ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Χρονοπρογραμματισμός
(ή Χρονοδρομολόγηση ή Δρομολόγηση)

Υλικό από:
Tanenbaum, *Modern Operating Systems,Structured Computer Organization*
Stallings, *Operating Systems: Internals and Design Principles*.
Silberschatz, Galvin and Gange, *Operating Systems Concepts.*
Deitel, Deitel and Choffnes, Operating Systems
Λειτουργικά Συστήματα, Γ.Α. Παπαδόπουλος, Πανεπιστήμιο Κύπρου
Λειτουργικά Συστήματα, Κ. Διαμαντάρας, ΤΕΙΘ
Systems Programming in C, A.D. Marshal, University of Cardiff

Σύνθεση
Κ.Γ. Μαργαρίτης, Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

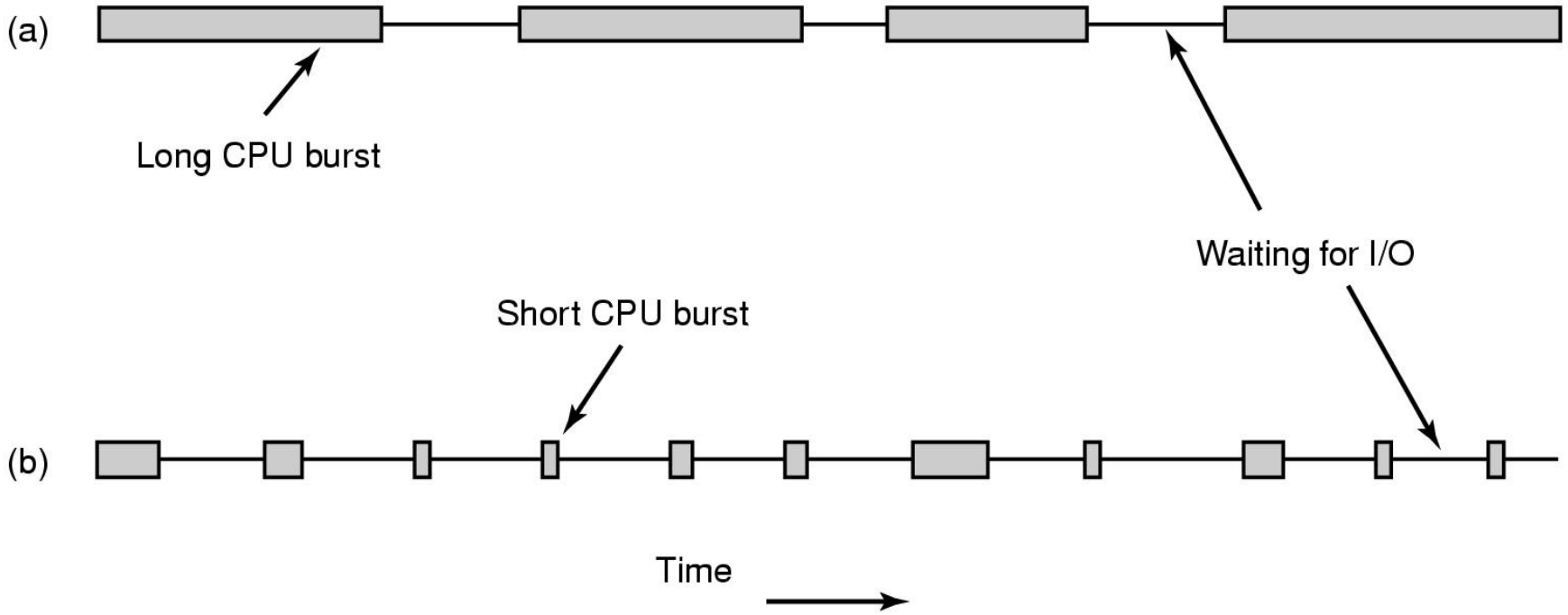
Χρονοπρογραμματισμός
(ή Χρονοδρομολόγηση ή Δρομολόγηση)

Γενικά
Συστήματα Δέσμης
Αλληλεπιδραστικά Συστήματα

Κατηγορίες χρονοπρογραμματισμού

- Για συστήματα δέσμης
- Για αλληλεπιδραστικά (διαδραστικά) συστήματα πολλών χρηστών
- Για συστήματα πραγματικού χρόνου
- Μικρότερη σημασία σε προσωπικούς υπολογιστές
- Μεγαλύτερη σημασία σε δικτυωμένους σταθμούς εργασίας
- Οι διακομιστές αντιμετωπίζονται ανάλογα με την κύρια εφαρμογή τους σαν απομακρυσμένα αλληλεπιδραστικά συστήματα ή συστήματα δέσμης.

Συμπεριφορά διεργασιών



Χρήση CPU και αναμονή για I/O.

(a) CPU-bound (εξαρτημένη από CPU) διεργασία: συχνοί, μεγάλοι καταιγισμοί CPU (CPU bursts), αραιοί, μικρές αναμονές Ε/Ε (I/O waits).

(b) I/O-bound (εξαρτημένη από Ε/Ε) διεργασία: συχνές, μεγάλες αναμονές Ε/Ε (I/O waits), αραιοί, μικροί καταιγισμοί CPU (CPU bursts).

CPU burst (καταιγισμός) – I/O wait (αναμονή)

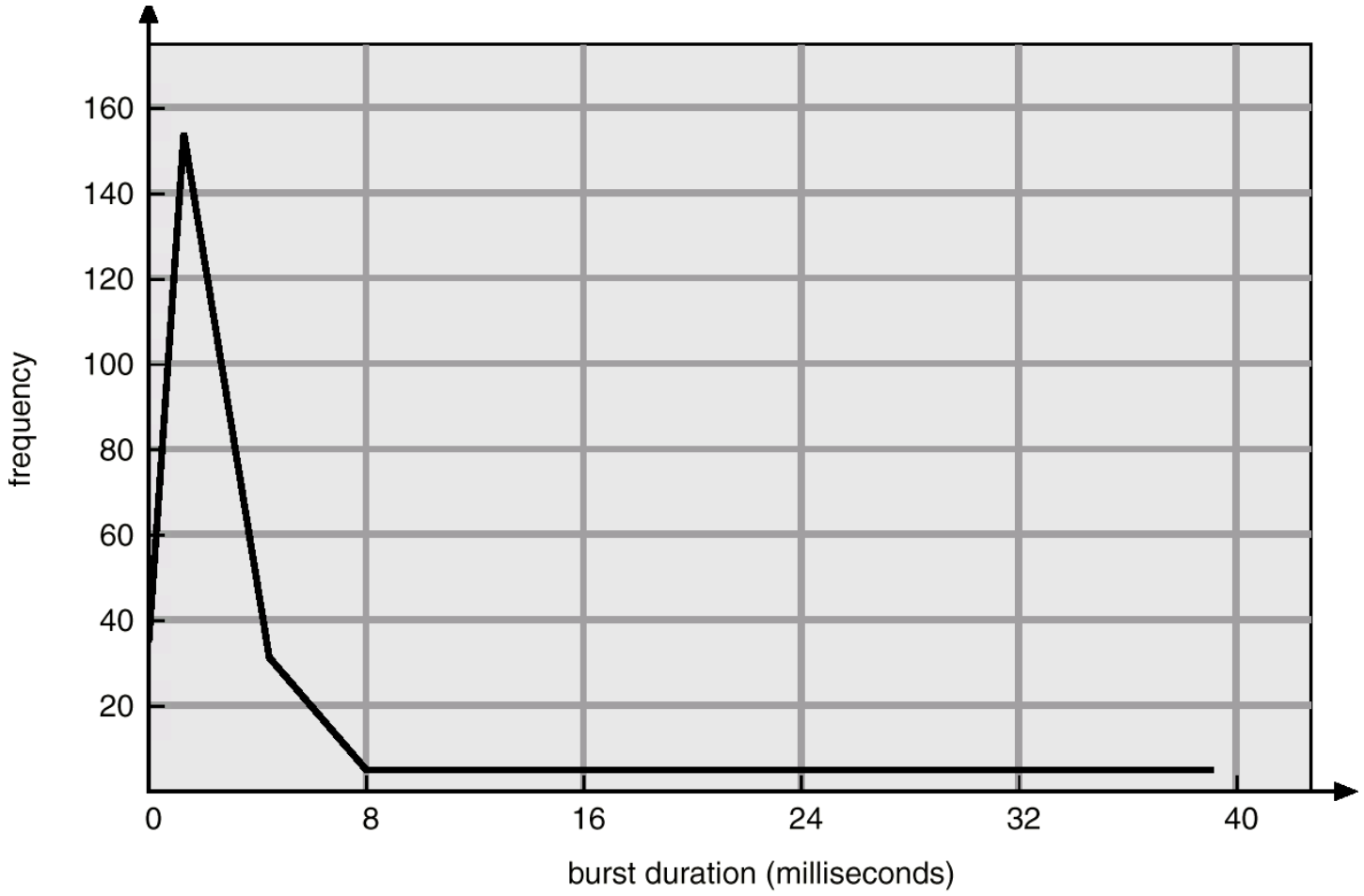
Χαρακτηρίζει την εκτέλεση μιας διεργασίας, που εναλλάσσεται μεταξύ των δραστηριοτήτων επεξεργαστή και Ε/Ε. Οι χρόνοι εξυπηρέτησης του επεξεργαστή είναι γενικά πολύ μικρότεροι των χρόνων για Ε/Ε.

Οι διεργασίες απαιτούν εναλλασσόμενη χρήση επεξεργαστή και Ε/Ε με επαναλαμβανόμενο τρόπο. Κάθε κύκλος αποτελείται από ένα CPU burst, διάρκειας συνήθως μερικών msecs, ακολουθούμενο από ένα I/O wait μεγαλύτερης διάρκειας (συνήθως).

Οι CPU-bound διεργασίες έχουν μεγαλύτερα CPU bursts από εκείνα των I/O-bound διεργασιών.

Όσο ταχύτεροι γίνονται οι επεξεργαστές σε σχέση με την Ε/Ε τόσο μειώνεται αναλογικά το μέγεθος του CPU burst σε σχέση με το I/O burst. Επομένως πρέπει να δίνεται προτεραιότητα στις I/O bound διεργασίες γιατί γρήγορα θα ανασταλούν.

Ενδεικτικό ιστόγραμμα συχνοτήτων των χρόνων καταιγισμού CPU



Πότε χρονοπρογραμματισμός CPU;

Ο χρονοπρογραμματισμός της CPU προσπαθεί να διαχειριστεί βέλτιστα τους καταιγισμού CPU και I/O, με βάση τα χαρακτηριστικά του συστήματος.

- Όταν τερματίζεται μια διεργασία
- Όταν αυτο-αναστέλλεται μια διεργασία
- Όταν δημιουργείται μια νέα διεργασία
- Όταν προκαλείται χρονοδιακοπή
- Όταν προκαλείται άλλου είδους διακοπή από το σύστημα

Μη-Προεκτόπιση (Non -Pre-emption): ο χρονοπρογραμματισμός εκτελείται μόνο για τερματισμό ή αναστολή.

Προεκτόπιση (Pre-emption): ο χρονοπρογραμματισμός εκτελείται σε όλες τις περιπτώσεις και κυρίως σε χρονοδιακοπές.

Υπάρχουν και υψηλότερα επίπεδα μακροπρόθεσμου χρονοπρογραμματισμού.

Μη- Προεκτόπιση ή μη- προεκχώρηση
Non- pre-emption

Μια διεργασία παραμένει στη CPU μέχρι να απελευθερώσει από μόνη της τη CPU. Πιθανοί λόγοι:

- Τερματισμός
- Εκτέλεση Ε/Ε
- Αναστολή λόγω πρόκλησης διακοπής
- Αναστολή λόγω κλήσης συστήματος

Προκαλεί μεγάλους χρόνους αναμονής και απόκρισης.

Ενδεχόμενο παρατεταμένης αναμονής για κάποιες διεργασίες.

Απλή και εύκολη στην υλοποίηση.

Δεν είναι κατάλληλη για συστήματα multi-user.

Προεκτόπιση ή προεκχώρηση pre-emption

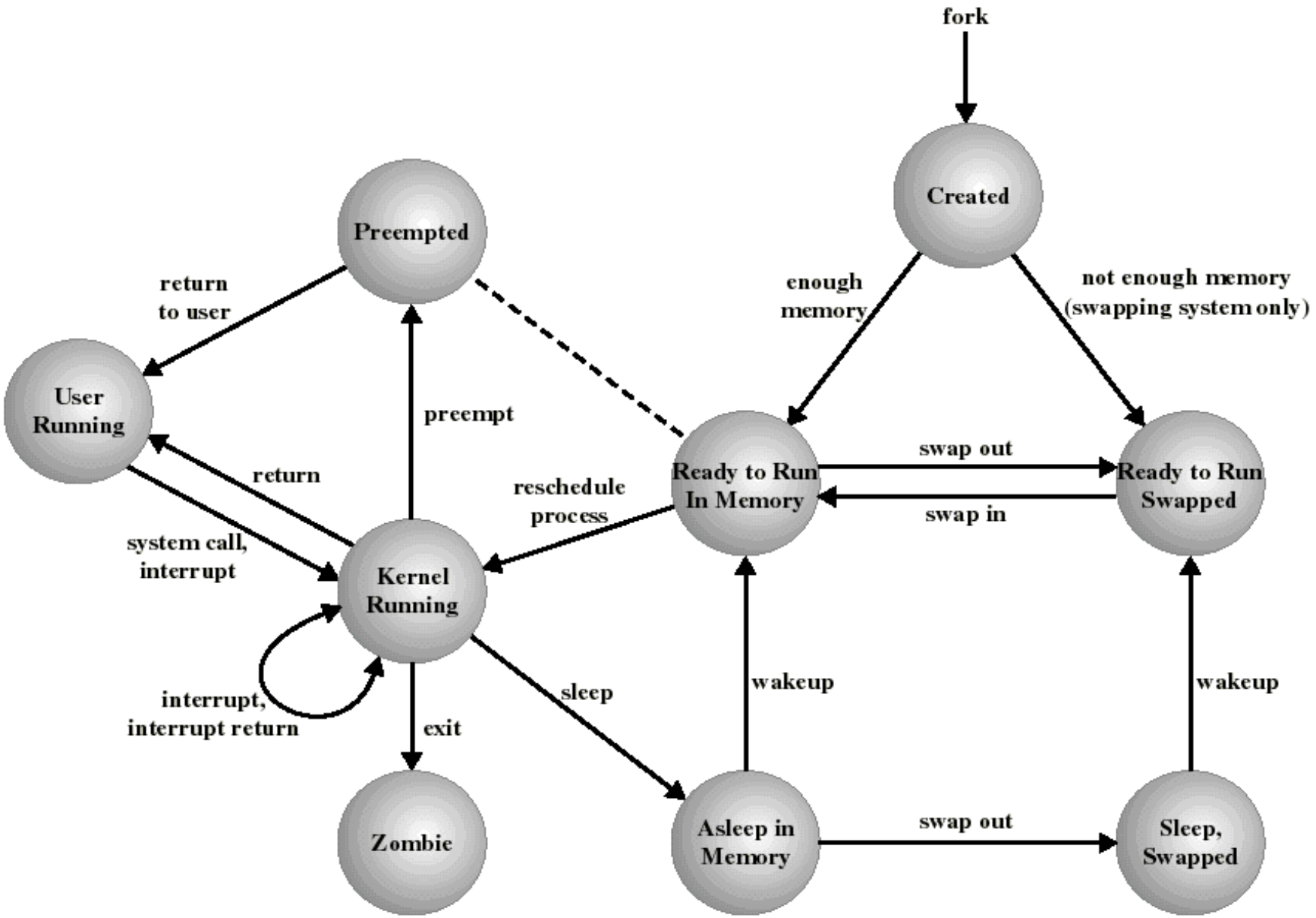
Η εκτέλεση μιας διεργασίας μπορεί να διακόπτεται από το λ.σ. οποιαδήποτε στιγμή. Πιθανές επιπλέον αιτίες :

- Η άφιξη μιας νέας διεργασίας με υψηλότερη προτεραιότητα
- Η πρόκληση μιας διακοπής από το σύστημα
- Η αλλαγή της κατάστασης μιας διεργασίας
- Η υπέρβαση ενός χρονικού ορίου (χρονοδιακοπή)

Αποτρέπει τη μονοπώληση της χρήσης της CPU από μία διεργασία

Μπορεί να οδηγήσει σε συνθήκες ανταγωνισμού, που επιλύονται χρησιμοποιώντας συγχρονισμό μεταξύ των διεργασιών

Διάγραμμα καταστάσεων με χρονοπρογραμματισμό



Γενικοί στόχοι

Δικαιοσύνη (Fairness): να εκχωρείται σε κάθε διεργασία ένα δίκαιο μερίδιο χρόνου της CPU

Επιβολή πολιτικής (Policy enforcement): να παρακολουθείται αν εφαρμόζεται η καθορισμένη πολιτική

Ισορροπία (Ballance): να διατηρούνται ενεργά όλα τα μέρη του συστήματος

Στόχοι: Συστήματα δέσμης

Χρησιμοποίηση CPU (CPU Utilization): να μεγιστοποιηθεί το ποσοστό ωφέλιμης χρήσης της CPU.

Διεκπεραιωτική Ικανότητα (Throughput Rate): να μεγιστοποιείται ο αριθμός των εξυπηρετούμενων εργασιών ανά ώρα

Χρόνος διεκπεραίωσης (Throughput Time): να ελαχιστοποιηθεί ο χρόνος που απαιτείται από την υποβολή μέχρι την περαίωση μιας εργασίας.

Στόχοι: Αλληλεπιδραστικά συστήματα

Χρόνος Απόκρισης (Response Time): να ελαχιστοποιηθεί το διάστημα μεταξύ της υποβολής αιτήματος χρήστη και εκκίνηση της εξυπηρέτησης από το σύστημα.

Τήρηση Αναλογιών (Proportionality): να τηρούνται αναλογίες στους χρόνους εξυπηρέτησης των χρηστών με βάση το 'βάρος' των αιτημάτων τους.

Στόχοι: Συστήματα πραγματικού χρόνου

Χρονικές Προθεσμίες (Deadlines): να τηρούνται συγκεκριμένες χρονικές προθεσμίες για την απόκριση, Είσοδο/Έξοδο και την επικοινωνία δεδομένων.

Προβλεψιμότητα (Predictability): σε ίδιες συνθήκες το σύστημα να φέρεται με τον ίδιο τρόπο, σε συνθήκες υπερφόρτωσης να τηρείται βαθμιαίος υποβιβασμός ποιότητας (όχι κατάρρευση).

Μετρικές απόδοσης (1)

Δικαιοσύνη (Fairness): συχνή χρήση της CPU από κάθε διεργασία (μέσος χρόνος χρήσης ανά διεργασία)

Χρησιμοποίηση (Utilization): το ποσοστό του χρόνου κατά το οποίο μια συσκευή χρησιμοποιείται (χρόνος χρήσης / συνολικός χρόνος)

Διεκπεραίωση (Throughput): ο αριθμός των εργασιών που ολοκληρώνονται σε μια χρονική περίοδο (εργασίες / second)

Χρόνος επιστροφής (Turnaround time): ο χρόνος εκτέλεσης μιας διεργασίας – από την αρχή μέχρι την ολοκλήρωση.

Χρόνος απόκρισης (Responce time): ο χρόνος από τότε που μια διεργασία υποβάλλεται μέχρι τη στιγμή που παράγεται η πρώτη απόκριση, όχι ο τερματισμός.

