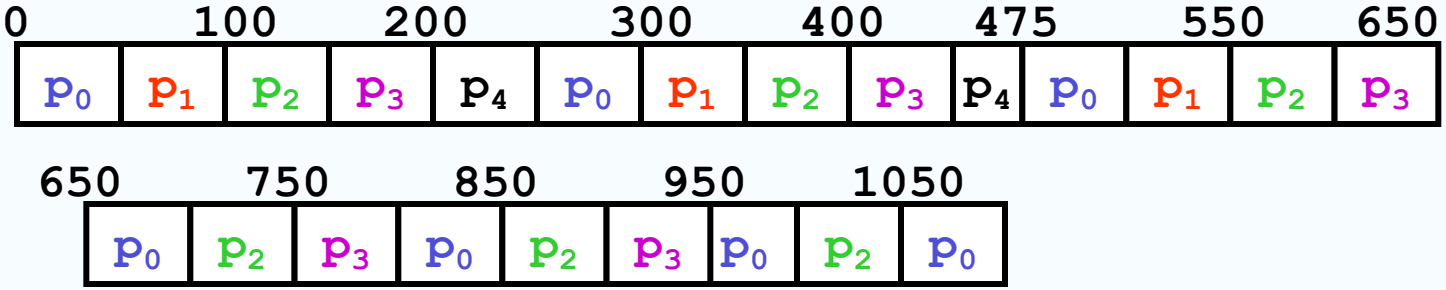
$W(p_3) = 150$

$W(p_4) = 200$



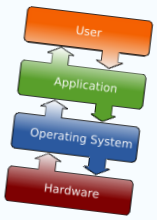
Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



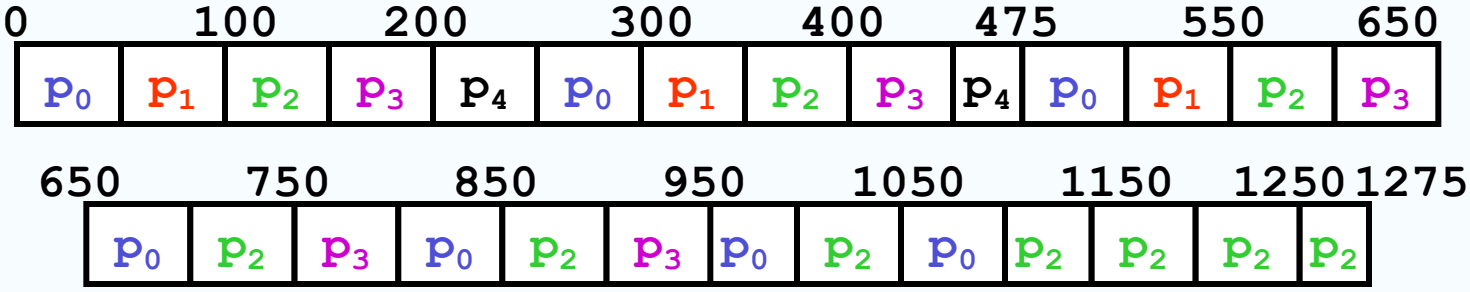
$T_{TRnd}(p_0) = 1100$
$T_{TRnd}(p_1) = 550$
$T_{TRnd}(p_3) = 950$
$T_{TRnd}(p_4) = 475$

$W(p_0) = 0$
$W(p_1) = 50$
$W(p_2) = 100$
$W(p_3) = 150$
$W(p_4) = 200$



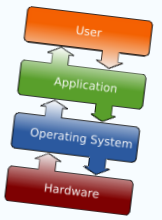
Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75



$T_{TRnd}(p_0) = 1100$
$T_{TRnd}(p_1) = 550$
$T_{TRnd}(p_2) = 1275$
$T_{TRnd}(p_3) = 950$
$T_{TRnd}(p_4) = 475$

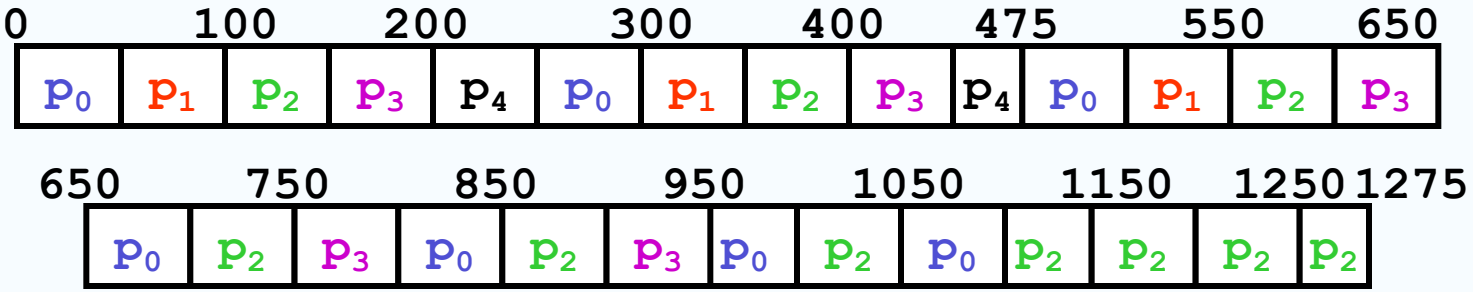
$W(p_0) = 0$
$W(p_1) = 50$
$W(p_2) = 100$
$W(p_3) = 150$
$W(p_4) = 200$



