ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Μεταπτυχιακή Εργασία

ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ

ΤΟ ΠΕΡΙΟΔΙΚΟ ΜΟΝΤΕΛΟ

υπό

ΣΤΑΜΑΤΙΟΥ Α. ΑΣΗΜΗ

Διπλωματούχου Μηχανικού Παραγωγής & Διοίκησης Δ.Π.Θ., 2006

Υπεβλήθη για την εκπλήρωση μέρους των

απαιτήσεων για την απόκτηση του

Μεταπτυχιακού Διπλώματος Ειδίκευσης

2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Μεταπτυχιακή Εργασία

ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ

ΤΟ ΠΕΡΙΟΔΙΚΟ ΜΟΝΤΕΛΟ

υπό

ΣΤΑΜΑΤΙΟΥ Α. ΑΣΗΜΗ

Διπλωματούχου Μηχανικού Παραγωγής & Διοίκησης Δ.Π.Θ., 2006

Υπεβλήθη για την εκπλήρωση μέρους των

απαιτήσεων για την απόκτηση του

Μεταπτυχιακού Διπλώματος Ειδίκευσης

2010

Η έγκριση της μεταπτυχιακής εργασίας από το Τμήμα Μηχανολόγων Μηχανικών της Πολυτεχνικής Σχολής του Πανεπιστημίου Θεσσαλίας δεν υποδηλώνει αποδοχή των απόψεων του συγγραφέα (Ν. 5343/32 αρ. 202 παρ. 2).

Εγκρίθηκε από τα Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής (Επιβλέπων)	Δρ. Δημήτριος Παντελής Επίκουρος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Γεώργιος Λυμπερόπουλος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Γεώργιος Κοζανίδης Λέκτορας, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Πρώτα απ' όλα, θέλω να ευχαριστήσω τον επιβλέποντα της μεταπτυχιακής εργασίας μου, Επίκουρο Καθηγητή κ. Δημήτριο Παντελή, για την πολύτιμη βοήθεια και καθοδήγησή του κατά τη διάρκεια της δουλειάς μου. Επίσης, είμαι ευγνώμων στα υπόλοιπα μέλη της εξεταστικής επιτροπής της μεταπτυχιακής εργασίας μου, Καθηγητές κκ. Γεώργιο Λυμπερόπουλο, Γεώργιο Κοζανίδη, για την προσεκτική ανάγνωση της εργασίας μου και για τις πολύτιμες υποδείξεις τους. Τέλος, ευχαριστώ την οικογένεια μου, τους φίλους και τους συναδέλφους μου για την υποστήριξή τους όλα αυτά τα χρόνια.

Σταμάτιος Α. Ασημής

ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ

ΤΟ ΠΕΡΙΟΔΙΚΟ ΜΟΝΤΕΛΟ

ΣΤΑΜΑΤΙΟΣ Α. ΑΣΗΜΗΣ

Πανεπιστήμιο Θεσσαλίας, Τμήμα Μηχανολόγων Μηχανικών, 2010

Επιβλέπων Καθηγητής: Δρ. Δημήτριος Παντελής, Επίκουρος Καθηγητής Στοχαστικών Προτύπων Επιχειρησιακής Έρευνας στη Βιομηχανική Διοίκηση

Περίληψη

Τα συστήματα πραγματικού χρόνου αποτελούν ένα πεδίο στο οποίο έχει αφιερωθεί μεγάλο μέρος έρευνας και μελέτης, ειδικά τα τελευταία χρόνια. Το φάσμα εφαρμογών των συστημάτων αυτών είναι, επίσης, αρκετά ευρύ. Όπως δηλώνει και το όνομά τους, κύριο στοιχείο στα εν λόγω συστήματα είναι ο χρόνος και διαδραματίζει τον πρωταγωνιστικό ρόλο σε όλα τα ζητήματα τα οποία έχουν να κάνουν με τον σχεδιασμό, τον χρονοπρογραμματισμό και γενικότερα με την δόμηση των συστημάτων αυτών. Σε αυτήν τη μεταπτυχιακή εργασία μελετάμε την περίπτωση του Περιοδικού Μοντέλου επιδιώκοντας να επιβεβαιώσουμε μια Βέλτιστη Πολιτική Απόφασης Προτεραιότητας μεταξύ των δύο Χρηστών με την συνεχή εκτέλεση ενός αλγορίθμου που υπολογίζει τον Αναμενόμενου Αριθμού Επιτυχημένης εκτέλεσης των εργασιών των χρηστών δίνοντας τυχαίες τιμές στα αρχικά δεδομένα.

Αρχικά, αναπτύσσουμε τις βασικές έννοιες των συστημάτων πραγματικού χρόνου καθώς και την κατηγοριοποίησή τους.

Στο δεύτερο κεφάλαιο, παρουσιάζεται μια βιβλιογραφική – ερευνητική ανασκόπηση του συγκεκριμένου ερευνητικού πεδίου.

Στη συνέχεια, αναπτύσσεται αναλυτικά ο αλγόριθμος που χρησιμοποιήθηκε για τους μαθηματικούς υπολογισμούς του μοντέλου.

Τέλος, παρουσιάζονται, συγκρίνονται και αναλύονται τα αριθμητικά αποτελέσματα που προέκυψαν από την εκτέλεση του αλγορίθμου.

Πίνακας Περιεχομένων

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Κεφάλαιο 1 Συστήματα πραγματικού χρόνου

1.1 Εισαγωγή

Το κύριο χαρακτηριστικό των συστημάτων πραγματικού χρόνου είναι οι ειδικές απαιτήσεις τους σε προθεσμίες υλοποίησης (deadlines). Συνεπώς η απόδοση, η αποτελεσματικότητα και η αξιοπιστία αυτών των συστημάτων δεν εξαρτάται μόνο από τα ακριβή αποτελέσματα των υπολογισμών, αλλά κατά κύριο λόγο από τον χρόνο στον οποίο παράγονται. Κύριος σκοπός, λοιπόν, των συστημάτων πραγματικού χρόνου είναι η ικανοποίηση των χρονικών και λειτουργικών απαιτήσεων των εργασιών που εκτελούνται.

Κάθε εργασία (task) στο σύστημα έχει μία προθεσμία υλοποίησης, στα χρονικά περιθώρια της οποίας θα πρέπει να ολοκληρωθεί. Γίνεται επομένως κατανοητό, πως η αποτελεσματικότητα των συστημάτων αυτών, αφορά τον χρόνο παράδοσης ενός σωστού αποτελέσματος, ο οποίος θα πρέπει να ικανοποιεί τον περιορισμό που του έχει τεθεί. Γενικότερα, σε τέτοια συστήματα αναμένεται μία έξοδος, μέχρι την προθεσμία υλοποίησης που έχει οριστεί, η οποία είναι το αποτέλεσμα της αντίδρασης του συστήματος σε κάποιο γεγονός ή σε κάποια ενέργεια που συνέβη στο περιβάλλον. Η χρονική περίοδος από την στιγμή που συμβαίνει το γεγονός μέχρι το τελικό αποτέλεσμα (μέχρι δηλαδή την αντίδραση του συστήματος), ονομάζεται χρόνος απόκρισης και πρέπει να ικανοποιεί έναν χρονικό περιορισμό.

Οι εφαρμογές των συστημάτων πραγματικού χρόνου εκτείνονται σε ένα ευρύ φάσμα, το οποίο περικλείει ενσωματωμένα συστήματα (embedded systems), τηλεπικοινωνιακά συστήματα, πυρηνικά συστήματα επίβλεψης, ρομποτική, τραπεζικές συναλλαγές, πολυμέσα (μεταφορά εικόνας και ήχου), οπτικοποίηση ή επίβλεψη για ιατρικές ανάγκες, συστήματα αεροπορικής πλοήγησης, δρομολόγηση αμαξοστοιχιών στις κατάλληλες ράγες. Στις εφαρμογές αυτές είναι φανερό πως ο χρόνος είναι η παράμετρος την οποία πρέπει να λάβουμε υπόψη περισσότερο από κάθε άλλη, καθώς μέσω αυτής, τελικά, αποτιμάται η ποιότητα υπηρεσίας (Quality of Service QoS) του συστήματος.

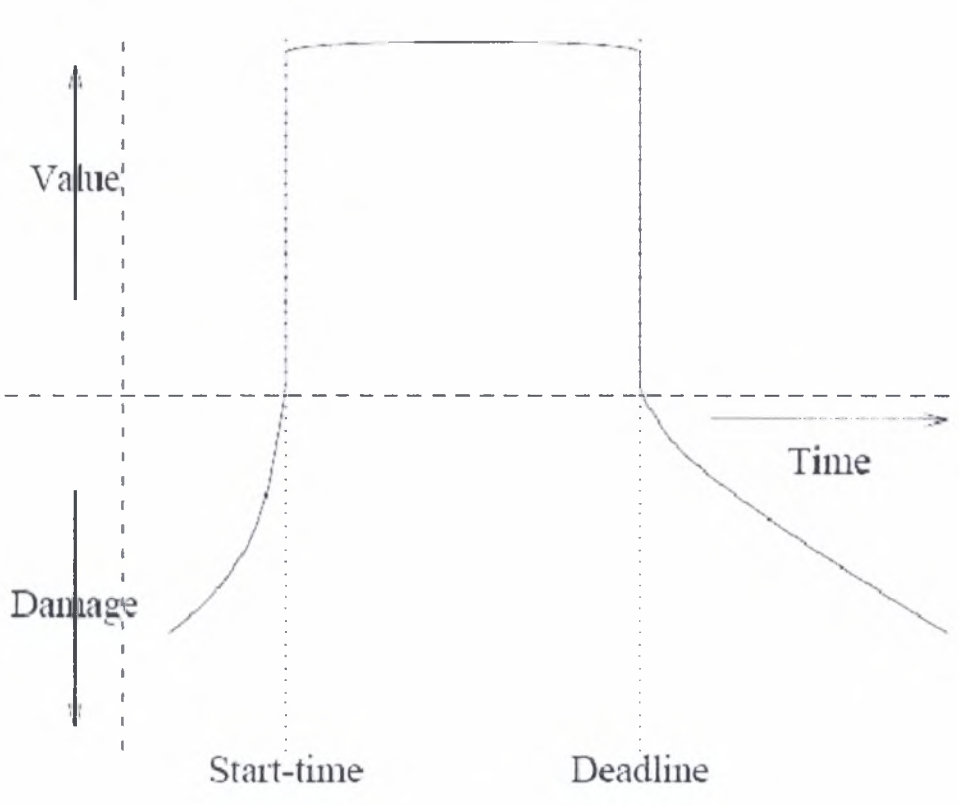
1.2 Κατηγοριοποίηση Συστημάτων Πραγματικού χρόνου

Οι απαιτήσεις της εκάστοτε εφαρμογής αλλά και οι συνέπειες σε υποτιθέμενη αποτυχία του συστήματος, οδηγούν στην κατηγοριοποίηση των συστημάτων πραγματικού χρόνου. Έτσι υπάρχουν τα αυστηρά, τα χαλαρά και τα σταθερά συστήματα πραγματικού χρόνου (hard / soft / firm real time systems).

1.2.1 Αυστηρά Συστήματα Πραγματικού Χρόνου

Ένα σύστημα ορίζεται ως αυστηρό σύστημα πραγματικού χρόνου (hard realtime system) όταν το αποτέλεσμα σε περίπτωση μη ικανοποίησης της προθεσμίας υλοποίησης είναι καταστροφικό και οι συνέπειές του ολέθριες. Τα χρονικά περιθώρια που τίθενται στα συστήματα αυτού του είδους είναι απολύτως αυστηρά και σε καμία περίπτωση δεν πρέπει να ξεπεραστούν, καθώς κάτι τέτοιο θα σήμαινε θανάσιμη καταστροφή για το σύστημα. Παραδείγματα τέτοιων συστημάτων είναι η δρομολόγηση αμαξοστοιχιών σε ράγες ή τα

ηλεκτρονικά συστήματα του αεροπλάνου. Στο παρακάτω σχήμα (Σχήμα 1-1), φαίνεται η αρχική στιγμή (start time) στην οποία συμβαίνει ένα γεγονός στο σύστημα και η προθεσμία υλοποίησης (deadline) μέχρι την οποία πρέπει να ολοκληρωθεί. Ως value, ορίζεται η συμβολή που έχει η εκτελούμενη εργασία στον γενικό αντικειμενικό σκοπό του συστήματος και μπορεί να είναι είτε θετική (θετική προσφορά στο σύστημα) είτε αρνητική (αρνητική προσφορά στο σύστημα δηλαδή καταστροφικές συνέπειες).

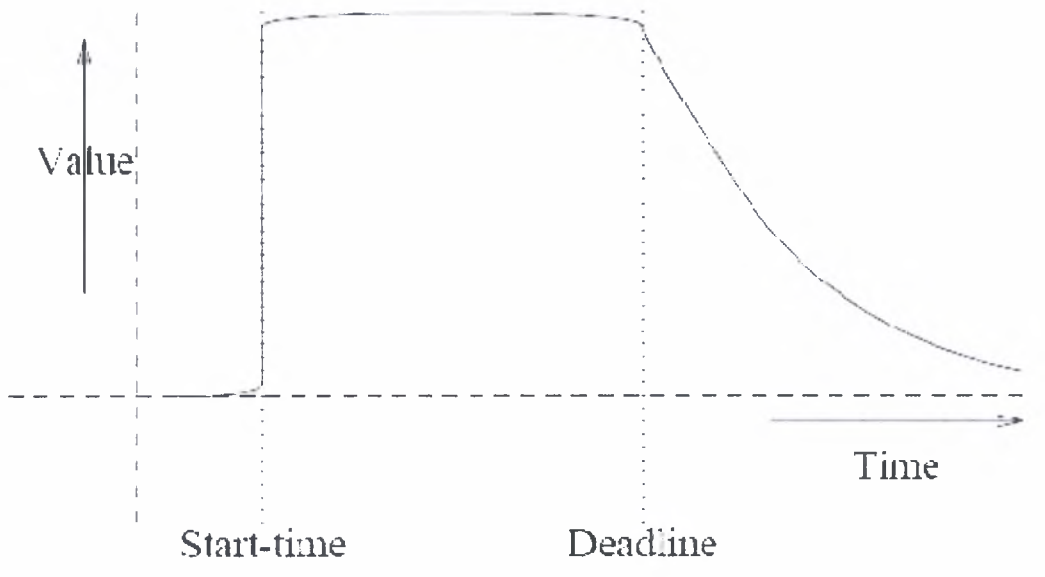


Σχήμα 1-1: Προθεσμία υλοποίησης εργασίας σε αυστηρό σύστημα.

1.2.2 Χαλαρά Συστήματα Πραγματικού Χρόνου

Ένα χαλαρό σύστημα πραγματικού χρόνου (soft real-time system) είναι αυτό στο οποίο η μη ικανοποίηση μιας προθεσμίας υλοποίησης δεν είναι σε θέση να καταστρέψει το

σύστημα ούτε να συνθλίψει την ακεραιότητά του. Στο Σχήμα 1-2 απεικονίζεται μία προθεσμία σε χαλαρό σύστημα. Όπως φαίνεται, ακόμα και μετά το πέρας της προθεσμίας υλοποίησης η συμβολή της εργασίας δεν μηδενίζεται για το σύστημα, αλλά ελαττώνεται με το χρόνο και είναι πάντα θετική. Μία υποτιθέμενη αποτυχία, λοιπόν, απλώς μειώνει την ποιότητα των παραγόμενων αποτελεσμάτων.

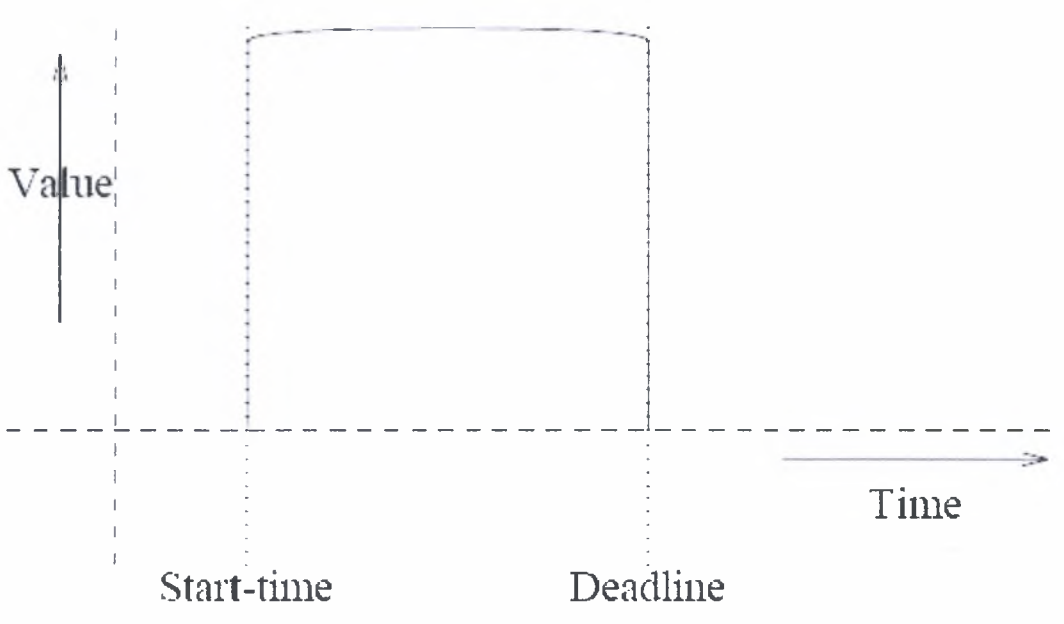


Σχήμα 1-2: Προθεσμία υλοποίησης εργασίας σε χαλαρό σύστημα.

1.2.3 Σταθερά Συστήματα Πραγματικού Χρόνου

Μεταξύ των αυστηρών και των χαλαρών συστημάτων υπάρχουν τα σταθερά συστήματα πραγματικού χρόνου (firm real-time systems). Στην περίπτωση των συστημάτων αυτών, η προθεσμία υλοποίησης δηλώνει συνήθως τον χρόνο αναμονής πριν από την έκδοση κάποιου αποτελέσματος. Η μη ικανοποίηση μιας προθεσμίας υλοποίησης δεν επιφέρει καταστροφικές συνέπειες, αλλά όταν κάποιο αποτέλεσμα λαμβάνεται πέραν αυτής, τότε είναι άχρηστο και δε λαμβάνεται υπόψη. Συνεπώς, στα σταθερά συστήματα οι προθεσμίες

υλοποίησης μπορούν να παραβιαστούν χωρίς ολέθριες συνέπειες, ωστόσο η πιθανότητα αυτή θα πρέπει να είναι αρκετά μικρή. Όπως φαίνεται και στο Σχήμα 1-3, η προσφορά της εργασίας στο σύστημα πέραν της λήξης της προθεσμίας υλοποίησης δεν είναι ούτε αρνητική ούτε θετική, αλλά μηδενική. Αυτό σημαίνει πως το αποτέλεσμα που έχει παραχθεί από την συγκεκριμένη εργασία θα επιφέρει μηδενική συμβολή στο σύστημα, εφόσον η προθεσμία έχει χαθεί. Παραδείγματα τέτοιων συστημάτων είναι τα τραπεζικά συστήματα ή τα συστήματα κράτησης αεροπορικών εισιτηρίων.

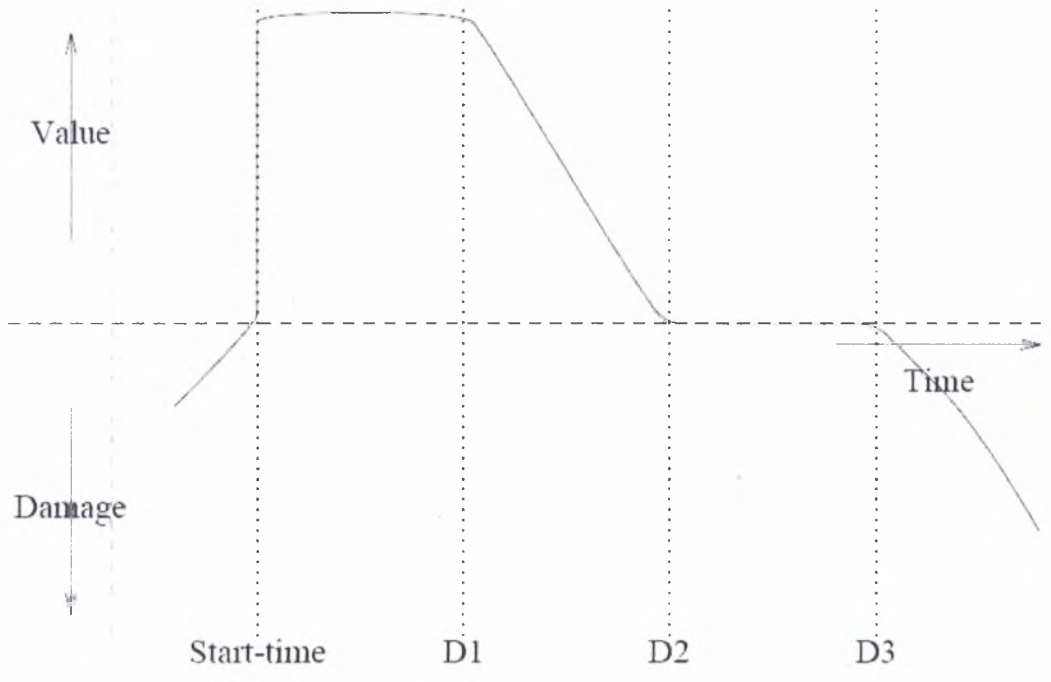


Σχήμα 1-3: Προθεσμία υλοποίησης εργασίας σε σταθερό σύστημα.

1.2.4 Υβριδικά Συστήματα Πραγματικού Χρόνου

Υπάρχουν, ωστόσο, εφαρμογές οι οποίες μπορούν να παρουσιάζουν υβριδική συμπεριφορά. Για παράδειγμα στο Σχήμα 1-4 υπάρχουν τρείς προθεσμίες υλοποίησης D1, D2 και D3. Η πρώτη αναπαριστά την «μέγιστη τιμή», η δεύτερη αναπαριστά την χρονική περίοδο στην

οποία η προσφορά θα είναι τουλάχιστον θετική στο σύστημα, ενώ η τρίτη ορίζει το σημείο στο οποίο θα υπάρξει ζημιά στο σύστημα.



Σχήμα 1-4: Υβριδικό σύστημα.

1.3 Περιοδικές και Μη περιοδικές Εργασίες

Κατά την ανάπτυξη προγραμμάτων για εφαρμογές, γίνεται συνήθως αντιστοίχιση των χρονικών απαιτήσεων με την μορφή χρονικών προθεσμιών. Το θέμα της ικανοποίησης των χρονικών προθεσμιών επαφίεται στον χρονοπρογραμματισμό των εργασιών. Υπάρχουν δύο ξεχωριστοί και απομονωμένοι τύποι εργασιών ανάλογα με την περιοδικότητά τους:

- περιοδικές εργασίες
- μη περιοδικές εργασίες

1.3.1 Περιοδικές Εργασίες

Οι περιοδικές εργασίες, όπως δηλώνει και το όνομά τους, εκτελούνται περιοδικά, και χαρακτηρίζονται από

i) την περίοδό τους T_i και

ii) τον απαιτούμενο χρόνο εκτέλεσης τους C_i (ανά περίοδο).

Στις εργασίες αυτές αναμένεται μία εκτέλεση για κάθε χρόνο T_i και για χρονικές μονάδες ίσες με C_i. Η προθεσμία υλοποίησης D_i της εργασίας αυτής είναι ίση με την περίοδο. Ο χρόνος εκτέλεσης δεν μπορεί να είναι μεγαλύτερος της περιόδου, αφού σε μία τέτοια περίπτωση θα χανόταν όλες οι προθεσμίες υλοποίησης και η λειτουργία του συστήματος πραγματικού χρόνου δε θα ήταν εφικτή. Ο χρόνος εκτέλεσης είναι ο χρόνος που αναμένεται στη χειρότερη περίπτωση (WCET – Worst Case Execution Time) που μπορεί να υπάρχει στο σύστημα, ώστε να διασφαλιστεί πως όλοι οι υπολογισμοί που γίνονται με βάση αυτό το χρόνο θα είναι ακριβείς ως προς τις χρονικές τους προθεσμίες. Ο χρόνος εκτέλεσης μπορεί να δίνεται ως το αποτέλεσμα ενός μέσου όρου μέτρησης ή/και της χειρότερης περίπτωσης του χρόνου εκτέλεσης.

1.3.2 Μη περιοδικές Εργασίες

Η ενεργοποίηση μίας μη περιοδικής εργασίας είναι ένα τυχαίο γεγονός και συνήθως πυροδοτείται από μία εξωτερική δράση στο σύστημα. Οι μη περιοδικές εργασίες έχουν επίσης χρονικούς περιορισμούς που τις αφορούν, δηλαδή αφού ξεκινήσουν την εκτέλεσή τους θα πρέπει να την ολοκληρώσουν μέσα σε μία προκαθορισμένη χρονική περίοδο. Συχνά,

αυτές οι εργασίες έχουν να κάνουν με συμβάντα μέσα στο περιβάλλον του συστήματος και έτσι οι χρονικές τους προθεσμίες είναι σημαντικές.

Οι εργασίες αυτές συνήθως αποκαλούνται event-driven / οδηγούμενες από γεγονότα, λόγω του ότι είναι απόρροια κάποιας ανεπιθύμητης ενέργειας που συμβαίνει στο σύστημα. Τα χαρακτηριστικά τους δεν είναι γνωστά εκ των προτέρων και μία μη περιοδική εργασία χαρακτηρίζεται από:

i. τον χρόνο άφιξής της A_i,
ii. τον χρόνο εκτέλεσης της C_i και
iii. την προθεσμία υλοποίησης της D_i.

Οι σποραδικές εργασίες (sporadic tasks) είναι μία ειδική κατηγορία των μη περιοδικών εργασιών. Αυτές εμφανίζονται τυχαία και απρόβλεπτα αλλά υπάρχει ένα χρονικό όριο μεταξύ των χρόνων εμφάνισης τους το οποίο δεν μπορεί να παραβιαστεί. Σε αντίθεση δηλαδή με τις μη περιοδικές εργασίες που η εμφάνισή τους είναι εντελώς απρόβλεπτη, υπάρχει ένα άνω όριο στον μέγιστο χρόνο επανάληψης.

Οι περιοδικές εργασίες με αυστηρές προθεσμίες υλοποίησης είναι αυτές που κυριαρχούν στα περισσότερα αυστηρά συστήματα πραγματικού χρόνου, ενώ οι μη περιοδικές εργασίες καταλαμβάνουν μικρό χρόνο επεξεργασίας εφόσον είναι μικρότερες αριθμητικά [2].

1.4 Χρονοδρομολόγηση Εργασιών σε Συστήματα Πραγματικού Χρόνου

1.4.1 Περιγραφή Εργασιών

Οι εργασίες πραγματικού χρόνου (tasks) είναι οι βασικές εκτελούμενες οντότητες προς χρονοδρομολόγηση. Μπορεί να είναι περιοδικές ή μη περιοδικές και να έχουν

αυστηρούς ή χαλαρούς χρονικούς περιορισμούς (time constraints). Μία εργασία ορίζεται από χρονικές παραμέτρους που δηλώνουν τις καθυστερήσεις και από χρονομετρικές παραμέτρους που δηλώνουν τον χρόνο. Το μοντέλο περιλαμβάνει βασικές και δυναμικές παραμέτρους. Οι βασικές παράμετροι είναι:

- r, χρόνος ανακοίνωσης της εργασίας, είναι ο χρόνος στον οποίον ενεργοποιείται η απαίτηση της εκτέλεσης της (task release time ή request times ή ready times),
- C, ο χρόνος εκτέλεσης της εργασίας στην χειρότερη περίπτωση, όταν ο επεξεργαστής είναι πλήρως αφοσιωμένος σε αυτήν,
- D, η σχετική προθεσμία υλοποίησης της εργασίας, π.χ. η μέγιστη αποδεκτή καθυστέρηση επεξεργασίας της,
- T , η περίοδος της εργασίας,
- Όταν η εργασία έχει αυστηρούς χρονικούς περιορισμούς, η σχετική χρονική προθεσμία επιτρέπει υπολογισμούς της απόλυτης προθεσμίας υλοποίησης (absolute deadline) $d = r + D$. Ενδεχόμενη παραβίαση της απόλυτης προθεσμίας υλοποίησης έχει ως συνέπεια χρονικό σφάλμα.