Μετρικές απόδοσης (2)

Χρόνος εξυπηρέτησης (Service time): ο χρόνος που απαιτείται από μια συσκευή για να διαχειριστεί μια αίτηση

Χρόνος αναμονής (Waiting time): Ο χρόνος σε μια ουρά αναμονής διεργασιών

Χρόνος παραμονής (Residence time): ο χρόνος που ξοδεύεται από μια αίτηση σε μια συσκευή

Residence time = Service time + Waiting time

Χρόνος / αριθμός θεματικών εναλλαγλων (Context Switch time / number): χρόνος που σπαταλάται στη θεματική εναλλαγή ή ο αριθμός των θεματικών εναλλαγών

Πολυπλοκότητα του αλγορίθμου δρομολόγησης: χρόνος που απαιτείται για την επιλογή της επόμενης διεργασίας

Τύποι χρονοπρογραμματιστών (1)

Μακροπρόθεσμος (Long-term) ή Αποδοχής (Admission)

για εργασίες batch

Μεσοπρόθεσμος (Medium-term) ή Μνήμης (Memory)

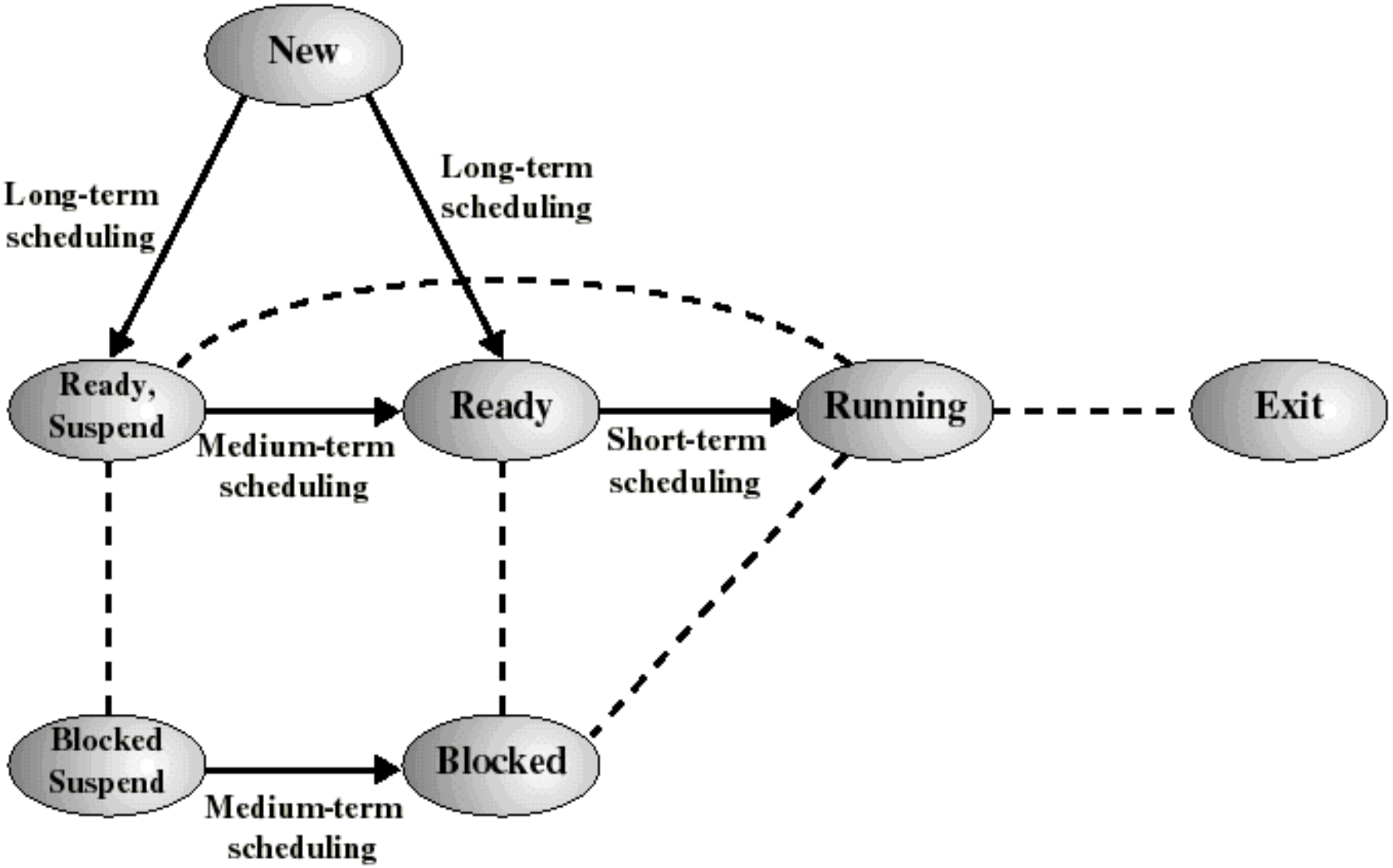
για διαχείριση μνήμης

Βραχυπρόθεσμος (Short-term) ή Επεξεργαστή (CPU)

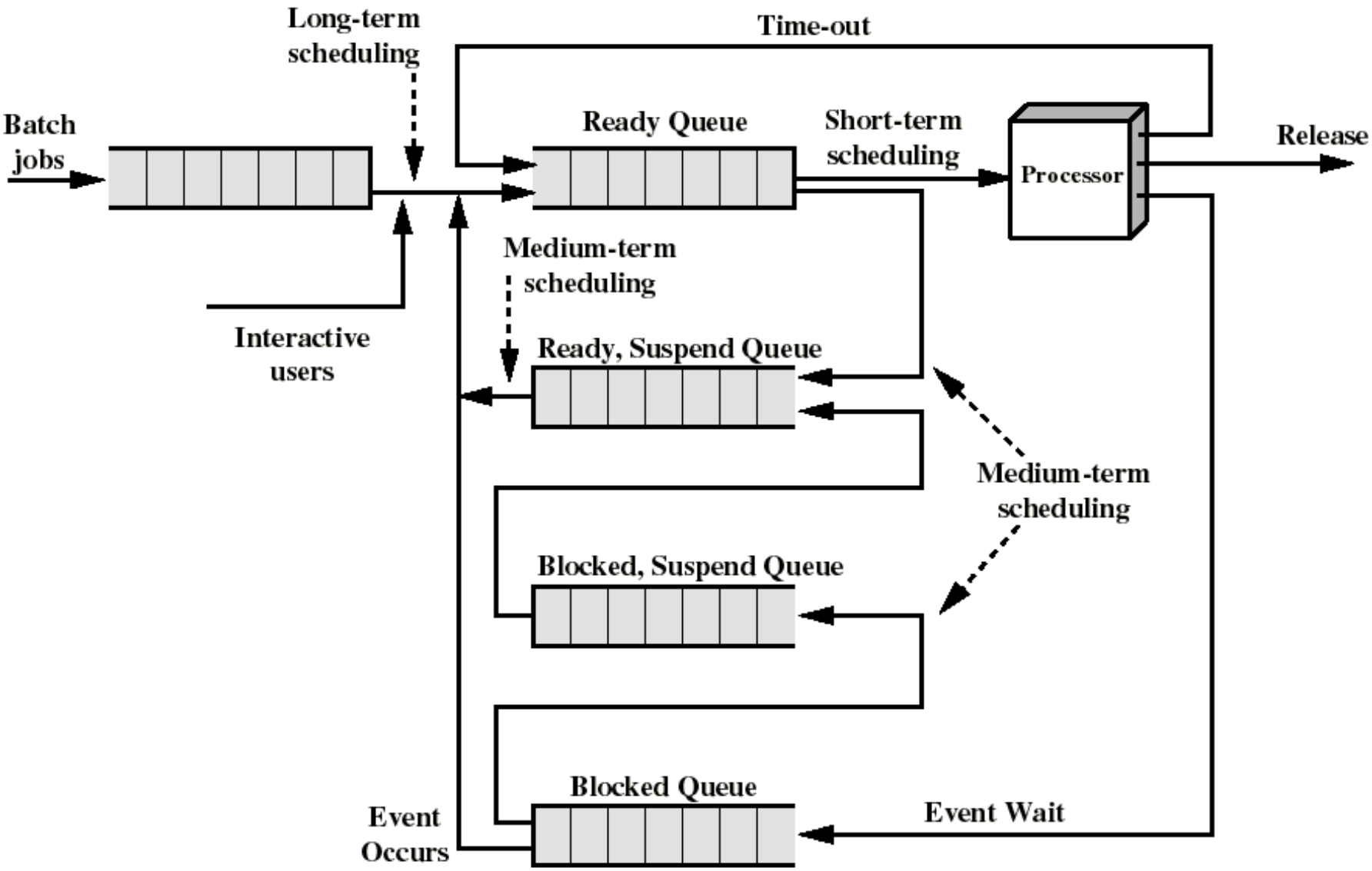
περιλαμβάνεται ο Διεκπεραιωτής (Dispatcher)

κυρίως δρομολόγηση

Τύποι χρονοπρογραμματιστών (2)



Διάγραμμα ουρών



Μακροπρόθεσμος

Καθορίζει ποιές διεργασίες θα γίνουν αποδεκτές στο σύστημα.

Σε συστήματα δέσμης κρατά ισορροπία μεταξύ CPU-bound και I/O-bound διεργασιών για βέλτιστη χρησιμοποίηση.

Σε διαδραστικά συστήματα δίνει προτεραιότητα στις μικρές εργασίες σε βάρος των μακροχρόνιων.

Σχετικά αραιή χρήση (όποτε υποβάλλεται εργασία, όχι διεργασίες που δημιουργούνται από άλλες διεργασίες, εκτός αν είναι batch) .

Μεσοπρόθεσμος (1)

Καθορίζει ποια διεργασία θα παραμείνει στην κύρια μνήμη (εν μέρει ή συνολικά) και ποια διεργασία θα μεταφερθεί στο δίσκο.

Στενή σχέση με διαχείριση εικονικής μνήμης.

Χρησιμοποιείται σχετικά συχνά (ανά sec) ώστε να μην αδικούνται οι εργασίες που βρίσκονται στο δίσκο. Απαιτείται ισορροπία μεταξύ κυκλοφορίας διεργασιών και Ε/Ε δεδομένων δίσκου (σε μεγάλα συστήματα διαφορετικοί δίσκοι συστήματος και χρηστών).

Καθορίζει το βαθμό πολυπρογραμματισμού του συστήματος (degree of multiprogramming), δηλαδή πόσες διεργασίες θα βρίσκονται ταυτόχρονα στη μνήμη.

Μεσοπρόθεσμος (2)

Μερικά κριτήρια:

Πόσο χρόνο βρίσκεται η διεργασία στη μνήμη (ή στο δίσκο)

Πόσο χρόνο CPU έλαβε πρόσφατα η διεργασία (CPU bound)

Πόσο χρόνο χρειάστηκε η διεργασία για E/E (I/O bound)

Τι προτεραιότητα έχει η διεργασία.

Βραχυπρόθεσμος (1)

Αποφασίζει ποια διεργασία πρόκειται να εκτελεστεί ως επόμενη.

Είναι το βασικό αντικείμενο αυτής της ενότητας

Ενεργοποιείται συνεχώς (msec) από γεγονότα που μπορεί να οδηγήσει στην επιλογή μιας άλλης διεργασίας για εκτέλεση :

- διακοπές ρολογιού (χρονοδιακοπή, timeout)
- διακοπές Ε/Ε
- κλήσεις του λ.σ. (πχ fork, sleep..)
- σήματα (signals)

Βραχυπρόθεσμος (2)

Κριτήρια προσανατολισμένα στο χρήστη

- Χρόνος αναμονής
- Χρόνος απόκρισης
- Χρόνος επιστροφής

Κριτήρια προσανατολισμένα στο σύστημα

- Χρησιμοποίηση επεξεργαστή
- Δικαιοσύνη
- Ρυθμός διεκπεραίωσης

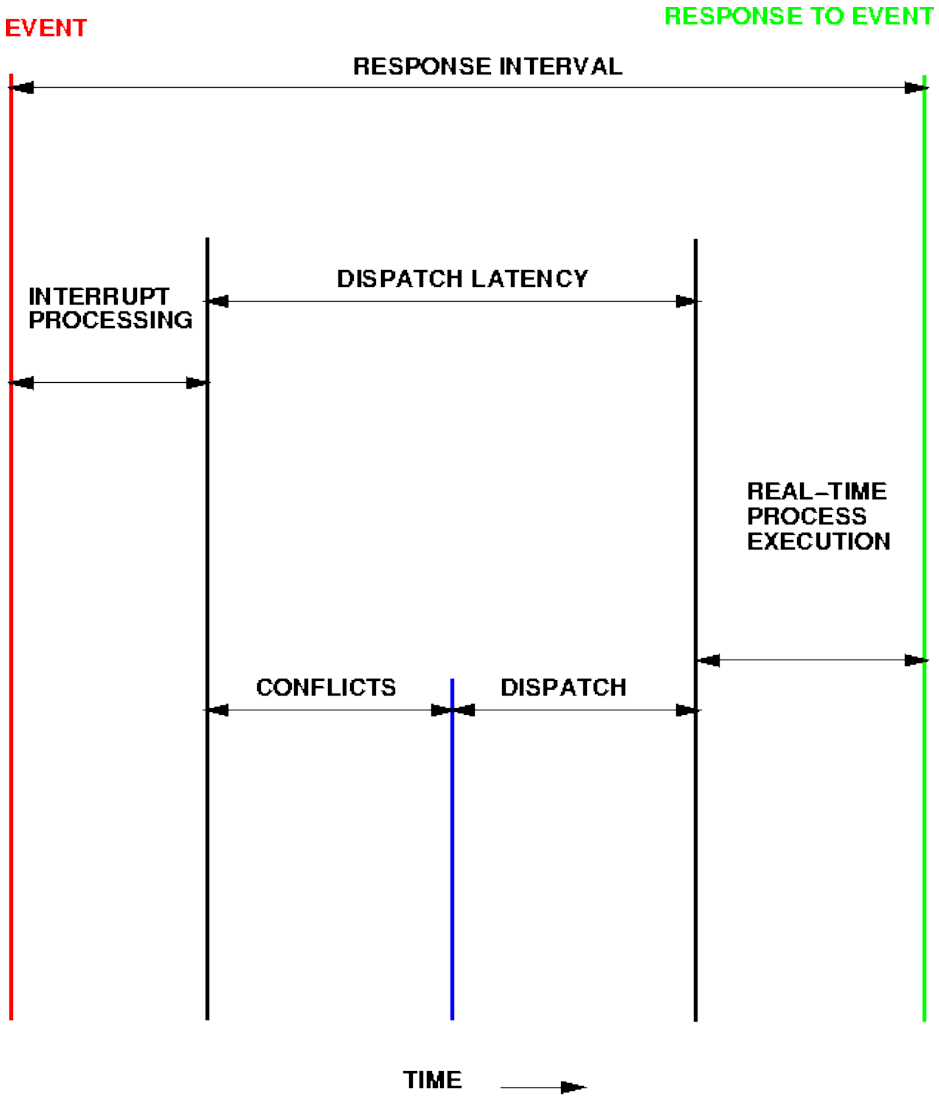
Είναι από μόνος του μια διεργασία, άρα χρησιμοποιεί τους πόρους του συστήματος (CPU και μνήμη). Δεν θα πρέπει να καταναλώνει σημαντικό χρόνο της CPU, διαφορετικά η συνολική επιβάρυνση θα είναι μεγάλη.

Βραχυπρόθεσμος (3)

Καθυστέρηση χρονοπρογραμματισμού

Dispatch latency

Ο χρόνος που χρειάζεται ο διεκπεραιωτής για να σταματήσει μια διεργασία και να αρχίσει την εκτέλεση μιας άλλης. Περιλαμβάνει την επεξεργασία διακοπών, την λήψη απόφασης, τη θεματική εναλλαγή, και την μεταφορά ελέγχου στην νέα διεργασία.



Προτεραιότητες

Έμμεσα: μέσω της πολιτικής του συστήματος ευνοούνται κάποιες διεργασίες με βάση πχ το προβλεπόμενο χρόνο εκτέλεσης ή αν είναι διεργασία πυρήνα, υπηρεσίας ή χρήστη.

Άμεσα: κάθε διεργασία λαμβάνει ρητά ένα αριθμό που καθορίζει το επίπεδο προτεραιότητας . Το λ.σ. υποστηρίζει πολλαπλές ουρές έτοιμων διεργασιών όπου κάθε ουρά αντιπροσωπεύει ένα επίπεδο προτεραιότητας.

Ο χρονοπρογραμματιστής επιλέγει πάντοτε μια διεργασία υψηλότερης προτεραιότητας έναντι μιας με μικρότερη προτεραιότητα.

Η χαμηλή προτεραιότητα μπορεί να υποφέρει από παρατεταμένη στέρηση (μεγάλος χρόνος αναμονής).

Ο χρονοπρογραμματιστής μπορεί να να αλλάζει την προτεραιότητα μιας διεργασίας, ανάλογα με το διάστημα που βρίσκεται σε αναμονή ή με το ιστορικό εκτέλεσής της.

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Χρονοπρογραμματισμός
(ή Χρονοδρομολόγηση ή Δρομολόγηση)

Γενικά
Συστήματα Δέσμης
Αλληλεπιδραστικά Συστήματα

Αλγόριθμοι: Συστήματα δέσμης

- Σειρά άφιξης (First Come First Served) FCFS
- Μικρότερη διάρκεια (Shortest Job First) Χωρίς προεκτόπιση (Non pre-emptive) SJFN
- Μικρότερη διάρκεια (Shortest Job First) Με προεκτόπιση (Pre-emptive) SJFP
- Μικρότερη διάρκεια που απομένει (Shortest Remaining Time Next) SRTN

First Come First Served (FCFS) (1)

Οι διεργασίες εξυπηρετούνται με τη σειρά που φθάνουν.

Η επιλογή της επόμενης διεργασίας είναι ταχύτατη και ανεξάρτητη από το πλήθος των διεργασιών στην ουρά των έτοιμων διεργασιών.

Αν φθάσουν πολλές διεργασίες την ίδια χρονική στιγμή, η διεργασία που θα εξυπηρετηθεί επιλέγεται τυχαία.

Χωρίς προεκτόπιση: η διεργασία εκτελείται μέχρι να ανασταλεί από μόνη της.

Πολύ απλός αλγόριθμος, υλοποιείται εύκολα. Χρησιμοποιεί μια ουρά FIFO (First In First Out).

First Come First Served (FCFS) (2)

Ευνοούνται οι CPU bound διεργασίες. Μια διεργασία που δεν χρησιμοποιεί Ε/Ε μπορεί να μονοπωλεί τη χρήση του επεξεργαστή.

Μια I/O bound διεργασία θα περιμένει μέχρι να ολοκληρωθεί η CPU bound διεργασία. Θα περιμένει ακόμη και αν οι λειτουργίες Ε/Ε έχουν ολοκληρωθεί.