Round Robin (TQ=50)

i	$t(p_i)$
0	350
1	125
2	475
3	250
4	75

- Δίκαιη
- Χρησιμοποιείται ευρέως

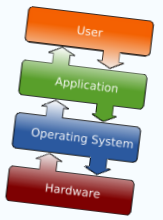


$T_{TRnd}(p_0) = 1100$
$T_{TRnd}(p_1) = 550$
$T_{TRnd}(p_2) = 1275$
$T_{TRnd}(p_3) = 950$
$T_{TRnd}(p_4) = 475$

$W(p_0) = 0$
$W(p_1) = 50$
$W(p_2) = 100$
$W(p_3) = 150$
$W(p_4) = 200$

$$T_{TRnd\_avg} = (1100+550+1275+950+475)/5 = 4350/5 = 870$$

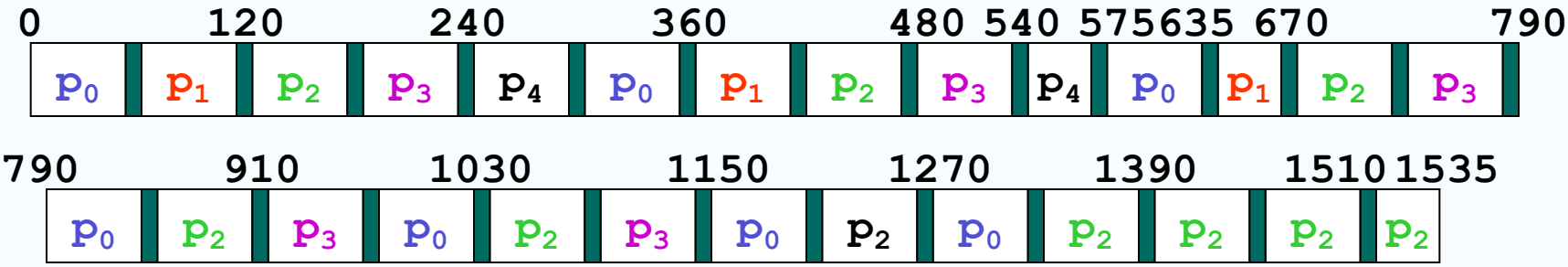
$$W_{avg} = (0+50+100+150+200)/5 = 500/5 = 100$$



RR with Overhead=10 (TQ=50)

i	$t(p_i)$	i	$t(p_i)$
0	350	2	475
1	125	3	250
		4	75

- Η επιβάρυνση είναι σημαντική
- Κάθε εναλλαγή κοστίζει κύκλους

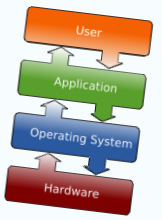


$T_{TRnd}(p_0) = 1320$
$T_{TRnd}(p_1) = 660$
$T_{TRnd}(p_2) = 1535$
$T_{TRnd}(p_3) = 1140$
$T_{TRnd}(p_4) = 565$

$W(p_0) = 0$
$W(p_1) = 60$
$W(p_2) = 120$
$W(p_3) = 180$
$W(p_4) = 240$

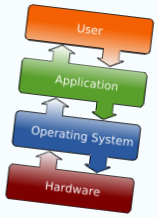
$$T_{TRnd\_avg} = (1320+660+1535+1140+565)/5 = 5220/5 = 1044$$

$$W_{avg} = (0+60+120+180+240)/5 = 600/5 = 120$$



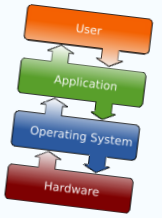
Χρόνος καταιγισμού

- Πρόβλεψη
- Εκτίμηση
- Υπολογισμός
- Παραδείγματα



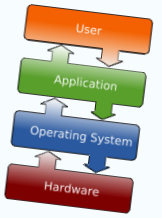
Πρόβλεψη του χρόνου καταιγισμού

- Στην πράξη, η διάρκεια του επόμενου καταιγισμού μιας διεργασίας στη CPU δεν είναι γνωστή
  - Ως συνέπεια, αλγόριθμοι όπως οι SJF, SRTF στην απλή τους μορφή δεν μπορούν να χρησιμοποιηθούν σε συστήματα
- Η διάρκεια καταιγισμού της CPU μιας διεργασίας μπορεί να υπολογιστεί ως συνάρτηση των προηγούμενων καταιγισμών της ίδιας διεργασίας
- Ο υπολογισμός απαιτεί την ανάλυση
  - Του κώδικα που θα εκτελεστεί ενώ θα λαμβάνεται η απόφαση για δρομολόγηση
  - Γενικά η διαδικασία είναι δύσκολη στον χειρισμό (intractable)



Εκτίμηση του χρόνου καταιγισμού

- Η διάρκεια του επόμενου καταιγισμού της CPU υπολογίζεται από τις διάρκειες των προηγούμενων καταιγισμών
- Συχνά, οι πρόσφατοι καταιγισμοί είναι περισσότερο σημαντικοί από τους παλαιότερους
- Ο υπολογισμός προσθέτει επιπλέον επιβάρυνση στη διαδικασία απόφασης για τη δρομολόγηση
  - Απαιτείται χρόνος για τους υπολογισμούς
  - Απαιτείται μνήμη για την αποθήκευση των τιμών των πρόσφατων καταιγισμών

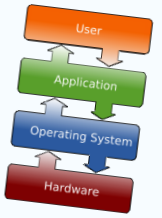


Υπολογισμός του χρόνου καταιγισμού

- Ο γενικός τύπος για την εκτίμηση της διάρκειας του επόμενου καταιγισμού της CPU από τις διάρκειες των προηγουμένων:

$$E_i = a * B_{i-1} + (1-a) * E_{i-1}$$

  - E_i εκτίμηση τη στιγμή *i*
  - B_{i-1} πραγματική διάρκεια καταιγισμού τη στιγμή *i-1*
  - a συντελεστής που εξισορροπεί τη σπουδαιότητα των πρόσφατων και των όχι πρόσφατων καταιγισμών