Η παράμετρος T δεν υφίσταται σε μη περιοδικές εργασίες. Μία περιοδική εργασία μοντελοποιείται σύμφωνα με τις τέσσερις παραπάνω παραμέτρους. Συνήθως χρησιμοποιείται η παρακάτω αναπαράσταση για μια εργασία τ:

$$\tau(r, C, D, T)$$

Κάθε στιγμή στην οποία μία εργασία είναι έτοιμη, εξάγει ένα περιοδικό αίτημα. Οι αλλεπάλληλοι χρόνοι ανακοίνωσης, γίνονται απαίτηση για χρόνο ανακοίνωσης γίνονται σε $r_{k=}r_0 + kT$ όπου r_0 είναι η πρώτη ανακοίνωση και r_k είναι η $(k + 1)$ οστή ανακοίνωση. Οι

αλλεπάλληλες απόλυτες προθεσμίες υλοποίησης είναι $d_k = r_k + D$. Αν $D = T$, η περιοδική εργασία έχει σχετική προθεσμία υλοποίησης ίση με την περίοδο. Μία εργασία είναι ορθώς μορφοποιημένη (well formed) εάν $0 < C \leq D \leq T$. Η ποιότητα της χρονοδρομολόγησης εξαρτάται από την ακρίβεια των παραπάνω παραμέτρων. Συνεπώς ο καθορισμός τους είναι ένας πολύ σημαντικός παράγοντας στον σχεδιασμό συστημάτων πραγματικού χρόνου. Αν οι χρονικές διάρκειες λειτουργιών όπως εναλλαγή εργασιών, κλήσεις λειτουργικού συστήματος, διακοπή επεξεργασίας ή εκτέλεσης δεν μπορούν να αγνοηθούν, η ανάλυση του σχεδιασμού πρέπει να προσθέσει αυτές τις χρονικές διάρκειες στους χρόνους υπολογισμού των εργασιών. Αυτός είναι και ο λόγος για τον οποίο απαιτείται μία ντετερμινιστική συμπεριφορά, η οποία θα εγγυάται μέγιστες τιμές για αυτές τις λειτουργίες.

Άλλες σημαντικές παράμετροι είναι:

- $u = \frac{C}{T}$, η χρησιμοποίηση του επεξεργαστή για την εργασία, πρέπει $u \leq 1$.
- $ch = \frac{C}{D}$, ο φόρτος του επεξεργαστή, πρέπει $ch \leq 1$.

Οι παρακάτω δυναμικές παράμετροι βοηθούν στην παρακολούθηση της εκτέλεσης της εργασίας:

- s, είναι ο χρόνος στον οποίο αρχίζει η εκτέλεση της εργασίας.
- e, είναι ο χρόνος στον οποίο ολοκληρώνεται η εκτέλεση της εργασίας.
- $D(t) = d - t$, είναι ο υπολειπόμενος χρόνος μέχρι την προθεσμία υλοποίησης την τυχαία στιγμή t: $0 \leq D(t) \leq D$.
- $C(t)$, είναι ο εκκρεμής χρόνος εκτέλεσης την χρονική στιγμή t: $0 \leq C(t) \leq C$.
- $L = D - C$ είναι η ονομαστική χαλαρότητα (laxity ή slack time) της εργασίας και προσδιορίζει τη μέγιστη καθυστέρηση από τον χρόνο έναρξης της όταν απασχολεί μόνο αυτή τον επεξεργαστή.

- $L(t) = D(t) - C(t)$, είναι η υπολειπόμενη ονομαστική χαλαρότητα (laxity) της εργασίας την χρονική στιγμή t και σηματοδοτεί την μέγιστη καθυστέρηση για επανεκκίνηση της εκτέλεσης, όταν η εργασία απασχολεί μόνη της τον επεξεργαστή.

Επίσης έχουμε $L(t) = D + r - t - C(t)$.

- $TR = e - r$ είναι ο χρόνος απόκρισης της εργασίας. Ισχύει $C \leq TR \leq D$ όταν δεν υπάρχει χρονικό σφάλμα.
- $CH(T) = \frac{C(t)}{D(t)}$ είναι ο υπολειπόμενος φόρτος. $0 \leq CH(t) \leq \frac{C}{T}$ (εξ' ορισμού, αν $e = d$, $CH(e) = 0$).

Οι περιοδικές εργασίες ενεργοποιούνται αλλεπάλληλα με απαιτήσεις χρόνων ανακοίνωσης (release time requests) και επιστρέφουν στην παθητική κατάσταση με την ολοκλήρωση του αιτήματός τους. Οι μη περιοδικές εργασίες μπορεί να έχουν την ίδια συμπεριφορά αν έχουν ενεργοποιηθεί περισσότερες της μίας φοράς. Μερικές φορές δημιουργούνται στον χρόνο ανακοίνωσης.

Μία εργασία, εφόσον δημιουργηθεί, μπορεί να βρίσκεται σε μία από τις δύο καταστάσεις: παθητική ή ενεργοποιημένη. Ωστόσο, ο διαμοιρασμός πόρων οδηγεί σε περισσότερες καταστάσεις για μία εργασία.

- elected/εκλεγμένη, ένας επεξεργαστής έχει προσδιοριστεί για την εργασία. Τα $C(t)$ και $D(t)$ αυξάνονται, ενώ το $L(t)$ μειώνεται.
- blocked/ μπλοκαρισμένη, η εργασία αναμένει για πόρους, για ένα μήνυμα ή για ένα σήμα συγχρονισμού. Τα $L(t)$ και $D(t)$ μειώνονται.
- ready/έτοιμη, η εργασία περιμένει να επιλεγεί. Τα $L(t)$ και $D(t)$μειώνονται.
- passive/ παθητική, η εργασία δεν έχει κάποιο τρέχον αίτημα.

- non existed/ μη υπάρχουσα, η εργασία δεν έχει δημιουργηθεί.

1.4.2 Χαρακτηριστικά Εργασιών

Οι εργασίες πέρα από τις χρονικές παραμέτρους, περιγράφονται και από άλλα χαρακτηριστικά.

Προεκχωρητικές και μη προεκχωρητικές εργασίες (preemptive and nonpreemptive tasks).

Κάποιες εργασίες από τη στιγμή που έχουν εκλεχθεί δεν είναι δυνατόν να διακοπούν αν δεν ολοκληρώσουν την εκτέλεση τους. Αυτές καλούνται μη προεκχωρητικές εργασίες ή άμεσες. Εν αντιθέσει, όταν μία εκλεγμένη εργασία μπορεί να σταματήσει την εκτέλεση της και να περάσει στην κατάσταση ready, προκειμένου να εκμεταλλευτεί τον επεξεργαστή της μία άλλη, ονομάζεται προεκχωρητική εργασία.

Στατικές και δυναμικές προτεραιότητες εργασιών.

Σε κάθε εργασία, σε ένα σύστημα πραγματικού χρόνου, ανατίθεται μία προτεραιότητα σε σχέση με τις υπόλοιπες εργασίες. Η εκτέλεση των εκάστοτε εργασιών χρονοδρομολογείται βάσει των προτεραιοτήτων τους. Αυτές οι προτεραιότητες μπορεί να είναι είτε στατικές είτε δυναμικές. Στατικές είναι οι προτεραιότητες οι οποίες ανατίθενται στις εργασίες σύμφωνα με σταθερές παραμέτρους που τις αφορούν όπως είναι π.χ. η περίοδός τους (για περιοδικές εργασίες) και παραμένουν σταθερές. Δυναμικές είναι οι προτεραιότητες οι οποίες ανατίθενται στις εργασίες βάσει δυναμικών παραμέτρων που τις αφορούν όπως π.χ. η απόλυτη προθεσμία υλοποίησης (absolute deadline) και μπορούν να μεταβάλλονται κατά τη διάρκεια της εκτέλεσης.

Εξάρτηση εργασιών (dependency of tasks).

Οι εργασίες μπορούν να αλληλεπιδρούν σύμφωνα με έναν κανόνα που καθορίζεται ή προκαλείται από μεταφορά μηνυμάτων ή από ρητό συγχρονισμό. Αυτό δημιουργεί μία σχέση προτεραιοτήτων μεταξύ των εργασιών. Οι σχέσεις προτεραιότητας είναι γνωστές πριν την εκτέλεση. Μπορεί π.χ. να είναι στατικές και να μπορούν να αναπαρασταθούν από ένα γράφημα στατικών προτεραιοτήτων. Οι εργασίες μπορούν να μοιράζονται και άλλους πόρους πέρα από τον επεξεργαστή και μερικοί πόροι μπορεί να είναι αποκλειστικοί ή κρίσιμοι (critical), π.χ. πρέπει να χρησιμοποιηθούν σε αμοιβαίο αποκλεισμό. Η ακολουθία των εντολών που μία εργασία πρέπει να εκτελέσει σε περίπτωση αμοιβαίου αποκλεισμού καλείται κρίσιμο τμήμα (critical section). Μόνο μία εργασία επιτρέπεται να εκτελεί το κρίσιμο τμήμα της για δεδομένο πόρο.

Ο διαμοιρασμός πόρων προκαλεί μία δυναμική σχέση όταν η σειρά χρήσης του πόρου εξαρτάται από την σειρά των εκλεγμένων εργασιών. Όταν οι εργασίες έχουν στατικές και δυναμικές εξαρτήσεις, που ενδεχομένως να τις διατάσσουν σειριακά, χρησιμοποιείται η έννοια του ολικού χρόνου απόκρισης, ή της end-to-end καθυστέρησης. Οι εργασίες είναι ανεξάρτητες όταν δεν υπάρχουν σχέσεις προτεραιότητας μεταξύ τους και δεν μοιράζονται κρίσιμους πόρους.

1.4.3 Χρονοδρομολόγηση

Στα συστήματα πραγματικού χρόνου, οι εργασίες έχουν χρονικούς περιορισμούς και η εκτέλεσή τους περιορίζεται σε μία μέγιστη επιτρεπτή καθυστέρηση η οποία πρέπει να γίνεται επιτακτικά σεβαστή το συχνότερο δυνατόν. Ο αντικειμενικός σκοπός της χρονοδρομολόγησης είναι να καλύπτονται πλήρως οι χρονικοί περιορισμοί όταν η εφαρμογή

τρέχει. Επίσης, πρέπει να έχει αποδειχτεί a priori, ότι όλοι οι χρονικοί περιορισμοί καλύπτονται. Όταν συμβεί μια βλάβη, ενδέχεται να ενεργοποιηθούν κάποιες εργασίες συναγερμού ή κάποιοι χρόνοι εκτέλεσης προφανώς θα αυξηθούν, υπερφορτώνοντας την εφαρμογή και αυξάνοντας τα χρονικά σφάλματα. Σε μία κατάσταση υπερφόρτωσης, ο σκοπός της χρονοδρομολόγησης είναι να επιτρέπει κάποιου είδους ανοχή, π.χ. να επιτρέπει την εκτέλεση εργασιών που κρατούν ασφαλή τον επεξεργαστή έστω και στο ελάχιστο επίπεδο απασχόλησης.

Μία εφαρμογή πραγματικού χρόνου μπορεί να προσδιορίστεί από την ομάδα των εργασιών της.

Προοδευτική ή ταυτόχρονη ενεργοποίηση (passive or simultaneous triggering).

Οι εργασίες μιας εφαρμογής ενεργοποιούνται ταυτόχρονα όταν έχουν τους ίδιους αρχικούς χρόνους ανακοίνωσης, σε αντίθετη περίπτωση ενεργοποιούνται προοδευτικά. Οι εργασίες οι οποίες ενεργοποιούνται ταυτόχρονα ονομάζονται και εργασίες σε φάση.

Παράγοντας χρησιμοποίησης επεξεργαστή (processor utilization factor).

Η χρησιμοποίηση του επεξεργαστή για μια ομάδα από n περιοδικές εργασίες είναι:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Παράγοντας φόρτου του επεξεργαστή (processor load factor).

Ο παράγοντας φόρτου του επεξεργαστή για μία ομάδα από n εργασίες είναι:

$$CH = \sum_{i=1}^{n} \frac{C_i}{D_i}$$

Χαλαρότητα του επεξεργαστή (processor laxity).

Εξαιτίας των χρονικών προθεσμιών, η χρησιμοποίηση αλλά και ο φόρτος του επεξεργαστή δεν είναι αρκετοί προκειμένου να αξιολογηθούν οι επιπτώσεις υπερφόρτωσης στους χρονικούς περιορισμούς. Έτσι, εισάγεται η χαλαρότητα του επεξεργαστή στον χρόνο t, $LP(t)$, σαν ο ανώτατος χρόνος που ο επεξεργαστής μπορεί να παραμείνει ανενεργός μετά από t χρόνο, χωρίς να προκαλεί παραβίαση προθεσμίας υλοποίησης για κάποια εργασία. Η $LP(t)$ μεταβάλλεται συναρτήσει του t. Για όλα τα t θα πρέπει να ισχύει $LP(t) \geq 0$. Για τον υπολογισμό της χαλαρότητας, θα πρέπει να είναι γνωστή η σειρά ανάθεσης των εργασιών στον επεξεργαστή και θα πρέπει να υπολογίζεται η υπό συνθήκη χαλαρότητα (conditional laxity) $LC_i(t)$ για κάθε εργασία i:

$$LC_i(t) = D_i - \sum C_j(t)$$

Όπου το άθροισμα στο j υπολογίζει τον υπολειπόμενο χρόνο εκτέλεσης όλων των εργασιών (συμπεριλαμβανομένης και της εργασίας i) που είναι ενεργοποιημένες την χρονική στιγμή t και προηγούνται της εργασίας i στην σειρά ανάθεσης. Η χαλαρότητα $LP(t)$ είναι η μικρότερη τιμή της υπό συνθήκη χαλαρότητας $LC_i(t)$.

Αδρανής χρόνος του επεξεργαστή (processor idle time).

Είναι ο χρόνος στον οποίο η χαλαρότητα του επεξεργαστή είναι αυστηρά θετική.

Κεφάλαιο 2 Βιβλιογραφική – Ερευνητική Ανασκόπηση

2.1 Εισαγωγή

Τα συστήματα πραγματικού χρόνου συνήθως διαχωρίζονται σε δύο βασικές κατηγορίες. Εκείνα στα οποία οι χρονικές απαιτήσεις είναι ρητά ντετερμινιστικές και εκείνα στα οποία οι χρονικές απαιτήσεις είναι στοχαστικές (πιθανοτικές). Ιστορικά, τα συστήματα πραγματικού χρόνου προγραμματιζόταν με τη μέθοδο της κυκλικής εκτέλεσης (cyclical executive), και ήταν κατασκευασμένα με πολύ ειδικό τρόπο. Κατά τη δεκαετία του '70 και του '80, υπήρξε μια αναπτυσσόμενη θεωρία ότι αυτή η στατική προσέγγιση, η προσέγγιση δηλαδή του χρονοπρογραμματισμού παραγωγικών συστημάτων δεν ήταν ιδιαιτέρως ευέλικτη και ήταν γενικά δύσκολο να πραγματοποιηθεί.

Τη δεκαετία του '80, με πρωτοπόρο τον Andre van Tilborg, οδηγηθήκαμε στην εκτόξευση της Πρωτοβουλίας των Συστημάτων Πραγματικού Χρόνου από το Γραφείο Ναυτικών Ερευνών των Ηνωμένων Πολιτειών. Τα περισσότερα και πιο σημαντικά από τα αποτελέσματα που θα αναφερθούν παρακάτω προέκυψαν ουσιαστικά από αυτήν την πρωτοβουλία έρευνας. Την περίοδο εκείνη, ένας μεγάλος αριθμός μεμονωμένων ερευνητών καθώς και ολόκληροι οργανισμοί συνεργάστηκαν για να αναπτύξουν μια νέα υπολογιστική υποδομή πραγματικού χρόνου, η οποία θα βασιζόταν σε μια θεωρία χρονοπρογραμματισμού με δεδομένη προτεραιότητα (fixed-priority scheduling theory), σε ένα σύνολο υποστηρικτικών ανοικτών προτύπων υλισμικού και λογισμικού, σε επιστημονικό και επαγγελματικό

εκπαιδευτικό υλικό και σε διαθέσιμα εμπορικά εργαλεία προγραμματιστικής ανάλυσης. Επιπλέον, την περίοδο εκείνη (ξεκινώντας από το 1979), ξεκίνησε μια σειρά συναντήσεων που με την πάροδο του χρόνου μετατράπηκαν σε διεθνές συνέδριο. Το IEEE Real-Time Systems Symposium έχει εξελιχθεί, τα τελευταία 30 χρόνια, το μέρος όπου δημοσιεύονται αποτελέσματα «κλειδιά» της θεωρίας χρονοπρογραμματισμού .

Στις αρχές της δεκαετίας του '90, χρηματοδοτήθηκε ένας αριθμός άλλων πρωτοβουλιών, συμπεριλαμβανομένων και μιας σειράς επιδραστικών σπουδών υπό την επίβλεψη της Ευρωπαϊκής Διαστημικής Υπηρεσίας (European Space Agency). Επιπλέον, ξεκίνησε η εμφάνιση των πρώτων γραπτών βιβλίων όπως για παράδειγμα το βιβλίο των Burns και Wellings το 1990.

Η μετάβαση από την υποδομή που ήταν βασισμένη στην κυκλική εκτέλεση, στην υποδομή που ήταν βασισμένη στη θεωρία χρονοπρογραμματισμού με δεδομένη προτεραιότητα αποτέλεσε την απαρχή μιας προσπάθειας για ενοποιημένη έρευνα και τεχνολογική μετάβαση, η οποία είχε καθοδηγητή τον Lui Sha του Carnegie Mellon University και του Software Engineering Institute. Η προσπάθεια αυτή απαρτιζόταν από:

- Τον προσδιορισμό των προβλημάτων του μοντέλου από τις προκλήσεις που έπρεπε να αντιμετωπιστούν κατά την ανάπτυξη ιδιαίτερα εξελιγμένων συστημάτων,
- Τη δημιουργία θεωρητικών αποτελεσμάτων των προβλημάτων του μοντέλου,
- Την παροχή συμβουλευτικής υποστήριξης στη βιομηχανία κατά την εφαρμογή των νέων αποτελεσμάτων,
- Την ανάπτυξη επαγγελματικού εκπαιδευτικού υλικού, όπως για παράδειγμα το "The handbook for practitioners" του Klein et al. το (1993) και

- Την αναθεώρηση των ανοιχτών προτύπων στον υπολογισμό πραγματικού χρόνου για την υποστήριξη της εφαρμογής της θεωρίας χρονοπρογραμματισμού με δεδομένη προτεραιότητα (fixed-priority scheduling theory)

Βασισμένη στην εξαιρετική δουλειά των Liu και Layland (1973), η ενοποιημένη προσπάθεια παρήγαγε το κύριο σώμα των αποτελεσμάτων του χρονοπρογραμματισμού με δεδομένη προτεραιότητα που παρουσιάζεται παρακάτω. Επιπλέον, υπήρξαν αξιοσημείωτες επιτυχίες στην εφαρμογή αυτής της θεωρίας σε εθνικά έργα υψηλής τεχνολογίας των Η.Π.Α. συμπεριλαμβανομένου της αναβάθμισης του λογισμικού για την Παγκόσμια Τοποθέτηση Δορυφόρων (Doyle and Elzey, 1994) και του Διαστημικού Σταθμού. Το 1991, η δημοσίευση ενός δελτίου του τμήματος Software Technology Strategy του Υπουργείου Άμυνας των Ηνωμένων Πολιτειών προκάλεσε την ανάπτυξη της γενικευμένης θεωρίας Χρονοπρογραμματισμού Μονοτονικού Ρυθμού (Rate Monotonic Scheduling theory) με σημαντικά οφέλη στην επιχορηγούμενη έρευνα του Υπουργείου Άμυνας. Το δελτίο ουσιαστικά παρέθετε ότι «οι σχεδιαστές συστημάτων θα μπορούσαν να χρησιμοποιήσουν τη θεωρία αυτή για να προβλέπουν αν τηρούνται οι προθεσμίες υλοποίησης των εργασιών πριν ξεκινήσει η δαπανηρή φάση υλοποίησης ενός έργου. Επιπλέον, διευκόλυνε τη διαδικασία πραγματοποίησης τροποποιήσεων στο λογισμικό». Στην ομιλία του ο Aaron Cohen (Αναπληρωτής Διαχειριστής της NASA) τον Οκτώβριο του 1992 με θέμα «Χαρτογραφώντας το Μέλλον: Προκλήσεις και Υποσχέσεις κατά την Εξερεύνηση του Διαστήματος» είπε: «...μέσω της ανάπτυξης του Μονοτονικού Ρυθμού Χρονοπρογραμματισμού (Rate Monotonic Scheduling), έχουμε ένα σύστημα που επιτρέπει στους υπολογιστές του Διεθνούς Διαστημικού Σταθμού "Freedom" να διαχειρίζονται τον χρόνο τους, να διαλέγουν μεταξύ ενός συνόλου

διαφορετικών εργασιών και να αποφασίζουν όχι μόνο ποια εργασία θα εκτελέσουν πρώτη αλλά και πόσο χρόνο θα δαπανήσουν σε κάθε διαδικασία».

Η επιτυχία στην πράξη παρείχε το κίνητρο για την αναθεώρηση των προτύπων ανοιχτών συστημάτων (open systems standards) και να δημιουργήσουν ένα λογικό σύνολο υλισμικών και λογισμικών προτύπων για να υποστηρίξουν τον υπολογισμό πραγματικού χρόνου βασισμένο στη θεωρία χρονοπρογραμματισμού με δεδομένη προτεραιότητα (fixed-priority-theory-based real-time computing).

Πριν από αυτές τις αναφορές, τα πρότυπα υπολογισμού πραγματικού χρόνου ήταν εξειδικευμένα. Υστερούσαν στα αντίστροφα προβλήματα προτεραιοτήτων (priority inversion problems) και είχαν έναν ανεπαρκή αριθμό επιπέδων προτεραιότητας για να υποστηρίξουν τις εφαρμογές πραγματικού χρόνου με χρονοπρογραμματισμό δεδομένης προτεραιότητας. Ένας μετασχηματισμός της υποδομής υπολογισμού πραγματικού χρόνου οδηγούμενος από πρότυπα (standards-driven) ξεκίνησε με τη λύση του προβλήματος αντίστροφης προτεραιότητας (Sha and Goodenough, 1990), και συνεχίστηκε με την παροχή των ικανών επιπέδων προτεραιότητας (Sha et al., 1991). Σήμερα, τα μεγαλύτερα ανοιχτά πρότυπα του υπολογισμού πραγματικού χρόνου υποστηρίζουν τον χρονοπρογραμματισμό με δεδομένη προτεραιότητα.

Μετά από αυτή την πρωτοβουλία, υπήρξε μια άνθηση του ενδιαφέροντος στα συστήματα πραγματικού χρόνου, στην έρευνα και στη δημοσίευση αναλύσεων πάνω στον χρονοπρογραμματισμό πραγματικού χρόνου. Παρακάτω παρουσιάζονται οι πέντε σημαντικότερες περιοχές της θεωρίας χρονοπρογραμματισμού πραγματικού χρόνου:

- Χρονοπρογραμματισμός δεδομένης προτεραιότητας (fixed-priority scheduling)
- Χρονοπρογραμματισμός δυναμικής προτεραιότητας (dynamic-priority scheduling)

- Ελαφριές εφαρμογές πραγματικού χρόνου (soft real-time applications)
- Χρονοπρογραμματισμός ανάδρασης (feedback scheduling)
- Εκτεταμένα μοντέλα χρονοπρογραμματισμού (extended scheduling models)

2.2 Χρονοπρογραμματισμός δεδομένης προτεραιότητας (fixed-priority scheduling)

Η αντίληψη της προτεραιότητας χρησιμοποιείται συνήθως για να υποδείξει την πρόσβαση στον επεξεργαστή και σε άλλους κοινούς πόρους όπως τα κανάλια επικοινωνίας. Κάθε εργασία έχει ένα χρόνο έναρξης (release time), έναν χρόνο υπολογισμού (computation time) και μία προθεσμία ολοκλήρωσης (deadline). Η προθεσμία υλοποίησης μπορεί να σχετίζεται με τον χρόνο έναρξης ή με έναν απόλυτο χρόνο. Στο χρονοπρογραμματισμό προτεραιοτήτων, κάθε εργασία επιφορτίζεται με μια προτεραιότητα δια μέσου κάποιας πολιτικής. Ο ανταγωνισμός για την χρήση των πόρων επιλύεται υπέρ της εργασίας με την υψηλότερη προτεραιότητα που είναι έτοιμη να εκτελεστεί.

Η φράση «χρονοπρογραμματισμός δεδομένης προτεραιότητας» έχει εφαρμογή γενικά σε εργασίες. Μία εργασία, η οποία μερικές φορές εμφανίζεται και ως διεργασία (process) ή ως νήμα (thread), είναι μια δυνητική άπειρη αλληλουχία υποεργασιών. Η διάρκεια του χρόνου μεταξύ των ενάρξεων διαδοχικών υποεργασιών μιας εργασίας τ_i είναι μία σταθερά T_i, η οποία αποκαλείται περίοδος της εργασίας. Η προθεσμία ολοκλήρωσης κάθε εργασίας είναι D_i χρονικές μονάδες μετά από το χρόνο έναρξης.

Στον χρονοπρογραμματισμό εργασιών δεδομένης προτεραιότητας, όλες οι υποεργασίες μιας εργασίας έχουν την ίδια προτεραιότητα. Συνήθως, οι εργασίες είναι αριθμημένες, έτσι ώστε η εργασια τ_i, έχει προτεραιότητα i, όπου η τιμή «1» υποδηλώνει την υψηλότερη

προτεραιότητα και μεγαλύτεροι ακέραιοι υποδηλώνουν χαμηλότερες προτεραιότητες. Οι αναθέσεις προτεραιότητας σε επίπεδο εργασιών (task-level priority) είναι γνωστές ως χρονοπρογραμματισμός «μονοτονικός γενικευμένου ρυθμού» (scheduling "generalized rate monotonic"). Σε αυτά τα συστήματα, η προτεραιότητα κάθε εργασίας υποτίθεται ότι είναι δεδομένη, αλλά ένα σύστημα μπορεί να αναλυθεί με την υπόθεση των εργασιών δεδομένης προτεραιότητας αν οι αλλαγές στην προτεραιότητα συμβαίνουν στις μεγάλες χρονικές περιόδους, ή αν οι αλλαγές εφαρμόζονται μόνο στα σύντομα χρονικά μεσοδιαστήματα, όπως τα κρίσιμα τμήματα.

2.2.1 Ανάλυση Liu και Layland

Το 1973, οι Liu και Layland δημοσίευσαν ένα άρθρο πάνω στον χρονοπρογραμματισμό περιοδικών εργασιών που γενικά θεωρείται θεμελιώδες και είναι αυτό που άσκησε τις μεγαλύτερες επιρροές πάνω στη θεωρία χρονοπρογραμματισμού δεδομένης προτεραιότητας πραγματικού χρόνου.

Οι Liu και Layland ξεκίνησαν με τις παρακάτω υποθέσεις:

1. Όλες οι εργασίες είναι περιοδικές
2. Όλες οι εργασίες ξεκινούν στην αρχή της περιόδου και έχουν προθεσμία υλοποίησης ίση με την περίοδό τους.
3. Όλες οι εργασίες είναι ανεξάρτητες, το οποίο σημαίνει ότι δεν έχουν κανένα πόρο ή σχέση προτεραιότητας.
4. Όλες οι εργασίες έχουν δεδομένο χρόνο υπολογισμού ή το τουλάχιστον ένα δεδομένο ανώτερο όριο στους χρόνους υπολογισμού, οι οποίοι είναι μικρότεροι ή ίσοι με την περίοδό τους.

5. Καμία εργασία δεν μπορεί να διακοπεί εθελοντικά από μόνη της.
6. Όλες οι εργασίες είναι πλήρως προεκχωρητικές
7. Όλα τα κόστη υποθέτουμε ότι είναι μηδέν
8. Υπάρχει ένας μόνο επεξεργαστής

Η διάρκεια του χρόνου μεταξύ των ενάρξεων διαδοχικών υποεργασιών των εργασιών τ_i, είναι μια σταθερά, T_i, η οποία αποκαλείται περίοδος της εργασίας. Κάθε εργασία έχει μια προθεσμία υλοποίησης D_i, χρονικές μονάδες μετά τον χρόνο έναρξης. Μία εργασία έχει αυστηρή προθεσμία υλοποίησης, αν κάθε εργασία πρέπει να ολοκληρωθεί μέσα στην προθεσμία υλοποίησής της.