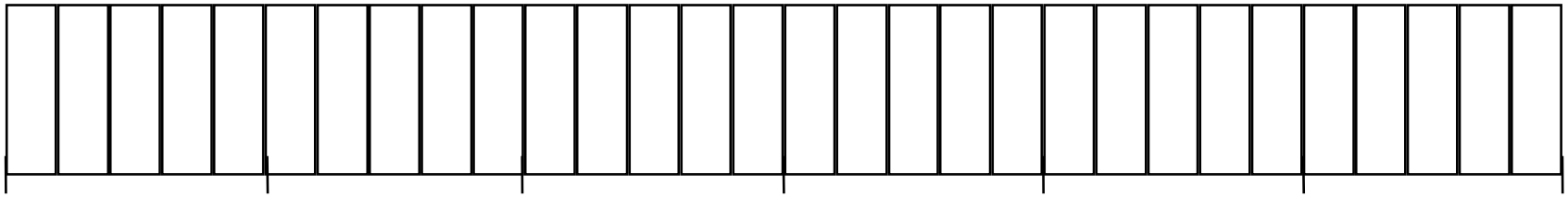
Υπάρχουν μεγάλες διακυμάνσεις στον μέσο χρόνο επιστροφής.

Συχνά προκύπτουν μεγάλοι χρόνοι αναμονής και απόκρισης.

Ακατάλληλος για αλληλεπιδραστικά συστήματα.

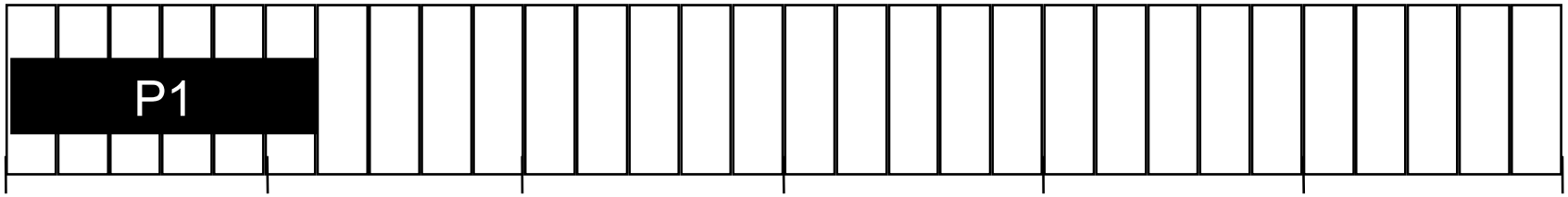
FCFS - παράδειγμα

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



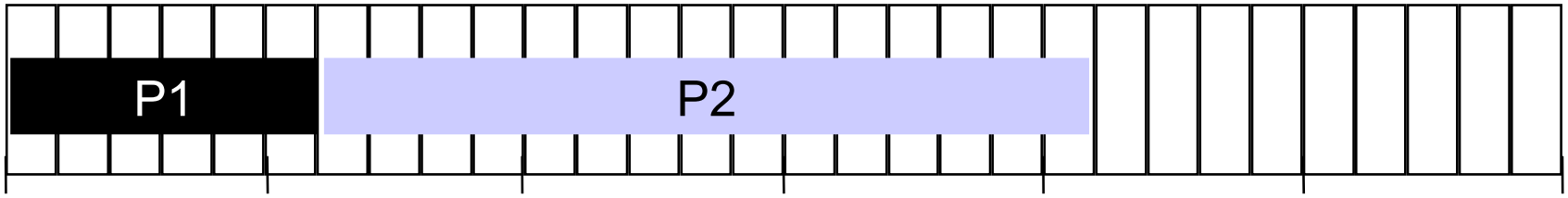
FCFS - παράδειγμα

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



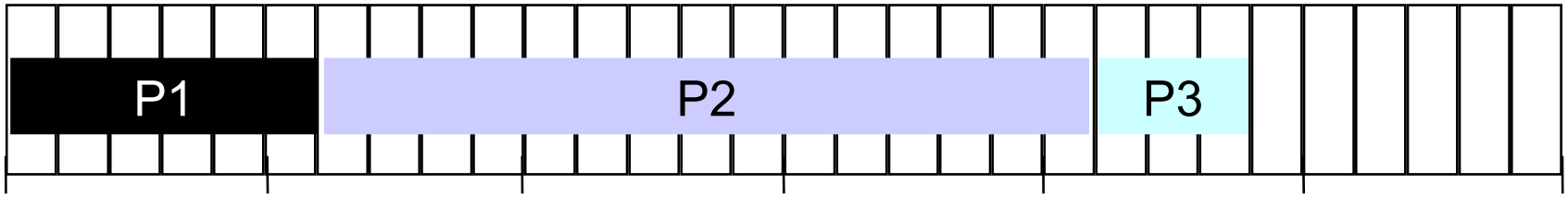
FCFS - παράδειγμα

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



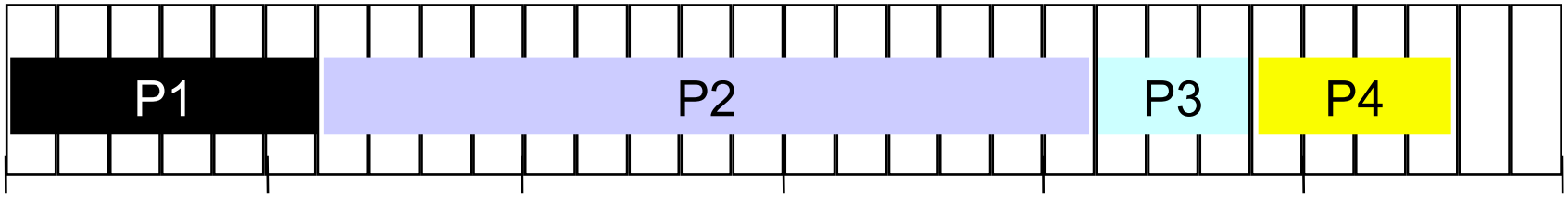
FCFS - παράδειγμα

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



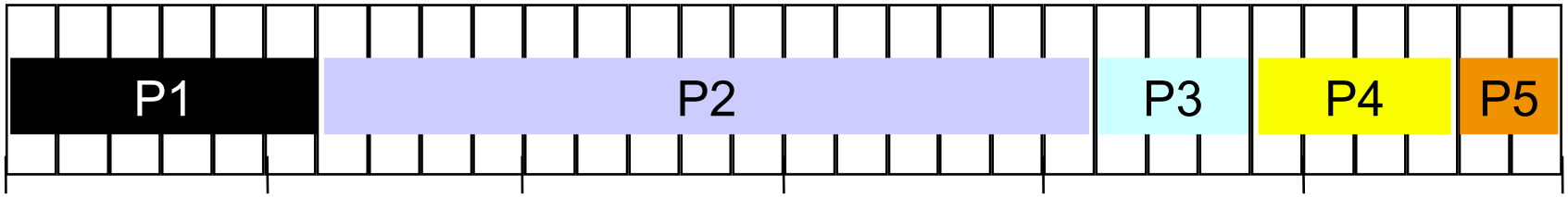
FCFS - παράδειγμα

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



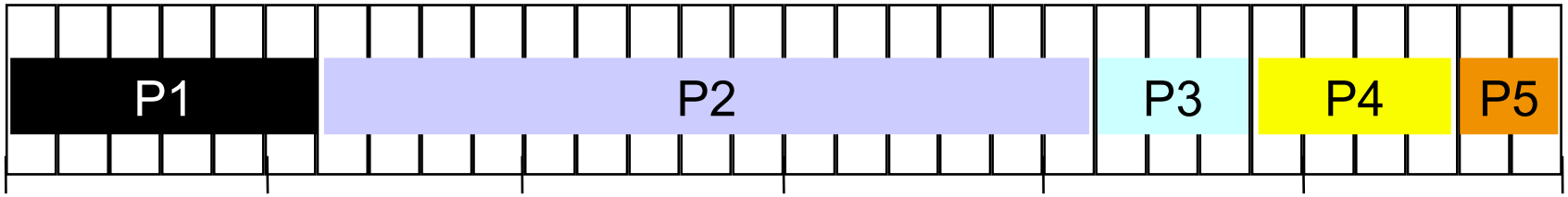
FCFS - παράδειγμα

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



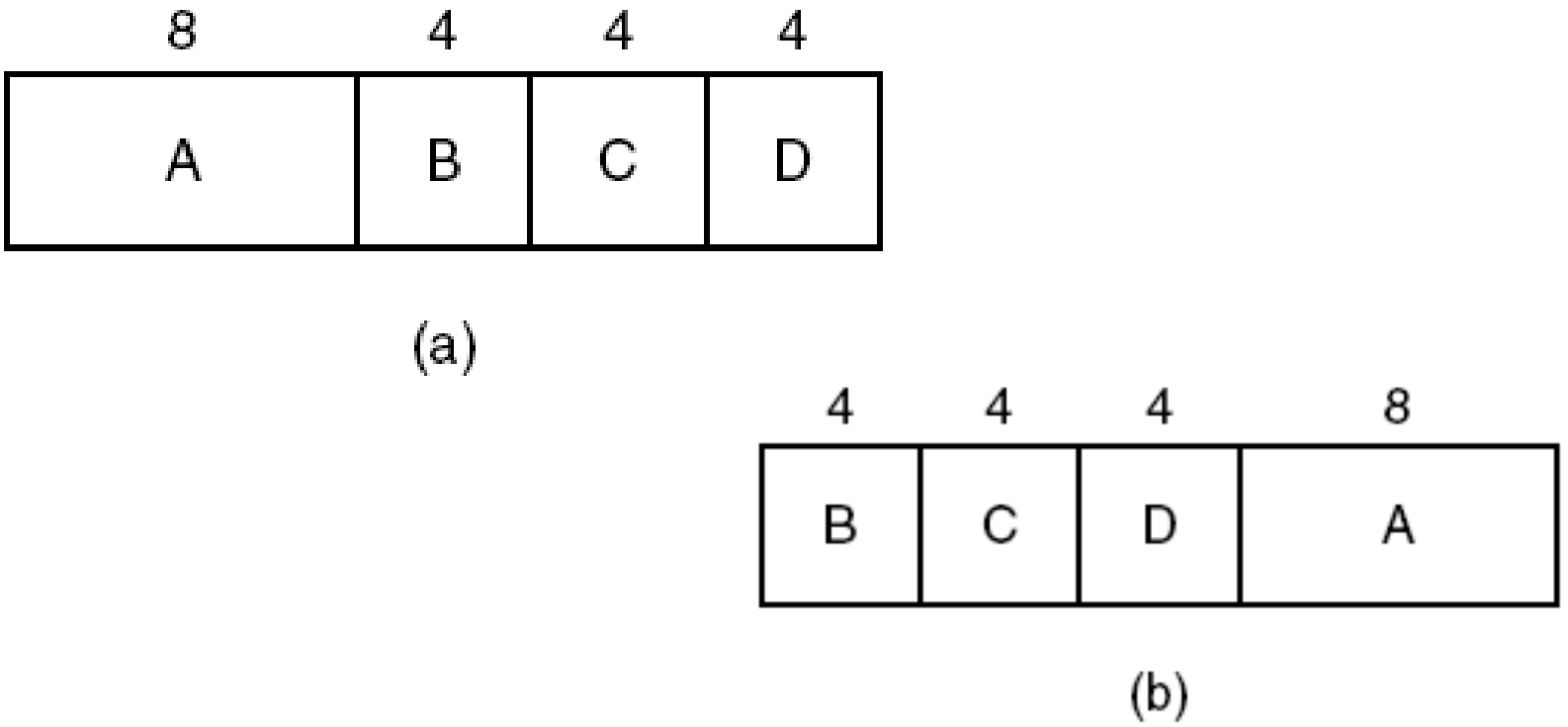
Process #	*Waiting Time*	*Response Time*	*Turnaround Time*	*#of Context Switches*
P1	0	0	6	1
P2	6	6	21	1
P3	21	21	24	1
P4	24	24	28	1
P5	28	28	30	1
Average	79/5 = 15.8	79/5 = 15.8	21.8	1

FCFS - παράδειγμα

Στο παραπάνω παράδειγμα, εισάγετε διακοπές λόγω Ε/Ε σε όλες τις διεργασίες. Αρχικά υποθέστε ίση κατανομή διακοπών Ε/Ε σε όλες τις διεργασίες. Επαναλάβετε τους υπολογισμούς. Τι παρατηρείτε;

Στη συνέχεια χωρίστε τις διεργασίες σε δύο κατηγορίες, CPU-bound και I/O-bound. Οι CPU-bound έχουν λίγες σύντομες αναμονές Ε/Ε και οι I/O bound έχουν πολλές μεγάλες αναμονές Ε/Ε. Ενδεικτικά θέστε πχ τη διεργασία P2 ώς CPU-bound και τις υπόλοιπες ως I/O-bound. Ορίστε 2 διακοπές του 1 slot και 5 διακοπές των 10 slots για τις CPU-bound και I/O-bound αντίστοιχα. Επαναλάβετε τους υπολογισμούς. Τι παρατηρείτε;

Shortest Job First (SJF)



(a) Εκτέλεση διεργασιών στην αρχική σειρά.

(b) Εκτέλεση εργασιών με βάση τη μικρότερη διάρκεια.

Shortest Job First Non Preemptive (SJFN)

Το λ.σ. γνωρίζει (η διεργασία δηλώνει) τον εκτιμώμενο χρόνο CPU burst (πχ επαναλαμβανόμενες εργασίες με γνωστή διάρκεια).

Όλες οι διεργασίες έχουν περίπου ταυτόχρονη άφιξη.

Ενσωματώνει αναμφίβολα προτεραιότητες : οι συντομότερες διεργασίες έχουν δεδομένη προτεραιότητα.

Χωρίς προεκτόπιση: αν εκχωρηθεί η CPU, η διεργασία δεν διακόπτεται μέχρι να ολοκληρωθεί ο καταιγισμός της. Δε προβλέπεται ενδιάμεση άφιξη διεργασιών.

Βέλτιση λύση για τις συγκεκριμένες παραδοχές: οι σύντομες εργασίες δεν επιβαρύνονται από αναμονές ενώ οι μακρόχρονες επηρρεάζουν μόνο τον εαυτό τους. Αποτελεί μια προφανή βελτίωση του αλγορίθμου FCFS.

Σύγκριση FCFS και SJFN

Από το παραπάνω σχήμα A=8, B= 4, C =4, D=4

FCFS	Response	Turnaround
A	0	8
B	8	12
C	12	16
D	16	20
	36/4	56/4

SJF	Response	Turnaround
B	0	4
C	4	8
D	8	12
A	12	20
	24/4	44/4

Shortest Job First Preemtive (SJFP)

Τροποποίηση του SJF για τη περίπτωση που όλες οι διεργασίες δεν έχουν περίπου ταυτόχρονη άφιξη.

Η απόφαση λαμβάνεται με βάση τις διεργασίες που είναι διαθέσιμες κατά το τερματισμό ή ανατολή της εκτελούμενης διεργασίας.

Κατάλληλη μέθοδος και για συστήματα χρονο-μερισμού.

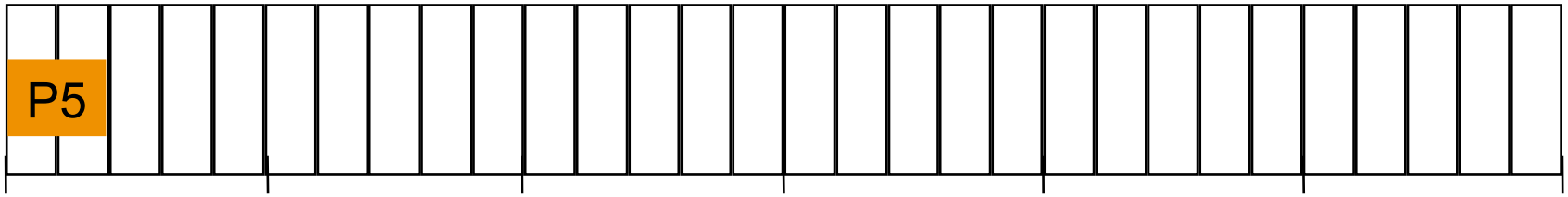
Και πάλι απαιτείται γνώση χρόνων CPU bursts.

Είναι πιθανή η παρατεταμένη στέρηση. Αν νέες μικρής διάρκειας διεργασίας φθάνουν στο σύστημα οι προγενέστερες, μεγάλης διάρκειας διεργασίες, δεν θα εξυπηρετηθούν ποτέ.

Με εξαίρεση τις ακραίες συνθήκες δίνει λύσεις καλύτερες του FCFS.