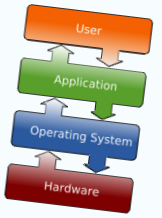


Παράδειγμα

- Τη χρονική στιγμή T_i γίνεται εκτίμηση της διάρκειας του επόμενου καταιγισμού της CPU που βασίζεται σε πληροφορίες που προέρχονται από τις διάρκειες των προηγουμένων, από τον τύπο:

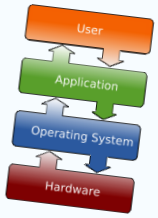
$$E_i = a * B_{i-1} + (1-a) * E_{i-1}$$

  - Πρέπει να επιλεγεί μια τιμή για το a (π.χ. 0.75)
  - Για την πρώτη τιμή B_0, γίνεται μια εικασία, ενώ στη συνέχεια χρησιμοποιείται η μετρηθείσα τιμή για τον προηγούμενο καταιγισμό
  - Οι μετρούμενες τιμές ίσως διαφέρουν σημαντικά από τις εκτιμώμενες



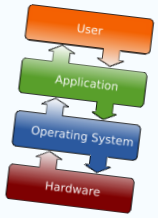
Παράδειγμα

Time	*Estimate*	*Previous Burst*
T_0	0.75 * 10 + 0.25 * 0 = 7.5	---



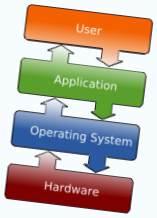
Παράδειγμα (συν.)

Time		*Estimate*			*Previous Burst*
T_0	0.75 * 10	+	0.25 * 0	= 7.5	6
T_1	0.75 * 6	+	0.25 * 7.5	= 6.375	



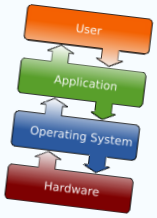
Παράδειγμα (συν.)

Time	*Estimate*	*Previous Burst*
T_0	0.75 * 10 + 0.25 * 0 = 7.5	6
T_1	0.75 * 6 + 0.25 * 7.5 = 6.375	3
T_2	0.75 * 3 + 0.25 * 6.375 = 3.85	



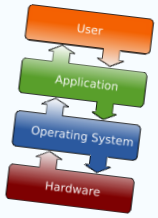
Παράδειγμα (συν.)

Time	Estimate			Previous Burst
T_0	0.75 * 10	+ 0.25 * 0	= 7.5	6
T_1	0.75 * 6	+ 0.25 * 7.5	= 6.375	3
T_2	0.75 * 3	+ 0.25 * 6.375	= 3.85	7
T_3	0.75 * 7	+ 0.25 * 3.85	= 6.2	



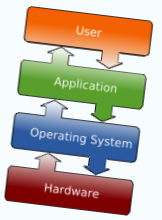
Παράδειγμα (συν.)

Time	*Estimate*				*Previous Burst*
T_0	0.75 * 10	+	0.25 * 0	= 7.5	6
T_1	0.75 * 6	+	0.25 * 7.5	= 6.375	3
T_2	0.75 * 3	+	0.25 * 6.375	= 3.85	7
T_3	0.75 * 7	+	0.25 * 3.85	= 6.2	12
T_4	0.75 * 12	+	0.25 * 6.2	= 9.55	