Η ανάλυση εφικτότητας (feasibility analysis) συνηθίζεται να προβλέπει χρονικές συμπεριφορές μέσω δοκιμών οι οποίες καθορίζουν αν οι χρονικοί περιορισμοί των εργασιών θα εκπληρώνονται κατά τον χρόνο εκτέλεσης. Τέτοια ανάλυση μπορεί να χαρακτηριστεί από έναν αριθμό παραγόντων συμπεριλαμβανομένων των περιορισμών του υπολογιστικού μοντέλου (όπως για παράδειγμα, μονοεπεξεργαστές (uniprocessor), ανεξαρτησία εργασιών κλπ.) και την κάλυψη των δοκιμών εφικτότητας. Οι ικανές και οι αναγκαίες δοκιμές είναι ιδανικές, αλλά για πολλά υπολογιστικά μοντέλα τέτοιες δοκιμές είναι απείθαρχες. Πράγματι, η πολυπλοκότητα τέτοιων δοκιμών είναι Μη Πολυνομιακή (NP-hard) για τα μη επουσιώδη υπολογιστικά μοντέλα (μια γενική επισκόπηση των αποτελεσμάτων πολυπλοκότητας δίνεται από τους Garey και Johnson (1978)). Οι ικανές αλλά όχι αναγκαίες δοκιμές είναι γενικά λιγότερο πολύπλοκες, αλλά ταυτόχρονα απαισιόδοξες. Η ανάλυση εφικτότητας είναι περισσότερο επιτυχημένη στα συστήματα όπου οι σχετικές προτεραιότητες των εργασιών (όπως στο χρονοπρογραμματισμό με δεδομένη προτεραιότητα), ή τουλάχιστον οι εργασίες (όπως με τον χρονοπρογραμματισμό EDI), δεν ποικίλουν. Οι Ha και Liu (1993) έδειξαν ότι εάν ένα σύστημα ανεξάρτητων προεκχωρητικών εργασιών με δεδομένους χρόνους έναρξης

και δεδομένες προτεραιότητες εργασιών είναι εφικτό με τους χρόνους εκτέλεσης της χειρότερης περίπτωσης, παραμένει εφικτό όταν οι χρόνοι εκτέλεσης των εργασιών μειώνονται.

Η θεμελιώδης απόδειξη των Liu και Layland που αφορούσε τη εφικτότητα των συνόλων εργασιών με δεδομένη προτεραιότητα είναι γνωστή ως το Θεώρημα Κρίσιμης Στιγμής (Critical Instant Theorem). Η κρίσιμη στιγμή για μια εργασία είναι ο χρόνος έναρξης για τον οποίο ο χρόνος απάντησης μεγιστοποιείται (είτε υπερβαίνει το περιθώριο υλοποίησης, στην περίπτωση που το σύστημα είναι αρκετά υπερφορτωμένο και οι χρόνοι απάντησης αυξάνονται χωρίς όριο). Το θεώρημα λέει ότι για ένα σύνολο περιοδικών εργασιών με δεδομένες προτεραιότητες, μια κρίσιμη στιγμή για μια εργασία συμβαίνει όταν καλείται για εκτέλεση ταυτόχρονα με εργασίες υψηλότερων προτεραιοτήτων. Αυτό το γεγονός δημιουργεί μια πολύ δύσκολη κατάσταση για την εργασία τ_i ώστε να εκπληρώσει την προθεσμία υλοποίησης της. Αυτό το θεώρημα έχει αποδειχθεί ότι είναι εύρωστο (robust) και ικανοποιείται όταν οι περισσότερες από τις αυστηρές υποθέσεις σχετικά με τις περιοδικές εργασίες, που παρουσιάστηκαν παραπάνω, είναι χαλαρές.

Αν και η επιπρόσθετη έρευνα θα έβρισκε περισσότερο αποτελεσματικές τεχνικές, το Θεώρημα της Κρίσιμης Ζώνης παρείχε μια άμεση και απολύτως αναγκαία και ικανή δοκιμή για την εφικτότητα, τέτοια ώστε να εξομοιώνει την εκτέλεση του συνόλου των εργασιών, υποθέτοντας ότι όλες αρχικά ξεκινάνε μαζί, μέχρι το σημείο όπου η εργασία με τη χαμηλότερη προτεραιότητα είτε ολοκληρώνει την εκτέλεση της είτε χάνει την πρώτη προθεσμία ολοκλήρωσης της. Το σύνολο των εργασιών είναι ικανό αν και μόνο αν όλες οι εργασίες έχουν ολοκληρωθεί μέσα στις προθεσμίες ολοκλήρωσής τους. Το μόνο που χρειάζεται η προσομοίωση είναι να σκεφτούμε σημεία στο χρόνο που να αντιστοιχούν τις προθεσμίες ολοκλήρωσης των εργασιών με τους χρόνους έναρξης. Εφόσον υπάρχουν

μόνο $[D_n \,/\, T_i]$ τέτοια σημεία για κάθε εργασία τ_i, η πολυπλοκότητα της προσομοίωσης είναι $O(\sum_{i=1}^{n} D_n/T_i)$.

Βασισμένοι στην ιδέα της κρίσιμης στιγμής οι Serlin, Liu και Layland το 1967 απέδειξαν μία επαρκή υπόθεση για την εφικτότητα του γεγονότος όταν ένα σύνολο εργασιών στα οποία έχουν ανατεθεί προτεραιότητες σύμφωνα με μια πολιτική μονοτονικού ρυθμού (rate-monotonic (RM)) δηλαδή, όταν στην εργασία με την μικρότερη περίοδο δίνεται η υψηλότερη προτεραιότητα, στην εργασία με την επόμενη μικρότερη περίοδο, η δεύτερη υψηλότερη προτεραιότητα κ.ο.κ., η οποία είναι βασισμένη στη αξιοποίηση των πόρων. Η αποτελεσματικότητα της RM ανάθεσης προτεραιοτήτων για τις περιοδικές εργασίες είχε προταθεί νωρίτερα από τους Liu και Layland το 1973 και τους Wyle και Burnett το 1967, αλλά χωρίς ανάλυση ή απόδειξη. Οι Liu και Layland απέδειξαν ότι είναι η βέλτιστη στατική ανάθεση προτεραιοτήτων των εργασιών, με την έννοια ότι αν μια εργασία μπορεί να προγραμματιστεί με οποιαδήποτε ανάθεση προτεραιοτήτων είναι εφικτό να γίνει με την ανάθεση RM. Ο Serlin και η επιστημονική του ομάδα, έδειξαν επίσης ότι με αυτήν την πολιτική ο χρονοπρογραμματισμός ενός συνόλου n περιοδικών εργασιών είναι εφικτός εάν:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$$

Για παράδειγμα, ένα ζευγάρι εργασιών είναι εφικτό εάν η συνδυαστική τους αξιοποίηση πόρων (combined utilization) δεν είναι μεγαλύτερη από 82,84%. Καθώς το n προσεγγίζει το άπειρο η τιμή του $n(2^{1/n} - 1)$ προσεγγίζει το ln(2) (προσεγγιστικά 69,31%).

Μια παρανόηση που παρατηρείται συχνά είναι ότι με τον χρονοπρογραμματισμό με δεδομένη προτεραιότητα και ειδικότερα με τον χρονοπρογραμματισμό μονοτονικού ρυθμού, δεν είναι δυνατό να εγγυηθείς την εφικτότητα για οποιοδήποτε σύνολο περιοδικών εργασιών

με αξιοποίηση του επεξεργαστή μεγαλύτερη από $ln2$. Αυτό το όριο είναι «σφικτό» με την έννοια ότι υπάρχουν κάποια μη εφικτά σύνολα εργασιών με αξιοποίηση πόρων αυθαίρετα κοντά στο $n(2^{1/n} - 1)$, αλλά είναι μόνο μια επαρκής συνθήκη. Αυτό σημαίνει, ότι πολλά σύνολα εργασιών με υψηλότερη αξιοποίηση πόρων από αυτή των Liu και Layland είναι ακόμα ικανά για να προγραμματιστούν. Το 1989, ο Lehoczky και η επιστημονική του ομάδα, έδειξαν ότι η μέση πραγματική εφικτή αξιοποίηση πόρων είναι περίπου 88% για μεγάλα σύνολα εργασιών τυχαίας επιλογής. Οι απομένοντες κύκλοι μπορούν να χρησιμοποιηθούν από την εκτέλεση των εργασιών μη πραγματικού χρόνου με δευτερεύοντες προτεραιότητες (background priorities). Η επιθυμία για πιο ακριβείς δοκιμές προγραμματιστικής ισχύος δεδομένης προτεραιότητας, δηλαδή, υποθέσεις που είναι ικανές και αναγκαίες, οδήγησαν στην ανάλυση εφικτότητας η οποία περιγράφηκε νωρίτερα.

Είναι επίσης σημαντικό να παρατηρήσουμε ότι μπορεί να είναι εγγυημένη η υψηλή αξιοποίηση πόρων με την κατάλληλη επιλογή των περιόδων των εργασιών. Ειδικότερα, αν οι περίοδοι των εργασιών είναι αρμονικές (δηλαδή, κάθε περίοδος είναι ένας ακριβής ακέραιος αριθμός, πολλαπλάσιο της επόμενης μικρότερης περιόδου), τότε η προγραμματιστική ισχύς (schedulability) είναι εγγυημένη για αξιοποίηση πόρων πάνων από 100%. Οι Sha και Goodenough το 1990 έδειξαν ότι μετασχηματίζοντας τις περιόδους σε περίπου αρμονικές (με μηδέν ή μικρό υπόλοιπο της διαίρεσης των περιόδων), η προγραμματιστική ισχύς μπορεί να βελτιωθεί σημαντικά. Για παράδειγμα, για δύο εργασίες με περιόδους 10 και 15, η εργασία με την περίοδο 10 μπορεί να μετατραπεί σε μια νέα εργασία με περίοδο 5 και χρόνο εκτέλεσης ίσο με το μισό του αρχικού του χρόνου εκτέλεσης. Η προγραμματιστική ισχύς τώρα γίνεται 100%, διότι το υπόλοιπο από τη διαίρεση των περιόδων είναι μηδέν. Η τεχνική αυτή, η οποία καλείται μετασχηματισμός περιόδων, μπορεί να πραγματοποιηθεί χωρίς να αλλαχτεί ο πηγαίος κώδικας από μέσα χρήσης μιας από τις τεχνικές χρονοπρογραμματισμού μη περιοδικού

εξυπηρετητή δεδομένης προτεραιότητας, όπως είναι ο Σποραδικός Εξυπηρετητής (Sporadic Server). Υπάρχει ωστόσο, κάποια αύξηση στις «δαπάνες» του συστήματος όταν χρησιμοποιείται ο εξυπηρετητής.

2.2.2 Περεταίρω βελτιώσεις

Η επιτυχία του χρονοπρογραμματισμού δεδομένης προτεραιότητας προήλθε από τη δουλειά μιας πολύ μεγάλης ομάδας ερευνητών, οι οποίοι επέκτειναν την αρχική ανάλυση των Liu και Layland με ποικίλους τρόπους. Η συνολική έρευνα συμπληρώνεται με ένα μεγάλο αριθμό δημοσιεύσεων. Ωστόσο, μπορούμε να αναγνωρίσουμε στην ιστορική αναδρομή των Audsley το 1995 και των Sha και Goodenough το 1990, κάποια βασικά αποτελέσματα όλης αυτής της έρευνας.

1. **Ακριβής ανάλυση εφικτότητας** – Οι ικανές και αναγκαίες δοκιμές (βασισμένες στον υπολογισμό της χειρότερης περίπτωσης του χρόνου απόκρισης της εργασίας) που επιτρέπουν εγγυημένα υψηλότερα επίπεδα αξιοποίησης πόρων και οδηγούν σε περεταίρω επεκτάσεις τις θεωρίας, τέτοιες ώστε να υπερβαίνουν την ανάλυση ευαισθησίας και την ανάλυση των εργασιών με offsets χρόνους έναρξης (tasks with start-time offsets)
2. **Ανάλυση αλληλεπίδρασης των εργασιών** – Η εισαγωγή της οικογένειας των Πρωτοκόλλων Κληρονομιάς Προτεραιότητας (Priority Inheritance Protocols) επέτρεψαν τον περιορισμό και την ανάλυση ενδεχόμενου μπλοκαρίσματος.

3. **Συμπερίληψη των μη περιοδικών εργασιών** – Η εισαγωγή των μη περιοδικών εξυπηρετητών επέτρεψαν την εξυπηρέτηση των μη περιοδικών εργασιών εντός του αυστηρού μοντέλου περιοδικών εργασιών.
4. **Διαχείριση υπερφόρτωσης** – Τεχνικές για τον χειρισμό των διακυμάνσεων στους χρόνους εκτέλεσης των εργασιών που ήταν πιθανόν να χαλαρώσει τις απαιτήσεις των γνωστών χρόνων εκτέλεσης της χειρότερης περίπτωσης και επιπλέον να διασφαλίσει ότι ένα κρίσιμο υποσύνολο εργασιών θα ολοκληρωθεί στην ώρα του.
5. **Εφαρμογή απλουστεύσεων** – Η απόδειξη ότι χρειάζεται για να πραγματοποιηθεί πρακτικά μόνο ένας μικρός αριθμός εφαρμοζόμενων επιπέδων προτεραιότητας για να εφαρμόσει ευρέως τον χρονοπρογραμματισμό με δεδομένη προτεραιότητα, περιλαμβάνοντας τις καθυστερήσεις του λογισμικού.
6. **Πολυεπεξεργαστές και συστήματα διανομής** – Προσαρμογή τεχνικών ανάλυσης σε συστήματα με πολλαπλούς επεξεργαστές

Στη συνέχεια θα συζητήσουμε αυτά τα αποτελέσματα μαζί με τα μεταγενέστερα αποτελέσματα της έρευνας που προέκυψε από αυτά.

2.2.3 Ανάλυση Εφικτότητας

Η δοκιμή εφικτότητας του ορίου αξιοποίησης πόρων που περιγράφηκε παραπάνω είναι απλή και στο θέμα της σύλληψης της ιδέας και στην υπολογιστική πολυπλοκότητα. Είναι ευρέως αναγνωρισμένη και παραθέτεται συχνά. Ωστόσο, έχει κάποιους περιορισμούς:

1. Η υπόθεση εφικτότητας είναι ικανή αλλά όχι αναγκαία (δηλαδή, απαισιόδοξη)

2. Επιβάλλει μη πραγματικούς περιορισμούς πάνω στα χρονικά χαρακτηριστικά των εργασιών (δηλαδή, $D_i = T_i$)
3. Οι προτεραιότητες των εργασιών πρέπει να ανατεθούν σύμφωνα με την πολιτική μονοτονικού ρυθμού (αν οι προτεραιότητες δεν έχουν ανατεθεί με αυτό τον τρόπο τότε η δοκιμή είναι ανεπαρκής).

Στα μέσα της δεκαετίας του '80, αναπτύχθηκαν πιο σύνθετες δοκιμές εφικτότητας για να καταπιαστούν περισσότερο με τους παραπάνω περιορισμούς.

Ο Lehoczky με την επιστημονική του ομάδα το 1989 αφαίρεσε την ιδέα πίσω από τη δοκιμή προσομοίωσης εφικτότητας της Κρίσιμη Ζώνης για τον RM χρονοπρογραμματισμό, παρατηρώντας εάν ένα σύνολο εργασιών ξεκινάει μαζί το χρόνο μηδέν, η εργασία με την υψηλότερη προτεραιότητα i θα ολοκληρώσει την πρώτη της εκτέλεση μέσα στο περιθώριο υλοποίησής της αν υπάρχει ένας χρόνος t, όπου $0 < t \leq T_i$, τέτοιος ώστε η ζήτηση, στον επεξεργαστή $W_i(t)$, των εργασιών με την υψηλότερη προτεραιότητα i, να είναι μικρότερη ή ίση του t. Δηλαδή,

$$W_i(t) = \sum_{j=1}^{i} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t$$

Όταν το (t/T_j) αυξάνεται αυστηρά, εκτός από τα σημεία έναρξης των εργασιών, οι μόνες τιμές του t που πρέπει να ελεγχθούν είναι τα πολλαπλάσια των περιόδων των εργασιών μεταξύ του μηδενός και του T_i. Αυτή η δοκιμή εφαρμόζεται επίσης σε σύνολα εργασιών με αυθαίρετες εντολές (orderings) δεδομένης προτεραιότητας, ωστόσο, δεν αναγνωρίστηκε μέχρι οι Nassor και Bres (1991) να επεκτείνουν την προσέγγιση με το να επιτρέπουν στα περιθώρια υλοποίησης των εργασιών να είναι μικρότερα από την περίοδό τους.

Κατά την ίδια χρονική περίοδο, μία άλλη ομάδα ερευνητών εξέτασε το πιο γενικό πρόβλημα του καθορισμού της χειρότερης περίπτωσης του χρόνου απόκρισης μιας εργασίας, δηλαδή, του μεγαλύτερου χρόνου μεταξύ της άφιξης μιας εργασίας και της ολοκλήρωσης της επόμενης. Άπαξ και είναι γνωστός ο χρόνος απόκρισης της χειρότερης περίπτωσης μιας εργασίας, μπορεί να ελεγχθεί και η εφικτότητα μιας εργασίας συγκρίνοντας τον χρόνο απόκρισης της χειρότερης περίπτωσής του με την προθεσμία υλοποίησής της.

Η ανάλυση του χρόνου απόκρισης δεδομένης προτεραιότητας ξεκίνησε από τον Harter το 1984 και το 1987 ο οποίος χρησιμοποίησε ένα ιστορικό λογικό σύστημα απόδειξης για να αποδείξει τον Αλγόριθμο Χρονικής Διαστολής (Time Dilation Algorithm). Ισάξια ανάλυση αναπτύχθηκε ανεξάρτητα και από τους Joseph και Pandya το 1986 και από τον Audsley και την ερευνητική του ομάδα το 1991. Ο αλγόριθμος υπολογίζει τον χρόνο απόκρισης της χειρότερης περίπτωσης R_i της εργασίας τ_i ως τη λύση τελευταίου σταθερού σημείου (least-fixed-point solution) της ακόλουθης κυρτής (recursive) εξίσωσης:

$$R_i = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Οι Harter, Joseph και η επιστημονική τους ομάδα, καθώς και ο Audsley με την επιστημονική του ομάδα, παρατήρησαν ότι πρέπει να εξεταστούν για την εφικτότητα μόνο οι χρόνοι έναρξης ενός υποσυνόλου εργασιών στο διάστημα μηδέν έως T_i. Δηλαδή, η παραπάνω εξίσωση μπορεί να επιλυθεί επαναληπτικά. Ο Audsley με την επιστημονική του ομάδα το 1991 έδωσε έναν αλγόριθμο για αυτό με την επίσημη μαθηματική παρουσίαση που κατατέθηκε το 1993.

Ο Leung το 1982, μελέτησε τον χρονοπρογραμματισμό δεδομένης προτεραιότητας συνόλων εργασιών που μπορεί να έχουν προθεσμίες υλοποίησης μικρότερες από την περίοδό

τους, δηλαδή, $C_i \leq D_i \leq T_i$. Ο Leung έδειξε ότι η βέλτιστη πολιτική για τέτοια συστήματα είναι να αναθέτεις στις εργασίες με τις συντομότερες προθεσμίες υλοποίησης υψηλότερες προτεραιότητες από αυτές με μεγαλύτερες προθεσμίες υλοποίησης και την ονόμασε Χρονοπρογραμματισμό μονοτονικής προθεσμίας υλοποίησης (Deadline Monotonic (DM) Scheduling). Ο RM και ο DM χρονοπρογραμματισμός, συμπίπτουν όταν οι χρόνοι υλοποίησης είναι ίσοι με τις περιόδους και η απόδειξη της βελτιστότητας ακολουθεί τον ίδιο τρόπο συλλογισμού. Όταν οι δοκιμές του χρόνου απόκρισης που περιγράφηκαν παραπάνω, δεν εξαρτώνται από κάποια ιδιαίτερη ανάθεση προτεραιοτήτων, μπορεί να εφαρμοστούν και στον DM χρονοπρογραμματισμό, όπως και στον RM χρονοπρογραμματισμό.

Το 1990 ο Lehoczky μελέτησε μία άλλη εκδοχή του μοντέλου των Liu και Layland, όπου επιτρέπεται σε μια εργασία να έχει μεγαλύτερη προθεσμία υλοποίησης από την περίοδό της. Σε αυτή την περίπτωση, ο Lehoczky πρότεινε δύο ικανές αλλά όχι αναγκαίες δοκιμές εφικτότητας. Και οι δύο δοκιμές βασίζονται στη αξιοποίηση των πόρων, επεκτείνοντας τις δοκιμές αξιοποίησης πόρων $D_i = T_i$ των Liu και Layland. Οι δοκιμές περιορίζουν όλες τις εργασίες τ_i να έχουν $D_i = kT_i$, όπου k είναι σταθερά για όλες τις εργασίες. Η μία δοκιμή περιορίζει το k να είναι ένας ακέραιος ενώ στην άλλη δεν υπάρχει τέτοιος περιορισμός. Στις αρχές της δεκαετίας του '90, ο Tindell (1994), επέκτεινε την ανάλυση του χρόνου απόκρισης αποδεικνύοντας μία ακριβή δοκιμή για τις εργασίες με αυθαίρετες προθεσμίες υλοποίησης. Σε αντίθεση με τις δοκιμές που προτάθηκαν από τον Lehoczky η ανάλυση του Tindell δεν έθετε κανέναν περιορισμό στις περιόδους των συσχετιζόμενων εργασιών (relative task periods).

Μια επιπρόσθετη εκδοχή του μοντέλου των Liu και Layland αποτελεί η περίπτωση όπου επιτρέπεται στις εργασίες να έχουν συγκεκριμένες κλιμακώσεις (offsets (phasing)), στην οποία οι εργασίες δεν μοιράζονται ποτέ έναν κοινό χρόνο έναρξης και η χειρότερη

περίπτωση που περιγράφεται από τους Liu και Layland (όταν οι εργασίες ξεκινούν ταυτόχρονα) δεν μπορεί ποτέ να συμβεί. Όταν οι χρόνοι απόκρισης και οι υποθέσεις εφικτότητας του ορίου αξιοποίησης πόρων βασίζονται στην υπόθεση της κλιμάκωσης της χειρότερης περίπτωσης, μπορεί να είναι υπερβολικά απαισιόδοξοι για τέτοια συστήματα. Για τις περιπτώσεις αυτών των συστημάτων, για να καθοριστεί η εφικτότητα, μπορεί να χρησιμοποιηθεί η προσέγγιση των Leung και Merrill (1980) (δηλαδή, η κατασκευή ενός προγράμματος για το Ελάχιστο Κοινό Πολλαπλάσιο των περιόδων των εργασιών). Εναλλακτικά, αναπτύχθηκε από τον Audsley (1994) μια προσέγγιση που πραγματοποιεί μια ικανή και αναγκαία δοκιμή, η οποία είναι πιο αποδοτική από την εκτίμηση του προγράμματος. Ωστόσο, η μέθοδος με τις γενικευμένες κλιμακώσεις (offsets) παραμένει ένα πρόβλημα για να αναλυθεί αποδοτικά, ειδικά αν περιέχονται στο σύστημα σποραδικές εργασίες. Δυστυχώς, σε πολλά πρακτικά προβλήματα, ένα σύστημα με κλιμακώσεις μπορεί να μετασχηματιστεί σε ένα άλλο χωρίς κλιμακώσεις για σκοπούς ανάλυσης (Bate και Burns, 1997).

Όταν ισχύει η υπόθεση των καθορισμένων κλιμακώσεων, δεν είναι βέλτιστοι οι αλγόριθμοι του μονοτονικού ρυθμού και μονοτονικού περιθωρίου υλοποίησης. Ο Leung το 1982 αναρωτήθηκε εάν υπάρχει ένας αλγόριθμος πολυνομιακού χρόνου που να βρίσκει τη βέλτιστη ανάθεση προτεραιοτήτων για τέτοια σύνολα εργασιών. Αργότερα, ο Audsley (1993 και 1994) έδειξε ότι η βέλτιστη ανάθεση προτεραιοτήτων για αυτή την περίπτωση μπορεί να επιτευχθεί με την εξέταση ταξινομήσεων προτεραιοτήτων πολυνομιακού αριθμού (δηλαδή, όχι $n!$).

Ωστόσο, οι μορφές της ανάλυσης που περιγράφηκαν παραπάνω είναι όλες αποδοτικές και μπορούν εύκολα να ενσωματωθούν σε εργαλεία τα οποία χρησιμοποιούν οι ερευνητές για να συνεχίσουν την εργασία τους πάνω σε αλγορίθμους και στην παραγωγή περισσότερο

αποδοτικών σχημάτων (όπως έκαναν οι Han και Tyan το 1997, οι Sjodin και Hansson το 1998 και οι Bini και Buttazzo το 2002). Για παράδειγμα, το 2003 ο Bini και η ερευνητική του ομάδα έβγαλαν το πόρισμα μιας νέας επαρκούς δοκιμής, η οποία είχε πολυνομιακή πολυπλοκότητα και ήταν λιγότερο απαισιόδοξη από αυτή των Liu και Layland και αποδείκνυε ότι ένα σύνολο από n περιοδικές εργασίες είναι εφικτό αν

$$\prod_{i=n}^{n}\left(\frac{C_i}{T_i}+1\right)\leq 2$$

Επίσης απέδειξαν ότι αυτή η δοκιμή είναι σφικτή (tight).

2.2.4 Αλληλεπίδραση Εργασιών

Το μοντέλων των Liu και Layland υποθέτει ότι οι εργασίες είναι ανεξάρτητες. Αυτό όμως δεν συμβαίνει στα πραγματικά συστήματα. Υπάρχουν πόροι όπως οι προσωρινές μνήμες (buffers) και συσκευές υλισμικού (hardware devices), όπου οι εργασίες πρέπει να έχουν δικαίωμα πρόσβασης με έναν τρόπο αμοιβαίου αποκλεισμού. Οι μηχανισμοί που επιβάλλουν τον αμοιβαίο αποκλεισμό, μερικές φορές προκαλούν μια εργασία να μπλοκαριστεί μέχρι μια άλλη να απελευθερώσει τον πόρο. Για να είναι εφικτό ένα σύστημα εργασιών, θα πρέπει να οριοθετείται η διάρκεια αυτού το μπλοκαρίσματος. Δεδομένου τέτοιου ορίου, η ανάλυση εφικτότητας μπορεί να μη λάβει υπ' όψιν της την επίπτωση της χειρότερης περίπτωσης του μπλοκαρίσματος.

Στα προεκχωρητικά συστήματα χρονοπρογραμματισμού δεδομένης προτεραιότητας, το μπλοκάρισμα αποκαλείται και αντιστροφή προτεραιοτήτων, διότι καταλήγει στην παραβίαση ενός κανόνα του χρονοπρογραμματισμού προτεραιοτήτων (αν η εργασία με την υψηλότερη προτεραιότητα μπλοκαριστεί, θα εκτελεστεί μια άλλη χαμηλότερης

προτεραιότητας). Ο Sha παρατήρησε ότι η διάρκεια της αντιστροφής των προτεραιοτήτων που οφείλεται στο μπλοκάρισμα ίσως να είναι αυθαίρετα μεγάλη. Για παράδειγμα, μια εργασία υψηλής προτεραιότητας μπορεί να προεκχωρήσει την προτεραιότητά της σε μια άλλη χαμηλότερης προτεραιότητας. Η εργασία υψηλής προτεραιότητας τότε μπορεί να μπλοκαριστεί όταν προσπαθήσει να κάνει χρήση ενός πόρου που είναι ήδη απασχολημένος από μια άλλη χαμηλότερης προτεραιότητας και τότε την εργασία χαμηλότερης προτεραιότητας μπορεί να την προλάβουν εργασίες ενδιάμεσης προτεραιότητας, έτσι ώστε να μην είναι δυνατόν να απελευθερώθει ο πόρος και να ξεμπλοκάρει την εργασία υψηλής προτεραιότητας. Το 1987, ο Sha και η ερευνητική του ομάδα εισήγαγε την οικογένεια πρωτοκόλλων κληρονομιάς προτεραιοτήτων ως μια προσέγγιση της λύσης στο πρόβλημα αντιστροφής προτεραιοτήτων (Sha, 1987, 1990, 1991 και Rajkumar 1994). Το Πρωτόκολλο Κληρονομιάς Προτεραιοτήτων (Priority Inheritance Protocol (PIP)) προδιαγράφει ότι αν μια εργασία υψηλής προτεραιότητας μπλοκαριστεί από μια άλλη χαμηλής προτεραιότητας, η εργασία που προκαλεί το μπλοκάρισμα πρέπει να εκτελεστεί με προτεραιότητα η οποία να είναι το μέγιστο της ονομαστικής της προτεραιότητας και της υψηλότερης προτεραιότητας των εργασιών που είναι ήδη μπλοκαρισμένες.