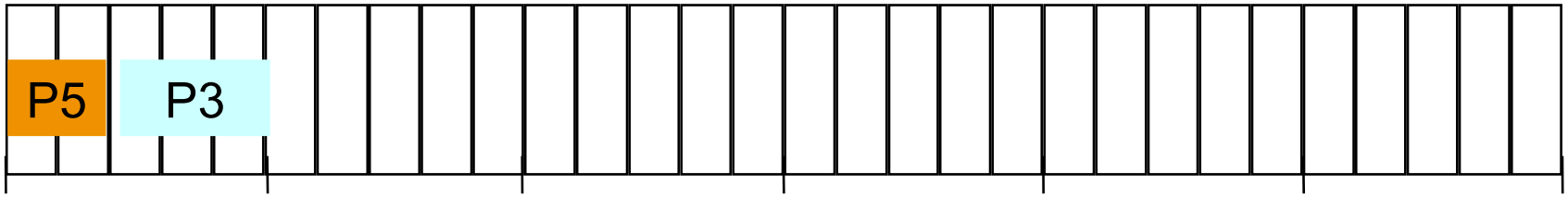
SJFN – παράδειγμα

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



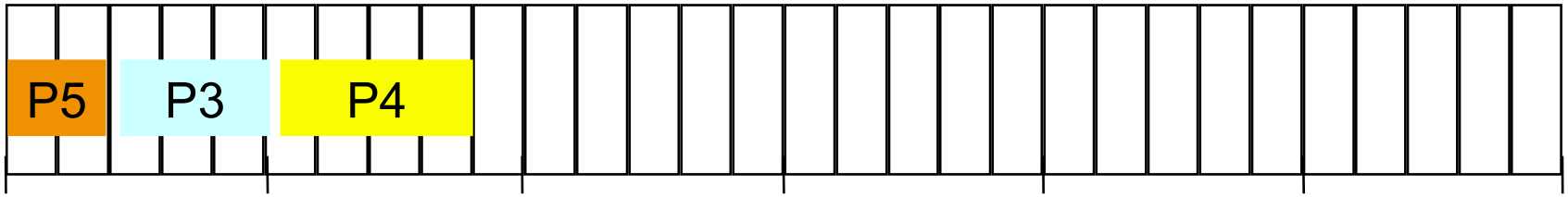
SJFN – παράδειγμα

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



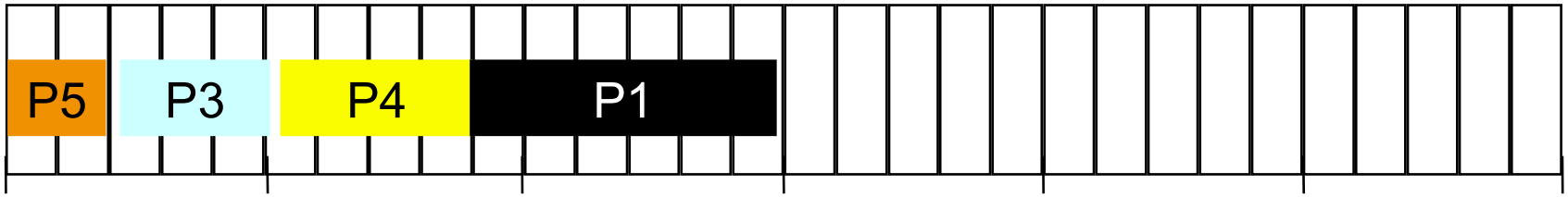
SJFN – παράδειγμα

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



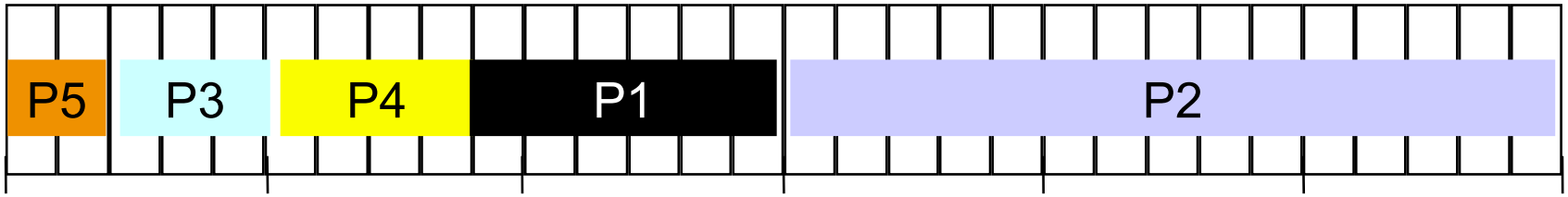
SJFN – παράδειγμα

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



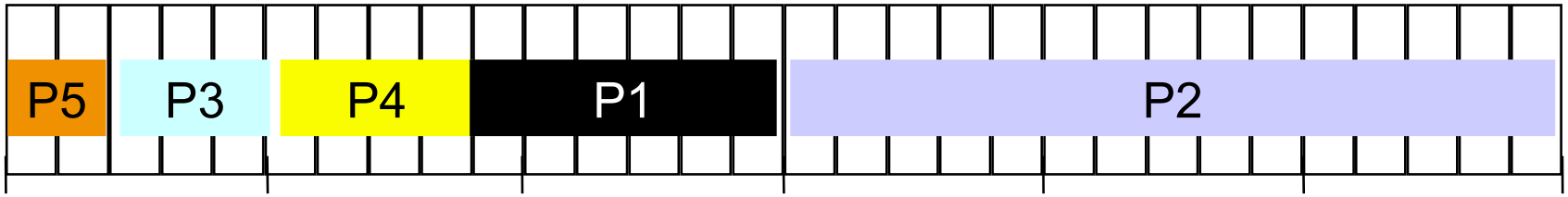
SJFN – παράδειγμα

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



SJFN – παράδειγμα

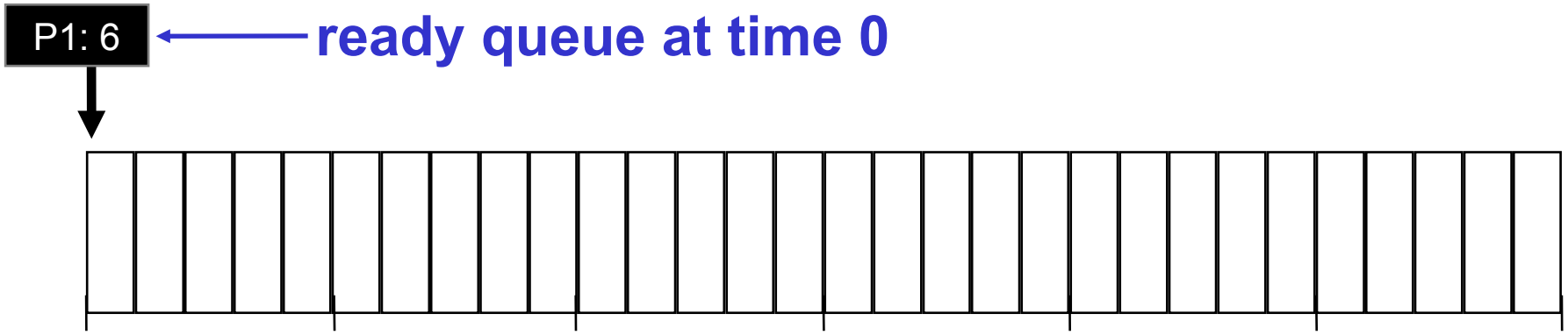
Process #	*Arrival Time*	*Burst Length*	*Priority*
P1	0	6	1
P2	0	15	1
P3	0	3	1
P4	0	4	1
P5	0	2	1



Process #	*Waiting Time*	*Response Time*	*Turnaround Time*	*#of Context Switches*
P1	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

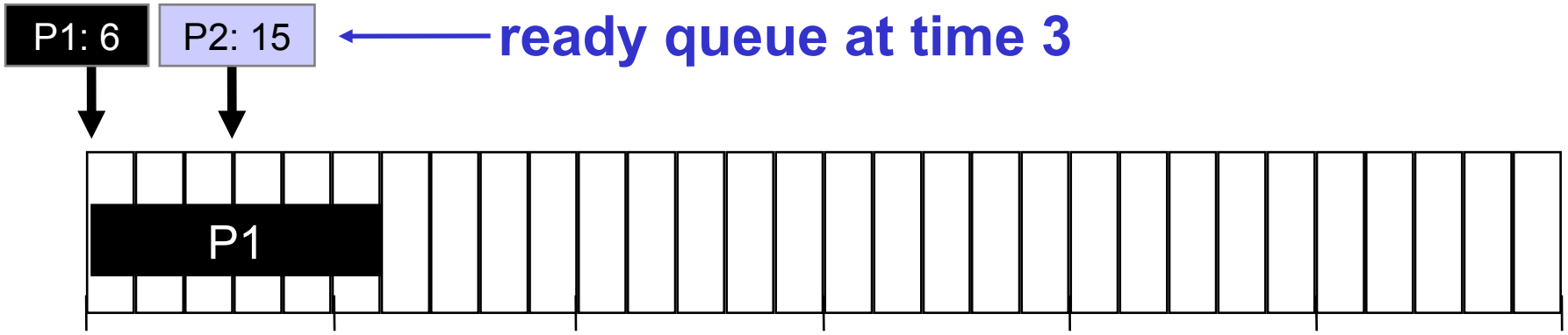
SJFP - παράδειγμα

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



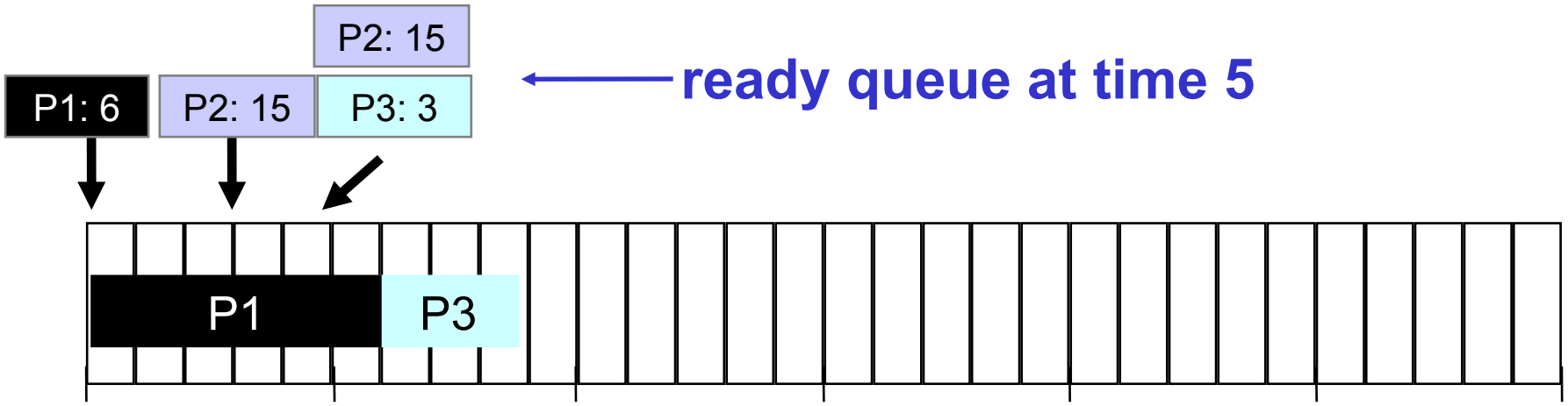
SJFP - παράδειγμα

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



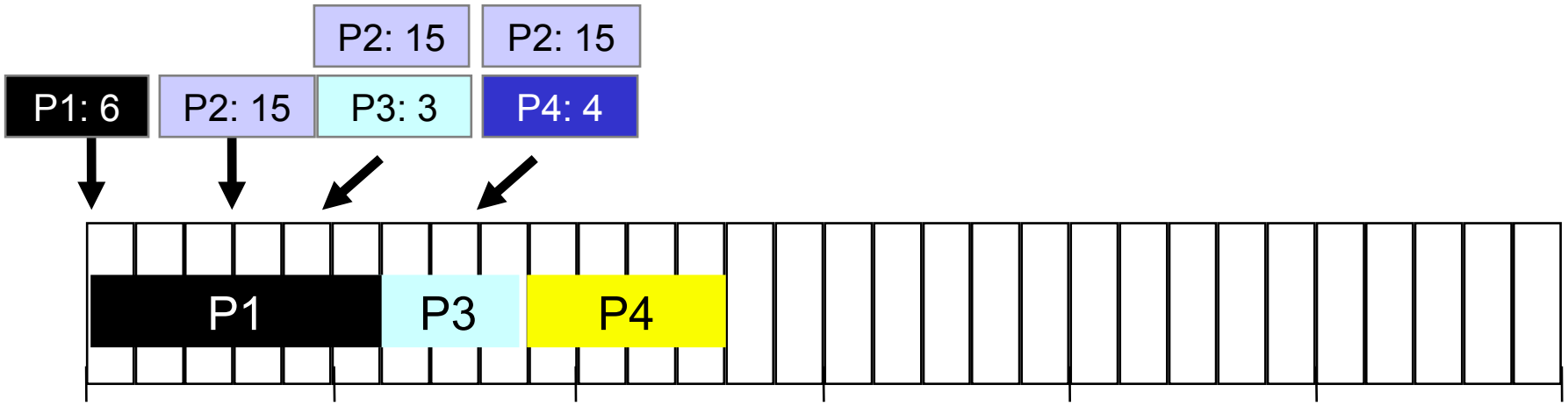
SJFP - παράδειγμα

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



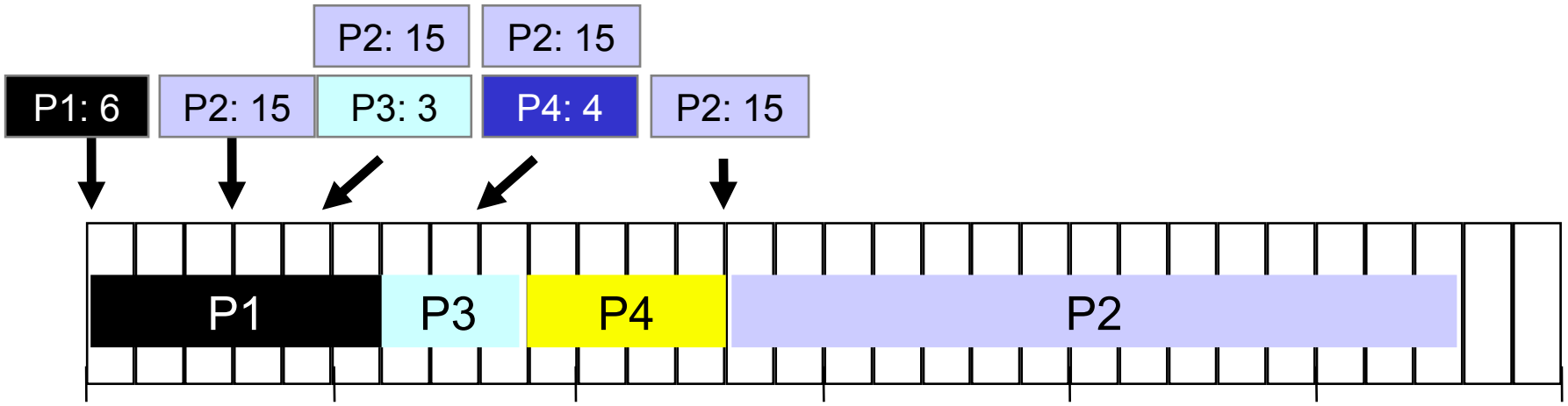
SJFP - παράδειγμα

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



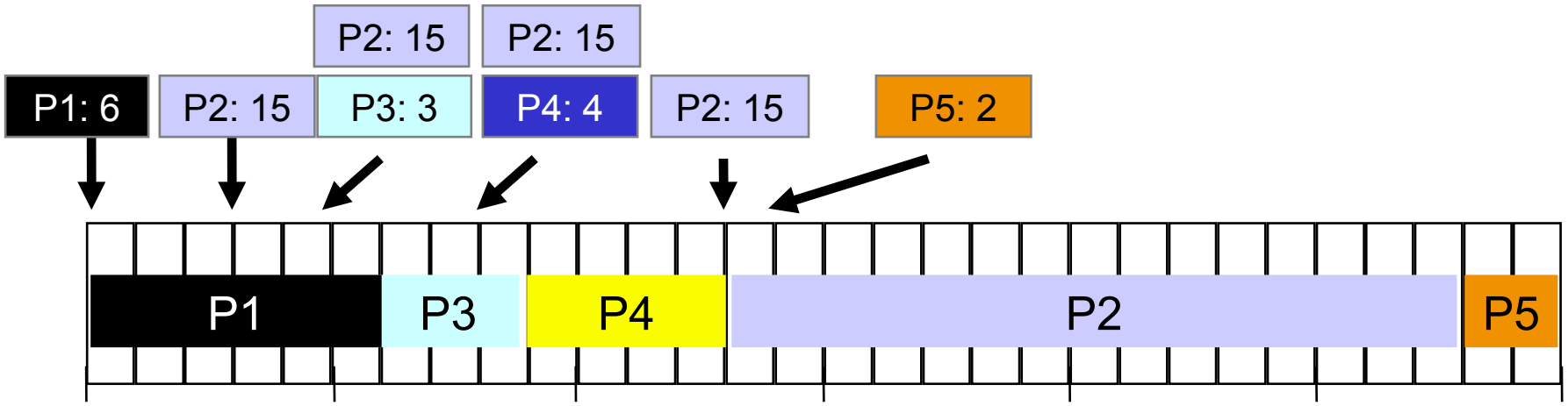
SJFP - παράδειγμα

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



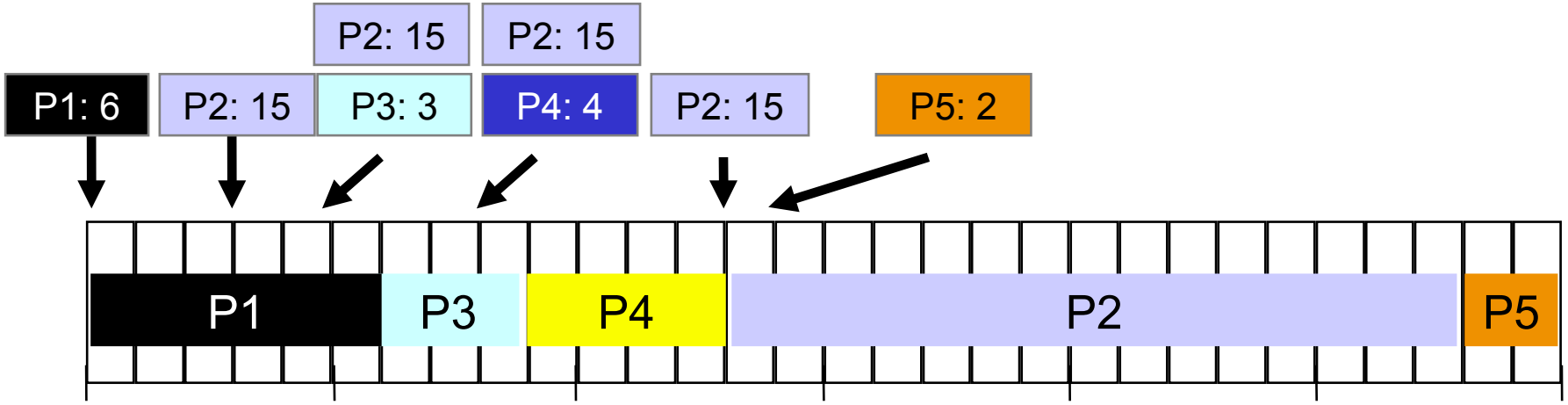
SJFP - παράδειγμα

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



SJFP- παράδειγμα

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

Shortest Remaining Time Next (SRTN) (1)

Επιλέγεται η διεργασία με το μικρότερο εναπομείναντα χρόνο.

Έμμεση εφαρμογή προτεραιότητας.

Ο εναπομένων χρόνος είναι ο συνολικός χρόνος καταιγισμού μείον το χρόνο που η διεργασία παρέμεινε προς εκτέλεση στη CPU.

Με προεκτόπιση: αν φθάσει μια διεργασία με μικρότερο χρόνο καταιγισμού από τον υπολειπόμενο χρόνο καταιγισμού της τρέχουσας διεργασίας, η τρέχουσα διεργασία διακόπτεται.

Δύσκολη υλοποίηση εξ αιτίας της αναγκαστικής πρόβλεψης των χρόνων καταιγισμού που απαιτείται .

Shortest Remaining Time Next (SRTN) (2)

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

Οι προεκτοπιζόμενες διεργασίες οδηγούνται στην ουρά των έτοιμων διεργασιών. Απαιτείται καταγραφή των χαρακτηριστκών των διεργασιών που βρίσκονται στην ουρά των έτοιμων διεργασιών.

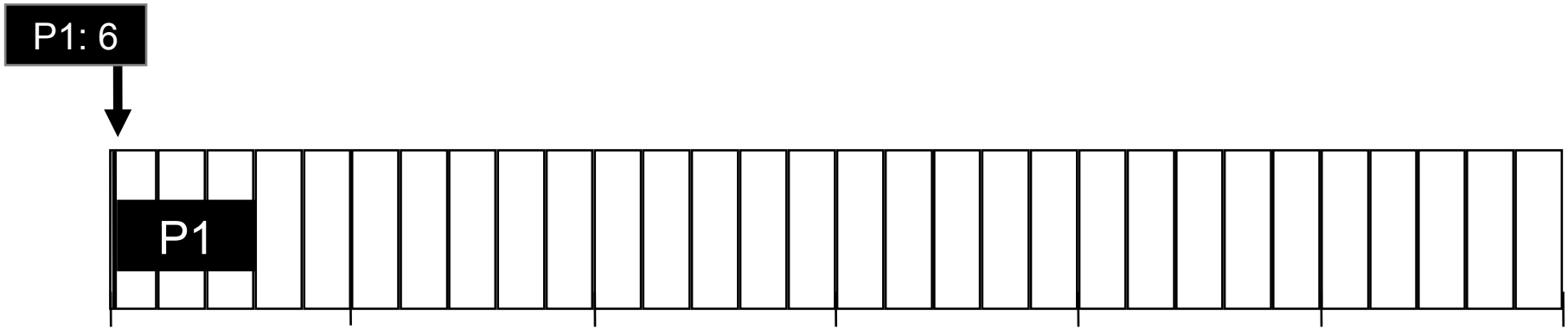
Δίνει καλούς χρόνους απόκρισης στις διεργασίες μικρής διάρκειας.

Αν ληφθούν υπόψη οι εναλλαγές πλαισίων η χρονική επιβάρυνση μπορεί να είναι σημαντική.

Η παρατεταμένη στέρηση είναι πιθανή. Ο χρόνος άφιξης μιας νέας διεργασίας είναι σημαντικός.