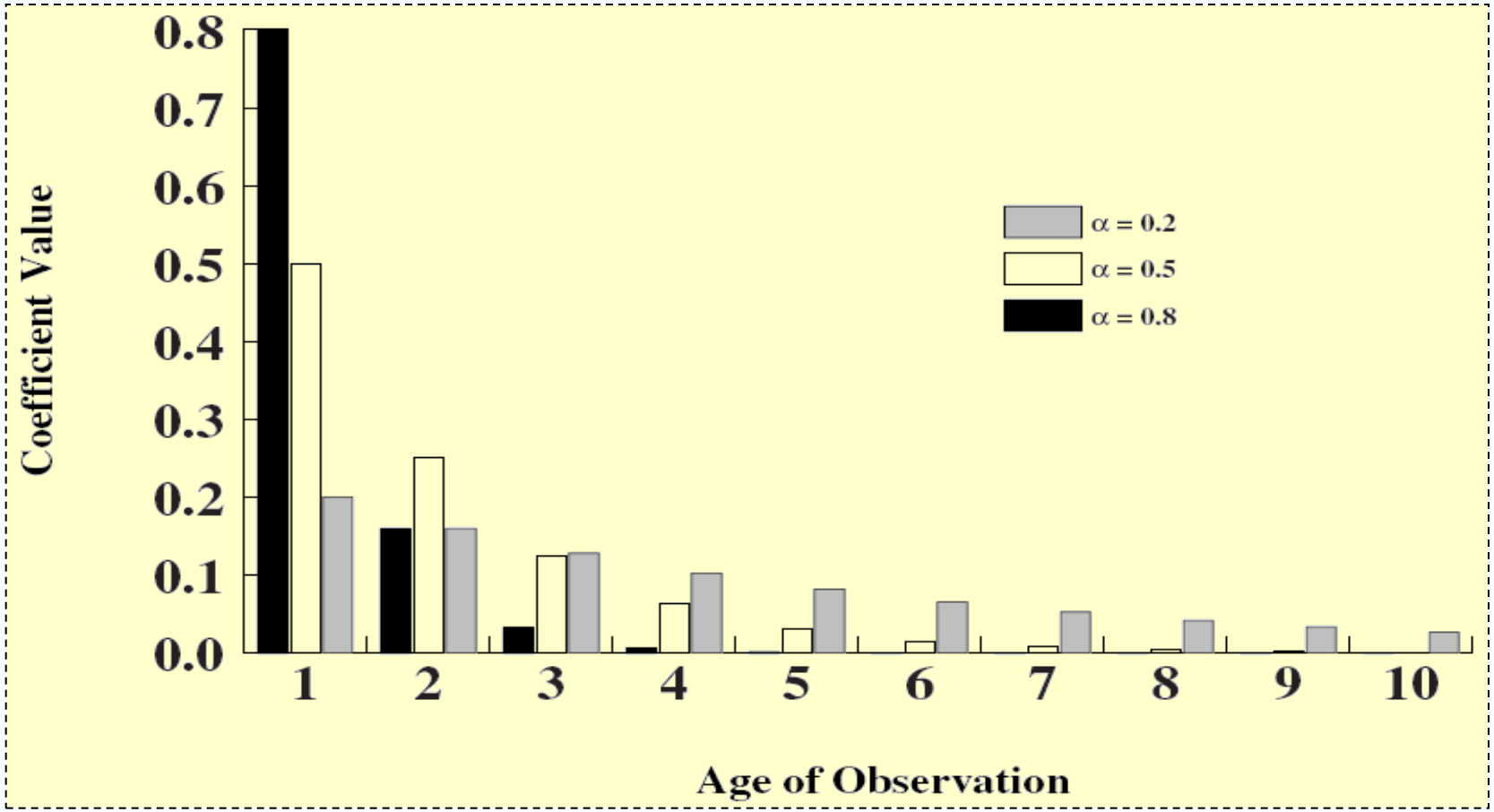


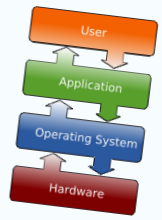
Υπολογισμός της διάρκειας καταιγισμού : σχόλια

- Ο υπολογισμός της διάρκειας καταιγισμού σύμφωνα με τα προηγούμενα βασίζεται στις εξής παραδοχές :
  - Τα πλέον πρόσφατα στιγμιότυπα είναι καταλληλότερα για να εκφράσουν τη μελλοντική συμπεριφορά, επομένως είναι επιθυμητό να δίνεται σε αυτά μεγαλύτερο βάρος
- Η χρήση του εκθετικού υπολογισμού του μέσου όρου της διάρκειας των πιο πρόσφατων καταιγισμών είναι μια κοινή τεχνική για την πρόβλεψη της μελλοντικής τιμής που προσεγγίζει καλύτερα την πραγματικότητα
- Συνήθως χρησιμοποιείται μια σταθερή τιμή για το *a*, ανεξάρτητη από το πλήθος των παλαιότερων παρατηρήσεων. Έτσι λαμβάνονται υπόψη όλες οι παλιές τιμές αλλά αυτές με τη μεγαλύτερη απόσταση από την τρέχουσα έχουν το μικρότερο βάρος
- Ο εκθετικός μέσος όρος παρακολουθεί τις αλλαγές στη συμπεριφορά της διεργασίας ταχύτερα από τον απλό μέσον όρο



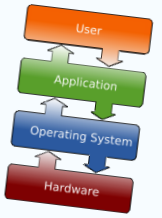
Εκθετικοί συντελεστές σταθεροποίησης





Δρομολόγηση με ουρές πολλαπλών επιπέδων

- Η ουρά των έτοιμων διεργασιών διαμοιράζεται σε αρκετές ξεχωριστές ουρές
  - Οι ουρές έχουν διαφορετικές προτεραιότητες
  - Οι ουρές χρησιμοποιούν διαφορετικούς αλγορίθμους δρομολόγησης
- Οι διεργασίες εκχωρούνται μόνιμα σε μια ουρά
  - Οι διεργασίες αλληλεπίδρασης ανήκουν στις ουρές υψηλής προτεραιότητας ενώ οι διεργασίες που εκτελούνται σωρηδόν (batch) στις ουρές χαμηλής προτεραιότητας

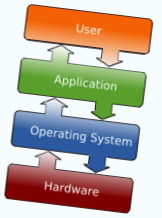


Δρομολόγηση με ανάδραση

- Δρομολόγηση με προεκχώρηση και δυναμικό μηχανισμό προτεραιότητας
- Διαφορετικές ουρές των έτοιμων διεργασιών με φθίνουσες προτεραιότητες:

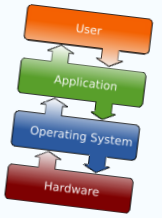
$$P(RQ0) > P(RQ1) > \ldots > P(RQn)$$

- Σε κάθε ουρά (εκτός από τη χαμηλότερης προτεραιότητας ουρά) χρησιμοποιείται ένας απλός μηχανισμός, τύπου FCFS.
- Στη χαμηλότερης προτεραιότητας ουρά μια διεργασία δεν μπορεί να μετακινηθεί χαμηλότερα αλλά επιστρέφει στην ουρά επανειλημμένα, μέχρι να ολοκληρωθεί η εκτέλεσή της (εξυπηρέτηση Round Robin)