Ωστόσο, το PIP δεν αποτρέπει τα οριστικά κλειδώματα (deadlocks). Αντίθετα, μια εργασία μπορεί να μπλοκαριστεί πολλές φορές. Μια λύση είναι να προστεθεί νέος κανόνας στο PIP: Η σύνδεση μιας προτεραιότητας με κάθε πόρο, η οποία αποκαλείται η προτεραιότητα – οροφή (priority ceiling) του πόρου. Μία εργασία μπορεί να μην εισαχθεί στο κρίσιμο τμήμα της αν η προτεραιότητα της είναι υψηλότερη από όλες τις προτεραιότητες – οροφή που είναι ήδη κλειδωμένες από άλλες εργασίες. Με αυτό τον κανόνα οροφής, μια εργασία, που μπορεί να μοιραστεί τους πόρους που ήδη αξιοποιούνται από άλλες εργασίς, δεν μπορεί να εισέλθει στο κρίσιμο τμήμα της. Αυτό αποτρέπει τα οριστικά κλειδώματα (deadlocks). Αντίθετα, με το

πρωτόκολλο προτεραιοτήτων – οροφής, μια εργασία μπορεί να μπλοκαριστεί το πολύ μία φορά. Η ιδέα των προτεραιοτήτων – οροφής είχε προταθεί νωρίτερα στα συμφραζόμενα της προγραμματιστικής γλώσσας Mesa το 1980, από τους Lampson και Redell, η οποία είναι η βάση διαφόρων πρωτοκόλλων κλειδώματος συμπεριλαμβανομένου του Πρωτοκόλλου Προτεραιοτήτων – Οροφής (Priority Ceiling Protocol (PCP)) (Sha το 1987, 1990 και 1991 και Rajkumar το 1994 και 1988), του Πρωτοκόλλου Συσσωρευμένων Πηγών (Stack Resource Protocol (SRP)) (Baker το 1991), του πρωτοκόλλου κλειδώματος της προγραμματιστικής γλώσσας Ada 95 (Baker και Pazy το 1991) και του Πρωτοκόλλου POSIX mutex PTHREAD_PRIO_PROTECT (IEEE PASC Committee το 2003).

Η ανάλυση εφικτότητας που παρουσιάστηκε παραπάνω, έχει επεκταθεί για να «λογοδοτήσει» για το μπλοκάρισμα που μπορεί να υποστούν οι εργασίες κατά τη διάρκεια του χρόνου εκτέλεσης. Στον υπολογισμό της εφικτότητας μιας εργασίας, ο χρόνος υπολογισμού φαίνεται να απαρτίζεται από τους χρόνους εκτέλεσης της χειρότερης περίπτωσής της. Η παρέμβαση της χειρότερης περίπτωσης προκύπτει από την εργασία υψηλότερης προτεραιότητας και το χρόνο της χειρότερης περίπτωσης για τον οποίο μπορεί να μπλοκαριστεί.

Το να επιτρέπεις στις εργασίες να επικοινωνούν και να συγχρονίζονται μέσω κοινόχρηστων πόρων, είναι μόνο μια μορφή αλληλεπίδρασης των εργασιών που δεν επιτρέπεται μέσα στο μοντέλο των Liu και Layland. Μια άλλη μορφή είναι οι περιορισμοί προτεραιοτήτων ή οι σχέσεις μεταξύ των εργασιών. Όταν δύο εργασίες έχουν μια σχέση προτεραιότητας μεταξύ τους, η διάδοχη εργασία δεν μπορεί να προβεί στην έναρξη της εκτέλεσης πριν την ολοκλήρωση της προαπαιτούμενης εργασίας. Ενώ η επιλογή των κλιμακώσεων και των περιόδων των εργασιών (task offsets) μπορεί να ενισχύσει αυτονόητα τις σχέσεις των προτεραιοτήτων, εμφανίστηκε σχετικά μικρός αριθμός ερευνητικών εργασιών

στη βιβλιογραφία αναφορικά με την κατηγορηματική ανάλυση των εργασιών που σχετίζονται με τις προτεραιότητες μέσα στα συστήματα δεδομένης προτεραιότητας. Ο Harbour και η επιστημονική του ομάδα το 1991 μελέτησαν τις εργασίες που υποδιαιρέθηκαν σε κομμάτια περιορισμού προτεραιότητας (precedence constrained parts), το καθένα με διαφορετική προτεραιότητα. Το 1992, Audsley και η επιστημονική του ομάδα μελέτησαν την ανάθεση προτεραιοτήτων μέσω των εργασιών περιορισμού προτεραιότητας, όπου η προτεραιότητα μιας προαπαιτούμενης εργασίας είναι τουλάχιστον αυτή όλων των διαδόχων της. Επιπροσθέτως, έκαναν επίσης μια ανάλυση για ομάδες εργασιών όπου κάθε ομάδα αποτελείται από ένα σύνολο εργασιών που συνδέονται από έναν γραμμικό περιορισμό προαπαιτούμενων. (Audsley, 1993).

2.3 Χρονοπρογραμματισμός Ανεξάρτητων Εργασιών

Στη συνέχεια περιγράφονται οι τέσσερις βασικοί αλγόριθμοι χρονοπρογραμματισμού των εργασιών. Αυτοί είναι οι:

- Μονότονου ρυθμού, Rate Monotonic RM,
- Αντίστροφης προθεσμίας υλοποίησης, Inverse Deadline ID,
- Συντομότερης προθεσμίας υλοποίησης, Earliest Deadline First EDF,
- Μικρότερης περιόδου χαλαρότητας, Least Laxity First LLF.

Αυτοί οι αλγόριθμοι έχουν να κάνουν με ομογενείς ομάδες περιοδικών ή μη περιοδικών εργασιών. Οι δυο πρώτοι αφορούν εργασίες με στατικές προτεραιότητες και οι δύο επόμενοι αφορούν εργασίες με δυναμικές προτεραιότητες. Ωστόσο οι εφαρμογές πραγματικού χρόνου απαιτούν, συνήθως, και τους δύο τύπους εργασιών.

Υπάρχουν δύο κλάσεις αλγόριθμων χρονοπρογραμματισμού:

- *Offline*. Ένας αλγόριθμος χρονοπρογραμματισμού χρησιμοποιείται offline αν εκτελείται σε όλη την ομάδα εργασιών πριν την ενεργοποίηση των εργασιών. Το πρόγραμμα που δημιουργείται με αυτόν τον τρόπο αποθηκεύεται σε ένα πίνακα και στην συνέχεια εκτελείται από έναν αποστολέα. Η ομάδα των εργασιών πρέπει να είναι γνωστή και καθορισμένη a priori, ώστε όλες οι ενεργοποιήσεις των εργασιών να μπορούν να υπολογιστούν offline. Το κύριο πλεονέκτημα αυτής της προσέγγισης είναι ότι η επιβάρυνση (overhead) του χρόνου εκτέλεσης είναι χαμηλή και δεν εξαρτάται από την πολυπλοκότητα του αλγόριθμου χρονοπρογραμματισμού. Παρόλα αυτά το σύστημα δεν είναι ευέλικτο σε ενδεχόμενες αλλαγές στο περιβάλλον.
- *Online*. Ένας αλγόριθμος προγραμματισμού χρησιμοποιείται online αν οι αποφάσεις χρονοπρογραμματισμού λαμβάνονται στον χρόνο εκτέλεσης κάθε φορά που μία εργασία εισέρχεται στο σύστημα ή όταν μία εκτελούμενη εργασία τερματίζει. Στους online αλγόριθμους σε κάθε εργασία αποδίδεται μία προτεραιότητα σύμφωνα με τις χρονικές παραμέτρους της. Αυτές οι προτεραιότητες μπορούν να είναι είτε προκαθορισμένες/στατικές (fixed) είτε δυναμικές. Οι στατικές προτεραιότητες βασίζονται σε στατικές παραμέτρους και αποδίδονται στις εργασίες πριν την ενεργοποίησή τους. Οι δυναμικές προτεραιότητες βασίζονται σε δυναμικές παραμέτρους οι οποίες ενδεχομένως να αλλάζουν κατά την εξέλιξη του συστήματος.

2.3.1 Βασικοί online αλγόριθμοι για περιοδικές εργασίες

Οι βασικοί online αλγόριθμοι σχεδιάζονται με έναν απλό κανόνα σύμφωνα με τον οποίο ανατίθενται προτεραιότητες με βάση χρονικές παραμέτρους των εργασιών. Σε περίπτωση που αυτές οι παράμετροι είναι στατικές/καθορισμένες, ο αλγόριθμος είναι στατικός διότι οι προτεραιότητες είναι στατικές. Οι προτεραιότητες ανατίθενται στις εργασίες πριν την εκτέλεση τους και δεν μπορούν να αλλάξουν. Οι βασικοί αλγόριθμοι με στατικές προτεραιότητες είναι οι μονότονου ρυθμού και αντίστροφης προθεσμίας υλοποίησης. Από την άλλη, αν ο αλγόριθμος χρονοπρογραμματισμού βασίζεται σε μεταβλητές παραμέτρους, χαρακτηρίζεται δυναμικός καθώς οι προτεραιότητες μεταβάλλονται.

2.3.1.1 Αλγόριθμοι για εργασίες με στατικές προτεραιότητες

Α) Χρονοδρομολόγηση μονότονου ρυθμού (Rate Monotonic RM)

Ο αλγόριθμος μονότονου ρυθμού αφορά προεκχωρητικές περιοδικές εργασίες με στατικές προτεραιότητες. Αυτός ο αλγόριθμος σχεδιάστηκε, αποδείχτηκε και επαληθεύτηκε από τους C. L. Liu και J. W. Layland το 1973. Σε αυτήν την κατηγορία αλγορίθμων έχουν γίνει πολλές μελέτες και έχουν προταθεί αρκετοί αλγόριθμοι.

Στην πιο απλή περίπτωση, στην οποία όλες οι εργασίες του συστήματος είναι περιοδικές, έχει αποδειχθεί ότι ο αλγόριθμος μονότονου ρυθμού είναι η βέλτιστη μέθοδος χρονοδρομολόγησης για ένα μοντέλο στατικών προτεραιοτήτων. Με τον όρο «βέλτιστος» εννοούμε ότι «κανένας άλλος στατικός αλγόριθμος δεν μπορεί να χρονοδρομολογήσει ένα σύνολο διεργασιών που ο αλγόριθμος μονότονου ρυθμού δεν μπορεί επίσης να

χρονοδρομολογήσει» και πως αν υπάρχει εφικτό πρόγραμμα για ένα δεδομένο σύνολο από εργασίες, τότε ο αλγόριθμος μονότονου ρυθμού μπορεί να το παράγει.

Στην χρονοδρομολόγηση μονότονου ρυθμού, κάθε εργασία είναι προεκχωρητική (preemptive) και της ανατίθεται μια προτεραιότητα σύμφωνα με την περίοδό της. Η πιο υψηλή προτεραιότητα αποδίδεται στην εργασία με τον πιο υψηλό ρυθμό, δηλαδή σε αυτήν που έχει τη μικρότερη περίοδο και εκτελείται συχνότερα. Η απόδοση προτεραιότητας γίνεται μόνο μία φορά στην αρχή και δεν μπορεί να αλλάξει, καθώς κάθε εργασία τη διατηρεί σε όλη τη διάρκεια λειτουργίας του συστήματος. Γι' αυτό αναφέρεται σαν αλγόριθμος που υποστηρίζει στατικές προτεραιότητες.

Ο αλγόριθμος μονότονου ρυθμού έχει δύο μέρη: το πρώτο αποφασίζει αν η δοθείσα ομάδα εργασιών μπορεί να χρονοπρογραμματιστεί με τον αλγόριθμο μονότονου ρυθμού και το δεύτερο αποδίδει στατικές προτεραιότητες στις εργασίες με την λογική που αναφέρθηκε παραπάνω.

Το πρώτο μέρος παρέχει χρονοπρογραμματιστικούς περιορισμούς στην ομάδα των εργασιών. Δίνεται ως:

$$U \leq n\left(2^{\frac{1}{n}} - 1\right)$$

Όπου n, ο αριθμός των εργασιών.

U, η συνολική χρησιμοποίηση του επεξεργαστή από τις εργασίες ώστε

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i}$$

Για πολύ μεγάλο αριθμό εργασιών n, η χρησιμοποίηση συγκλίνει στο 69%. Έτσι αν μία ομάδα εργασιών έχει χρησιμοποίηση λιγότερη του 69%, είναι εγγυημένο ότι μπορεί να

χρονοπρογραμματιστεί με τον αλγόριθμο μονότονου ρυθμού. Ωστόσο, ο περιορισμός χρονοπρογραμματισμού είναι μία αναγκαία αλλά όχι και ικανή συνθήκη. Δηλαδή, υπάρχουν ομάδες εργασιών που μπορούν να χρονοπρογραμματιστούν με τον αλγόριθμο μονότονου ρυθμού αλλά δεν ικανοποιούν το όριο της χρησιμοποίησης.

Βρέθηκε, λοιπόν, ένας ικανός και αναγκαίος περιορισμός χρονοπρογραμματισμού. Η ομάδα εργασιών περιγράφηκε παραπάνω αποτελείται από n εργασίες.

$$\{\tau_1, \tau_2 \dots, \tau_j, \dots, \tau_n\}$$

όπου οι εργασίες είναι με κατιούσα σειρά προτεραιότητας. Η περίοδος της εργασίας τ_j είναι T_j. Ο φόρτος εργασίας στο διάστημα $t[0,t]$ (για αυθαίρετο $t > 0$) λόγω όλων των εργασιών που έχουν μικρότερη ή ίση προτεραιότητα με την τ_j δίνεται ως:

$$W_j(t) = \sum_{i=1}^{j} C_j \left\lceil \frac{t}{T_j} \right\rceil$$

Η εργασία τ_j είναι χρονοπρογραμματίσιμη αν για όλες τις εργασίες κάτω από αυτήν ισχύει

$$min_{(0<t\leq T_j)} \frac{W_j(t)}{t} \leq 1$$

ολόκληρη η ομάδα εργασιών είναι χρονοπρογραμματίσιμη αν η παραπάνω συνθήκη ισχύει για όλες τις εργασίες.

Ένα ακόμη πρόβλημα με τον αλγόριθμο μονότονου ρυθμού είναι ότι δεν επιτρέπει δυναμικές αλλαγές στις προτεραιότητες των εργασιών. Αν για κάποιο λόγο κατά τη διάρκεια λειτουργίας του συστήματος πραγματικού χρόνου αλλάξει η περίοδος εκτέλεσης μιας εργασίας, ο αλγόριθμος μονότονου ρυθμού δεν μπορεί να κάνει ξανά ανάθεση προτεραιοτήτων, καθώς κάτι τέτοιο παραβιάζει την παραδοχή ότι οι προτεραιότητες είναι

στατικές. Έχουν προταθεί αρκετές παραλλαγές του αλγόριθμου μονότονου ρυθμού, όπως είναι ο αλγόριθμος μονότονης προθεσμίας υλοποίησης (deadline-monotonic scheduling algorithm) ή ο αλγόριθμος μονότονου βάρους (weight-monotonic scheduling algorithm).

Β) Χρονοδρομολόγηση αντίστροφης ή μονότονης προθεσμίας υλοποίησης (Inverse deadline or deadline monotonic algorithm)

Ο αλγόριθμος αντίστροφης προθεσμίας υλοποίησης αποδίδει προτεραιότητες στις εργασίες βάσει των χρονικών τους προθεσμιών. Στην εργασία με την μικρότερη προθεσμία υλοποίησης αποδίδεται η μεγαλύτερη προτεραιότητα, ενώ στην εργασία με την μεγαλύτερη προθεσμία υλοποίησης αποδίδεται η χαμηλότερη προτεραιότητα. Ο αλγόριθμος αντίστροφης προθεσμίας υλοποίησης είναι βέλτιστος, με την έννοια ότι αν κάποιος αλγόριθμος στατικών προτεραιοτήτων μπορεί να χρονοπρογραμματίσει μία ομάδα εργασιών με προθεσμίες υλοποίησης μικρότερες των περιόδων, τότε και ο αλγόριθμος αντίστροφης προθεσμίας υλοποίησης μπορεί επίσης να χρονοπρογραμματίσει αυτή την ομάδα εργασιών. Για έναν τυχαίο αριθμό από n εργασίες με προθεσμίες υλοποίησης μικρότερες των περιόδων μία ικανή συνθήκη είναι η εξής:

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n\left(2^{\frac{1}{n}} - 1\right)$$

2.3.1.2 Αλγόριθμοι για εργασίες με δυναμικές προτεραιότητες

Α) Αλγόριθμος συντομότερης προθεσμίας υλοποίησης (Earliest DeadlineFirst, EDF)

Ο αλγόριθμος αυτός σχεδιάστηκε και επαληθεύτηκε από τους Liu και Layland το 1973. Σύμφωνα με αυτόν οι προτεραιότητες στις εργασίες αποδίδονται με βάση την απόλυτη χρονική τους προθεσμία. Η εργασία με την μικρότερη προθεσμία υλοποίησης έχει την μεγαλύτερη προτεραιότητα. Ο αλγόριθμος είναι επίσης βέλτιστος με την έννοια που αναλύθηκε παραπάνω. Όταν μία νέα εργασία φτάνει στο σύστημα, ο χρονοδρομολογητής ξαναϋπολογίζει τις προτεραιότητες όλων των εργασιών που είναι σε εκκρεμότητα και στη συνέχεια αναδιοργανώνει τη σειρά της επεξεργασίας. Εάν η νέα άφιξη έχει την πιο υψηλή προτεραιότητα, η εργασία που εκτελείται εκείνη τη στιγμή προεκχωρείται (γίνεται preempted) και η νέα εργασία εξυπηρετείται αμέσως. Η εργασία που διακόπηκε θα ξεκινήσει αργότερα από το σημείο διακοπής.

Διαφορετικά, αν η προτεραιότητα της νέας άφιξης είναι μικρότερη, θα τοποθετηθεί σε κατάλληλη θέση στη σειρά των εργασιών. Μία ομάδα περιοδικών εργασιών, με προθεσμίες υλοποίησης ίσες με την περίοδο, είναι χρονοπρογραμματίσιμη με τον EDF αν και μόνο αν η χρησιμοποίηση του επεξεργαστή είναι μικρότερη ή ίση της μονάδας:

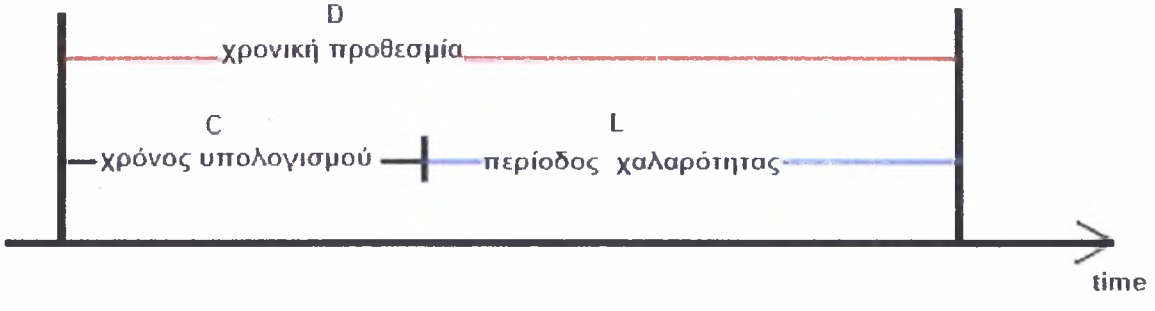
$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

Τα δύο μεγάλα πλεονεκτήματα του αλγόριθμου EDF είναι αφενός ότι μπορεί να επιτύχει μέχρι και 100% χρησιμοποίηση του επεξεργαστή και αφετέρου ότι υποστηρίζει δυναμικές προτεραιότητες. Επίσης, ο EDF αλγόριθμος δεν κάνει καμία υπόθεση περί των περιόδων των εργασιών. Έτσι μπορεί να χρησιμοποιείται για τον χρονοπρογραμματισμό τόσο περιοδικών όσο και μη περιοδικών εργασιών.

Υπάρχουν και σε αυτήν την περίπτωση μελέτες που επεκτείνουν τον αρχικό αλγόριθμο, όπως για παράδειγμα ο αλγόριθμος πρόβλεψης προθεσμίας υλοποίησης (Predictive Deadline algorithm) ή ο πολύ ενδιαφέρων αλγόριθμος του εξυπηρετητή ολικού εύρους ζώνης (Total Bandwidth Server), ο οποίος είναι πολύ σημαντικός, γιατί βασίζεται στο γεγονός ότι πολλά συστήματα πραγματικού χρόνου αποτελούνται από αυστηρές περιοδικές εργασίες (hard periodic tasks) και σταθερές μη περιοδικές εργασίες (firm aperiodic tasks).

Β) Αλγόριθμος της μικρότερης περιόδου χαλαρότητας (Least Laxity First, LLF)

Ο αλγόριθμος αυτός είναι μία παραλλαγή του αλγόριθμου συντομότερης προθεσμίας υλοποίησης (EDF). Εισάγει την έννοια της περιόδου χαλαρότητας. Όταν λέμε περίοδος χαλαρότητας, εννοούμε τη χρονική διάρκεια μέχρι τη στιγμή που η εργασία δεν έχει άλλα περιθώρια να καθυστερήσει την εκτέλεσή της. Όπως επισημάνθηκε και παραπάνω η περίοδος χαλαρότητας ορίζεται ως $L = D - C$. Δηλαδή είναι το αποτέλεσμα της αφαίρεσης του χρόνου υπολογισμού χειρότερης περίπτωσης μιας εργασίας, μείον την χρονική της προθεσμία.



Σχήμα 2-1: Περίοδος χαλαρότητας / laxity για μία εργασία.

Ο αλγόριθμος αποδίδει την υψηλότερη προτεραιότητα στην εργασία με την μικρότερη περίοδο χαλαρότητας. Είναι βέλτιστος με την έννοια που αναλύθηκε παραπάνω και μπορεί να χρονοπρογραμματίσει όποια ομάδα εργασιών χρονοπρογραμματίζει και ο αλγόριθμος EDF.

Μια περίοδος χαλαρότητας t_i υποδηλώνει ότι ακόμα και αν η εργασία καθυστερήσει για t_i χρονικές μονάδες, θα μπορέσει να προλάβει τη προθεσμία υλοποίησης της. Αν η περίοδος χαλαρότητας μιας εργασίας είναι 0, αυτό σημαίνει ότι η εργασία πρέπει να αρχίσει να εκτελείται αμέσως. Σε διαφορετική περίπτωση η χρονική της προθεσμία δεν θα ικανοποιηθεί. Το μέτρο της "περιόδου χαλαρότητας" επιτρέπει την απόρριψη των εργασιών που σίγουρα δε θα προλάβαιναν να εκτελεστούν πριν από τη χρονική τους προθεσμία.

Η κύρια διαφορά μεταξύ του LLF και του EDF είναι ότι ο LLF λαμβάνει υπόψη τους χρόνους εκτέλεσης κάθε εργασίας. Αξίζει να σημειωθεί ότι ο αλγόριθμος αυτός παρουσιάζει υψηλή χρησιμοποίηση και συμπεριφέρεται καλύτερα από τον EDF σε κατάσταση υπερφόρτωσης του συστήματος.

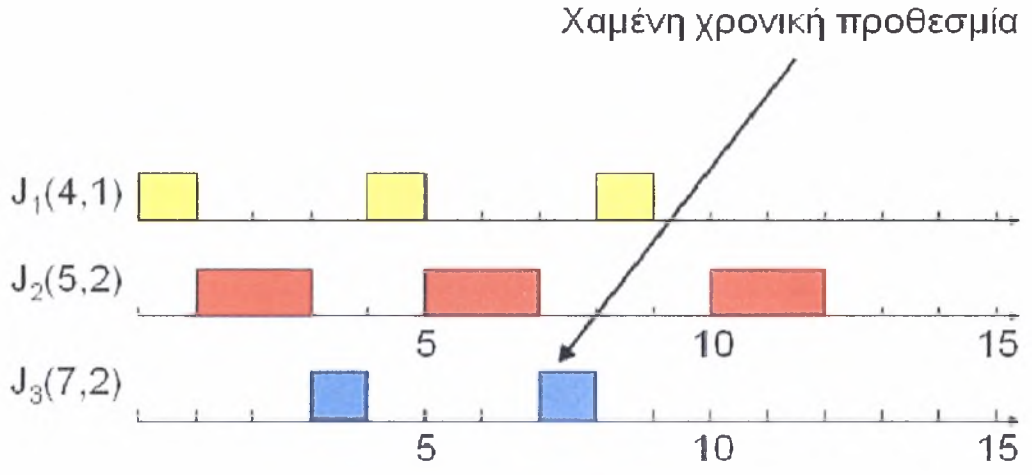
2.3.1.3 Παραδείγματα των RM και EDF αλγορίθμων χρονοδρομολόγησης - Σύγκριση

Α) Παράδειγμα RM αλγορίθμου

Έστω ότι σε ένα σύστημα πραγματικού χρόνου υπάρχει ένα σύνολο από 3 περιοδικές προεκχωρητικές εργασίες J_1, J_2 και J_3 με περιόδους και χρόνους εκτέλεσης αντίστοιχα όπως δίνονται παρακάτω:

- $J_1 = (4,1)$
- $J_2 = (5,2)$
- $J_3 = (7,2)$

Στο παρακάτω Σχήμα 2-2 φαίνεται ο χρονοπρογραμματισμός των εργασιών σύμφωνα με τον αλγόριθμο RM :



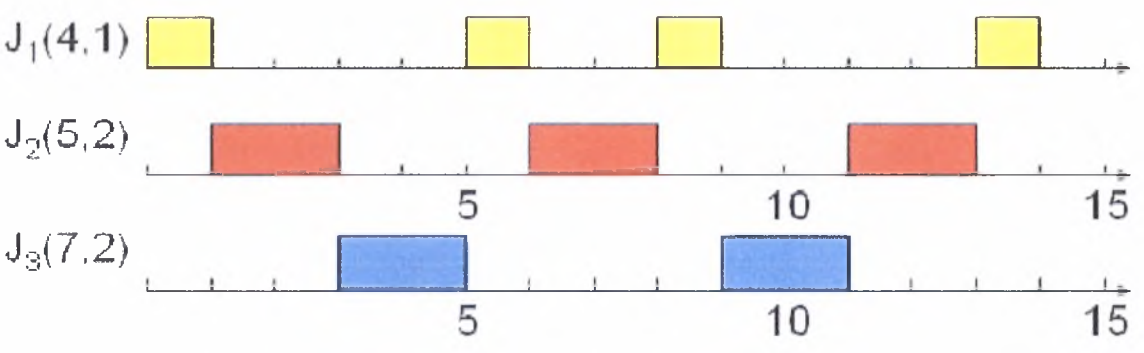
Σχήμα 2-2: Χρονοπρογραμματισμός εργασιών με τον RM αλγόριθμο

Η μεγαλύτερη προτεραιότητα βάσει του αλγορίθμου (απόδοση υψηλότερης προτεραιότητας στην εργασία με τη μικρότερη περίοδο) δίνεται στην εργασία J_1, η μεσαία προτεραιότητα στην εργασία J_2 και η χαμηλότερη προτεραιότητα στην εργασία J_3, αφού οι περίοδοι είναι αντίστοιχα 4, 5 και 7. Στην αρχή εκτελείται η J_1 και καθώς ο χρόνος εκτέλεσής της είναι μία χρονική μονάδα, επιλέγεται μετά η J_2. Η J_2 έχει χρόνο εκτέλεσης ίσο με 2 χρονικές μονάδες και μόλις τελειώσει την εκτέλεσή της η μόνη που απομένει είναι η J_3, καθώς οι άλλες εργασίες έχουν ήδη ολοκληρώσει την εκτέλεσή τους για την τρέχουσα περιοδό τους. Κατά τη χρονική στιγμή 4 συμβαίνει κάτι που δημιουργεί πρόβλημα στο σύστημα. Ενώ η J_3 δεν έχει ολοκληρώσει την εκτέλεσή της, αρχίζει ο νέος κύκλος της περιόδου για την J_1, οπότε λόγω του ότι έχει μεγαλύτερη προτεραιότητα από την J_3, θα την εκτοπίσει (η J_3 θα γίνει preempted) και θα αρχίσει την εκτέλεσή της η J_1. Για την J_3 έχει απομείνει μία χρονική μονάδα χρόνου εκτέλεσης. Ωστόσο, μόλις τελειώσει την εκτέλεσή της η J_1, έχει ξεκινήσει ο δεύτερος κύκλος για την εκτέλεση της J_2 και καθώς έχει μεγαλύτερη προτεραιότητα από την J_3, θα ξεκινήσει την εκτέλεσή της. Όταν ολοκληρώνεται η εκτέλεση της J_2 μετά από δύο χρονικές μονάδες, τη χρονική στιγμή 7 είναι πλέον αργά για την εργασία

J_3, καθώς έχει ξεπεραστεί η προθεσμία υλοποίησης που είχε για την εκτέλεσή της μέσα στην πρώτη της περίοδο. Θα μπορούσε να προβλεφθεί η χρήση μιας χρονικής μονάδας, για να προλάβει να εκτελεστεί η J_3, αλλά σύμφωνα με τον αλγόριθμο οι προτεραιότητες ανατίθενται στην αρχή και δεν μπορούν να αλλάξουν.