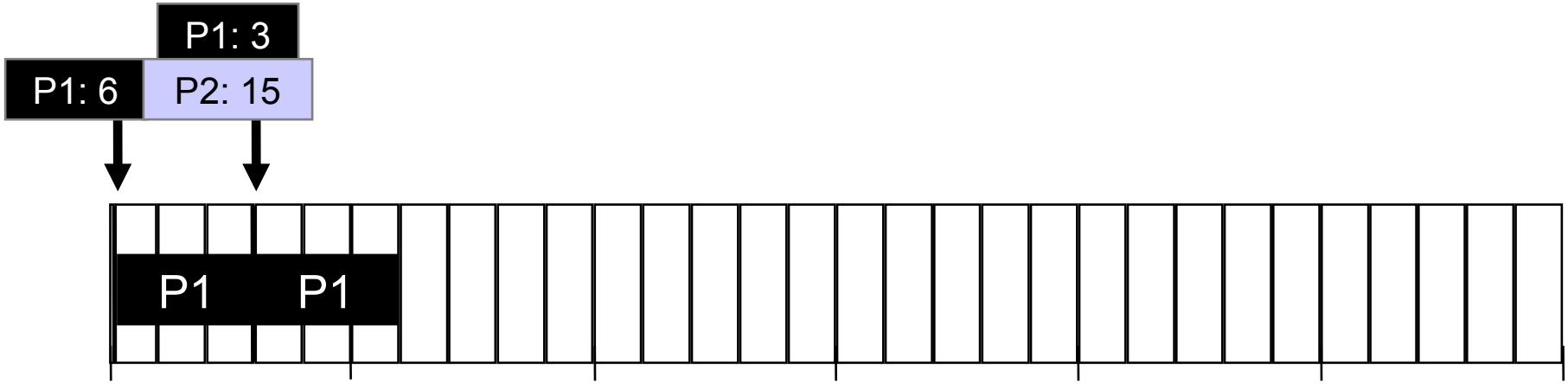
SRTN - παράδειγμα

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



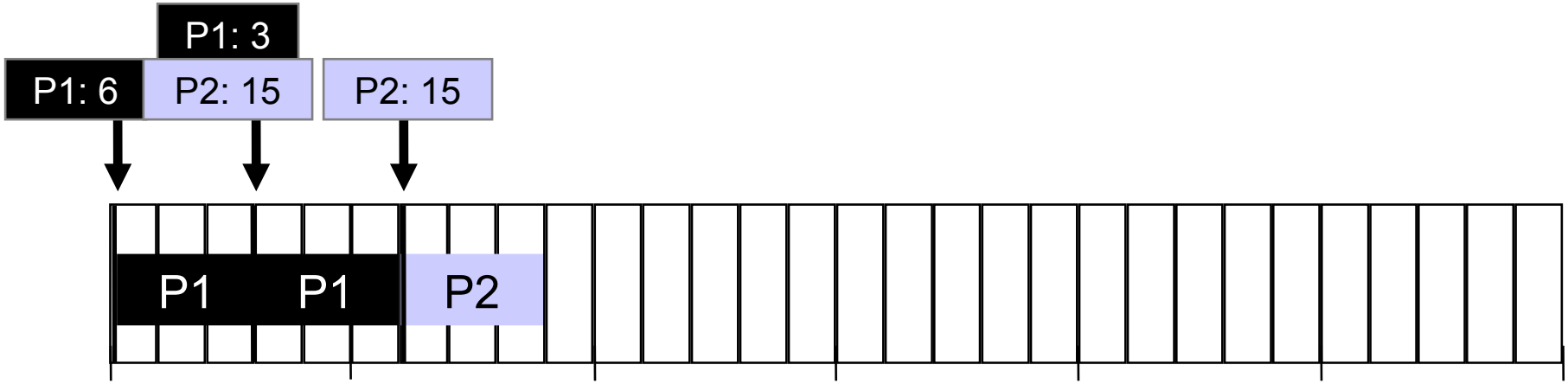
SRTN - παράδειγμα

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



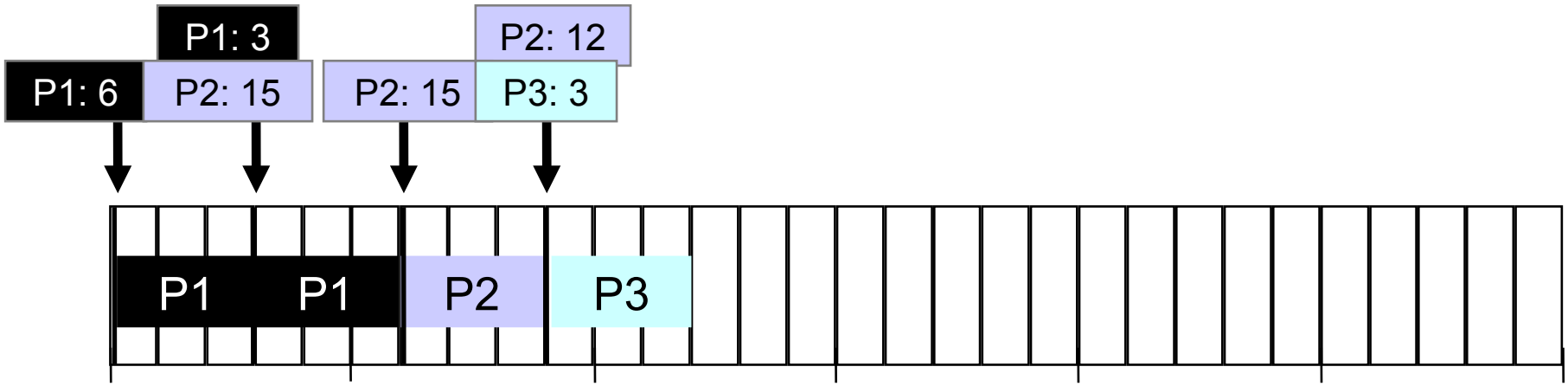
SRTN - παράδειγμα

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



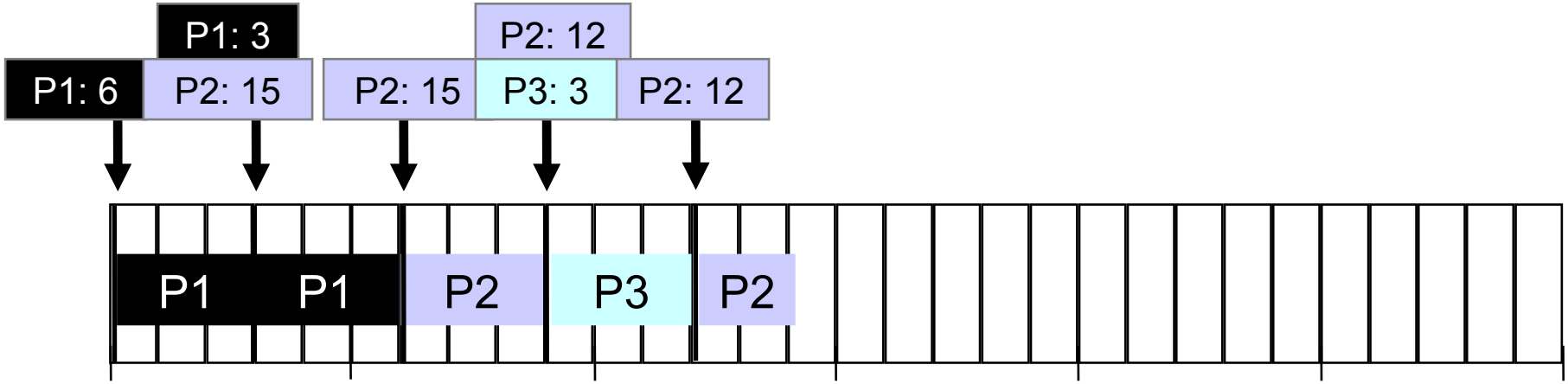
SRTN - παράδειγμα

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



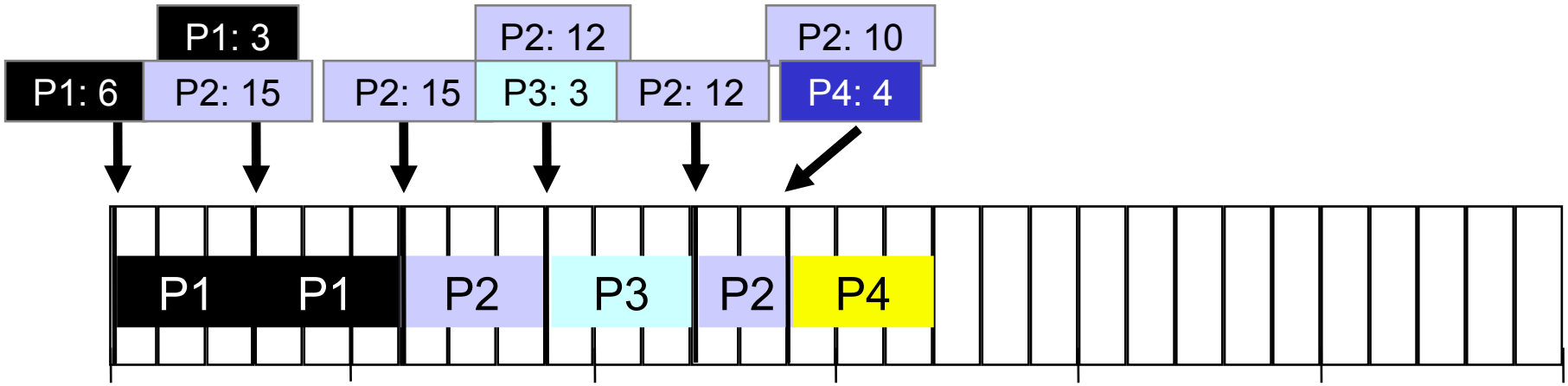
SRTN - παράδειγμα

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



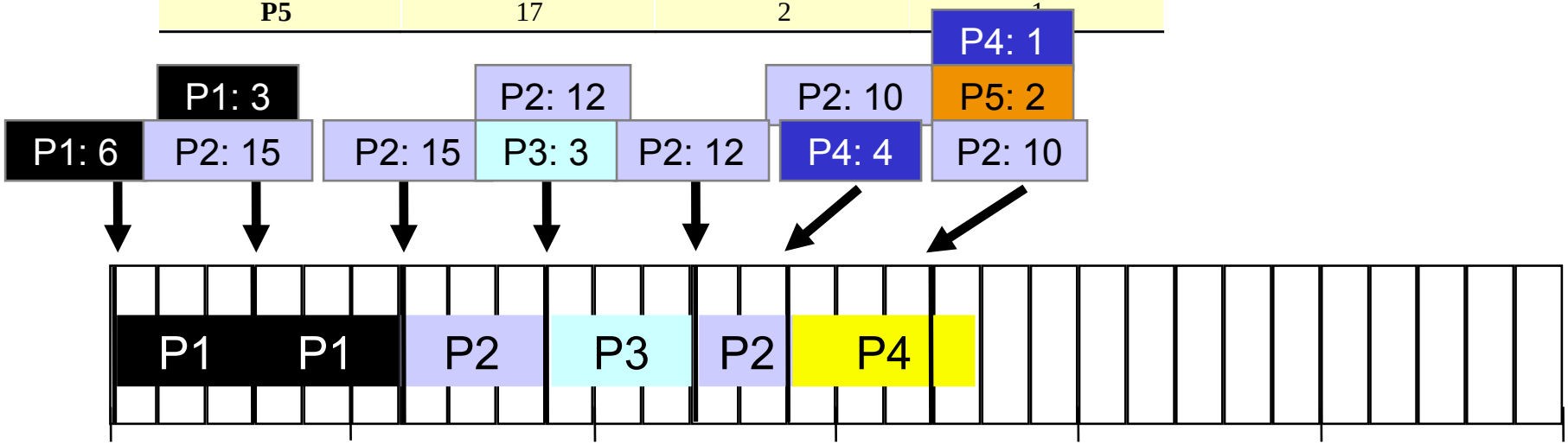
SRTN - παράδειγμα

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



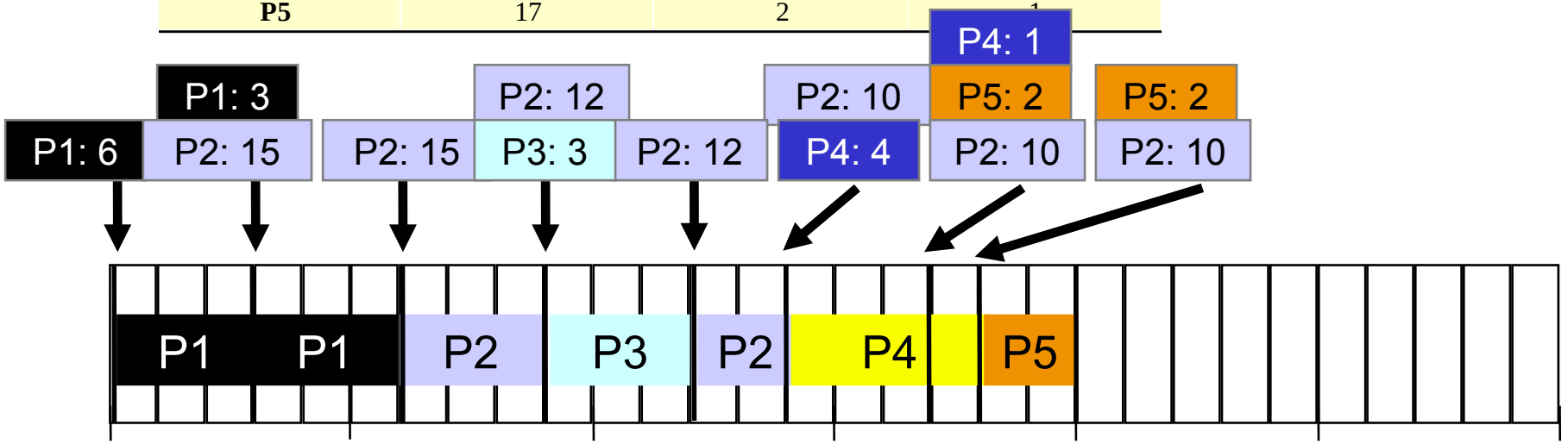
SRTN - παράδειγμα

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	



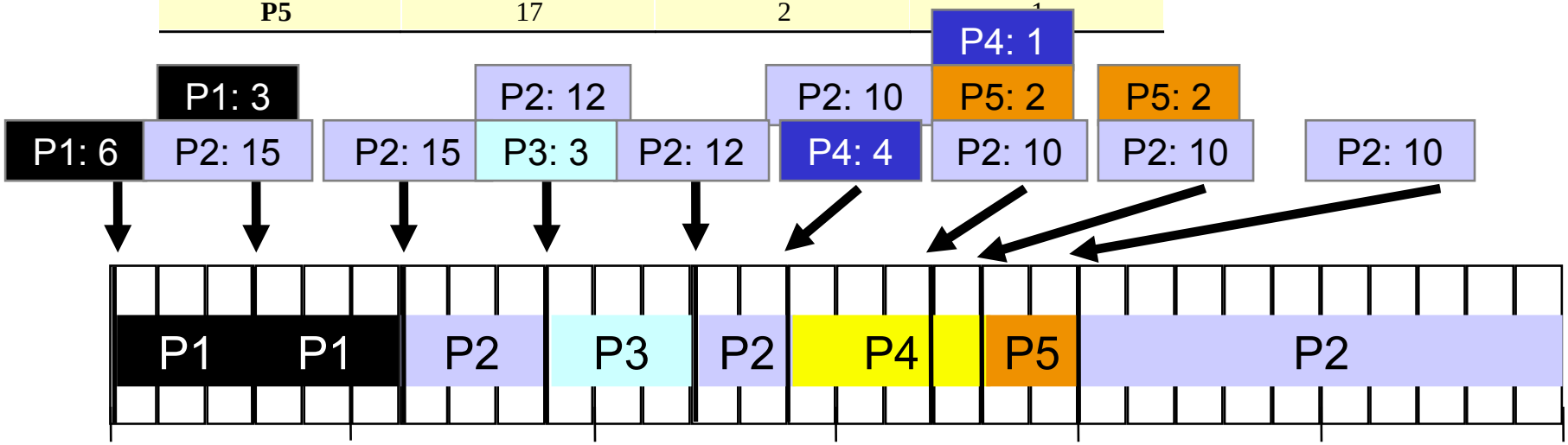
SRTN - παράδειγμα

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



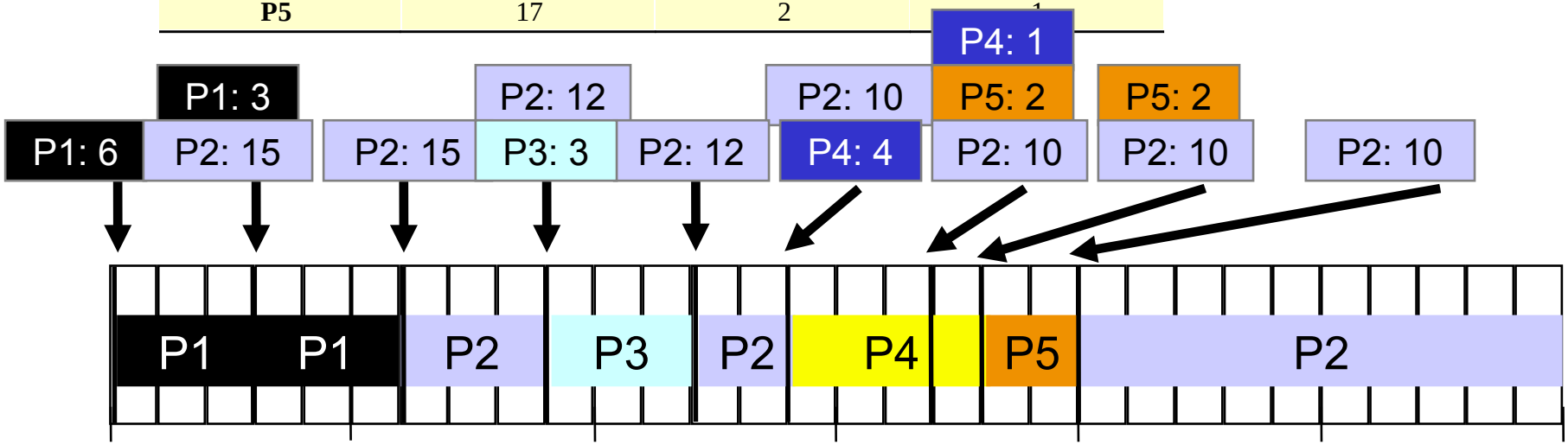
SRTN - παράδειγμα

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



SRTN - 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

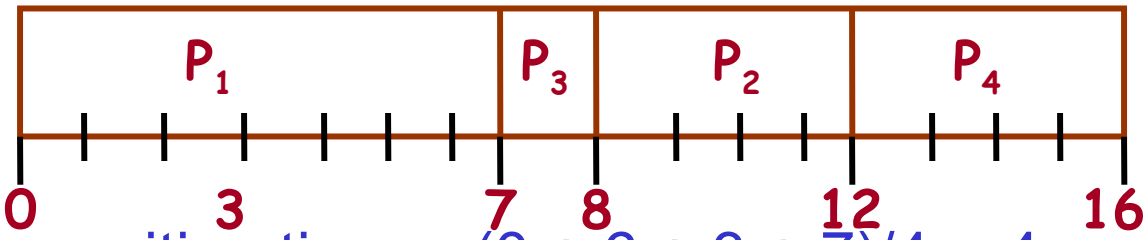


Process #	*Waiting Time*	*Response Time*	*Turnaround Time*	*#of Context Switches*
P1	0	0	6	1 (2)
P2	(6-3) + (12-9) + (20-14) = 12	(6-3) = 12	(30 - 3) = 27	3
P3	0	0	(12-9) = 3	1
P4	0	0	(18-14) = 3	1 (2)
P5	(18-17) = 1	(18-17) = 1	(20-17) = 3	1
Average	13/5 = 2.6	13/5 = 2.6	36/5 = 7.2	7