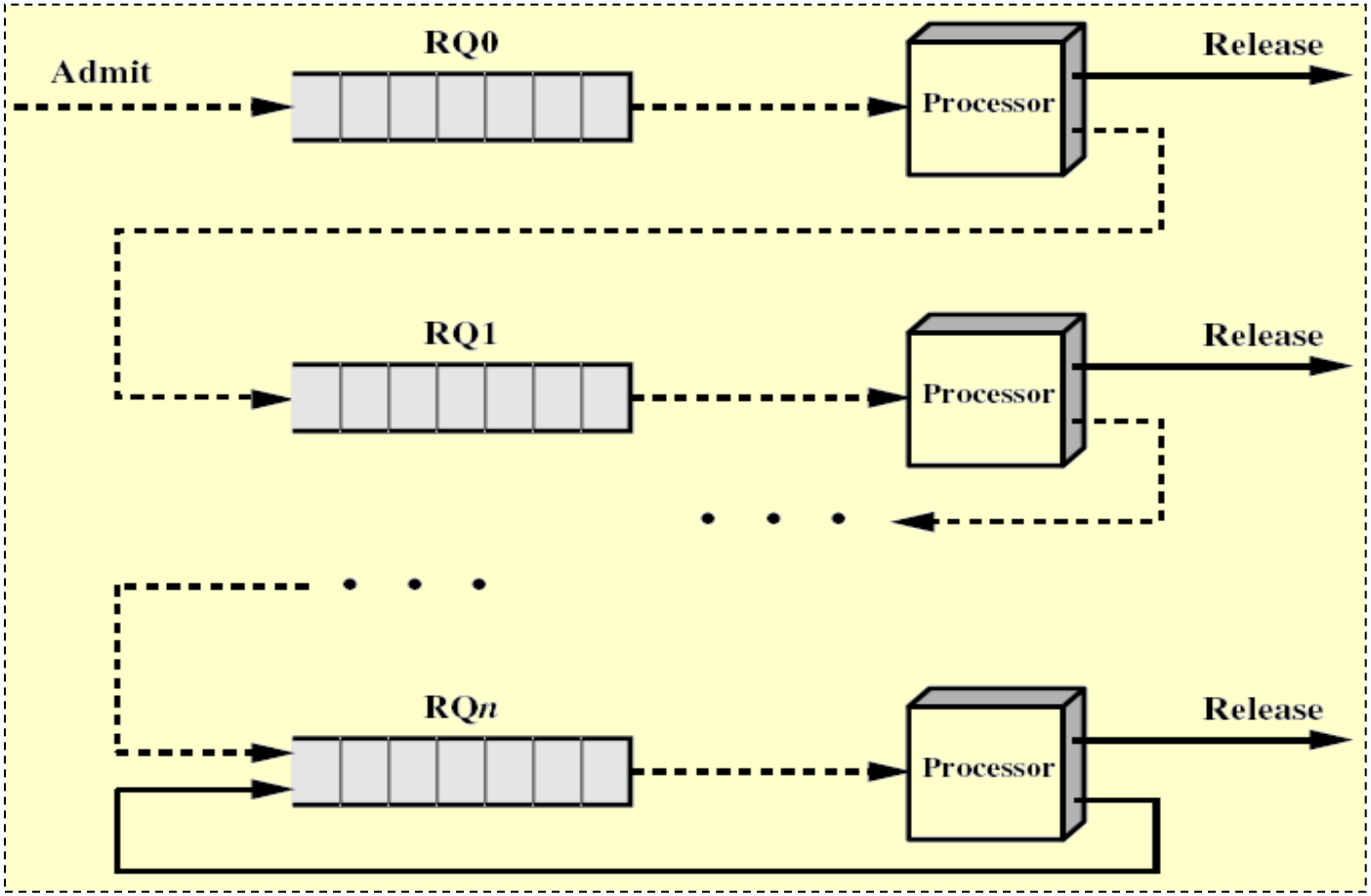


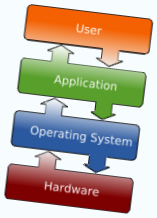
Παράδειγμα

- Μια νέα διεργασία τοποθετείται στην **RQ0**
- Όταν φθάσει στο quantum χρόνου, τοποθετείται στην **RQ1**. Όταν αυτό ξανασυμβεί τοποθετείται στην **RQ2**... μέχρι να φθάσει στην **RQn**. Οι προοριζόμενες για διεργασίες Ε/Ε θα παραμένουν σε υψηλότερης προτεραιότητας ουρές. Οι προοριζόμενες για CPU διεργασίες θα κατευθύνονται χαμηλότερα.
- Ο διεκπεραιωτής επιλέγει μια διεργασία για εκτέλεση στην **RQi** μόνον αν **RQi-1** έως **RQ0** είναι κενές
- Και εδώ οι μεγάλες διεργασίες μπορούν να παρουσιάσουν παρατεταμένη στέρηση



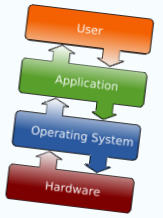
Δρομολόγηση με ανάδραση





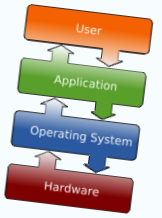
Δρομολόγηση με ανάδραση με ουρές πολλαπλών επιπέδων

- Η ουρά των έτοιμων διεργασιών διαμοιράζεται σε αρκετές ξεχωριστές ουρές
  - Οι ουρές έχουν διαφορετικές προτεραιότητες
  - Οι ουρές χρησιμοποιούν διαφορετικούς αλγορίθμους δρομολόγησης
- Υπάρχει ένας αλγόριθμος για τη δρομολόγηση των ουρών
  - Συνήθως σε κάθε ουρά εκχωρείται ένα καθορισμένο ποσοστό του χρόνου της CPU
- Οι διεργασίες μπορούν να μετακινούνται από ουρά σε ουρά
  - Παρέχεται ευελιξία στις διεργασίες των οποίων τα χαρακτηριστικά μεταβάλλονται κατά τη διάρκεια ζωής τους
  - Προλαμβάνεται η πιθανότητα παρατεταμένης στέρησης