Β) Παράδειγμα EDF αλγορίθμου

Στο παρακάτω σχήμα φαίνεται ο χρονοπρογραμματισμός των εργασιών με τον αλγόριθμο EDF:



Σχήμα 2-3: Χρονοπρογραμματισμός εργασιών με τον EDF αλγόριθμο

Στο σχήμα αυτό δεν υπάρχουν χαμένες προθεσμίες υλοποίησης. Στη χρονική στιγμή 0 αποφασίζεται η εκτέλεση της εργασίας J_1 όχι επειδή έχει τη μικρότερη περίοδο, αλλά επειδή το τέλος της προθεσμίας υλοποίησης της είναι στη χρονική στιγμή 4, ενώ για τις άλλες δύο είναι 5 και 7 αντίστοιχα. Παρομοίως στη δεύτερη χρονική στιγμή, αποφασίζεται η εκτέλεση της εργασίας J_2, επειδή η J_1 έχει ολοκληρώσει την εκτέλεσή της και η προθεσμία υλοποίησης της J_3 αργεί περισσότερο. Παρατηρείται κάτι πολύ σημαντικό. Κατά τη χρονική στιγμή 4 θα ξεκινήσει ο νέος κύκλος για την εργασία J_1. Ωστόσο, αντίθετα με τον αλγόριθμο RM, δε

σταματά η εκτέλεση της εργασίας J_3, αλλά συνεχίζεται κανονικά, καθώς η προθεσμία υλοποίησης της εργασίας J_1 αργεί περισσότερο. Σε αυτήν την περίπτωση η εργασία J_1 έχει μικρότερη προτεραιότητα από την J_3. Παρατηρείται λοιπόν ότι η εργασία J_3 έχει προλάβει να εκτελεστεί πριν από το πέρας της προθεσμίας υλοποίησης και αυτό επιτυγχάνεται με τη δυναμική αλλαγή προτεραιότητας. Ενώ στην αρχή (στη χρονική στιγμή 0) η εργασία J_1 έχει μεγαλύτερη προτεραιότητα από την J_3, τα πράγματα αλλάζουν στη χρονική στιγμή 4, όπου η εργασία J_1 έχει μικρότερη προτεραιότητα από την J_3.

Πρέπει να πούμε ότι ο αλγόριθμος της μικρότερης "περιόδου χαλαρότητας" λειτουργεί με τον ίδιο τρόπο όπως και ο EDF, μόνο που αντί για να ελέγχεται η προθεσμία υλοποίησης, ελέγχεται η "περίοδος χαλαρότητας".

Θα μπορούσε αβίαστα να εξαχθεί το συμπέρασμα ότι ο αλγόριθμος EDF είναι καλύτερος από τον RM. Ωστόσο, δεν παρέχει την απαιτούμενη προβλεψιμότητα και έχει το πρόβλημα του ντόμινο (domino effect), καθώς σε περίπτωση υπερφορτώσεων του συστήματος αρχίζουν διαδοχικές χαμένες προθεσμίες, κάτι που δε θα γινόταν με τον RM, όπου κάποιες εργασίες έχουν σταθερά υψηλή προτεραιότητα και σίγουρα δε θα χάσουν τη προθεσμία υλοποίησης τους. Ακολουθεί μία σύγκριση των δύο βασικών αλγορίθμων, κατά την οποία γίνονται κατανοητά τα πλεονεκτήματα και τα μειονεκτήματά τους.

Γ) Σύγκριση RM και EDF

Συγκρίνοντας τους δύο αυτούς αλγόριθμους διαπιστώνει κανείς ότι ο EDF είναι πιο δυναμικός. Πρέπει όμως να εκτελείται συχνά και έτσι προκαλεί μεγαλύτερη χρονική επιβάρυνση χρονοδρομολόγησης. Το πλεονέκτημα είναι ότι μπορεί να επιτύχει χρησιμοποίηση του επεξεργαστή και μέχρι 100%, ενώ ο αλγόριθμος RM μέχρι 69%. Ο

αλγόριθμος του μονότονου ρυθμού αντίθετα είναι στατικός, αφού η προτεραιότητα που αποδίδεται στις εργασίες υπολογίζεται μόνο μια φορά. Αυτό όμως προσδίδει προβλεψιμότητα στην εκτέλεση των εργασιών. Επειδή οι προτεραιότητες αποδίδονται σύμφωνα με το ρυθμό αίτησης των διεργασιών, περισσότερες εναλλαγές περιεχομένου συμβαίνουν κατά τη χρονοδρομολόγηση μονότονου ρυθμού παρά στον EDF. Επειδή η χρονοδρομολόγηση μονότονου ρυθμού δεν εισάγει καθόλου καθυστερήσεις χρονοδρομολόγησης και είναι βέλτιστη για περιοδικές εργασίες, είναι κατάλληλη για εφαρμογές συνεχών μέσων. Ο EDF έχει κακή συμπεριφορά σε κατάσταση υπερφόρτωσης του συστήματος. Ο RM έχει απλή υλοποίηση ακόμα και σε συστήματα χωρίς σαφή υποστήριξη για χρονικούς περιορισμούς (περιόδους και προθεσμίες υλοποίησης).

2.4 Χρονοπρογραμματισμός εξαρτημένων εργασιών

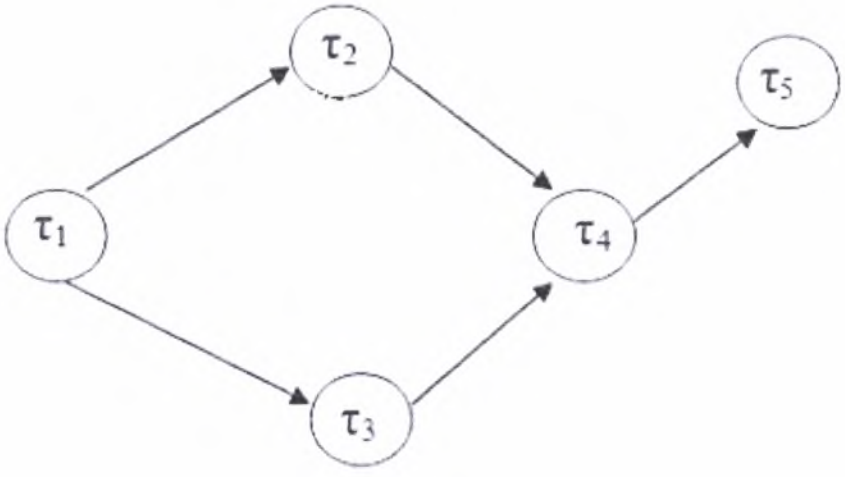
Μέχρι τώρα μιλήσαμε για ανεξάρτητες εργασίες οι οποίες δεν έχουν σχέσεις μεταξύ τους. Όμως, σε πάρα πολλά συστήματα πραγματικού χρόνου, οι διαεργασιακές εξαρτήσεις είναι απαραίτητες. Η διαεργασιακή επικοινωνία μπορεί να εκφραστεί με διάφορους τρόπους: κάποιες εργασίες πρέπει να σέβονται την σειρά επεξεργασίας, την ανταλλαγή μηνυμάτων μεταξύ των εργασιών, την χρησιμοποίηση διαφόρων πόρων. Σε εργασίες πραγματικού χρόνου υπάρχουν δύο είδη τυπικών εξαρτήσεων που μπορούν να οριστούν.

1. Εξάρτηση των περιορισμών που αντικατοπτρίζονται στον συγχρονισμό ή την επικοινωνία μεταξύ των εργασιών
2. Αμοιβαίοι απαγορευτικοί περιορισμοί για την προστασία των διαμοιραζόμενων πόρων.

2.4.1 Εργασίες με σχέσεις προτεραιότητας

Ο πρώτος τύπος περιορισμού είναι οι σχέσεις προτεραιότητας μεταξύ των εργασιών πραγματικού χρόνου. Ορίζουμε έναν περιορισμό προτεραιότητας μεταξύ δύο εργασιών τ_i και τ_j, σημειώνοντας τον ως $\tau_i \longrightarrow \tau_j$, αν η εκτέλεση της εργασίας τi προηγείται από αυτή της τ_j. Με άλλα λόγια η εργασία τ_j θα πρέπει να περιμένει την εκτέλεση της εργασίας τ_i πριν ξεκινήσει την δική της εκτέλεση.

Αυτές οι σχέσεις μπορούν να περιγραφούν με ένα γράφημα όπου οι κόμβοι αναπαριστούν τις εργασίες και τα βέλη τις σχέσεις προτεραιότητας μεταξύ των κόμβων όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 2-4: Παράδειγμα γραφήματος σε σχέσεις προτεραιότητας ανάμεσα σε 5 εργασίες

2.4.2 Διαμοιρασμός πόρων από εργασίες

Όταν επιτρέπεται στις εργασίες να διαμοιράζονται πόρους, η πρόσβασή τους σε αυτούς πρέπει να ελέγχεται. Ας υποθέσουμε έναν πόρο R διαμοιραζόμενο από δύο εργασίες

τ_1 και τ_2. Πρέπει να εξασφαλίσουμε ότι η εκτέλεση των δύο προεκχωρητικών εργασιών πάνω στον πόρο γίνεται κάτω από συνθήκες αμοιβαίου αποκλεισμού.

Ας υποθέσουμε ότι η εργασία τ_1 ενεργοποιείται πρώτη και χρησιμοποιεί τον πόρο R, δηλαδή εκτελεί το κρίσιμο τμήμα της. Έπειτα η εργασία τ_2, έχουσα μεγαλύτερη προτεραιότητα, αιτείται της χρησιμοποίησης τους πόρου. Εφόσον έχει μεγαλύτερη προτεραιότητα από την εργασία τ_1, προκύπτει προεκχώρηση. Η εργασία τ_1 μπλοκάρεται και η τ_2 αρχίζει την εκτέλεσή της. Ωστόσο, όταν η τ_2 επιθυμεί πρόσβαση στον διαμοιραζόμενο πόρο R, αποκλείεται λόγω του αμοιβαίου αποκλεισμού. Επομένως η τ_1 μπορεί να συνεχίσει την εκτέλεσή της. Όταν η τ_1 τελειώσει την εκτέλεση του κρίσιμου τμήματος της, η μεγαλύτερης προτεραιότητας εργασία τ_2 μπορεί να εκτελεστεί χρησιμοποιώντας τον πόρο. Αυτή η διεργασία μπορεί να οδηγήσει σε ανεξέλεγκτο μπλοκάρισμα της εργασίας τ_2.

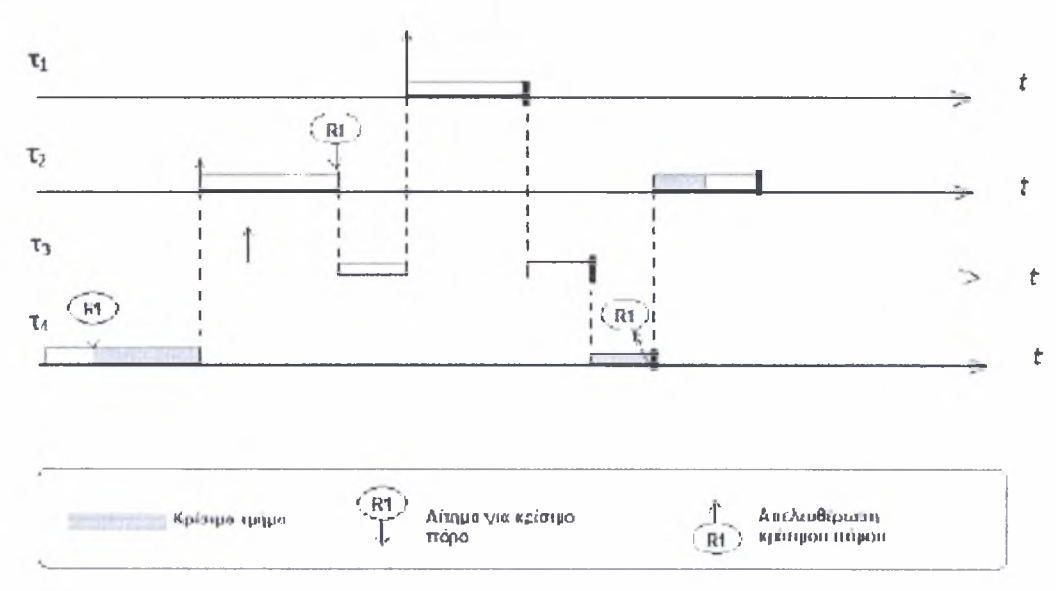
Για την ικανοποίηση αυστηρών απαιτήσεων πραγματικού χρόνου, μία εφαρμογή πρέπει να ελέγχεται από έναν χρονοπρογραμματιστικό αλγόριθμο ο οποίος να μπορεί πάντα να εγγυηθεί ένα προβλέψιμο χρόνο απόκρισης του συστήματος. Είναι επιτακτική ανάγκη λοιπόν να διασφαλίζεται ένας προβλέψιμος χρόνος απόκρισης μέσω ενός προεκχωρητικού χρονοπρογραμματιστικού αλγορίθμου, ο οποίος να λαμβάνει υπόψη τους περιορισμούς που τίθενται από τους πόρους.

2.4.2.1 Φαινόμενο αντιστροφής προτεραιοτήτων (Priority inversion phenomenon)

Το φαινόμενο της αντιστροφής προτεραιότητας μπορεί να συμβεί στον προεκχωρητικό χρονοπρογραμματισμό οδηγούμενο από καθορισμένες προτεραιότητες όπου οι κρίσιμοι πόροι προστατεύονται από τον μηχανισμό του αμοιβαίου αποκλεισμού (Kaiser, 1981, Rajkumar, 1990). Προκειμένου να εξηγηθεί αυτό το πρόβλημα, ας υποθέσουμε μία

ομάδα εργασιών αποτελούμενη από τέσσερις εργασίες $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ με μειούμενη σειρά προτεραιοτήτων. Οι εργασίες τ_2 και τ_4 μοιράζονται έναν κρίσιμο πόρο, R_1, η πρόσβαση στον οποίο είναι αμοιβαίως αποκλειόμενη.

Ας επικεντρώσουμε την προσοχή μας στον χρόνο απόκρισης της εργασίας τ_2. Η σειρά χρονοπρογραμματισμού φαίνεται στο παρακάτω σχήμα. Η χαμηλότερης προτεραιότητας εργασία τ_4 ξεκινά πρώτη την εκτέλεσή της και μετά από κάποιο χρονικό διάστημα εισέρχεται στο κρίσιμο τμήμα της χρησιμοποιώντας τον πόρο R_1. Όταν η εργασία τ_4 βρίσκεται στο κρίσιμο τμήμα της, η υψηλότερης προτεραιότητας εργασία τ_2 ελευθερώνεται και προεκχωρεί την εργασία τ_4. Κατά την διάρκεια της εκτέλεσης της τ_2 ελευθερώνεται η τ_3, όμως λόγω της χαμηλότερης προτεραιότητάς της σε σχέση με την τ_2 υποχρεούται να περιμένει. Όταν η εργασία τ_2 πρέπει να εισέλθει στο κρίσιμο τμήμα της που συνδέεται με τον κρίσιμο πόρο R_1 τον οποίο διαμοιράζεται με την τ_4, ανακαλύπτει ότι ο πόρος R_1 δεσμεύεται από την τ_4. Έτσι, μπλοκάρεται. Η υψηλότερης προτεραιότητας εργασία η οποία είναι σε θέση να εκτελεστεί είναι η τ_3. Συνεπώς, η τ_3 δεσμεύει τον πόρο και εκτελείται.



Σχήμα 2-5: Παράδειγμα του φαινομένου αντίστροφης προτεραιότητας

Κατά τη διάρκεια αυτής της εκτέλεσης η εργασία τ_1 ξυπνάει, με αποτέλεσμα η εργασία τ_3 να αναβληθεί και ο πόρος να ανατεθεί στην εργασία τ_1. Στο τέλος της εκτέλεσης της εργασίας τ_1, η εργασία τ_3 μπορεί να συνεχίσει την εκτέλεσή της. Τώρα μόνο η εργασία τ_4 μπορεί να εκτελεστεί, η οποία υπέστη προεκχώρηση στο κρίσιμο τμήμα της. Συνεχίζει λοιπόν την εκτέλεσή της μέχρι να απελευθερώσει τον πόρο για την μεγαλύτερης προτεραιότητας εργασία τ_2. Έπειτα αυτή η εργασία μπορεί να συνεχίσει την εκτέλεση της κρατώντας τον πόρο για τον εαυτό της.

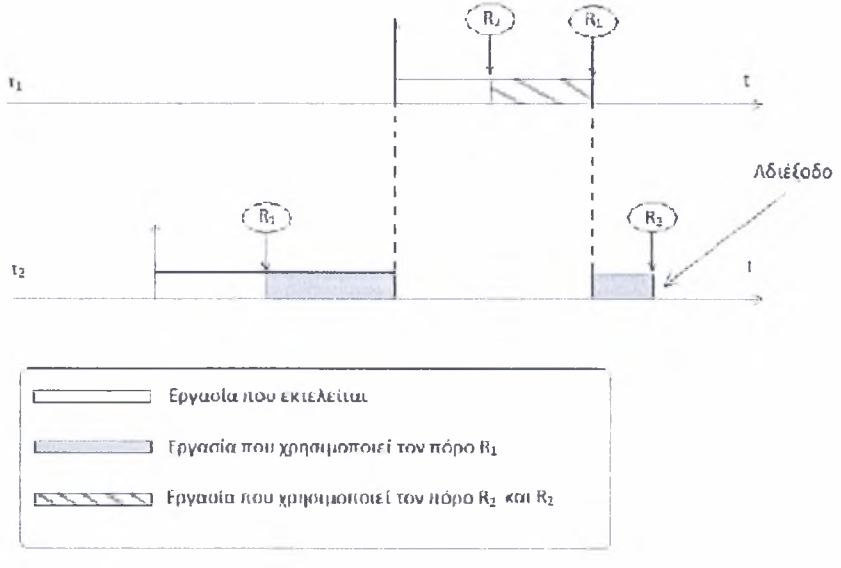
Ο μέγιστος χρόνος μπλοκαρίσματος στον οποίο υπόκειται η εργασία τ_2, εξαρτάται από τη διάρκεια των κρίσιμων τμημάτων των εργασιών χαμηλότερης προτεραιότητας οι οποίες διαμοιράζονται κάποιον πόρο μαζί της, όπως η τ_4, και από την άλλη από τους χρόνους εκτέλεσης των εργασιών υψηλότερης προτεραιότητας, όπως η τ_1. Αυτοί οι δύο παράγοντες αύξησης του χρόνου απόκρισης της εργασίας τ_2 είναι απολύτως σύμφωνοι με τους κανόνες χρονοπρογραμματισμού. Αλλά η εργασία τ_3 που έχει χαμηλότερη προτεραιότητα και δεν μοιράζεται κάποιον κρίσιμο πόρο με την εργασία τ_2, συμμετέχει στην αύξηση του χρόνου μπλοκαρίσματός της. Αυτό ονομάζεται αντιστροφή προτεραιότητας και αντικρούεται με τους χρονοπρογραμματιστικούς ορισμούς.

2.4.2.2 Φαινόμενο αδιέξοδου (Deadlock phenomenon)

Όταν κάποιες εργασίες μοιράζονται δύο ή περισσότερους κρίσιμους πόρους μπορεί να εμφανιστεί το φαινόμενο του αδιέξοδου. Στην περίπτωση αυτή η εφαρμογή πραγματικού χρόνου αποτυγχάνει.

Προς κατανόηση του φαινομένου δίνουμε το παρακάτω παράδειγμα. Ας σκεφτούμε δυο εργασίες, έστω τ_1 και τ_2 , οι οποίες μοιράζονται δύο κρίσιμους πόρους R_1 και R_2. Οι

εργασίες τ_1 και τ_2 έχουν πρόσβαση στους κρίσιμους πόρους με αντίστροφη σειρά. Ακόμη η εργασία τ_1 έχει μεγαλύτερη προτεραιότητα από την εργασία τ_2. Ας υποθέσουμε τώρα ότι η εργασία τ_2 εκτελείται πρώτη και δεσμεύει τον πόρο R_1. Κατά τη διάρκεια του κρίσιμου τμήματος της εργασίας τ_2 η οποία χρησιμοποιεί τον πόρο R_1, η εργασία τ_1 ξυπνάει και προεκχωρεί την εργασία τ_2 πριν μπορέσει να δεσμεύσει τον πόρο R_2 Η εργασία τ_1 χρειάζεται πρώτα τον πόρο R_2 ο οποίος είναι ελεύθερος και έτσι τον δεσμεύει. Έπειτα, η εργασία τ_1 χρειάζεται τον πόρο R_1 που είναι δεσμευμένος από την εργασία τ_2. Έτσι, η εργασία τ_2 ξαναρχίζει και ζητά τον πόρο R_2 ο οποίος δεν είναι ελεύθερος. Το τελικό αποτέλεσμα είναι ότι η εργασία τ_2 έχει στην διάθεσή της τον πόρο R_1, αλλά περιμένει για τον πόρο R_2 και η εργασία τ_1 έχει στη διάθεσή της τον πόρο R_2, αλλά περιμένει τον R_1. Καμία από τις δύο εργασίες δεν είναι διατεθειμένη να απελευθερώσει τον πόρο της αν δεν ικανοποιηθεί το εκκρεμές αίτημά της. Επομένως, η κατάσταση αυτή οδηγεί σε αδιέξοδο και για τις δύο εργασίες. Η κατάσταση αυτή μπορεί να επεκταθεί και σε μεγαλύτερο αριθμό εργασιών και πόρων οδηγώντας σε αλυσιδωτά μπλοκαρίσματα.



Σχήμα 2-6: Παράδειγμα του φαινομένου του αδιεξόδου

Το φαινόμενο του αδιέξοδου είναι ένα πολύ σημαντικό ζήτημα για κρίσιμες εφαρμογές πραγματικού χρόνου, γι' αυτό και πολλές μελέτες έχουν αφιερωθεί στην εξεύρεση λύσης για το πρόβλημα. Μία μέθοδος είναι να διασφαλιστεί μία σειρά για την πρόσβαση στους πόρους (Havender, 1968). Δεν είναι όμως πάντα δυνατό να εφαρμόζεται αυτή η τεχνική διότι είναι απαραίτητο να γνωρίζουμε όλους τους πόρους που θα χρησιμοποιήσει μία εργασία κατά τη διάρκεια της δραστηριότητάς της.

Μία άλλη τεχνική η οποία μπορεί να χρησιμοποιηθεί on line είναι γνωστή ως ο αλγόριθμος του τραπεζίτη (banker's algorithm, Haberman 1969) και απαιτεί κάθε εργασία να δηλώνει εκ των προτέρων τον μέγιστο αριθμό πόρων που ενδέχεται να δεσμεύσει ταυτόχρονα.

Μία άλλη μέθοδος, έχει να κάνει με την χρησιμοποίηση ενός φύλακα χρονομετρητή (watchdog timer). Η χρήση ενός τέτοιου χρονομετρητή επιτρέπει την ανακάλυψη αδρανών εργασιών, καθώς μία αδρανής εργασία είναι πιθανώς μία εργασία σε αδιέξοδο. Έτσι, η λύση είναι να γίνει επανεκκίνηση της εργασίας ή και όλης της ομάδας των εργασιών στην οποία ανακαλύφθηκαν αδρανείς εργασίες οφειλόμενες σε αδιέξοδο. Η τεχνική αυτή χρησιμοποιείται αρκετά συχνά όταν τα αδιέξοδα προκύπτουν σπάνια, ωστόσο δεν είναι αποδεκτή για συστήματα υψηλού βαθμού κρισιμότητας.

2.5 Χρονοπρογραμματιστικές μέθοδοι σε καταστάσεις φόρτου

Σε καταστάσεις φόρτου ενός πραγματικού συστήματος ο χρόνος υπολογισμού μίας ομάδας εργασιών υπερβαίνει τον διαθέσιμο χρόνο κατά τον οποίο είναι διαθέσιμος ο επεξεργαστής, με συνέπεια την μη ικανοποίηση χρονικών προθεσμιών. Ακόμη και σε περιπτώσεις που τα συστήματα πραγματικού χρόνου και οι εφαρμογές τους έχουν σχεδιαστεί

κατάλληλα, ενδέχεται να προκύψουν καθυστερήσεις για διάφορους λόγους, όπως είναι η απώλεια ενός σήματος ενεργοποίησης μίας εργασίας που οφείλεται σε σφάλμα κάποιας συσκευής, ή η επέκταση του χρόνου υπολογισμού κάποιων εργασιών που οφείλεται στον παράλληλο διαμοιρασμό κρίσιμων πόρων. Ακόμη, οι ταυτόχρονες αφίξεις πολλών μη περιοδικών εργασιών μπορούν να οδηγήσουν το σύστημα σε κατάσταση φόρτου.

Στην περίπτωση που το σύστημα δεν είναι σχεδιασμένο να αντιμετωπίζει τέτοιες καταστάσεις, τα αποτελέσματα μπορεί να είναι καταστροφικά. Οι βασικοί αλγόριθμοι (EDF & RM), όπως αυτοί αναλύθηκαν παραπάνω, παρουσιάζουν φτωχές αποδόσεις σε καταστάσεις φόρτου και δεν είναι σε θέση να διαχειριστούν ομάδες καθυστερημένων εργασιών. Ακόμη περισσότερο, με αυτούς τους αλγόριθμους αν μία εργασία χάσει την χρονική της προθεσμία προκαλεί και τις υπόλοιπες στην ίδια κατεύθυνση (domino effect).

2.5.1 Διαχείριση εργασιών πραγματικού χρόνου με μεταβλητές χρονικές παραμέτρους

Ένα σύστημα πραγματικού χρόνου διαχειρίζεται πολλές εργασίες και η απόφαση της εκτέλεσής τους βασίζεται στον χρονοπρογραμματιστή. Ο χρονοπρογραμματιστής, στη συνέχεια, βασίζεται στην γνώση για τον χρόνο υπολογισμού της κάθε εργασίας, στις σχέσεις εξάρτησης που ενδεχομένως υπάρχουν μεταξύ τους και στις προθεσμίες υλοποίησης, ώστε να πάρει τις σωστές χρονοπρογραμματιστικές αποφάσεις. Κάτι τέτοιο λειτουργεί ορθώς εφόσον ο χρόνος εκτέλεσης της κάθε εργασίας είναι προκαθορισμένος. Όμως, ένας χρονοπρογραμματισμός βασιζόμενος σε προκαθορισμένες παραμέτρους μάλλον δεν θα είναι αποτελεσματικός για ένα δυναμικό περιβάλλον. Για την αντιμετώπιση ενός τέτοιου περιβάλλοντος, λοιπόν, ο χρονοπρογραμματισμός εκτέλεσης ενός συστήματος πραγματικού χρόνου πρέπει να είναι αρκετά ευλύγιστος.

Για παράδειγμα, σε συστήματα πολυμέσων, οι χρονικοί περιορισμοί μπορούν να είναι πολύ πιο ευλύγιστοι και δυναμικοί απ' όσο επιτρέπεται. Δραστηριότητες όπως η διαχείριση εικόνας και ήχου εκτελούνται περιοδικά, αλλά οι ρυθμοί και οι χρόνοι εκτέλεσής τους μπορούν να μην είναι και τόσο αυστηροί όσο σε άλλες πιο αυστηρές εφαρμογές. Αν μία εργασία διαχειρίζεται συμπιεσμένες δομές, ο χρόνος συμπίεσης και αποσυμπίεσης της κάθε δομής μεταβάλλεται ανάλογα με την πολυπλοκότητα της εικόνας. Έτσι, μπορεί να είναι μεγαλύτερος από τον μέσο χρόνο εκτέλεσης. Συνεπώς, ο χρόνος εκτέλεσης της χειρότερης περίπτωσης μία εργασίας μπορεί να είναι αρκετά μεγαλύτερος από τον μέσο χρόνο εκτέλεσης. Εφόσον οι αυστηρές εργασίες πραγματικού χρόνου βασίζονται στον χρόνο εκτέλεσης της χειρότερης περίπτωσης, οι δραστηριότητες που αφορούν τα πολυμέσα μπορούν να οδηγήσουν σε σπατάλη των επεξεργαστικών πόρων, αν αντιμετωπιστούν σαν αυστηρές εργασίες πραγματικού χρόνου.

Ένα άλλο παράδειγμα έχει να κάνει με τα συστήματα ραντάρ, όπου ο αριθμός των αντικειμένων που οπτικοποιούνται μπορεί να μεταβάλλεται με τον χρόνο. Έτσι, ο φόρτος του επεξεργαστή μπορεί να αλλάζει λόγω της αύξησης της διάρκειας εκτέλεσης μίας εργασίας η οποία συνδέεται με τον αριθμό των αντικειμένων. Κάποιες φορές, για τους υπολογισμούς πραγματικού χρόνου, είναι καλό να μην επιδιώκεται η μέγιστη δυνατή ακρίβεια, ούτως ώστε ο χρόνος και οι πόροι να διαφυλάσσονται προκειμένου να χρησιμοποιηθούν από άλλες εργασίες.

Αρκετές έρευνες έχουν επικεντρωθεί στη διαχείριση των εργασιών με μεταβλητούς χρόνους εκτέλεσης. Τρείς κύριοι τρόποι για να αντιμετωπιστεί το πρόβλημα αυτό είναι οι εξής:

- ειδικά μοντέλα εργασιών ικανά να διαχειριστούν μεταβλητές παραμέτρους εργασιών, όπως προθεσμίες υλοποίησης, περιόδους ή χρόνους εκτέλεσης και υπολογισμού
- on-line προσαρμόσιμο μοντέλο, που υπολογίζει τις μέγιστες πιθανές παραμέτρους μίας εργασίας σε οποιονδήποτε χρόνο.
- μηχανισμός ανεκτικότητας σε σφάλματα που εξασφαλίζει συμμόρφωση με προκαθορισμένες χρονικές απαιτήσεις κάτω από οποιαδήποτε συνθήκη.