Σύγκριση SJFP και SRTN (1)

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

- **SJFP**

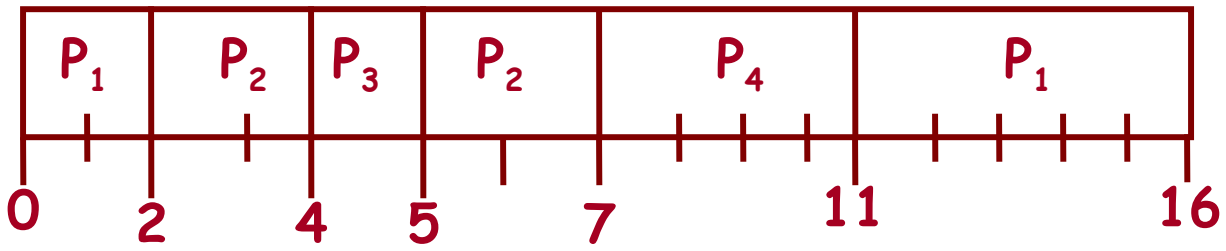


- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Σύγκριση SJFP και SRTN (2)

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

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

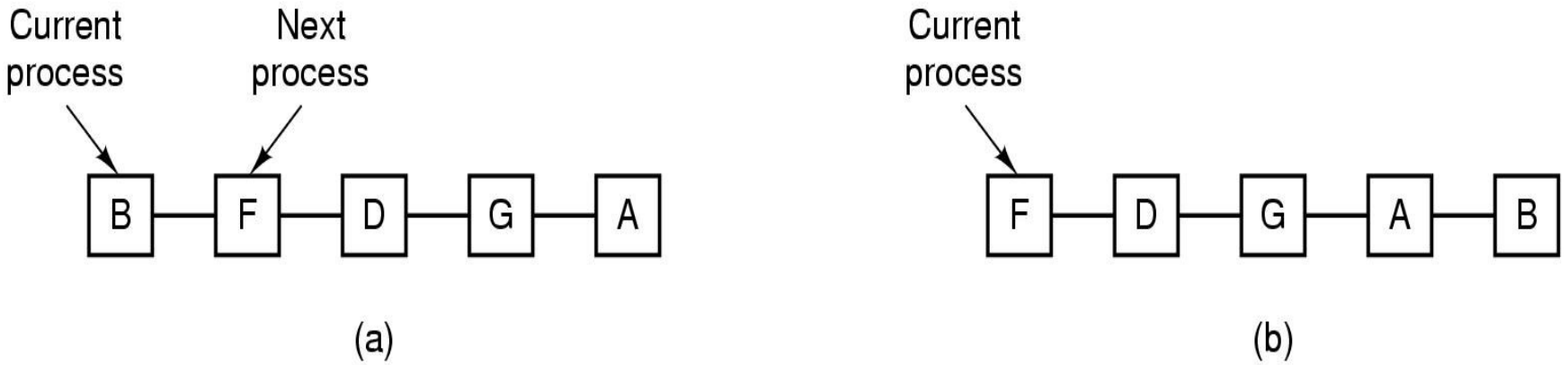
Χρονοπρογραμματισμός
(ή Χρονοδρομολόγηση ή Δρομολόγηση)

Γενικά
Συστήματα Δέσμης
Αλληλεπιδραστικά Συστήματα

Αλγόριθμοι: Αλληλεπιδραστικά συστήματα

- Εκ περιτροπής (Round Robin) RR
- Προτεραιότητας (Priority)
- Πολλαπλές ουρές (Multiple Queues)
- Μικρότερη διάρκεια (Shortest Process Next) SPN
- Άλλοι αλγόριθμοι

Round Robin (RR) (1)



(a) Ουρά FIFO έτοιμων διεργασιών πρίν την εκτέλεση της διεργασίας B (b) Η ίδια ουρά μετά την εκτέλεση της διεργασίας B.. σειρά έχει η διεργασία F.

Στις διεργασίες δίνεται ένα σταθερό ποσό χρόνου της CPU και αναφέρεται ως κβάντο χρόνου (time quantum, time slice, slot).

Με προεκτόπιση: μια διεργασία επιτρέπεται να εκτελείται μέχρι να συμπληρωθεί το κβάντο χρόνου. Τότε το λ.σ προκαλεί μια χρονοδιακοπή (timeout) και η εκτελούμενη διεργασία τίθεται στην ουρά των έτοιμων διεργασιών.

Round Robin (RR) (2)

Χρησιμοποιείται συχνά σε αλληλεπιδραστικά συστήματα χρονομερισμού.

Απλή υλοποίηση, επιδέχεται tuning μέσω του κβάντου χρόνου.

Βέλτιστο κβάντο χρόνου:

Χρόνος θεματικής εναλλαγής: επιβάλλεται σε κάθε χρονοδιακοπή, άρα προκαλεί αντίστχοιχη επιβάρυνση: πχ κβάντο 5msec και θεματική εναλλαγή 1msec σημαίνει επιβάρυνση 1/(1+5)~15%.

Μέσος χρόνος CPU burst: αν το κβάντο χρόνου είναι πολύ μεγάλο σε σχέση με το μέσο CPU burst τότε η προεκτόπιση θα συμβαίνει συχνότερα από τη χρονοδιακοπή, ενώ αν είναι πολύ μικρό, θα συμβαίνει το αντίθετο.

Μεγάλο κβάντο χρόνου: ευνοούνται οι CPU bound διεργασίες.

Μικρό κβάντο χρόνου: ευνοούνται οι I/O bound διεργασίες (?)

Κανόνας: 80-90% των διεργασιών να έχουν μέσο CPU burst < quantum.

Τυπικό κβάντο χρόνου: 20-50 msec (ανάλογα με το χρονισμό και τη CPU).

Round Robin (RR) (3)

Γενικά περισσότερο ευνοούνται οι CPU bound διεργασίες.

Μια I/O bound διεργασία χρησιμοποιεί την CPU για χρονικό διάστημα μικρότερο ή ίσο του κβάντου χρόνου και στη συνέχεια αναστέλλεται περιμένοντας για Ε/Ε.

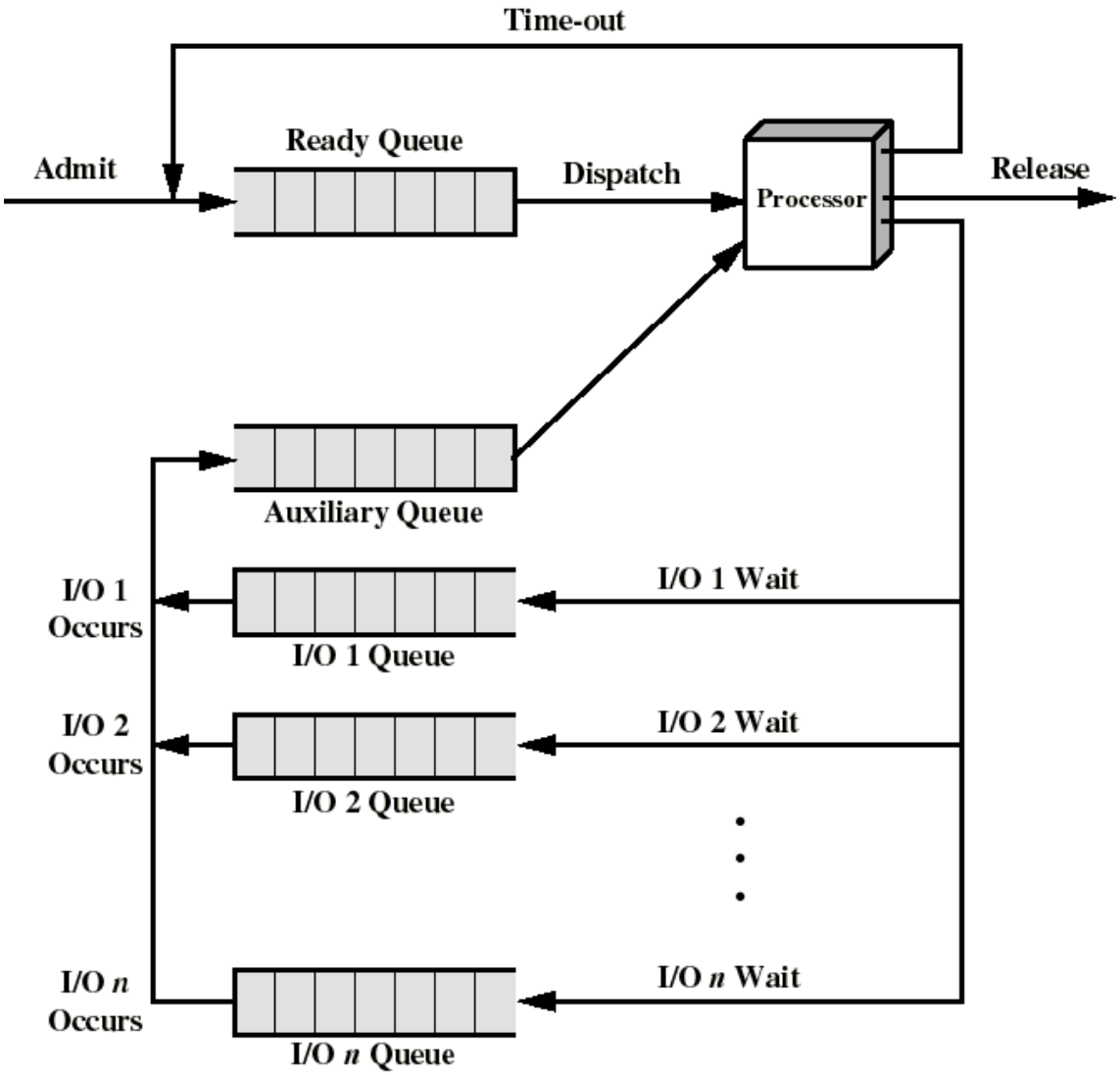
Μια CPU bound διεργασία εκτελείται για όλο το κβάντο χρόνου και τίθεται μετά στην ουρά των έτοιμων διεργασιών.

Λύση (ιδεατό Round Robin): όταν μια λειτουργία 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 για χρόνο = κβάντο χρόνου - το χρόνο της προηγούμενης εκτέλεσης.

Υπάρχουν λύσεις με μεταβλητό ή προσαρμοζόμενο ή εξειδικευμένο κβάντο.