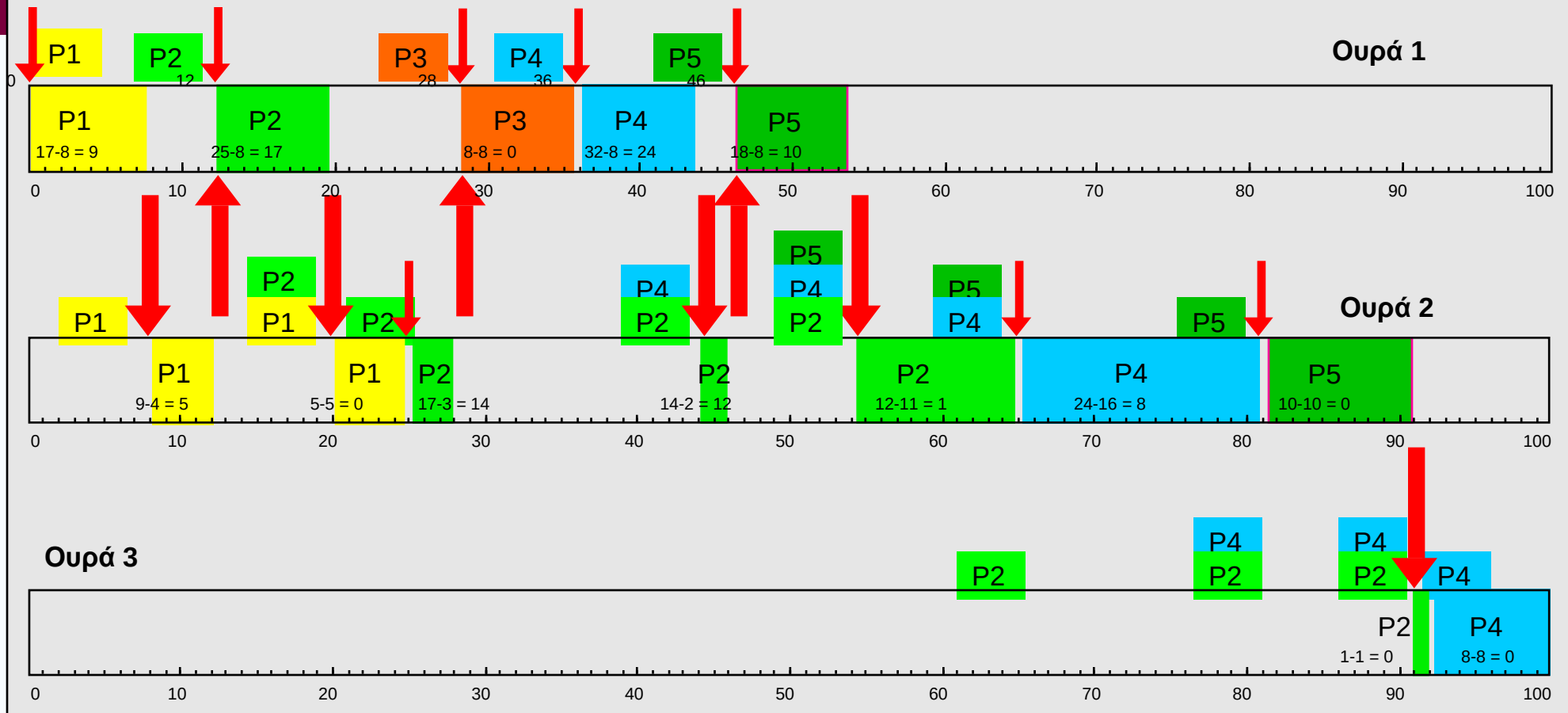


Παράδειγμα: Ανάδραση

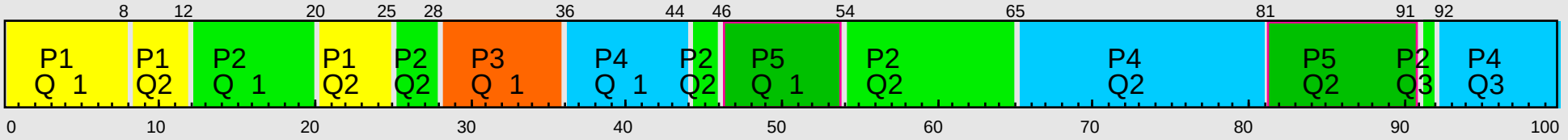
- Τρεις ουρές:
  - Q_1 – quantum 8 msecs – υψηλή προτεραιότητα
  - Q_2 – quantum 16 msecs – μεσαία προτεραιότητα
  - Q_3 – FCFS – χαμηλή προτεραιότητα
- Δρομολόγηση
  - Οι νέες διεργασίες εισέρχονται στην ουρά Q_1
  - Οι διεργασίες με μικρούς χρόνους καταιγισμού έχουν υψηλή προτεραιότητα
  - Οι διεργασίες με μεγάλους χρόνους καταιγισμού ολοκληρώνονται στην ουρά χαμηλής προτεραιότητας



Παράδειγμα Πολύ-επίπεδων Ουρών

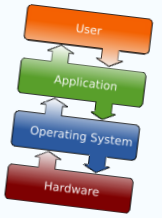


Gantt Chart



Σύγκριση αλγορίθμων δρομολόγησης

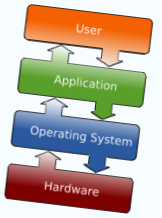
- Εξαρτάται από:
  - Το φορτίο του συστήματος (ισχυρά μεταβαλλόμενο)
  - Την υποστήριξη υλικού για το δρομολογητή
  - Το σχετικό βάρος των κριτηρίων απόδοσης (χρόνος απόκρισης, χρήση της, ρυθμο-απόδοση...)
  - Τη μέθοδο αποτίμησης που χρησιμοποιείται (καθεμιά έχει τους περιορισμούς της...)