2.5.1.1 Ειδικά μοντέλα για μεταβλητή εκτέλεση

Στο πλαίσιο των ειδικών μοντέλων για εργασίες με μεταβλητούς χρόνους εκτέλεσης, έχουν προταθεί δύο προσεγγίσεις:

- στατικός χρονοπρογραμματισμός μονότονου ρυθμού (statical rate monotonic algorithm, Atlas & Bestavros 1998) και
- πολυδομικό μοντέλο για εργασίες πραγματικού χρόνου (multiframe model for real-time tasks, (Mok & Chen 1997).

Το πρώτο μοντέλο είναι μία γενίκευση του κλασικού αλγόριθμου μονότονου ρυθμού. Για κάθε εργασία ορίζεται μία ποιότητα υπηρεσίας (Quality of Service) σαν η πιθανότητα ότι σε μία αυθαίρετη μακρά περίοδο εκτέλεσης, ένα τυχαίο στιγμιότυπο της εργασίας θα ικανοποιήσει την χρονική του προθεσμία. Ο στατικός χρονοπρογραμματισμός μονότονου ρυθμού αποτελείται από δυο μέρη: την είσοδο της εργασίας και έναν χρονοπρογραμματιστή. Ο ελεγκτής εισόδου της εργασίας διαχειρίζεται την ποιότητα υπηρεσίας που παραδίδεται στις μεταβλητές εργασίες, μέσω μίας επιλογής admit/reject και μέσω αποφάσεων που λαμβάνει που έχουν να κάνουν με το θέμα των προτεραιοτήτων. Πιο συγκεκριμένα δεν σπαταλάει

καθόλου πόρους σε εργασίες οι οποίες προφανώς θα χάσουν τις χρονικές τους προθεσμίες, μειώνοντας τον φόρτο. Ο χρονοπρογραμματιστής είναι ένας απλός, προεκχωρητικός χρονοπρογραμματιστής καθορισμένων προτεραιοτήτων. Ο στατικός χρονοπρογραμματισμός μονότονου ρυθμού είναι απόλυτα κατάλληλος για εφαρμογές πολυμέσων.

Το πολυδομικό μοντέλο επιτρέπει στον χρόνο εκτέλεσης μίας εργασίας να μεταβάλλεται από το ένα στιγμιότυπο στο άλλο. Στο μοντέλο αυτό οι χρόνοι εκτέλεσης των διαδοχικών στιγμιότυπων μιας εργασίας καθορίζονται από έναν πεπερασμένο πίνακα ακεραίων αριθμών και όχι από έναν συγκεκριμένο αριθμό ο οποίος αντιπροσωπεύει τον χρόνο εκτέλεσης της χειρότερης περίπτωσης, όπως συμβαίνει στο κλασικό μοντέλο. Βήμα προς βήμα, το άνω όριο της χρησιμοποίησης παράγεται μέσα σε μία στρατηγική χρονοπρογραμματισμού προτεραιοτήτων υπό τον διαθέσιμο πεπερασμένο πίνακα. Έτσι, το μοντέλο αυτό βελτιώνει την φόρτο της χρησιμοποίησης του επεξεργαστή.

2.5.1.2 On-line προσαρμόσιμο μοντέλο

Στο πλαίσιο του on-line προσαρμόσιμου μοντέλου προτάθηκαν δύο προσεγγίσεις: το ελαστικό μοντέλο εργασιών (Elastic task model, Buttazzo et.al.,1998) και το προγραμματιστικό προσαρμόσιμο μοντέλο εργασιών (scheduling adaptive task model, Wang & Lin, 1994).

Στο ελαστικό μοντέλο εργασιών, οι περίοδοι των εργασιών αντιμετωπίζονται ως ελαστικές με δεδομένες ελαστικές παραμέτρους: μέγιστο και ελάχιστο μήκος χρόνου και έναν αυστηρό συντελεστή. Οι περιοδικές εργασίες μπορούν να αλλάζουν σκοπίμως τις περιόδους και τον ρυθμό εκτέλεσής τους για να παρέχουν διαφορετική ποιότητα υπηρεσίας, ώστε οι άλλες εργασίες να μπορούν αυτόματα να προσαρμόσουν τις περιόδους τους κρατώντας το

σύστημα σε συνθήκες χαμηλού φόρτου. Το μοντέλο αυτό είναι πολύ χρήσιμο για την διαχείριση εφαρμογών και πολυμέσων, στα οποία οι ρυθμοί εκτέλεσης κάποιων υπολογιστικών δραστηριοτήτων θα πρέπει να είναι δυναμικά προσαρμοσμένοι με την εκάστοτε κατάσταση του συστήματος.

Για παράδειγμα, ας σκεφτούμε μία ομάδα τριών εργασιών $\tau_i(r_i, C_i, D_i, T_i)$ με τα παρακάτω χαρακτηριστικά $\tau_1(0,10,20,20)$, $\tau_2(0,10,40,40)$ και $\tau_3(0,15,70,70)$. Με αυτές τις περιόδους η συγκεκριμένη ομάδα εργασιών είναι χρονοπρογραμματίσιμη από τον αλγόριθμο EDF εφόσον

$$U = \frac{10}{20} + \frac{10}{40} + \frac{15}{70} = 0{,}964 < 1$$

Αν η εργασία τ_3 ελαττώνει τον ρυθμό εκτέλεσής της σε 50 δεν υφίσταται εφικτός αλγόριθμος αφού ο φόρτος του επεξεργαστή θα πρέπει να είναι μικρότερος της μονάδας:

$$U = \frac{10}{20} + \frac{10}{40} + \frac{15}{50} = 1{,}05 > 1$$

Ωστόσο το σύστημα μπορεί να αποδεχτεί τον υψηλότερο ρυθμό της εργασίας τ_3 μειώνοντας την εκτέλεση των δύο άλλων εργασιών. Για μία περίοδο της τάξεως του 22 για την εργασία τ_1 και 45 για την εργασία τ_2 παίρνουμε φόρτο του επεξεργαστή μικρότερο της μονάδας:

$$U = \frac{10}{22} + \frac{10}{45} + \frac{15}{50} = 0{,}977 < 1$$

2.5.1.3 Μηχανισμός ανεκτικότητας σε σφάλματα (Fault tolerant mechanism)

Η βασική ιδέα του μηχανισμού ανεκτικότητας σε σφάλματα, βασίζεται στο μοντέλο των ανακριβών υπολογισμών. Δηλαδή στην εξαγωγή αποτελεσμάτων φτωχής αλλά αποδεκτής ποιότητας στον επιθυμητό χρόνο, όταν τα ακριβή αποτελέσματα δεν μπορούν να παραχθούν εντός του καθορισμένου χρόνου. Στο πλαίσιο αυτό, έχουν προταθεί δύο μηχανισμοί: το μοντέλο μηχανισμού προθεσμίας υλοποίησης (deadline mechanism model, Campel et al., 1979, Chetto & Chetto, 1991) και το μοντέλο των ανακριβών υπολογισμών (imprecise computation model, Chung et al., 1990).

Α) Μοντέλο μηχανισμού προθεσμίας υλοποίησης (deadline mechanism model)

Το μοντέλο μηχανισμού προθεσμίας υλοποίησης απαιτεί κάθε εργασία τi να έχει ένα βασικό (primary) τ_i^p και ένα εναλλακτικό (alternative) πρόγραμμα τ_i^a. Ο βασικός αλγόριθμος παρέχει καλής ποιότητας υπηρεσία, η οποία είναι φυσικά και περισσότερο επιθυμητή, αλλά σε άγνωστο μήκος χρόνου. Το εναλλακτικό πρόγραμμα παράγει ένα αποδεκτό αποτέλεσμα, αλλά σαφώς λιγότερο επιθυμητό, σε ένα ντετερμινιστικό και γνωστό μήκος χρόνου. Σε ένα σύστημα ελέγχου το οποίο χρησιμοποιεί τον μηχανισμό προθεσμίας υλοποίησης, ο χρονοπρογραμματιστικός αλγόριθμος διασφαλίζει ότι όλες οι προθεσμίες υλοποίησης ικανοποιούνται είτε από το βασικό πρόγραμμα, είτε από το εναλλακτικό πρόγραμμα.

Το μοντέλο βρίσκει εφαρμογή στα συστήματα αεροπορικών σκαφών. Στα συστήματα αυτά, ένα από τα θέματα που μας ενδιαφέρει είναι ο καθορισμός της θέσης του αεροπλάνου κατά τη διάρκεια της πτήσης. Η πιο ακριβής μέθοδος είναι να χρησιμοποιηθεί μία δορυφορική επικοινωνία μέσω της τεχνικής GPS (Global Position System). Αλλά το πρόγραμμα το οποίο ανταποκρίνεται σ' αυτή την λειτουργία, έχει μία άγνωστη διάρκεια

εκτέλεσης λόγω πολλαπλής χρήσης του δορυφόρου από διάφορους χρήστες. Από την άλλη, υπάρχει η πιθανότητα το αεροπλάνο να βρίσκεται στην σωστή θέση χρησιμοποιώντας την προηγούμενη καθορισμένη θέση του, δεδομένης της ταχύτητας και της κατεύθυνσης του, μέσα σε ένα καθορισμένο χρονικό βήμα. Η πρώτη τεχνική αντιστοιχεί στο βασικό πρόγραμμα ενώ η δεύτερη, αλλά και πιο ανακριβής, στο εναλλακτικό πρόγραμμα.

Β) Μοντέλο ανακριβών υπολογισμών (imprecise computation model)

Ο χρονοπρογραμματισμός περιοδικών εργασιών με ανακριβή αποτελέσματα παρουσιάστηκε επίσημα για πρώτη φορά από τον Kwei-Jay Lin και τους συνεργάτες του το 1987. Στα συστήματα πραγματικού χρόνου, εστιάζουμε στο γεγονός ότι κάθε εργασία ολοκληρώνει την εκτέλεση της πριν την λήξη της χρονικής της προθεσμίας. Όμως, διάφορα σφάλματα και λάθη μπορούν να προκαλέσουν μη ικανοποίηση των χρονικών προθεσμιών. Έτσι, εισήχθη η ιδέα της εκμετάλλευσης μερικών-ανακριβών αποτελεσμάτων, όταν τα ακριβή δεν μπορούν να παραχθούν στο επιτρεπόμενο χρονικό περιθώριο. Αναπτύχθηκαν, λοιπόν, ειδικές τεχνικές, οι οποίες μπορούν να παράγουν και να χρησιμοποιούν μερικά αποτελέσματα για συστήματα τα οποία μπορούν τα υποστηρίξουν τεχνικές ανακριβών υπολογισμών.

Η έννοια των ανακριβών υπολογισμών εισάγει μία τεχνική, στην οποία εξασφαλίζεται ότι ένα μερικό αποτέλεσμα αποδεκτής ποιότητας είναι διαθέσιμο στον χρήστη, όταν τα ακριβή αποτελέσματα δεν μπορούν να ληφθούν εγκαίρως. Για να υποστηρίξει ανακριβείς υπολογισμούς ένα σύστημα πραγματικού χρόνου, κάθε εργασία στο σύστημα πρέπει να δομηθεί με τέτοιο τρόπο ώστε να μπορεί να αποσυντεθεί σε δύο λογικά μέρη: το υποχρεωτικό (mandatory) και το προαιρετικό (optional) μέρος. Το υποχρεωτικό μέρος για μία εργασία θα

πρέπει να εκτελείται πριν την λήξη της προθεσμίας υλοποίησης, προκειμένου να έχουμε ένα αρκετά αξιόλογο αποτέλεσμα. Το προαιρετικό μέρος της εργασίας απλώς βελτιώνει την ποιότητα του τελικώς παραγόμενου αποτελέσματος.

Μία εργασία λέγεται ότι παράγει ένα ακριβές αποτέλεσμα (precise result), όταν εκτελεστούν το υποχρεωτικό και προαιρετικό μέρος πριν από τη προθεσμία υλοποίησης. Αντίθετα, λέγεται ότι παράγει ανακριβές/προσεγγιστικό (imprecise) αποτέλεσμα όταν εκτελείται μόνο το υποχρεωτικό μέρος.

Σχήμα 2-7: Εργασία στο συμβατικό μοντέλο και στο μοντέλο ανακριβών υπολογισμών

Η κεντρική ιδέα των ανακριβών υπολογισμών, είναι η διαπραγμάτευση της ποιότητας για χάρη του χρόνου, με σκοπό την αποφυγή των χαμένων χρονικών προθεσμιών, πάντα με τη διασφάλιση ότι ένα προσεγγιστικό αποτέλεσμα μιας αποδεκτής ποιότητας είναι πάντοτε διαθέσιμο, όταν το ακριβές αποτέλεσμα δεν μπορεί να εξαχθεί στα πλαίσια της προθεσμίας υλοποίησης. Για να διασφαλιστεί κάτι τέτοιο, υπάρχει μία σαφής προτεραιότητα του υποχρεωτικού μέρους, το οποίο πρέπει πάντα να ολοκληρώνεται, πριν αρχίσει η εκτέλεση του αντίστοιχου προαιρετικού μέρους. Η χρήση της τεχνικής των ανακριβών υπολογισμών είναι ένας τρόπος να αποφεύγονται τα χρονικά λάθη κατά τη διάρκεια παροδικών υπερφορτώσεων του συστήματος. Αυτό το πετυχαίνει με τον πρόωρο τερματισμό ή την ακύρωση εκτέλεσης ολόκληρου του προαιρετικού μέρους, όταν δεν υπάρχουν αρκετοί διαθέσιμοι πόροι στο σύστημα ή όταν δεν υπάρχουν χρονικά περιθώρια. Αυτό φυσικά έχει ως αποτέλεσμα

χαμηλότερη ποιότητα, αλλά αυτό μπορεί να γίνει αποδεκτό, αν ολοκληρωθεί το υποχρεωτικό μέρος. Έτσι, λοιπόν, κερδίζεται πολύτιμος χρόνος με το ανακριβές αλλά χρήσιμο αποτέλεσμα.

Στο μοντέλο των ανακριβών υπολογισμών υπάρχουν τα παρακάτω δύο είδη εργασιών:

- μονότονες εργασίες (monotone)
- εργασίες με περιορισμό 0/1 (0/1 constraint).

Ο διαχωρισμός μεταξύ των δύο παραπάνω έγκειται στην ποιότητα του αποτελέσματος που επιφέρει η εκτέλεση του προαιρετικού μέρους. Μονότονη είναι μία εργασία στην οποία όσο περισσότερο εκτελείται το προαιρετικό μέρος τόσο περισσότερο βελτιώνεται η ποιότητα του αποτελέσματος. Σαφώς, το ακριβές αποτέλεσμα επιτυγχάνεται όταν ολοκληρωθούν και τα δύο λογικά μέρη μιας εργασίας. Όμως, ακόμα και αν εκτελεστεί ένα μικρό μέρος του προαιρετικού μέρους, στις μονότονες εργασίες, η ποιότητα του τελικού αποτελέσματος βελτιώνεται και συνεχίζει να βελτιώνεται όσο συνεχίζεται η εκτέλεση.

Αντίθετα στις εργασίες με περιορισμό 0/1 για να έχουμε βελτίωση της ποιότητας του αποτελέσματος από την εκτέλεσή τους, θα πρέπει το προαιρετικό μέρος να ολοκληρωθεί. Μία μερική εκτέλεση του προαιρετικού μέρους δεν προσφέρει τίποτα στην ποιότητα του τελικά παραγόμενου αποτελέσματος. Ο χρονοπρογραμματισμός αυτών των εργασιών γίνεται με διαφορετικό τρόπο σε σχέση με τις μονότονες εργασίες και έχουν δημοσιευτεί ειδικοί αλγόριθμοι χρονοπρογραμματισμού. Στην περίπτωση αυτών των εργασιών γίνεται κατανοητό ότι πιθανή διακοπή στην εκτέλεση του προαιρετικού μέρους συνιστά χαμένο χρόνο εκτέλεσης.

Κεφάλαιο 3 Το Περιοδικό Μοντέλο

3.1 Εισαγωγή

Η ενδεχόμενη χρήση του ασύρματου μέσου για τον έλεγχο δικτύωσης (networked control) έχει φέρει στο προσκήνιο την ανάγκη για έρευνα στον χρονοπρογραμματισμό των ασύρματων συστημάτων. Οι εφαρμογές πραγματικού χρόνου δεν απαιτούν μόνο ικανή και αξιόπιστη παραλαβή δεδομένων, αλλά επίσης η εγκαιρότητα (timeliness) είναι υπέρτατης σημαντικότητας, καθώς η επικοινωνία των δεδομένων είναι συχνά μέρος μιας σύνθετης διαδοχής κρισίμων χρονικά πράξεων που εκτελούνται από το σύστημα ελέγχου δικτύωσης σε απάντηση σε εξωτερικά γεγονότα. Τέτοιου είδους εγγυήσεις πραγματικού χρόνου απαιτούνται επίσης για εφαρμογές πολυμέσων σε ασύρματα δίκτυα βρόχου (wireless mesh networks). Στον παραδοσιακό χρονοπρογραμματισμό πραγματικού χρόνου, πάνω στον οποίο βασίστηκε το πεδίο του κλασσικού ψηφιακού ελέγχου, προέκυψε το πρόβλημα του χρονοπρογραμματισμού των εργασιών που απαιτούσε συγκεκριμένους (ντετερμινιστικούς) χρόνους εκτέλεσης ώστε να τηρήσουν τις αυστηρές προθεσμίες ολοκλήρωσης. Ωστόσο, εξαιτίας της αναξιοπιστίας του ασύρματου μέσου που οφείλεται στη διάλειψη (fading). Είναι απαραίτητο να επιληφθούμε του προβλήματος του χρονικού χρονοπρογραμματισμού για τις εργασίες με τους αβέβαιους χρόνους εκτέλεσης.

Παρακινούμενοι από αυτό και χρησιμοποιώντας το πλαίσιο του στοχαστικού ελέγχου, μελετήσαμε το πρόβλημα της ελαχιστοποίησης του αναμενόμενου λόγου αστοχίας της

προθεσμίας υλοποίησης (deadline miss ratio) για τυχαίους χρόνους εκτέλεσης. Συγκεκριμένα, μελετήσαμε ένα κανονικό πρόγραμμα χρονοπρογραμματισμού για ασύρματα συστήματα με απαιτήσεις εφαρμογών πραγματικού χρόνου. Παραδοσιακά, τέτοιες λανθάνουσες εγγυήσεις παρέχονται κατά τη χρησιμοποίηση του συντονισμού ανταγωνισμού παραθύρων (contention window modulation) IEEE 802.11 και τη διάταξη πλαισίων (frame spacing). Η επίπτωση των ασύρματων απωλειών και της ποικιλομορφίας της διάλειψης (fading diversity) στον σχεδιασμό του χρονοπρογραμματισμού πραγματικού χρόνου έχει αγνοηθεί και θα προσπαθήσουμε να αναπτύξουμε μια τεκμηριωμένη άποψη για τον ασύρματο χρονοπρογραμματισμό πραγματικού χρόνου.

Υπάρχουν δύο βασικά θέματα σε κάθε σχεδιασμό χρονοπρογραμματισμού:

α) σχεδιασμός των βέλτιστων πολιτικών χρονοπρογραμματισμού και

β) σχεδιασμός των μηχανισμών πρωτοκόλλων διανομής

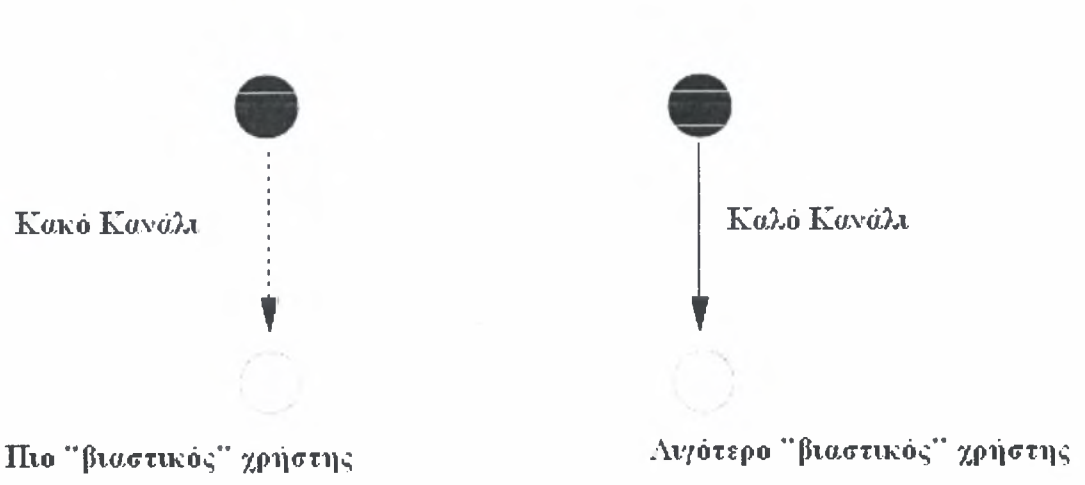
Εμείς εστιάσαμε στο θέμα των πολιτικών, παρατηρώντας ότι η βέλτιστη πολιτική δομή επηρεάζει σημαντικά τον μηχανισμό σχεδιασμού. Ευλόγως, η σημαντική ανακάλυψη στη δουλειά των Liu και Layland πάνω στον αυστηρό χρονοπρογραμματισμό πραγματικού χρόνου, ήταν η απόδειξη της επάρκειας των στατικών προτεραιοτήτων μηχανισμών χρονοπρογραμματισμού CPU, με το επακόλουθο ότι τα συστήματα πραγματικού χρόνου κυρίως χρησιμοποιούν σήμερα χρονοπρογραμματισμό στατικής προτεραιότητας.

Τώρα θα περιγράψουμε πως η ασύρματη εκδοχή αυτού του προβλήματος διαφέρει από το κλασσικό αυστηρό χρονοπρογραμματισμό πραγματικού χρόνου. Η προγραμματιστική ανάλυση για τα αυστηρά συστήματα πραγματικού χρόνου είναι τυπικά βασισμένη στο κλασσικό πλαίσιο των Liu και Layland. Είναι γνωστό ότι ο χρονοπρογραμματισμός μονοτονικού ρυθμού (RM) και η προσέγγιση χρονοδρομολόγησης της ενωρίτερης

προθεσμίας υλοποίησης (earliest-deadline-first (EDF)) είναι στατικές και δυναμικές πολιτικές προτεραιότητας αντίστοιχα, με την έννοια της μεγιστοποίησης της προγραμματιστικής περιοχής για περιοδικές αφίξεις. Το ευρύτερο πλαίσιο μιας τέτοια ανάλυσης είναι ο χρονοπρογραμματισμός CPU και είναι απόλυτα ντετερμινιστικός. Δεν υπάρχει αιτιολόγηση για την πιθανότητα ότι ένας χρήστης να χρονοπρογραμμάτισε σε μια χρονική περίοδο (slot), μπορεί και να μην ολοκληρωθεί σε αυτή τη χρονική περίοδο εξαιτίας, για παράδειγμα, της απώλειας ενός πακέτου. Από την άλλη, τα ασύρματα συστήματα είναι γνωστά για τέτοιες απώλειες που οφείλονται σε μεγάλης κλίμακας εξασθένηση του σήματος (scale signal attenuation), σε μικρής κλίμακας διάλειψη των πολυμονοπατιών (multipath fading) και από φαινόμενα ασύρματων παρεμβολών. Έτσι, είναι κοινά στην πράξη μη αξιόπιστα ενδιάμεσης ποιότητας συνδέσεις, ειδικά με τα δίκτυα IEEE 802.11 και 802.15.4. Μια διαδεδομένη τεχνική για την καταπολέμηση της διακύμανσης της ποιότητας του συνδέσμου είναι ο καιροσκοπικός χρονοπρογραμματισμός (opportunistic scheduling), στον οποίο ένας αποστολέας εκπέμπει επιλεκτικά στους δέκτες με το καλύτερο ποιοτικά κανάλι.

Μελετήσαμε ένα κανονικό πρόβλημα ασύρματου χρονοπρογραμματισμού πραγματικού χρόνου με δύο χρήστες πραγματικού χρόνου που ο καθένας συνδέεται με μια προθεσμία υλοποίησης. Ο ένας χρήστης είναι λιγότερο «βιαστικός» από τον άλλο και έχει μακρύτερη προθεσμία υλοποίησης. Υποθέτοντας ότι ο λιγότερο «βιαστικός» χρήστης έχει ένα «καλό» ασύρματο κανάλι και ο πιο «βιαστικός» έχει ένα «κακό» ασύρματο κανάλι, όπως φαίνεται στο Σχήμα 3-1. Θέτοντας την ερώτηση: Πως θα πρέπει να προγραμματίσουμε αυτούς τους χρήστες ώστε να ελαχιστοποιήσουμε τον αναμενόμενο αριθμό των χαμένων προθεσμιών υλοποίησης; Η θεωρία του πραγματικού χρόνου προτείνει τη χρήση της EDF προσέγγισης. Από την άλλη, η θεωρία των ασύρματων δικτύων προτείνει τη χρήση του καιροσκοπικού χρονοπρογραμματισμού. Το πρόβλημα του ασύρματο

χρονοπρογραμματισμού πραγματικού χρόνου απαιτεί μια ανταλλαγή μεταξύ των δύο συναγωνιζόμενων θεωριών.



Σχήμα 3-1: Δύο εφαρμογές πραγματικού χρόνου σε ένα ασύρματο περιβάλλον με ανομοιότητα καναλιών.

Στην παρούσα εργασία μελετήσαμε το περιοδικό μοντέλο, όπου οι αφίξεις είναι περιοδικές, δηλαδή, οι εργασίες του χρήστη i καταφθάνουν κάθε t_i χρονικές περιόδους με προθεσμία υλοποίησης ίση με την περίοδο, ελπίζοντας να βρούμε την πολιτική που ελαχιστοποιεί τον αναμενόμενο αριθμό των χαμένων προθεσμιών υλοποίησης.

Τα κύρια αποτελέσματα συνοψίζονται ακολούθως:

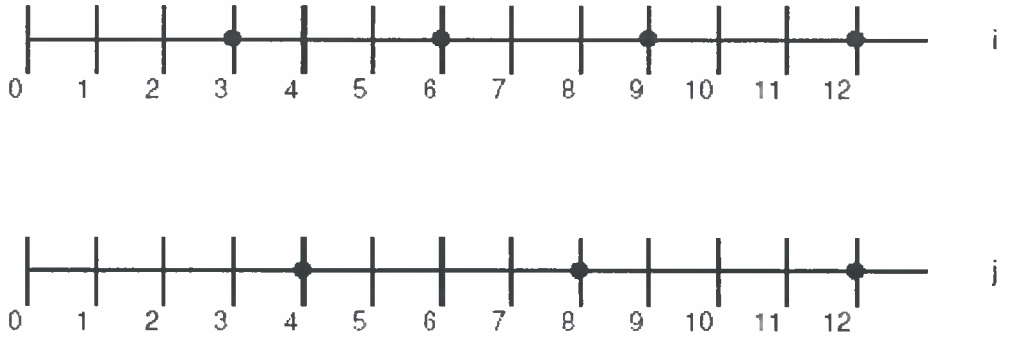
1. Η βέλτιστη πολιτική εξαρτάται από τις πιθανότητες επιτυχίας και από τη διαφορά των δύο προθεσμιών ολοκλήρωσης. Δηλαδή, στο συγκεκριμένο πρόβλημα η βέλτιστη πολιτική μπορεί να αλλάξει μεταξύ των χρηστών μόνο στις χρονικές περιόδους που είναι πολλαπλάσια της μιας ή της άλλης περιόδου.

2. Η πολιτική πάντα επιλέγει το χρήστη που έχει τη μεγαλύτερη πιθανότητα επιτυχίας όταν είναι πιο κοντά στη δική του προθεσμία ολοκλήρωσης σε σχέση με τον άλλο χρήστη.

3.2 Μορφοποίηση του προβλήματος

Μελετήσαμε ένα σύστημα με ίσες χρονικές περιόδους και συνδέσαμε κάθε χρήστη με ένα ζεύγος πηγής-προορισμού (source-destination pair). Δηλαδή, ένα κανονικό σενάριο με δύο χρήστες πραγματικού χρόνου που μοιράζονται ένα ασύρματο κανάλι, όπως φαίνεται στο Σχήμα 3-2. Κάθε εργασία του χρήστη i έχει μια προθεσμία ολοκλήρωσης t_i και μια απαίτηση του χρόνου εξυπηρέτησης ίσο ακριβώς με μια χρονική περίοδο. Μια εργασία αφήνει το σύστημα αν προγραμματιστεί για εκπομπή σε μία χρονική περίοδο και ολοκληρωθεί επιτυχώς αυτή τη χρονική περίοδο ή αν λήξει ο προθεσμία ολοκλήρωσής της.