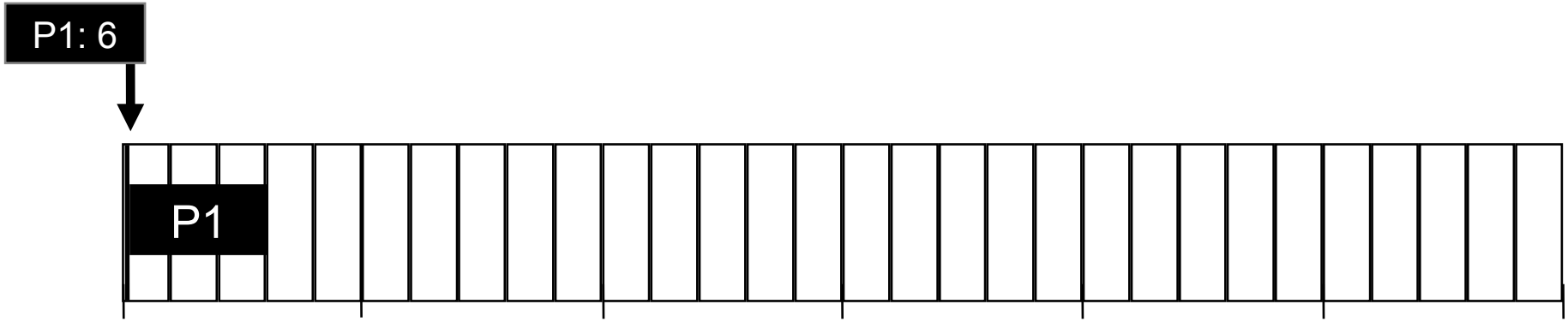
Διάγραμμα ουράς για ιδεατό Round Robin



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

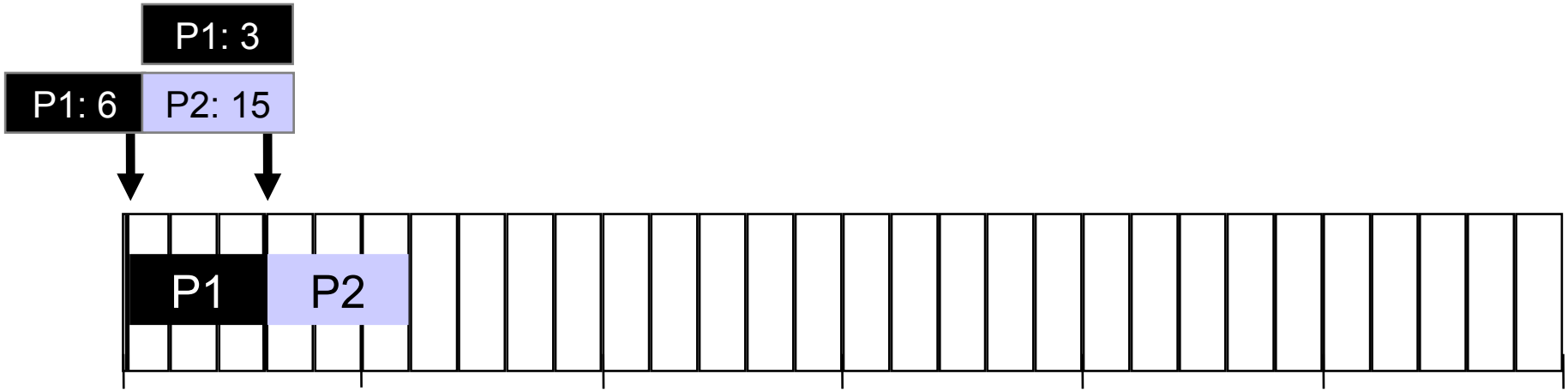
time quantum: 3 units



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

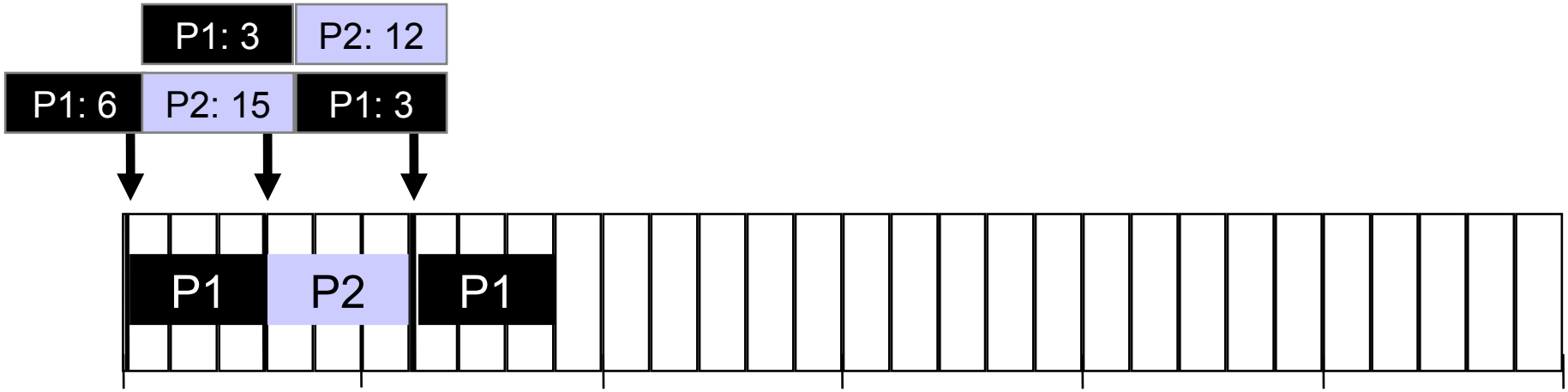
time quantum: 3 units



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

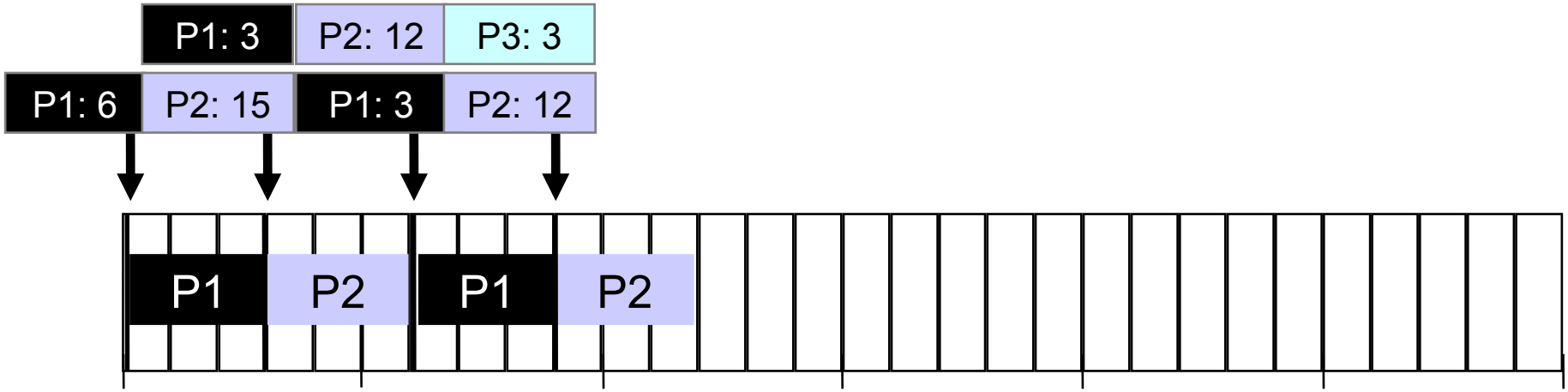
time quantum: 3 units



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

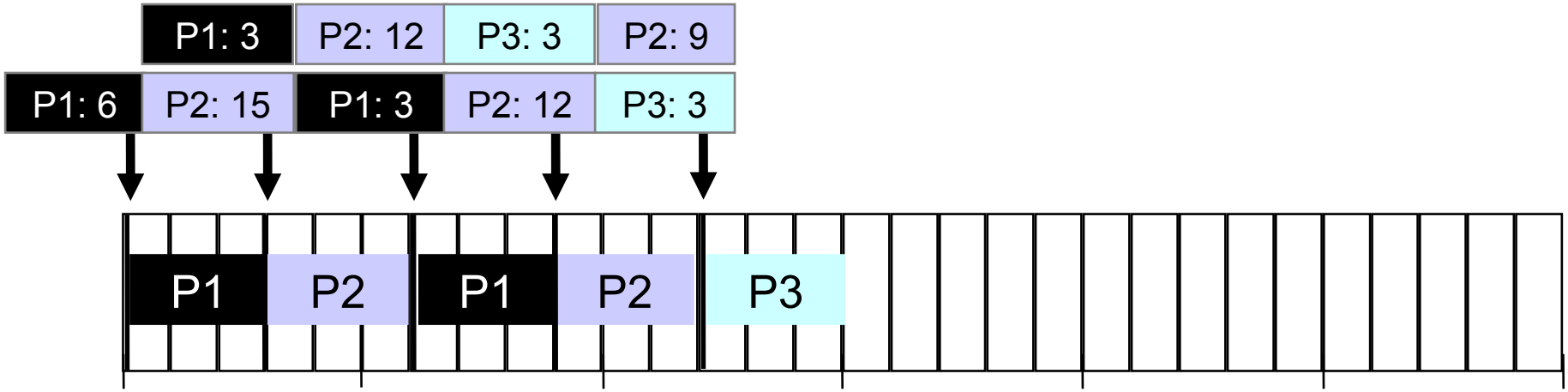
time quantum: 3 units



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

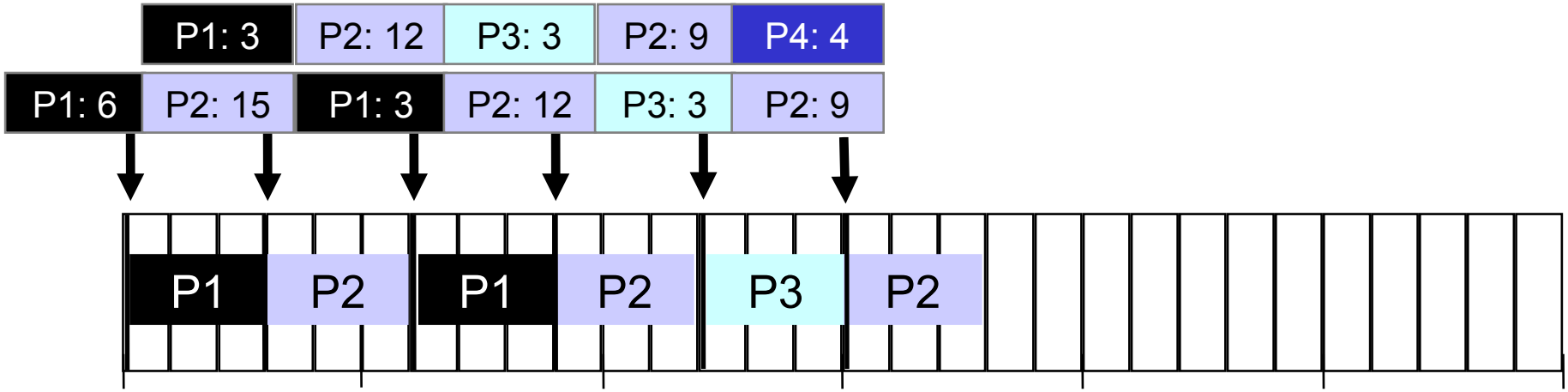
time quantum: 3 units



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

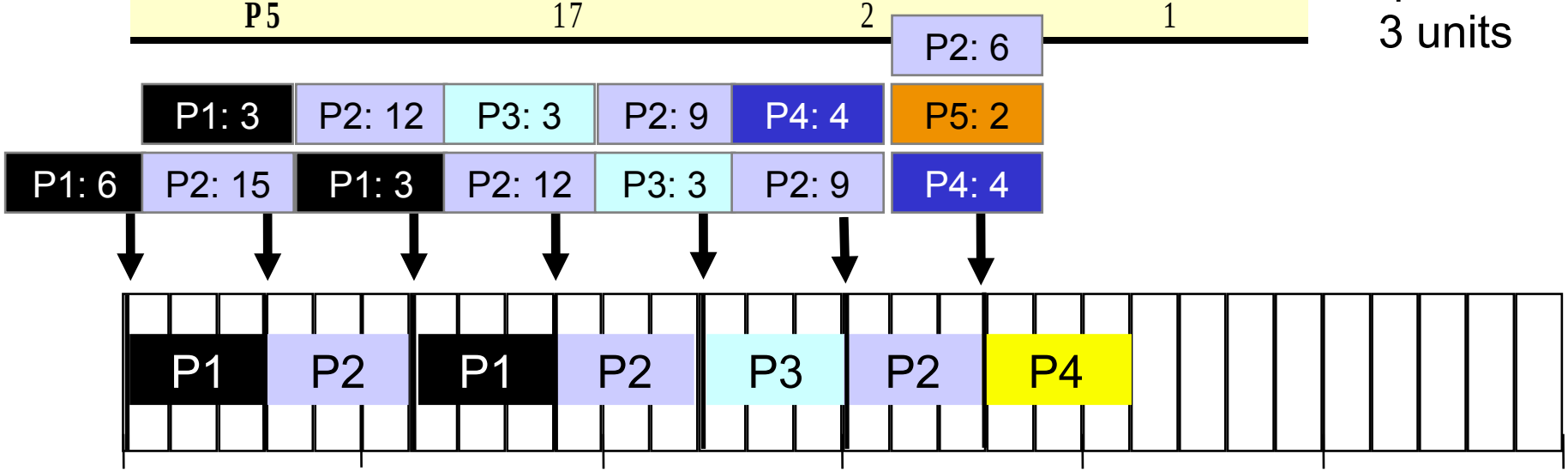
time quantum: 3 units



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

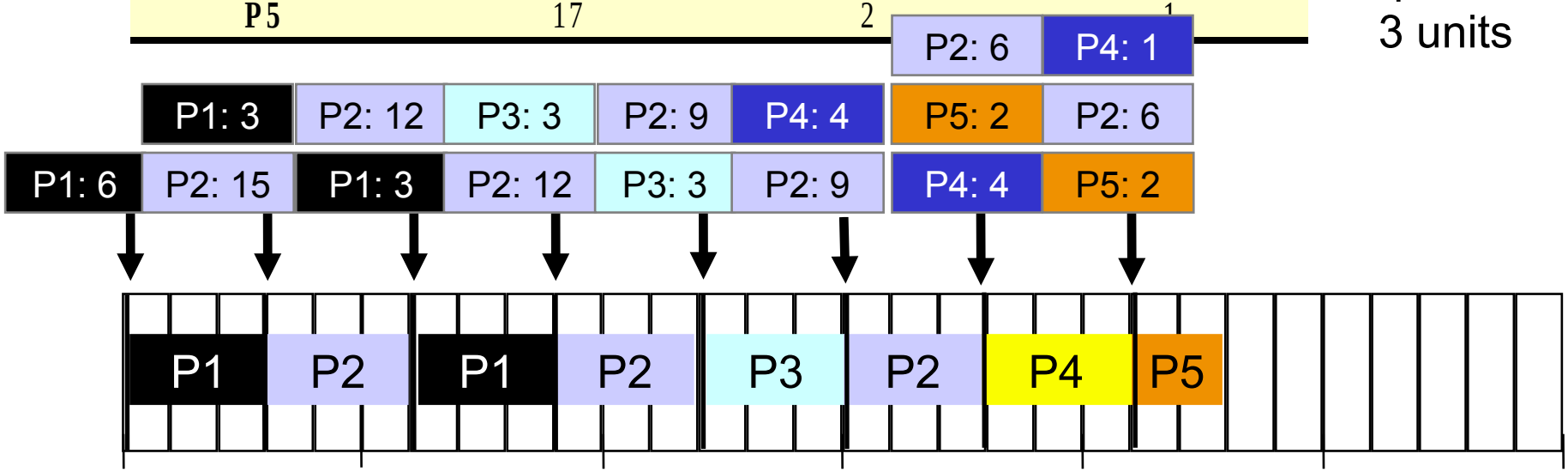
time quantum: 3 units



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

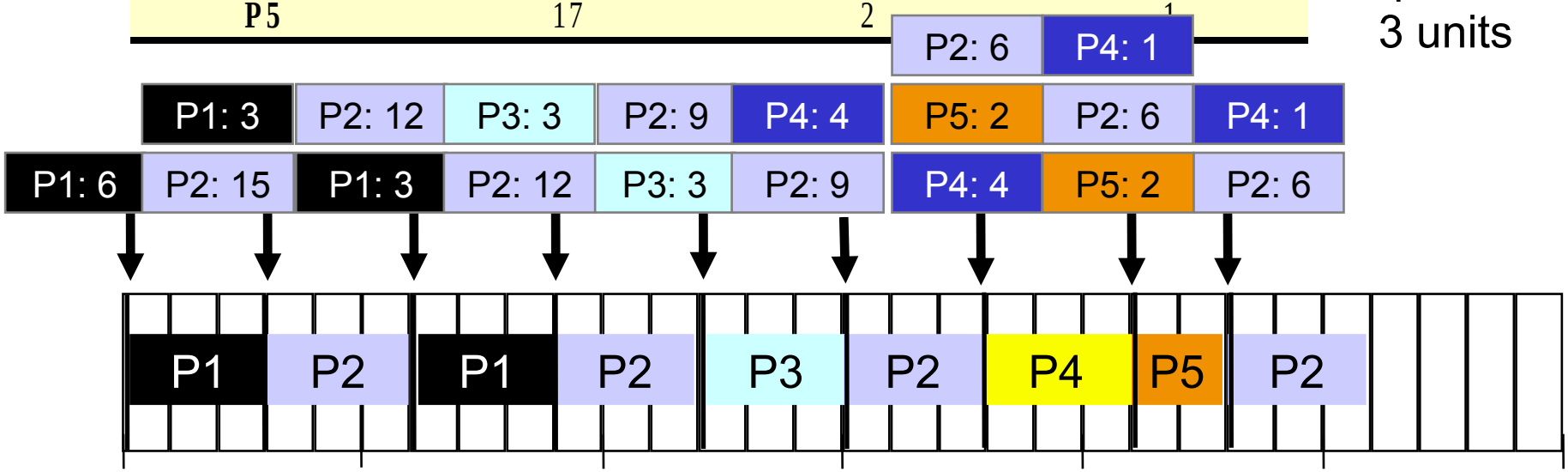
time quantum: 3 units



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

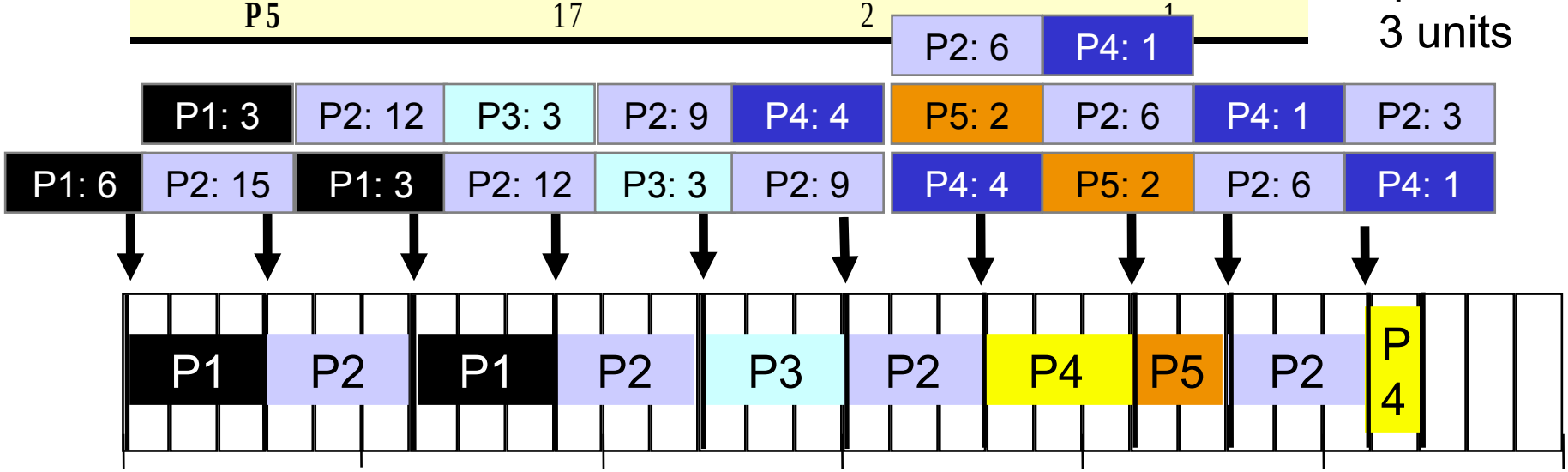
time quantum: 3 units



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

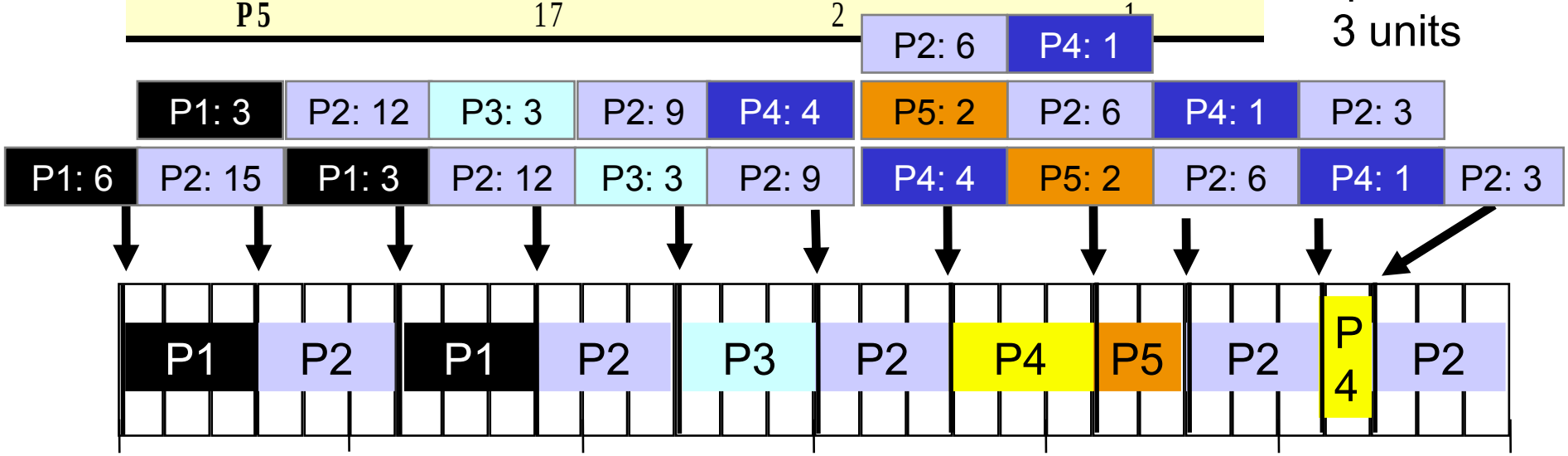
time quantum: 3 units



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

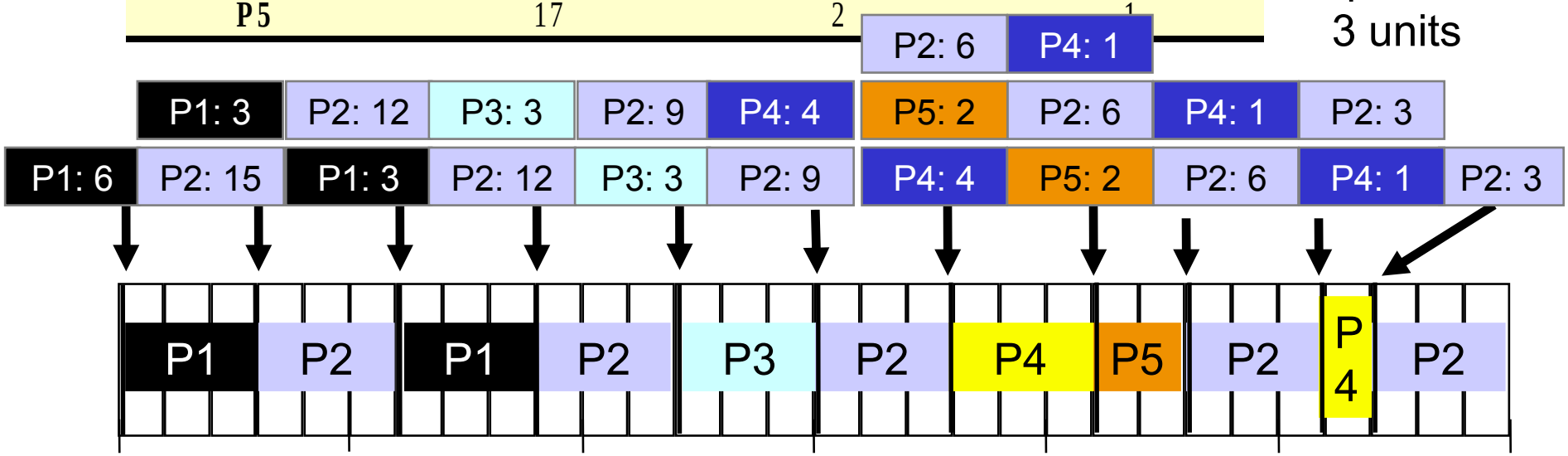
time quantum: 3 units



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

time quantum: 3 units



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

Προτεραιότητες (1)

Οι διεργασίες λαμβάνουν ένα επίπεδο προτεραιότητας με βάση τη πολιτική του συστήματος, πχ

Επείγουσες διεργασίες λ.σ. > διεργασίες λ.σ. παρασκηνίου > αλληλεπιδραστικές διεργασίες I/O-bound > αλληλεπιδραστικές διεργασίες CPU-bound > > διεργασίες δέσμης (δες task manager Windows ή εντολή top/nice Linux)

Η εφαρμογή μπορεί να είναι έμμεση ή άμεση (ρητή αντιστοίχιση επιπέδου με κάποιο αριθμό) ή έμμεση (επιλογή του χρονοδρομολογητή σε κάθε εκτέλεσή του με βάση τις διαθέσιμες διεργασίες).

Για την αποφυγή πιθανής στέρησης ή λιμοκτονίας (παρατεταμένης μη εκτέλεσης) μιας διεργασίας το επίπεδο προτεραιότητας μπορεί να μειώνεται σε κάθε Ν κτύπους ρολογιού ή κάθε Ν context switches.