Άσκηση – 9.1

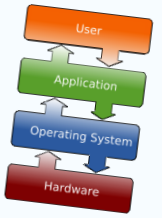
- Θεωρείστε το ακόλουθο σύνολο διεργασιών, στο οποίο το μήκος των CPU burst times είναι σε milliseconds
- Υποθέστε ότι οι διεργασίες έχουν φθάσει με τη σειρά P1, P2, P3, P4, και P5, όλες τη χρονική στιγμή 0

process	Burst time	priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2



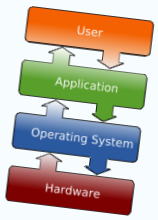
Άσκηση – 9.1 (συν.)

- Σχεδιάστε το διάγραμμα εκτέλεσης για καθεμία από τις πολιτικές δρομολόγησης : FCFS, SJF, RR (TQ: 1)
- Βρείτε για κάθε διεργασία τον turnaround time και τον waiting time καθώς και τις μέσες τιμές τους.



Λύση άσκησης 9.1

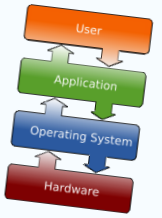
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
FCFS:	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P2	P3	P3	P4	P5	P5	P5	P5	P5
SJF:	P2	P4	P3	P3	P5	P5	P5	P5	P5	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1
RR:	P1	P2	P3	P4	P5	P1	P3	P5	P1	P5	P1	P5	P1	P5	P1	P1	P1	P1	P1



Λύση άσκησης 9.1 (συν.)

Turnaround time = waiting time + execution time

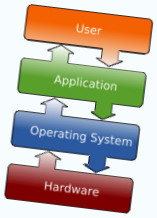
	FCFS	SJF	RR
P1	10	19	19
P2	11	1	2
P3	13	4	7
P4	14	2	4
P5	19	9	14
Average	13.4	7.0	9.2



Λύση άσκησης 9.1 (συν.)

Waiting time = turnaround time - burst time

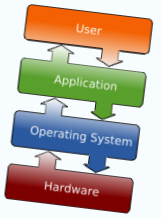
	FCFS	SJF	RR
P1	0	9	9
P2	10	0	1
P3	11	2	5
P4	13	1	3
P5	14	4	9
Average	9.6	3.2	5.4



Άσκηση – 9.2

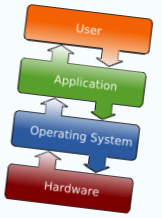
- Θεωρείστε τις παρακάτω δύο διεργασίες. Κάθε διεργασία εκτελεί ένα CPU burst μετά ένα I/O burst, άλλο ένα CPU burst, άλλο ένα I/O burst και τερματίζει με ένα CPU burst. Τα μήκη των CPU burst και I/O burst χρόνων σε milliseconds δίνονται στον παρακάτω πίνακα:

Process	CPU-burst1	I/O-burst1	CPU-burst2	I/O-burst2	CPU-burst3	Arrival
P1	3	2	2	3	2	0
P2	2	2	3	3	2	1



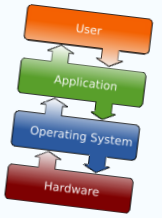
Άσκηση – 9.2 (συν.)

1. Σχεδιάστε δύο διαγράμματα που θα δείχνουν την εκτέλεση των διεργασιών χρησιμοποιώντας τον αλγόριθμο δρομολόγησης round robin (RR) με quantum = 1 και quantum = 3. Αν μια ολοκλήρωση Ε/Ε και μια CPU timeout των δύο διεργασιών συμβαίνει την ίδια στιγμή εξυπηρετείται πρώτα η ολοκλήρωση Ε/Ε.
2. Ποιος είναι ο waiting time κάθε διεργασίας σε κάθε περίπτωση; Ποια είναι η μέση τιμή;
3. Ποιος είναι ο turn-around time κάθε διεργασίας σε κάθε περίπτωση; Ποια είναι η μέση τιμή;
4. Τι θα συμβεί αν αντιστοιχηθεί προτεραιότητα σε κάθε διεργασία και η προτεραιότητα της P2 είναι υψηλότερη της P1. Να εξετάσετε και τις δύο περιπτώσεις (quantum=1 και quantum=3)



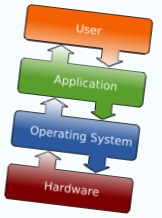
Λύση άσκησης 9.2

```
xxx   Η διεργασία εκτελείται
---   Η διεργασία είναι έτοιμη
III   Η διεργασία αναστέλλεται στη συσκευή Ε/Ε
iii   Η CPU είναι άεργη

Time  0 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
q=1
P1:   xxx---xxx---xxxIIIIIIxxx---xxxIIIIIIIIIxxx---xxxTerm
P2:      xxx---xxxIIIIIIxxx---xxx---xxxIIIIIIIIIxxx---xxxTerm
CPU:                  iii                iiiiii

q=3
P1:   xxxxxxxxxIIIIIIxxxxxxIIIIIIIIIxxxxxxTerm
P2:      ------xxxxxxIIIIIIxxxxxxxxxIIIIIIIIIxxxxxxTerm
```



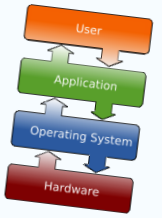
Λύση άσκησης 9.2 (συν.)

```
Waiting times:
   (i)  q = 1:  P1:4;  P2:4;  Mean value: 4
   (ii) q = 3:  P1:0;  P2:2;  Mean value: 1

Turn around times:
   (i)  q = 1:  P1:16;  P2:16;  Mean value:16
   (ii) q = 3:  P1:12;  P2:14;  Mean value:13
```



Λύση άσκησης 9.2 (συν.)