Στο περιοδικό μοντέλο, κρατάμε το ίχνος των ηλικιών των δύο χρηστών καθώς και των εργασιών τους που βρίσκονται στο σύστημα.



Σχήμα 3-2: Δύο χρήστες πραγματικού χρόνου που μοιράζονται ένα ασύρματο κανάλι

Στην περίπτωσή μας, έχουμε 2 χρήστες (i,j). Ο χρήστης i έχει μια προθεσμία υλοποίησης ti = a χρονικές περιόδους και ο χρήστης j έχει προθεσμία υλοποίησης tj = b χρονικές περιόδους. Ο κάθε χρήστης έχει μια πιθανότητα επιτυχίας εκτέλεσης της εργασίας του (pi για το χρήστη i και pj για το χρήστη j).

3.3 Αλγόριθμος υπολογισμού.

Το συγκεκριμένο πρόβλημα δεν χρειάζεται να εξεταστεί σε άπειρο ορίζοντα καθώς μετά από τον αριθμό των χρονικών περιόδων που ισούται με το Ελάχιστο Κοινό Πολλαπλάσιο των δύο προθεσμιών υλοποίησης επανερχόμαστε στην ακριβώς στην αρχική κατάσταση. Συνεπώς, πρόκειται για μια Μαρκοβιανή Αλυσίδα με 4 δυνατές καταστάσεις.

Αρχικά, αφού δηλώσουμε και αρχικοποιήσουμε τις μεταβλητές που θα χρησιμοποιήσουμε στο πρόγραμμα, το πρόγραμμα μας ζητά να καθορίσουμε τις τιμές των περιόδων – προθεσμιών υλοποίησης των δύο χρηστών (T_1 και T_2) καθώς και τις πιθανότητες επιτυχίας των δύο χρηστών (p_1 και p_2). Επίσης, συμβολίζοντας την κάθε πιθανή κατάσταση με (i, j), ορίζουμε τις τέσσερις πιθανές καταστάσεις ως εξής:

- $(\mathbf{0}, \mathbf{0})$: Όταν ούτε ο $1^{ος}$ αλλά ούτε και ο $2^{ος}$ Χρήστης έχουν εργασία προς επεξεργασία στο σύστημα
- $(\mathbf{1}, \mathbf{0})$: Όταν ο $1^{ος}$ Χρήστης έχει εργασία προς επεξεργασία στο σύστημα, ενώ ο $2^{ος}$ Χρήστης δεν έχει.
- $(\mathbf{0}, \mathbf{1})$: Όταν ο $2^{ος}$ Χρήστης έχει εργασία προς επεξεργασία στο σύστημα, ενώ ο $1^{ος}$ Χρήστης δεν έχει.

- $(1,1)$: Όταν και ο $1^{ος}$ αλλά και ο $2^{ος}$ Χρήστης έχουν εργασία προς επεξεργασία στο σύστημα.

Επιπλέον, αν βρισκόμαστε στη χρονική περίοδο t και οι χρήστες βρίσκονται στην κατάσταση (i,j), συμβολίζουμε τον Αναμενόμενο Αριθμό Επιτυχιών με $V_t(i,j)$.

Το πρώτο βήμα είναι ο υπολογισμός του Ελάχιστου Κοινού Πολλαπλασίου των δύο περιόδων (T_1 και T_2) χρησιμοποιώντας τη συνάρτηση lcm.

Στη συνέχεια , δηλώνουμε και αρχικοποιούμε (μηδενίζουμε όλες τις τιμές) τους παρακάτω πίνακες:

1. Πίνακας V: Ο Πίνακας V έχει διάσταση $lcm \times 4$, όπου το lcm είναι το Ελάχιστο Κοινό Πολλαπλάσιο των περιόδων – προθεσμιών υλοποίησης των δύο χρηστών και το 4 είναι ουσιαστικά οι πιθανές καταστάσεις που μπορεί να βρίσκεται ο κάθε Χρήστης. Για παράδειγμα, αν ο $1^{ος}$ Χρήστης έχει προθεσμία υλοποίησης $T_1 = 3$ και ο $2^{ος}$ Χρήστης έχει προθεσμία υλοποίησης $T_2 = 4$, τότε το Ελάχιστο Κοινό Πολλαπλάσιο των δύο περιόδων είναι 12, άρα ο Πίνακας V θα είναι διάστασης 12×4. Ο πίνακας αυτός θα περιέχει τα αποτελέσματα των υπολογισμών του αναμενόμενου αριθμού επιτυχιών για κάθε κατάσταση σε κάθε χρονική περίοδο, όταν εφαρμόζεται ο βασικός αλγόριθμος Βέλτιστης Πολιτικής.
2. Πίνακας Vk: Ο Πίνακας Vk έχει επίσης διάσταση $lcm \times 4$, όπου το lcm είναι το Ελάχιστο Κοινό Πολλαπλάσιο των περιόδων – προθεσμιών υλοποίησης των δύο χρηστών και το 4 είναι ουσιαστικά οι πιθανές καταστάσεις που μπορεί να βρίσκεται ο κάθε Χρήστης. Ο πίνακας αυτός θα περιέχει τα αποτελέσματα των υπολογισμών του αναμενόμενου αριθμού επιτυχιών για

κάθε κατάσταση σε κάθε χρονική περίοδο, όταν εφαρμόζουμε ως Πολιτική Απόφασης την προτεραιότητα στο Χρήστη 1 εάν έχει μικρότερη προθεσμία υλοποίησης και ισχύει:

$$p_1 > p_2(1-p_2)^k,$$

προτεραιότητα στο Χρήστη 2 εάν έχει μικρότερη προθεσμία υλοποίησης και ισχύει:

$$p_2 > p_1(1-p_1)^k \text{ και}$$

στην περίπτωση όπου και οι δύο Χρήστες έχουν την ίδια προθεσμία υλοποίησης, προτεραιότητα έχει αυτός με την μεγαλύτερη πιθανότητα επιτυχίας.

3. Πίνακας $Apofasi$: Ο Πίνακας $Apofasi$ είναι ένας μονοδιάστατος πίνακας διάστασης $lcm \times 1$, όπου το lcm είναι το Ελάχιστο Κοινό Πολλαπλάσιο των περιόδων – προθεσμιών υλοποίησης των δύο χρηστών και θα περιέχει τις αποφάσεις που θα προκύπτουν σε κάθε χρονική περίοδο σύμφωνα με τους υπολογισμούς του βασικού αλγορίθμου Βέλτιστης Πολιτικής.
4. Πίνακας $Apofasi2$: Ο Πίνακας $Apofasi2$ είναι επίσης ένας μονοδιάστατος πίνακας διάστασης $lcm \times 1$, όπου το lcm είναι το Ελάχιστο Κοινό Πολλαπλάσιο των περιόδων – προθεσμιών υλοποίησης των δύο χρηστών και θα περιέχει τις αποφάσεις που θα προκύπτουν σε κάθε χρονική περίοδο σύμφωνα με τους υπολογισμούς του αλγορίθμου όταν εφαρμόζουμε την Πολιτική Απόφασης σύμφωνα με τον κανόνα:

$$p_1 > p_2(1-p_2)^k,$$

που αναλύσαμε παραπάνω.

5. Πίνακες $Pollap1$ και $Pollap2$: Πρόκειται για δύο μονοδιάστατους πίνακες που θα περιέχουν τις περιόδους που θα είναι πολλαπλάσια των T_1 και T_2. Αυτοί οι πίνακες βοηθούν σε διάφορους υπολογισμούς που θα πραγματοποιηθούν στη συνέχεια.

Το επόμενο βήμα περιλαμβάνει την εκτέλεση των υπολογισμών του αναμενόμενου αριθμού επιτυχιών σε κάθε χρονική περίοδο για κάθε πιθανή κατάσταση. Αυτό πραγματοποιείται ξεκινώντας από την τελευταία χρονική περίοδο (η χρονική περίοδος που ισούται με το Ελάχιστο Κοινό Πολλαπλάσιο των T_1 και T_2) και προχωρώντας αντίστροφα.

Σε αυτό το σημείο να τονίσουμε ότι όταν ο Χρήστης βρίσκεται σε μια χρονική περίοδο που αποτελεί πολλαπλάσιο της περιόδου – προθεσμίας υλοποίησής του, τότε σε αυτή τη χρονική περίοδο αποκλείεται να βρίσκεται στην κατάσταση $(\mathbf{0},\mathbf{0})$ και στην κατάσταση όπου το αντίστοιχο i ή j είναι ίσο με το μηδέν (ανάλογα με τον Χρήστη) . Για παράδειγμα όταν ο Χρήστης 1 έχει περίοδο – προθεσμίας υλοποίησης $T_1 = 3$, τότε τις χρονικές περιόδους 3, 6, 9, 12,… κλπ, δεν μπορεί να βρίσκεται στις καταστάσεις $(\mathbf{0},\mathbf{0})$ και $(\mathbf{0},\mathbf{1})$, οπότε εκείνες τις χρονικές περιόδους τα αντίστοιχα $V_t(i,j)$ θα είναι ίσα με το μηδέν.

Συνεπώς, αν μελετάμε το πρόβλημα για n χρονικές περιόδους, όπου $n = lcm(T_1, T_2)$, τότε για τη χρονική στιγμή n θα έχουμε: $V_n(0,0) = 0, V_n(1,0) = 0, V_n(0,1) = 0$. Επίσης, όπως είναι λογικό και ο αναμενόμενος αριθμός επιτυχιών τη χρονική περίοδο n θα ισούται με το μηδέν, δηλαδή, $V_n(1,1) = 0$.

Έτσι, στο πρόγραμμά μας, εφαρμόζουμε έναν επαναληπτικό βρόχο από τη χρονική στιγμή $n-1$ έως τη χρονική στιγμή 0. Μέσα στο συγκεκριμένο βρόχο και ανάλογα με τη χρονική περίοδο που βρισκόμαστε θα πρέπει να γίνουν διαφορετικοί υπολογισμοί. Συγκεκριμένα, μελετούμε τις παρακάτω περιπτώσεις:

1. Την περίπτωση όπου η χρονική περίοδος t που βρισκόμαστε είναι πολλαπλάσιο και των δύο περιόδων T_1 και T_2.

   Σε αυτήν την περίπτωση είναι πιθανή μόνο η κατάσταση $(\mathbf{1},\mathbf{1})$. Ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,1) = \max\begin{Bmatrix} p_1[1 + V_{t+1}(0,1)] + (1-p_1)[0 + V_{t+1}(1,1)] \\ p_2[1 + V_{t+1}(1,0)] + (1-p_2)[0 + V_{t+1}(1,1)] \end{Bmatrix}$$

   Αν το πρώτο μέγεθος (πρώτη γραμμή) είναι μεγαλύτερο, τότε η απόφαση τη χρονική στιγμή t θα είναι η εξυπηρέτηση του Χρήστη 1. Αν το δεύτερο μέγεθος (δεύτερη γραμμή) είναι μεγαλύτερο τότε η απόφαση τη χρονική στιγμή t θα είναι η εξυπηρέτηση του Χρήστη 2. Όπως είναι λογικό, σε περίπτωση ισότητας των δύο μεγεθών η απόφαση δεν παίζει κανένα ρόλο.

2. Την περίπτωση όπου η χρονική περίοδος t που βρισκόμαστε είναι πολλαπλάσιο μόνο της περιόδου T_1 και όχι της περιόδου T_2. Αυτή την περίπτωση τη μελετούμε ως δύο διαφορετικές υποπεριπτώσεις:

   a. Την περίπτωση όπου η χρονική περίοδος $t+1$, δεν είναι πολλαπλάσιο της περιόδου T_2.

   Σε αυτήν την περίπτωση είναι πιθανές οι καταστάσεις $(\mathbf{1},\mathbf{0})$ και $(\mathbf{1},\mathbf{1})$.

   Για την κατάσταση $(\mathbf{1},\mathbf{0})$, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,0) = p_1[1 + V_{t+1}(0,0)] + (1-p_1)[0 + V_{t+1}(1,0)]$$

Ενώ, για την κατάσταση $(\mathbf{1},\mathbf{1})$, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,1) = \max\begin{Bmatrix} p_1[1 + V_{t+1}(0,1)] + (1 - p_1)[0 + V_{t+1}(1,1)] \\ p_2[1 + V_{t+1}(1,0)] + (1 - p_2)[0 + V_{t+1}(1,1)] \end{Bmatrix}$$

Όπως και στην προηγούμενη περίπτωση, η απόφαση εξαρτάται από το ποιο από τα δύο μεγέθη είναι μεγαλύτερο.

b. Την περίπτωση όπου η χρονική περίοδος $t + 1$, είναι πολλαπλάσιο της περιόδου T_2.

Σε αυτήν την περίπτωση είναι πιθανές οι καταστάσεις $(\mathbf{1},\mathbf{0})$ και $(\mathbf{1},\mathbf{1})$. Για την κατάσταση $(\mathbf{1},\mathbf{0})$, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,0) = p_1[1 + V_{t+1}(0,1)] + (1 - p_1)[0 + V_{t+1}(1,0)]$$

Ενώ, για την κατάσταση $(\mathbf{1},\mathbf{1})$, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,1) = \max\begin{Bmatrix} p_1[1 + V_{t+1}(0,1)] + (1 - p_1)[0 + V_{t+1}(1,1)] \\ p_2[1 + V_{t+1}(1,1)] + (1 - p_2)[0 + V_{t+1}(1,1)] \end{Bmatrix}$$

Όπως και στις προηγούμενες περιπτώσεις, η απόφαση εξαρτάται από το ποιο από τα δύο μεγέθη είναι μεγαλύτερο.

3. Την περίπτωση όπου η χρονική περίοδος t που βρισκόμαστε είναι πολλαπλάσιο μόνο της περιόδου T_2 και όχι της περιόδου T_1. Ομοίως με την προηγούμενη περίπτωση και αυτήν τη μελετούμε ως δύο διαφορετικές υποπεριπτώσεις:

   a. Την περίπτωση όπου η χρονική περίοδος $t+1$, δεν είναι πολλαπλάσιο της περιόδου T_1.

      Σε αυτήν την περίπτωση είναι πιθανές οι καταστάσεις $(\mathbf{0},\mathbf{1})$ και $(\mathbf{1},\mathbf{1})$.

      Για την κατάσταση $(\mathbf{0},\mathbf{1})$, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,1) = p_2[1 + V_{t+1}(0,0)] + (1 - p_2)[0 + V_{t+1}(0,1)]$$

      Ενώ, για την κατάσταση $(\mathbf{1},\mathbf{1})$, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,1) = \max\begin{Bmatrix} p_1[1 + V_{t+1}(0,1)] + (1 - p_1)[0 + V_{t+1}(1,1)] \\ p_2[1 + V_{t+1}(1,0)] + (1 - p_2)[0 + V_{t+1}(1,1)] \end{Bmatrix}$$

      Και σε αυτή την περίπτωση, ομοίως με τις προηγούμενες περιπτώσεις, η απόφαση εξαρτάται από το ποιο από τα δύο μεγέθη είναι μεγαλύτερο.

   b. Την περίπτωση όπου η χρονική περίοδος $t+1$, είναι πολλαπλάσιο της περιόδου T_1.

      Και σε αυτήν την περίπτωση είναι πιθανές οι καταστάσεις $(\mathbf{0},\mathbf{1})$ και $(\mathbf{1},\mathbf{1})$.

Για την κατάσταση $(\mathbf{0},\mathbf{1})$, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,1) = p_2[1 + V_{t+1}(1,0)] + (1 - p_2)[0 + V_{t+1}(1,1)]$$

Ενώ, για την κατάσταση $(\mathbf{1},\mathbf{1})$, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,1) = \max\begin{Bmatrix} p_1[1 + V_{t+1}(1,1)] + (1 - p_1)[0 + V_{t+1}(1,1)] \\ p_2[1 + V_{t+1}(1,0)] + (1 - p_2)[0 + V_{t+1}(1,1)] \end{Bmatrix}$$

Και σε αυτή την περίπτωση, ομοίως με τις προηγούμενες περιπτώσεις, η απόφαση εξαρτάται από το ποιο από τα δύο μεγέθη είναι μεγαλύτερο.

4. Την περίπτωση όπου η χρονική περίοδος t που βρισκόμαστε δεν είναι πολλαπλάσιο ούτε της περιόδου T_1 αλλά ούτε και της περιόδου T_2. Αυτή την περίπτωση τη μελετούμε ως δεκαέξι διαφορετικές υποπεριπτώσεις (τέσσερις περιπτώσεις για κάθε κατάσταση:

   a. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{0},\mathbf{0})$ και η επόμενη χρονική περίοδος $t + 1$ δεν είναι πολλαπλάσια ούτε της περιόδου T_1 αλλά ούτε και της περιόδου T_2.

      Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,0) = V_{t+1}(0,0) + 0$$

b. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(0,0)$ και η επόμενη χρονική περίοδος $t+1$ είναι πολλαπλάσιο της περιόδου T_1 αλλά όχι της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,0) = V_{t+1}(1,0) + 0$$

c. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{0},\mathbf{0})$ και η επόμενη χρονική περίοδος $t+1$ δεν είναι πολλαπλάσιο της περιόδου T_1 αλλά είναι πολλαπλάσιο της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,0) = V_{t+1}(0,1) + 0$$

d. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{0},\mathbf{0})$ και η επόμενη χρονική περίοδος $t+1$ είναι πολλαπλάσιο και της περιόδου T_1 αλλά και της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,0) = V_{t+1}(1,1) + 0$$

e. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{1},\mathbf{0})$ και η επόμενη χρονική περίοδος $t+1$ δεν είναι πολλαπλάσια ούτε της περιόδου T_1 αλλά ούτε και της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,0) = p_1[1 + V_{t+1}(0,0)] + (1 - p_1)[0 + V_{t+1}(1,0)]$$

f. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{1},\mathbf{0})$ και η επόμενη χρονική περίοδος $t+1$ είναι πολλαπλάσιο της περιόδου T_1 αλλά όχι της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,0) = p_1[1 + V_{t+1}(1,0)] + (1 - p_1)[0 + V_{t+1}(1,0)]$$

g. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{1},\mathbf{0})$ και η επόμενη χρονική περίοδος $t+1$ δεν είναι πολλαπλάσιο της περιόδου T_1 αλλά είναι πολλαπλάσιο της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,0) = p_1[1 + V_{t+1}(0,1)] + (1 - p_1)[0 + V_{t+1}(1,1)]$$

h. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{1},\mathbf{0})$ και η επόμενη χρονική περίοδος $t+1$ είναι πολλαπλάσιο και της περιόδου T_1 αλλά και της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,0) = p_1[1 + V_{t+1}(1,1)] + (1 - p_1)[0 + V_{t+1}(1,1)]$$

i. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{0},\mathbf{1})$ και η επόμενη χρονική περίοδος $t+1$ δεν είναι πολλαπλάσια ούτε της περιόδου T_1 αλλά ούτε και της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,1) = p_2[1 + V_{t+1}(0,0)] + (1 - p_2)[0 + V_{t+1}(0,1)]$$

j. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{0},\mathbf{1})$ και η επόμενη χρονική περίοδος $t+1$ είναι πολλαπλάσιο της περιόδου T_1 αλλά όχι της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,1) = p_2[1 + V_{t+1}(1,0)] + (1 - p_2)[0 + V_{t+1}(1,1)]$$

k. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{0},\mathbf{1})$ και η επόμενη χρονική περίοδος $t+1$ δεν είναι πολλαπλάσιο της περιόδου T_1 αλλά είναι πολλαπλάσιο της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,1) = p_2[1 + V_{t+1}(0,1)] + (1 - p_2)[0 + V_{t+1}(0,1)]$$

l. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{0},\mathbf{1})$ και η επόμενη χρονική περίοδος $t+1$ είναι πολλαπλάσιο και της περιόδου T_1 αλλά και της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(0,1) = p_2[1 + V_{t+1}(11)] + (1 - p_2)[0 + V_{t+1}(1,1)]$$

m. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{1},\mathbf{1})$ και η επόμενη χρονική περίοδος $t+1$ δεν είναι πολλαπλάσια ούτε της περιόδου T_1 αλλά ούτε και της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,1) = \max\begin{Bmatrix} p_1[1 + V_{t+1}(0,1)] + (1 - p_1)[0 + V_{t+1}(1,1)] \\ p_2[1 + V_{t+1}(1,0)] + (1 - p_2)[0 + V_{t+1}(1,1)] \end{Bmatrix}$$

Και σε αυτή την περίπτωση, ομοίως με τις αντίστοιχες προηγούμενες περιπτώσεις, η απόφαση εξαρτάται από το ποιο από τα δύο μεγέθη είναι μεγαλύτερο.

n. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{1},\mathbf{1})$ και η επόμενη χρονική περίοδος $t+1$ είναι πολλαπλάσιο της περιόδου T_1 αλλά όχι της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,1) = \max\begin{Bmatrix} p_1[1 + V_{t+1}(1,1)] + (1-p_1)[0 + V_{t+1}(1,1)] \\ p_2[1 + V_{t+1}(1,0)] + (1-p_2)[0 + V_{t+1}(1,1)] \end{Bmatrix}$$

Και σε αυτή την περίπτωση, ομοίως με τις αντίστοιχες προηγούμενες περιπτώσεις, η απόφαση εξαρτάται από το ποιο από τα δύο μεγέθη είναι μεγαλύτερο.

ο. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{1},\mathbf{1})$ και η επόμενη χρονική περίοδος $t+1$ δεν είναι πολλαπλάσιο της περιόδου T_1 αλλά είναι πολλαπλάσιο της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,1) = \max\begin{Bmatrix} p_1[1 + V_{t+1}(0,1)] + (1-p_1)[0 + V_{t+1}(1,1)] \\ p_2[1 + V_{t+1}(1,1)] + (1-p_2)[0 + V_{t+1}(1,1)] \end{Bmatrix}$$

Και σε αυτή την περίπτωση, ομοίως με τις αντίστοιχες προηγούμενες περιπτώσεις, η απόφαση εξαρτάται από το ποιο από τα δύο μεγέθη είναι μεγαλύτερο.

p. Την περίπτωση όπου βρισκόμαστε στην κατάσταση $(\mathbf{1},\mathbf{1})$ και η επόμενη χρονική περίοδος $t+1$ είναι πολλαπλάσιο και της περιόδου T_1 αλλά και της περιόδου T_2.

Σε αυτή την περίπτωση, ο αναμενόμενος αριθμός επιτυχιών στη χρονική περίοδο t θα είναι:

$$V_t(1,1) = \max\begin{Bmatrix} p_1[1+V_{t+1}(1,1)] + (1-p_1)[0+V_{t+1}(1,1)] \\ p_2[1+V_{t+1}(1,1)] + (1-p_2)[0+V_{t+1}(1,1)] \end{Bmatrix}$$

Και σε αυτή την περίπτωση, ομοίως με τις αντίστοιχες προηγούμενες περιπτώσεις, η απόφαση εξαρτάται από το ποιο από τα δύο μεγέθη είναι μεγαλύτερο.

Στον ίδιο επαναληπτικό βρόχο πραγματοποιείται και ο υπολογισμός του αναμενόμενου αριθμού επιτυχιών για κάθε κατάσταση σε κάθε χρονική περίοδο, όταν εφαρμόζουμε ως Πολιτική Απόφασης την προτεραιότητα στο Χρήστη 1 εάν έχει μικρότερη προθεσμία υλοποίησης και ισχύει:

$p_1 > p_2(1-p_2)^k$,

προτεραιότητα στο Χρήστη 2 εάν έχει μικρότερη προθεσμία υλοποίησης και ισχύει:

$p_2 > p_1(1-p_1)^k$ και

στην περίπτωση όπου και οι δύο Χρήστες έχουν την ίδια προθεσμία υλοποίησης, προτεραιότητα έχει αυτός με την μεγαλύτερη πιθανότητα επιτυχίας. Σε αυτή την περίπτωση, εξετάζονται ξανά όλες οι παραπάνω περιπτώσεις και χρησιμοποιούνται όλοι οι παραπάνω τύποι με τη διαφορά όμως ότι η Πολιτική Απόφασης εξαρτάται καθαρά από τον κανόνα $p_1 > p_2(1-p_2)^k$ και όχι από το ποιο από τα δύο μεγέθη είναι μεγαλύτερο. Συγκεκριμένα, για κάθε περίπτωση και για κάθε χρονική περίοδο, υπολογίζουμε, για κάθε Χρήστη, τον αριθμό των χρονικών περιόδων μέχρι την επόμενη προθεσμία υλοποίησης του. Αν ο πρώτος Χρήστης έχει μικρότερο αριθμό χρονικών περιόδων μέχρι την επόμενη προθεσμία υλοποίησης του και ικανοποιείται η συνθήκη $p_1 > p_2(1-p_2)^k$, τότε ο Χρήστης 1 έχει προτεραιότητα. Στην περίπτωση που ο δεύτερος Χρήστης έχει μικρότερο αριθμό χρονικών περιόδων μέχρι την επόμενη προθεσμία υλοποίησης του και ικανοποιείται η συνθήκη $p_2 > p_1(1-p_1)^k$, τότε ο Χρήστης 2 έχει προτεραιότητα και τέλος στην περίπτωση όπου και οι δύο Χρήστες έχουν τον ίδιο αριθμό περιόδων μέχρι την επόμενη προθεσμία υλοποίησης, προτεραιότητα έχει αυτός με την μεγαλύτερη πιθανότητα επιτυχίας. Αυτός ο έλεγχος πραγματοποιείται με την χρήση του k που ουσιαστικά είναι η διαφορά των δύο προθεσμιών υλοποίησης των δύο χρηστών και του οποίου η απόλυτη τιμή χρησιμοποιείται στον υπολογισμό των συνθηκών που παρουσιάστηκαν παραπάνω.

Μετά την ολοκλήρωση του υπολογιστικού κομματιού του αλγορίθμου και αφού εκτυπωθούν τα αποτελέσματα, ακολουθούν οι έλεγχοι της ορθότητας των αποτελεσμάτων. Συγκεκριμένα, αρχικά ακολουθεί ο έλεγχος εάν αλλάζει η Βέλτιστη Πολιτική απόφασης στις χρονικές περιόδους που δεν είναι πολλαπλάσια των δύο περιόδων – προθεσμιών ολοκλήρωσης (T_1 και T_2). Ο δεύτερος έλεγχος που πραγματοποιείται αφορά τις χρονικές περιόδους όπου ο Χρήστης με τη μεγαλύτερη πιθανότητα επιτυχίας (έστω ο Χρήστης Α) έχει μικρότερο αριθμό χρονικών περιόδων μέχρι την επόμενη προθεσμία υλοποίησης του, σε

σχέση με το Χρήστη Β. Σε αυτές τις χρονικές περιόδους ελέγχεται εάν ο Χρήστης Α έχει προτεραιότητα σε σχέση με το Χρήστη Β.

Και για τους δύο ελέγχους, γίνεται χρήση ενός επαναληπτικού βρόχου από το 0 έως το lcm (όπου lcm, το Ελάχιστο Κοινό Πολλαπλάσιο των T_1 και T_2). Για τον πρώτο έλεγχο και συγκεκριμένα για κάθε χρονική περίοδο που δεν είναι πολλαπλάσιο της μίας (T_1) ή της άλλης (T_2) περιόδου, συγκρίνεται η Απόφαση που προέκυψε από τους υπολογισμούς, με αυτήν της προηγούμενης περιόδου. Για το δεύτερο έλεγχο και συγκεκριμένα για κάθε χρονική περίοδο όπου ο Χρήστης με τη μεγαλύτερη πιθανότητα επιτυχίας (έστω ο Χρήστης Α) έχει μικρότερο αριθμό χρονικών περιόδων μέχρι την επόμενη προθεσμία υλοποίησης του, σε σχέση με το Χρήστη Β, ελέγχεται εάν η απόφαση που έχει προκύψει από τους υπολογισμούς πράγματι δίνει προτεραιότητα στο Χρήστη Α.

Τέλος, αφού ολοκληρωθούν οι δύο έλεγχοι που παρουσιάζονται παραπάνω, εκτυπώνονται στην οθόνη οι τιμές $V_0(1,1)$ και $Vk_0(1,1)$ που ουσιαστικά είναι οι Αναμενόμενοι Αριθμοί Επιτυχιών και για τις δύο πολιτικές απόφασης που αναφέρονται παραπάνω.

Ο Αλγόριθμος γράφηκε σε γλώσσα C και παρουσιάζεται στο Παράρτημα.

Κεφάλαιο 4 Συμπεράσματα

Στη συνέχεια θα παρουσιαστούν αναλυτικά τα αποτελέσματα που εξάγει το πρόγραμμα όταν για ένα τυχαίο αριθμητικό παράδειγμα.

Συγκεκριμένα, παρουσιάζουμε την περίπτωση όπου ο Χρήστης 1 έχει περίοδο $T_1 = 3$ και πιθανότητα επιτυχίας $p_1 = 0{,}6$ και ο Χρήστης 2 έχει περίοδο $T_2 = 4$ και πιθανότητα επιτυχίας $p_2 = 0{,}7$.