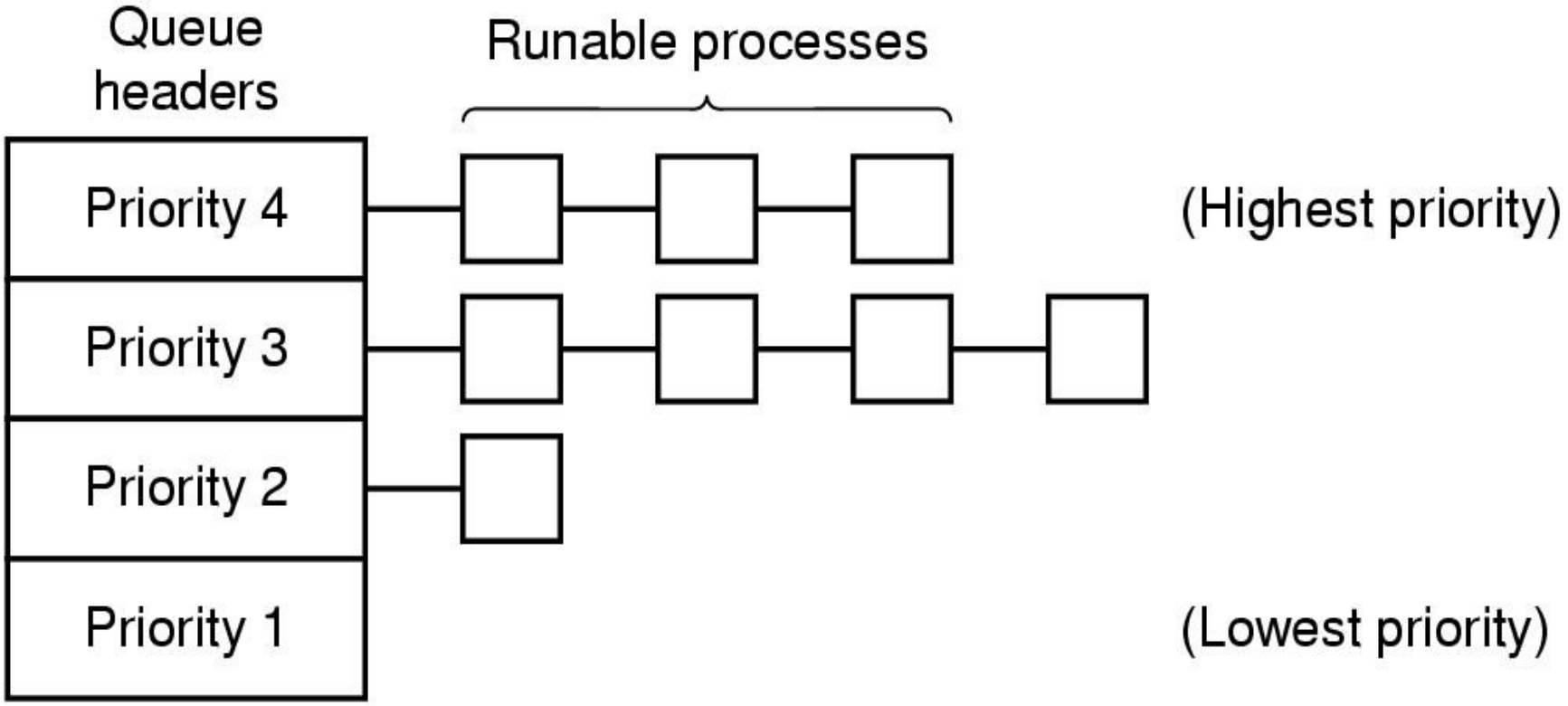
Προτεραιότητες (2)

Εναλλακτικά κάθε διεργασία μπορεί να λαμβάνει ένα μέγιστο συνολικό quantum από το οποίο αφαιρείται ένα ποσό κάθε φορά που εκτελείται.

Η προτεραιότητα μπορεί να μεταβάλλεται δυναμικά με βάση τη τρέχουσα συμπεριφορά της διεργασίας (μετάπτωση από CPU-bound σε I/O-bound ή I/O-bound μετά από μεγάλη αναμονή Ε/Ε).

Συνήθως στα σύγχρονα συστήματα τα επίπεδα προτεραιότητας υλοποιούνται με διαφορετικές ουρές. Οι διεργασίες εκχωρούνται κατ' αρχήν σε μια ουρά: Οι αλληλεπιδραστικές διεργασίες ανήκουν στις ουρές υψηλής προτεραιότητας ενώ οι διεργασίες δέσμης στις ουρές χαμηλής προτεραιότητας. Οι προσαρμογές των επιπέδων προτεραιότητας σημαίνουν μετακίνηση της διεργασίας από μια ουρά σε άλλη.

Προτεραιότητες (3)



Σύστημα με τέσσερα επίπεδα προτεραιότητας: κάθε ουρά έτοιμων εργασιών εκτελεί ξεχωριστό αλγόριθμο χρονοπρογραμματισμού (ή τον ίδιο αλγόριθμο με διαφορετικό tuning). Οι ουρές χαμηλότερης προτεραιότητας εξυπηρετούνται μόνο αν οι ουρές ανώτερης προτεραιότητας είναι άδειες.

Πολλαπλές ουρές

Δρομολόγηση με προεκτόπιση και δυναμικό μηχανισμό μεταβολής προτεραιότητας.

Διαφορετικές ουρές των έτοιμων διεργασιών με φθίνουσα σειρά προτεραιότητας :

P(RQ0) > P(RQ1) > ... > 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 (Round Robin).

Πολλαπλές ουρές - παράδειγμα

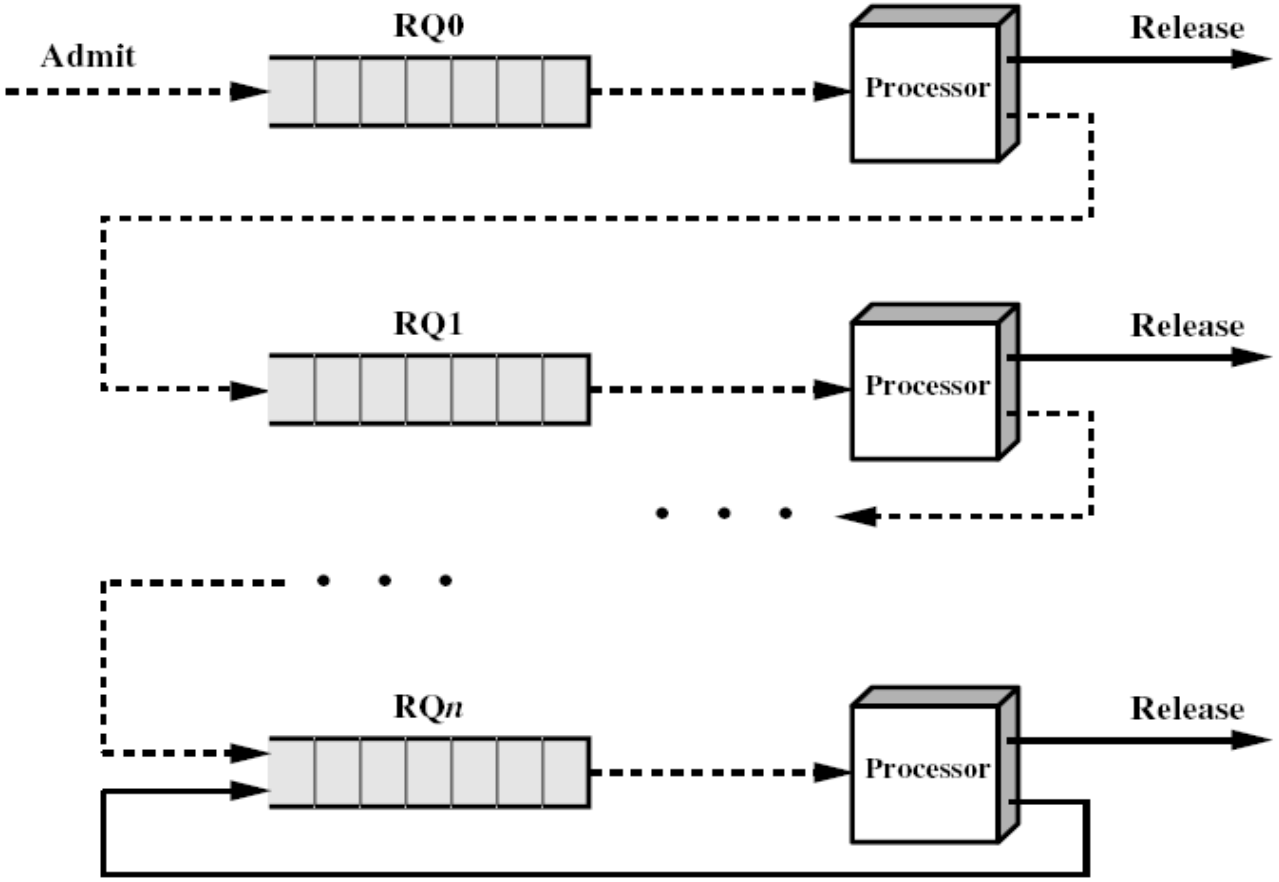
Μια νέα διεργασία τοποθετείται στην RQ0.

Όταν φθάσει στο συνολικό quantum χρόνου, τοποθετείται στην RQ1. Όταν αυτό ξανασυμβεί τοποθετείται στην RQ2... μέχρι να φθάσει στην RQn. Οι I/O-bound διεργασίες παραμένουν σε υψηλότερης προτεραιότητας ουρές. Οι CPU-bound διεργασίες κατευθύνονται χαμηλότερα.

Ο διεκπεραιωτής επιλέγει μια διεργασία για εκτέλεση στην RQi μόνον αν RQi-1 έως RQ0 είναι κενές.

Λαμβάνεται η πιθανότητα παρατεταμένης στέρησης: Παρέχεται ευελιξία στις διεργασίες των οποίων τα χαρακτηριστικά μεταβάλλονται κατά τη διάρκεια ζωής τους ώστε να μετακινηθούν σε ουρές ανώτερης προτεραιότητας.

Πολλαπλές ουρές - παράδειγμα



Πολλαπλές ουρές - παράδειγμα

Τρεις ουρές:

Q_1 – quantum 8 msecs – υψηλή προτεραιότητα

Q_2 – quantum 16 msecs – μεσαία προτεραιότητα

Q_3 – FCFS – χαμηλή προτεραιότητα

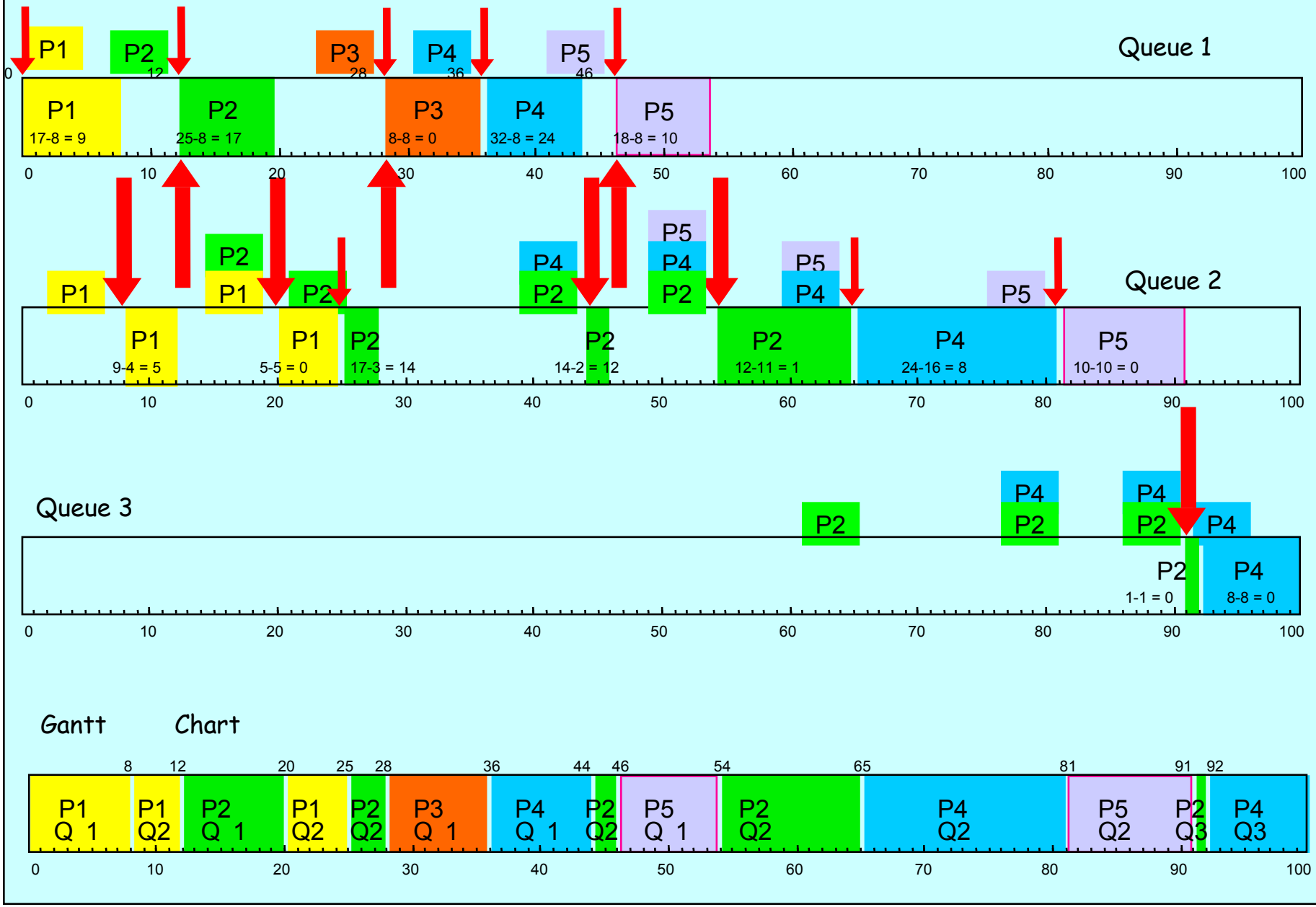
Χρονοπρογραμματισμός:

Οι νέες διεργασίες εισέρχονται στην ουρά Q_1

Οι διεργασίες με μικρούς χρόνους καταιγισμού CPU έχουν υψηλή προτεραιότητα.

Οι διεργασίες με μεγάλους χρόνους καταιγισμού μετακινούνται στην ουρά χαμηλής προτεραιότητας.

Πολλαπλές ουρές - παράδειγμα



Shortest Process Next (SPN) (1)

Τροποποίηση των αλγορίθμων SJFP ή SRTN για αλληλεπιδραστικά συστήματα.

Εκτίμηση της διάρκειας του επομένου καταιγισμού (CPU burst) κάθε διεργασίας στην ουρά έτοιμων με βάση το ιστορικό της κάθε διεργασίας,πχ

$E(i) = a * B(i-1) + (1-a) * E(i-1)$

με E(i) το εκτιμώμενο χρόνο του επομένου CPU burst,
B(i-1)το πραγματικό χρόνο του τελευταίου CPU burst
E(i-1) την προηγούμενη εκτίμηση που είχε κάνει το λ.σ.
a το συντελεστή 'γήρανσης'.

Συνήθως a = ½ (shift right 1 bit).

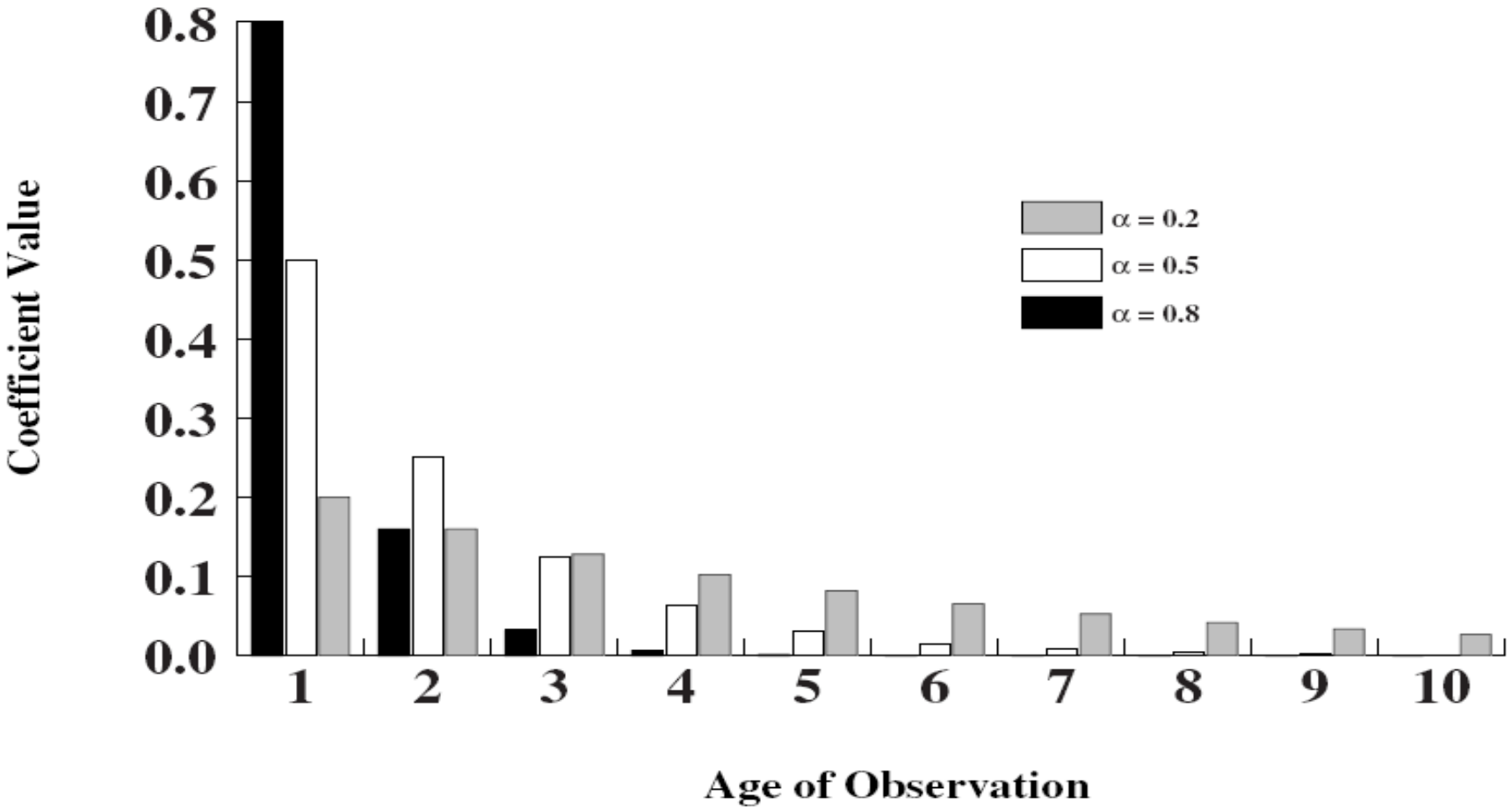
Shortest Process Next (SPN) (2)

Ο υπολογισμός της διάρκειας καταιγισμού βασίζεται στη παραδοχή οτι τα πλέον πρόσφατα στιγμιότυπα είναι καταλληλότερα για να εκφράσουν τη μελλοντική συμπεριφορά, επομένως είναι επιθυμητό να δίνεται σε αυτά μεγαλύτερο βάρος.

Η χρήση του συντελεστή γήρανσης a είναι μια κοινή τεχνική για την πρόβλεψη της μελλοντικής τιμής που προσεγγίζει καλύτερα την πραγματικότητα. Ο βεβαρυμένος μέσος όρος παρακολουθεί τις αλλαγές στη συμπεριφορά της διεργασίας ταχύτερα από τον απλό μέσον όρο.

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.

Shortest Process Next (SPN) (3)



Επίδραση του συντελεστή γήρανσης στη συμμετοχή παλαιότερων τιμών μετρήσεων και παρατηρήσεων.

Εκτίμηση χρόνου καταιγισμού

Στην πράξη, η διάρκεια του επόμενου καταιγισμού μιας διεργασίας στη CPU δεν είναι γνωστή. Ως συνέπεια, αλγόριθμοι όπως οι SJF, SRTN, SPN στην απλή τους μορφή δεν είναι εύχρηστοι.

Η διάρκεια καταιγισμού (CPU busrt) μιας διεργασίας μπορεί να υπολογιστεί ως συνάρτηση των προηγούμενων καταιγισμών της ίδιας διεργασίας.

Ο ακριβής (αναλυτικός) υπολογισμός είναι δύσκολος. Συνήθως χρησιμοποιείται απλός τύπος που δίνει έμφαση στους πρόσφατους καταιγισμούς της διεργασίας.

$E(i) = a * B(i-1) + (1-a) * E(i-1)$

δες ορισμούς στο SPN. Εδώ a = 0.75. Αρχικό B(0) τυχαίο, εδώ 10.

Παράδειγμα (1)

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

Παράδειγμα (2)

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 (3)

Time		*Estimate*		*PreviousBurst*
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 (4)

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 (5)

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**	