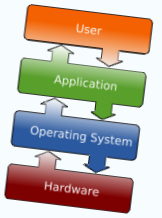
```
xxx   Η διεργασία εκτελείται
---   Η διεργασία είναι έτοιμη
III   Η διεργασία αναστέλλεται στη συσκευή Ε/Ε
iii   Η CPU είναι άεργη

Time  0 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
q=1
P1:   xxx------xxxxxxIIIIII---xxxxxxIIIIIIIIIxxxxxxTerm
P2:      xxxxxxIIIIIIxxxxxxxxxIIIIIIIIIxxxxxxTerm

q=3
P1:   xxx------xxxxxxIIIIII---xxxxxxIIIIIIIIIxxxxxxTerm
P2:      xxxxxxIIIIIIxxxxxxxxxIIIIIIIIIxxxxxxTerm
```



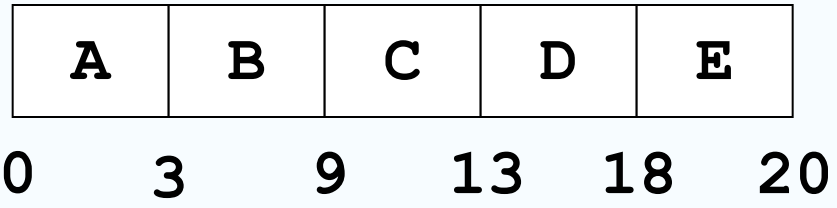
Άσκηση – 9.3

- Να σχεδιάσετε το διάγραμμα Gantt για καθένα από τους παρακάτω αλγορίθμους δρομολόγησης:
  - FCFS,
  - RR με quantum=2 χρονικές μονάδες
  - Preemptive SJF
  - Non-preemptive SJF
- Να υπολογίσετε τον μέσο χρόνο επιστροφής για τον κάθε αλγόριθμο.

Process	Arrival time	Burst time	Priority
A	0	3	2
B	2	6	4
C	4	4	1
D	6	5	5
E	8	2	3

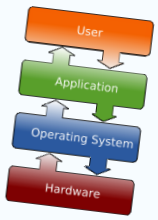


FCFS : Gantt Chart

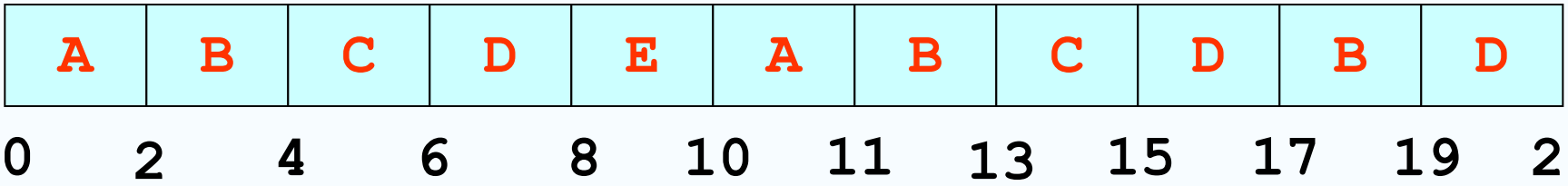


$$\text{Ave. TAT} = \frac{3 + 9 + 13 + 18 + 20}{5} = \frac{63}{5}$$

$$= 12.6 \text{ time units}$$

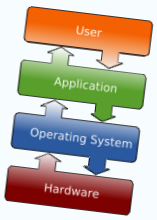


RR :
Χρονικό κβάντο = 2 χρονικές μονάδες

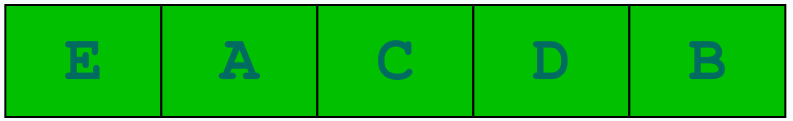


$$\text{Ave. TAT} = \frac{11 + 19 + 15 + 20 + 10}{5} = \frac{75}{5}$$

$$= 15 \text{ time units}$$

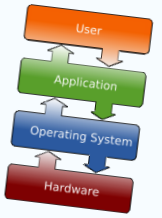


Nonpreemptive SJF Gantt Chart :

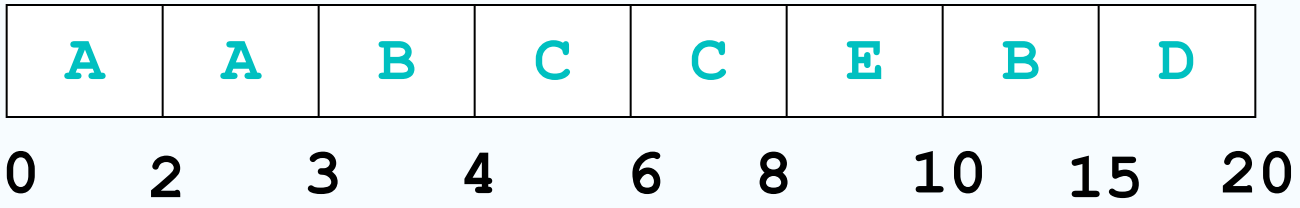


$$\text{Ave. TAT} = \frac{2 + 5 + 9 + 14 + 20}{5} = \frac{50}{5}$$

$$= 10 \text{ time units}$$



Preemptive SJF Gantt Chart :



$$\text{Ave.TAT} = \frac{(3 - 0) + (15 - 2) + (8 - 4) + (20 - 6) + (10 - 8)}{5}$$

$$= 7.2 \text{ time units}$$