Ο πίνακας των υπολογισμών της Βέλτιστης Πολιτικής παρουσιάζεται παρακάτω:

Περίοδος	**Αναμενόμενος Αριθμός Επιτυχιών**			
	Καταστάσεις			
	(0,0)	**(1,0)**	**(0,1)**	**(1,1)**
12	0	0	0	0
11	0	0,6	0,7	0,7
10	0	0,84	0,91	1,33
9	0	0,936	0	1,687
8	0	0	1.861	2,287
7	1.861	2,632	2,561	2,987
6	0	2,769	0	3,438
5	2.769	3,369	3,67	4,038
4	0	0	3,74	4,417
3	0	4,611	0	5,117
2	4,611	5,211	5,463	5,717
1	4,611	5,451	5,566	6,165
0	0	0	0	**6,406**

Πίνακας 4-1: Υπολογισμοί του Αναμενόμενου Αριθμού Επιτυχιών σύμφωνα με τη Βέλτιστη Πολιτική - $V_n(i,j)$

Στη συνέχεια παρουσιάζεται ο πίνακας των υπολογισμών που έγιναν κάνοντας χρήση την Πολιτική Απόφασης που εξαρτάται από τον κανόνα $p_1 > p_2(1-p_2)^k$ που παρουσιάστηκε παραπάνω.

Περίοδος	**Αναμενόμενος Αριθμός Επιτυχιών**			
	Καταστάσεις			
	(0 , 0)	**(1 , 0)**	**(0 , 1)**	**(1 , 1)**
12	0	0	0	0
11	0	0,6	0,7	0,7
10	0	0,84	0,91	1,33
9	0	0,936	0	1,678
8	0	0	1.859	1,859
7	1.859	2,459	2,559	2,559
6	0	2,699	0	3,159
5	2.699	3,299	3,537	3,759
4	0	0	3,65	4,137
3	0	4,445	0	4,445
2	4,445	5,045	5,145	5,045
1	4,445	5,285	5,355	5,705
0	0	0	0	**6,095**

Πίνακας 4-2: Υπολογισμοί του Αναμενόμενου Αριθμού Επιτυχιών σύμφωνα με την Πολιτική Απόφασης από τον κανόνα του Bambos - $V_{kn}(i,j)$

Οι αποφάσεις των δύο Πολιτικών παρουσιάζονται στον παρακάτω πίνακα.

Χρονική Περίοδος	Απόφαση Βέλτιστη Πολιτική	Απόφαση Κανόνας Bambos
0	Χρήστης 1	Χρήστης 1
1	Χρήστης 1	Χρήστης 1
2	Χρήστης 1	Χρήστης 1
3	Χρήστης 2	Χρήστης 1
4	Χρήστης 1	Χρήστης 2
5	Χρήστης 1	Χρήστης 1
6	Χρήστης 2	Χρήστης 1
7	Χρήστης 2	Χρήστης 2
8	Χρήστης 1	Χρήστης 2
9	Χρήστης 2	Χρήστης 1
10	Χρήστης 2	Χρήστης 2
11	Χρήστης 2	Χρήστης 2

Πίνακας 4-3: Οι αποφάσεις για κάθε Χρονική Περίοδο σύμφωνα με τις δύο Πολιτικές

Στη συνέχεια ελέγχεται αν σε όλες τις χρονικές περιόδους που δεν είναι πολλαπλάσια της μίας ή της άλλης περιόδου (T_1 και T_2), άλλαξε η πολιτική απόφασης σε σχέση με την απόφαση που είχε παρθεί την προηγούμενη χρονική περίοδο. Δηλαδή, για το συγκεκριμένο παράδειγμα, ελέγχεται η αν άλλαξε η πολιτική απόφασης για τις χρονικές περιόδους 1, 2, 5, 7, 10 και 11.

Επιπλέον, πραγματοποιείται ο έλεγχος προτεραιότητας του Χρήστη με τη μεγαλύτερη πιθανότητα επιτυχίας στις περιόδους όπου αυτός έχει μικρότερη ή ίση προθεσμία υλοποίησης σε σχέση με τον άλλο Χρήστη. Στο συγκεκριμένο αριθμητικό παράδειγμα ο Χρήστης 2 είναι αυτός με την μεγαλύτερη πιθανότητα επιτυχίας ($p_2 = 0,7$) και ελέγχεται αν έχει προτεραιότητα στις χρονικές περιόδους 3, 6, 7, 9 και 11 στις οποίες έχει μικρότερη προθεσμία υλοποίησης από τον Χρήστη 1.

Για να ελέγξουμε και να επιβεβαιώσουμε την ορθότητα των συμπερασμάτων μας, εκτελέσαμε τον αλγόριθμο πάρα πολλές φορές κάνοντας χρήση ενός επαναληπτικού βρόχου, με τον αριθμό τον επαναλήψεων που επιθυμούσαμε. Επίσης διαμορφώσαμε καταλλήλως και τα αποτελέσματα της εξόδου ώστε να μας δίνει συνοπτικά τα αποτελέσματα των ελέγχων.

Στον παρακάτω πίνακα παρουσιάζονται ενδεικτικά τα αποτελέσματα των 100 πρώτων δοκιμών. Συγκεκριμένα, παρουσιάζονται τα αρχικά στοιχεία (T_1, T_2, p_1 και p_2), η στήλη ΕΚΠ περιέχει το Ελάχιστο Κοινό Πολλαπλάσιο των δύο περιόδων, στη στήλη «Task Max» παρουσιάζονται για κάθε περίπτωση ο μέγιστος αριθμός εργασιών που μπορούν να εκτελεστούν, παρουσιάζεται δηλαδή η περίπτωση που δεν χάνεται καμία εργασία καθώς ολοκληρώνεται η εκτέλεσή της πριν να συναντήσει την προθεσμία ολοκλήρωσής της.

Test	T_1	T_2	**ΕΚΠ**	p_1	p_2	**Task Max**	V_n	**Ποσοστιαία Διαφορά V_n με Task Max**	V_{kn}	**Ποσοστιαία Διαφορά V_{kn} με Task Max**	**Ποσοστιαία Διαφορά V_{kn} με V_n**
1	3	9	9	0,8	0,5	4	3,943	- 1,43%	3,943	- 1,43%	0%
2	11	6	66	0,4	0,1	17	10,294	- 39,45%	9,675	- 43,09%	- 6,01%
3	4	6	12	0,9	0,3	5	4,621	- 7,58%	4,564	- 8,72%	- 1,23%
4	3	9	9	0,8	0,6	4	3,964	- 0,90%	3,964	- 0,90%	0%
5	7	4	28	0,4	0,1	11	5,530	- 49,73%	4,767	- 56,66%	- 13,80%
6	3	6	6	0,6	0,1	3	2,129	- 29,03%	2,129	- 29,03%	0%
7	4	4	4	0,7	0,6	2	1,861	- 6,95%	1,861	- 6,95%	0%
8	10	7	70	0,3	0,1	17	10,668	- 37,25%	9,850	- 42,06%	- 7,67%
9	3	10	30	0,7	0,5	13	12,651	- 2,68%	12,496	- 3,88%	- 1,23%
10	9	11	99	0,8	0,4	20	19,953	- 0,24%	19,928	- 0,36%	- 0,13%
11	5	3	15	0,3	0,4	8	5,199	- 35,01%	4,999	- 37,51%	- 3,85%
12	5	6	30	0,4	0,8	11	10,420	- 5,27%	10,122	- 7,98%	- 2,86%
13	5	10	10	0,6	0,6	3	2,972	- 0,93%	2,909	- 3,03%	- 2,12%
14	4	9	36	0,4	0,8	13	11,781	- 9,38%	11,488	- 11,63%	- 2,49%
15	5	3	15	0,8	0,5	8	7,181	- 10,24%	6,891	- 13,86%	- 4,04%

Test	T_1	T_2	ΕΚΠ	p_1	p_2	Task Max	V_n	Ποσοστιαία Διαφορά V_n με Task Max	V_{kn}	Ποσοστιαία Διαφορά V_{kn} με Task Max	Ποσοστιαία Διαφορά V_{kn} με V_n
16	8	7	56	0,6	0,7	15	14,984	- 0,11%	14,966	- 0,23%	- 0,12%
17	10	8	40	0,5	0,6	9	8,983	- 0,19%	8,972	- 0,31%	- 0,12%
18	6	10	30	0,6	0,1	8	6,588	- 17,65%	6,498	- 18,78%	- 1,37%
19	2	11	22	0,8	0,6	13	12,518	- 3,71%	12,347	- 5,02%	- 1,37%
20	8	3	24	0,4	0,3	11	7,257	- 34,03%	6,981	- 36,54%	- 3,80%
21	11	5	55	0,1	0,2	16	8,877	- 44,52%	8,830	- 44,81%	- 0,53%
22	8	2	8	0,8	0,6	5	4,304	- 13,92%	4,298	- 14,04%	- 0,14%
23	3	10	30	0,7	0,4	13	12,535	- 3,58%	12,363	- 4,90%	- 1,37%
24	8	5	40	0,3	0,2	13	8,593	- 33,90%	8,227	- 36,72%	- 4,26%
25	10	4	20	0,7	0,9	7	6,999	- 0,01%	6,998	- 0,03%	- 0,01%
26	5	7	35	0,5	0,4	12	11,230	- 6,42%	10,965	- 8,63%	- 2,36%
27	6	2	6	0,1	0,3	4	1,617	- 59,58%	1,617	- 59,58%	0%
28	5	8	40	0,9	0,9	13	13,000	0%	12,999	- 0,01%	- 0,01%
29	6	4	12	0,7	0,7	5	4,953	- 0,94%	4,858	- 2,84%	- 1,92%
30	9	5	45	0,4	0,7	14	13,826	- 1,24%	13,734	- 1,90%	- 0,67%
31	8	2	8	0,9	0,7	5	4,625	- 7,50%	4,624	- 7,52%	- 0,02%
32	7	9	63	0,4	0,5	16	15,593	- 2,54%	15,411	- 3,68%	- 1,17%
33	3	4	12	0,9	0,8	7	6,963	- 0,53%	6,885	- 1,64%	- 1,12%
34	3	6	6	0,8	0,2	3	2,522	- 15,93%	2,522	- 15,93%	0%
35	9	3	9	0,8	0,7	4	3,915	- 2,13%	3,915	- 2,13%	0%
36	9	9	9	0,7	0,7	2	2,000	0%	0	- 100,00%	- 100,00%
37	7	11	77	0,8	0,3	18	17,787	- 1,18%	17,713	- 1,59%	- 0,42%
38	3	10	30	0,6	0,8	13	12,351	- 4,99%	12,151	- 6,53%	- 1,62%
39	8	2	8	0,7	0,9	5	4,941	- 1,18%	4,941	- 1,18%	0%
40	2	3	6	0,7	0,9	5	4,440	- 11,20%	4,119	- 17,62%	- 7,23%
41	2	11	22	0,4	0,9	13	8,970	- 31,00%	8,687	- 33,18%	- 3,15%
42	10	5	10	0,2	0,1	3	1,363	- 54,57%	1,280	- 57,33%	- 6,09%
43	4	2	4	0,5	0,3	3	1,507	- 49,77%	1,416	- 52,80%	- 6,04%
44	6	2	6	0,7	0,1	4	1,441	- 63,98%	1,233	- 69,18%	- 14,43%

Test	T_1	T_2	ΕΚΠ	p_1	p_2	Task Max	V_n	Ποσοστιαία Διαφορά V_n με Task Max	V_{kn}	Ποσοστιαία Διαφορά V_{kn} με Task Max	Ποσοστιαία Διαφορά V_{kn} με V_n
45	10	11	110	0,4	0,5	21	20,882	- 0,56%	20,818	- 0,87%	- 0,31%
46	3	6	6	0,9	0,2	3	2,565	- 14,50%	2,565	- 14,50%	0%
47	10	10	10	0,7	0,9	2	2,000	0%	2,000	0%	0%
48	9	9	9	0,3	0,8	2	1,935	- 3,25%	1,935	- 3,25%	0%
49	10	5	10	0,6	0,1	3	1,730	- 42,33%	1,721	- 42,63%	- 0,52%
50	2	9	18	0,8	0,2	11	9,724	- 11,60%	9,584	- 12,87%	- 1,44%
51	11	8	88	0,9	0,1	19	14,057	- 26,02%	13,845	- 27,13%	- 1,51%
52	2	11	22	0,5	0,3	13	9,429	- 27,47%	9,246	- 28,88%	- 1,94%
53	10	5	10	0,6	0,2	3	2,262	- 24,60%	2,257	- 24,77%	- 0,22%
54	6	11	66	0,5	0,6	17	16,811	- 1,11%	16,724	- 1,62%	- 0,52%
55	7	7	7	0,7	0,7	2	1,996	- 0,20%	0	- 100,00%	- 100,00%
56	5	9	45	0,2	0,9	14	10,896	- 22,17%	10,636	- 24,03%	- 2,39%
57	10	2	10	0,4	0,4	6	3,782	- 36,97%	3,662	- 38,97%	- 3,17%
58	2	8	8	0,3	0,9	5	2,943	- 41,14%	2,919	- 41,62%	- 0,82%
59	11	10	110	0,2	0,5	21	19,774	- 5,84%	19,583	- 6,75%	- 0,97%
60	4	4	4	0,8	0,8	2	1,971	- 1,45%	0	- 100,00%	- 100,00%
61	10	11	110	0,2	0,3	21	18,634	- 11,27%	18,336	- 12,69%	- 1,60%
62	10	3	30	0,7	0,2	13	7,595	- 41,58%	7,173	- 44,82%	- 5,56%
63	8	3	24	0,3	0,7	11	10,130	- 7,91%	9,930	- 9,73%	- 1,97%
64	4	7	28	0,7	0,3	11	10,193	- 7,34%	9,991	- 9,17%	- 1,98%
65	4	8	8	0,3	0,3	3	2,095	- 30,17%	1,697	- 43,43%	- 19,00%
66	11	4	44	0,4	0,4	15	13,179	- 12,14%	12,752	- 14,99%	- 3,24%
67	3	10	30	0,9	0,4	13	12,893	- 0,82%	12,839	- 1,24%	- 0,42%
68	3	11	33	0,3	0,1	14	8,000	- 42,86%	7,673	- 45,19%	- 4,09%
69	4	11	44	0,9	0,3	15	14,807	- 1,29%	14,748	- 1,68%	- 0,40%
70	2	2	2	0,7	0,5	2	1,260	- 37,00%	1,260	- 37,00%	0%
71	3	10	30	0,1	0,7	13	5,442	- 58,14%	5,325	- 59,04%	- 2,15%
72	7	5	35	0,8	0,2	12	9,420	- 21,50%	9,110	- 24,08%	- 3,29%
73	4	7	28	0,8	0,8	11	10,987	- 0,12%	10,956	- 0,40%	- 0,28%
74	9	9	9	0,5	0,9	2	1,996	- 0,20%	1,996	- 0,20%	0%

Test	T_1	T_2	ΕΚΠ	p_1	p_2	Task Max	V_n	Ποσοστιαία Διαφορά V_n με Task Max	V_{kn}	Ποσοστιαία Διαφορά V_{kn} με Task Max	Ποσοστιαία Διαφορά V_{kn} με V_n
75	11	6	66	0,9	0,9	17	17,000	0%	17,000	- 0,00%	0%
76	8	11	88	0,2	0,1	19	12,361	- 34,94%	11,766	- 38,07%	- 4,81%
77	7	7	7	0,6	0,8	2	1,993	- 0,35%	1,993	- 0,35%	0%
78	3	4	12	0,5	0,7	7	6,040	- 13,71%	5,686	- 18,77%	- 5,86%
79	10	5	10	0,3	0,4	3	2,652	- 11,60%	2,652	- 11,60%	0%
80	11	8	88	0,2	0,2	19	14,537	- 23,49%	13,581	- 28,52%	- 6,58%
81	11	9	99	0,4	0,8	20	19,953	- 0,24%	19,928	- 0,36%	- 0,13%
82	5	7	35	0,4	0,5	12	11,034	- 8,05%	10,727	- 10,61%	- 2,78%
83	8	7	56	0,9	0,2	15	13,169	- 12,21%	12,716	- 15,23%	- 3,44%
84	7	6	42	0,7	0,2	13	10,785	- 17,04%	10,310	- 20,69%	- 4,40%
85	8	3	24	0,9	0,6	11	10,477	- 4,75%	10,279	- 6,55%	- 1,89%
86	5	7	35	0,1	0,2	12	5,300	- 55,83%	5,156	- 57,03%	- 2,72%
87	4	10	20	0,6	0,3	7	6,627	- 5,33%	6,559	- 6,30%	- 1,03%
88	5	8	40	0,8	0,2	13	11,787	- 9,33%	11,620	- 10,62%	- 1,42%
89	8	4	8	0,5	0,5	3	2,777	- 7,43%	2,563	- 14,57%	- 7,71%
90	8	4	8	0,2	0,2	3	1,510	- 49,67%	1,122	- 62,60%	- 25,70%
91	7	2	14	0,5	0,9	9	8,728	- 3,02%	8,590	- 4,56%	- 1,58%
92	6	11	66	0,4	0,7	17	16,463	- 3,16%	16,302	- 4,11%	- 0,98%
93	5	6	30	0,1	0,9	11	7,166	- 34,85%	6,497	- 40,94%	- 9,34%
94	7	11	77	0,2	0,7	18	15,528	- 13,73%	15,263	- 15,21%	- 1,71%
95	6	4	12	0,9	0,6	5	4,914	- 1,72%	4,868	- 2,64%	- 0,94%
96	9	7	63	0,4	0,3	16	14,660	- 8,38%	14,336	- 10,40%	- 2,21%
97	4	7	28	0,9	0,5	11	10,920	- 0,73%	10,863	- 1,25%	- 0,52%
98	6	6	6	0,4	0,9	2	1,916	- 4,20%	1,916	- 4,20%	0%
99	4	11	44	0,8	0,9	15	14,982	- 0,12%	14,963	- 0,25%	- 0,13%
100	9	9	9	0,4	0,6	2	1,970	- 1,50%	1,970	- 1,50%	0%

Πίνακας 4-4: Αναλυτικά αποτελέσματα υπολογισμών αλγορίθμου για 100 διαφορετικές περιπτώσεις

Παρατηρήσεις Αποτελεσμάτων:

1. Από τα παραπάνω αποτελέσματα επιβεβαιώνεται η Βέλτιστη Πολιτική Απόφασης, δηλαδή ότι η απόφαση δεν αλλάζει στις περιόδους που δεν είναι πολλαπλάσια των περιόδων T_1 και T_2. Επιπλέον, επιβεβαιώνεται η υπόθεση ότι ο Χρήστης με την μεγαλύτερη πιθανότητα επιτυχίας έχει προτεραιότητα στις χρονικές περιόδους που έχει μικρότερη ή ίση προθεσμία υλοποίησης σε σχέση με τον άλλο Χρήστη.
2. Το εύρος των χρονικών περιόδων που χρησιμοποιήθηκε στα αριθμητικά παραδείγματα είναι από 1 – 11 καθώς για μεγαλύτερους αριθμούς ο Αναμενόμενος Αριθμός Επιτυχιών της Βέλτιστης Πολιτικής (V_n) ήταν ίσος με το μέγιστο αριθμό εργασιών που μπορούσαν να εκτελεστούν ακόμα και για μικρές πιθανότητας επιτυχίας.
3. Ο Αναμενόμενος Αριθμός Επιτυχιών της Πολιτικής Απόφασης σύμφωνα με τον κανόνα του Bambos (V_{kn}) στις περιπτώσεις όπου ο μέγιστος αριθμός εργασιών που μπορούν να εκτελεστούν είναι πολύ μικρός (ειδικά στις περιπτώσεις που οι δύο Χρήστες έχουν την ίδια περίοδο) πολλές φορές να είναι και 0.
4. Η μέση ποσοστιαία διαφορά του Αναμενόμενου Αριθμού Επιτυχιών της Βέλτιστης Πολιτικής (V_n) σε σχέση με το μέγιστο αριθμό εργασιών που μπορούσαν να εκτελεστούν είναι 18,95%.
5. Η μέση ποσοστιαία διαφορά του Αναμενόμενου Αριθμού Επιτυχιών της Πολιτικής Απόφασης σύμφωνα με τον κανόνα του Bambos (V_{kn}) σε σχέση με το μέγιστο αριθμό εργασιών που μπορούσαν να εκτελεστούν είναι 21,56%.
6. Η μέση ποσοστιαία διαφορά του Αναμενόμενου Αριθμού Επιτυχιών της Πολιτικής Απόφασης σύμφωνα με τον κανόνα του Bambos (V_{kn}) σε σχέση με τον Αναμενόμενο Αριθμό Επιτυχιών της Βέλτιστης Πολιτικής (V_n) είναι 3,58%.

Παράρτημα

```
#include <stdio.h>
#include<stdlib.h>
#include <math.h>


int lcm (int a , int b);
int main()

{
int t1, t2, ekp, n, i, j, l, m, sit, r, z, x, dl1, dl2, deadline1, deadline2, k, test1, test2, test3;

float p11, p22, p1,p2,riza,bskelos;

float d1, d2, Vn, dd1, dd2, max11, max12, max21, max22, max31, max32, max41, max42,
max51, max52, max61, max62, max71, max72, max81, max82, max91, max92, maxx11,
maxx12, maxx21, maxx22, maxx31, maxx32, maxx41, maxx42, maxx51, maxx52, maxx61,
maxx62, maxx71, maxx72, maxx81, maxx82, maxx91, maxx92;

FILE *f;
f=fopen("test.txt","w");

/*for (i=0;i<1000;i++)
{
                t1=rand() % 10+2;
                t2=rand() % 10+2;
                p11=rand() % 9+1;
                p22=rand() % 9+1;
                p1=(p11/10);
                p2=(p22/10);

Ean theloume na to treksoume polles fores aytomata kai gia tyxaious arithmous */


printf("DWSE THN PERIODO TOY 1ou PAKETOY (T1)\n\n");
scanf("%d",&t1);

printf("\nDWSE THN PITHANOTHTA EPITYXIAS GIA TO 1o PAKETO\n\n");
scanf("%f",&p1);
```

```
printf("\nDWSE THN PERIODO TOY 2ou PAKETOY (T2)\n\n");
scanf("%d",&t2);

printf("\nDWSE THN PITHANOTHTA EPITYXIAS GIA TO 2o PAKETO\n\n");
scanf("%f",&p2);

ekp=lcm(t1,t2);
n=ekp;
sit=4;
r=sit;
float V[ekp][sit];                    /* Dilwsi kai arxikopoiisi pinaka*/
for(l=0;l<=ekp;l++)
{
                for(m=0;m<=sit;m++)
                {
                        V[l][m]=0;
                }
}

float Vk[ekp][sit];                   /* Dilwsi kai arxikopoiisi pinaka*/
for(l=0;l<=ekp;l++)
{
                for(m=0;m<=sit;m++)
                {
                        Vk[l][m]=0;
                }
}

int Apofasi[ekp];                     /* Dilwsi kai arxikopoiisi pinaka*/
for(z=0;z<ekp+1;z++)
{
                Apofasi[z]=0;
}

int Apofasi2[ekp];                    /* Dilwsi kai arxikopoiisi pinaka*/
for(z=0;z<ekp+1;z++)
{
                Apofasi2[z]=0;
}

int Pollap1[(ekp/t1)];                /* Dilwsi kai arxikopoiisi pinaka*/
for(z=0;z<((ekp/t1)+1);z++)
{
                Pollap1[z]=0;
}

for(z=1;z<((ekp/t1)+1);z++)
```

```
{
		Pollap1[z]=z*t1;
}

int Pollap2[(ekp/t2)];			/* Dilwsi kai arxikopoiisi pinaka*/
for(z=0;z<((ekp/t2)+1);z++)
{
		Pollap2[z]=0;
}
for(z=1;z<((ekp/t2)+1);z++)
{
		Pollap2[z]=z*t2;
}

fprintf(f,"\nT1=%d\n\n",t1);
fprintf(f,"p1=%1.2f\n\n",p1);
fprintf(f,"T2=%d\n\n",t2);
fprintf(f,"p2=%1.2f\n\n",p2);
fprintf(f,"EKP=%d\n\n",ekp);
/*printf("n=%d\n\n",n);					Testing n */

for(n=(ekp-1);n>=0;n--)
{
/* printf("n=%d\n\n",n);					n indicator*/
	if (n%t1==0 && n%t2==0)
 /*Otan i xroniki stigmi n einai polaplasio kai tou t1 kai tou t2 moni pitnani katastasi h (1,1) */
 {	V[n][0]=0;
		V[n][1]=0;
		V[n][2]=0;
		max11=(p1*(1 + V[n+1][2]))+((1-p1)*(0 + V[n+1][3]));
		max12=(p2*(1 + V[n+1][1]))+((1-p2)*(0 + V[n+1][3]));
		 if (max11>max12)
		{	V[n][3]= max11;
			Apofasi[n]=1;
		}
		if (max11<max12)
		{	V[n][3]= max12;
			Apofasi[n]=2;
		}
		if (max11==max12)
		{	V[n][3]= max11;
			Apofasi[n]=0;
		}
	}

if (n%t1==0 && n%t2==0)
/*Otan i xroniki stigmi n einai polaplasio kai tou t1 kai tou t2 moni pitnani katastasi h (1,1) */
 {	Vk[n][0]=0;
```

```
Vk[n][1]=0;
Vk[n][2]=0;
maxx11=(p1*(1 + Vk[n+1][2]))+((1-p1)*(0 + Vk[n+1][3]));
maxx12=(p2*(1 + Vk[n+1][1]))+((1-p2)*(0 + Vk[n+1][3]));

for (x=1;x<=ekp;x++)
{
        if (n<=Pollap1[x])
        {
                deadline1=Pollap1[x]-n;
                /* fprintf(f,"\nPeriodos %4d: Deadline tou
                user 1=%d\n",n,deadline1); */
                break;
        }
}
for (x=1;x<=ekp;x++)
{
        if (n<=Pollap2[x])
        {
                deadline2=Pollap2[x]-n;
                /* fprintf(f,"\nPeriodos %4d: Deadline tou
                user 2=%d\n",n,deadline2); */
                break;
        }
}

k1=deadline1-deadline2;
k2=abs(k1);
/* fprintf(f,"\nk=%d\n",k);                    /*testing k*/
riza=1-p2;
bskelos=p2*(pow(riza,k2));

if (k1<0 && p1>bskelos)
{   Vk[n][3]=maxx11;
   Apofasi2[n]=1;
}
if (k1<0 && p1<=bskelos)
{   Vk[n][3]=maxx12;
   Apofasi2[n]=2;
}
if (k1>0 && p2<=bskelos)
{   Vk[n][3]=maxx11;
   Apofasi2[n]=1;
}
if (k1>0 && p2>bskelos)
{   Vk[n][3]=maxx12;
   Apofasi2[n]=2;
}
```

```
                if (k1==0 && p1>p2)
                {   Vk[n][3]=maxx11;
                    Apofasi2[n]=1;
                }
                if (k1==0 && p1<p2)
                {   Vk[n][3]=maxx12;
                    Apofasi2[n]=2;
                }
        }


if (n%t1==0 && n%t2!=0 && (n+1)%t2 != 0)
/*Otan i xroniki stigmi n einai polaplasio tou t1 kai oxi tou t2 kai to n+1 den einai pollaplasio
tou t2 - pitnanes katastaseis (1,0) kai (1,1)*/
        {       V[n][0]=0;
                V[n][2]=0;
                V[n][1]=(p1*(1 + V[n+1][0]))+((1-p1)*(0 + V[n+1][1]));
                max21=(p1*(1 + V[n+1][2]))+((1-p1)*(0 + V[n+1][3]));
                max22=(p2*(1 + V[n+1][1]))+((1-p2)*(0 + V[n+1][3]));
        if (max21>max22)
                {       V[n][3]= max21;
                        Apofasi[n]=1;
                }
                 if (max21<max22)
                {       V[n][3]= max22;
                        Apofasi[n]=2;
                }
                if (max21==max22)
                {        V[n][3]= max21;
                        Apofasi[n]=0;
                }
        }


        if (n%t1==0 && n%t2!=0 && (n+1)%t2 != 0)
/*Otan i xroniki stigmi n einai polaplasio tou t1 kai oxi tou t2 kai to n+1 den einai pollaplasio
tou t2 - pitnanes katastaseis (1,0) kai (1,1)*/
        {       Vk[n][0]=0;
Vk[n][2]=0;
                Vk[n][1]=(p1*(1 + Vk[n+1][0]))+((1-p1)*(0 + Vk[n+1][1]));
                maxx21=(p1*(1 + Vk[n+1][2]))+((1-p1)*(0 + Vk[n+1][3]));
                maxx22=(p2*(1 + Vk[n+1][1]))+((1-p2)*(0 + Vk[n+1][3]));

                for(x=1;x<=ekp;x++)
                {
                        if (n<=Pollap1[x])
                        {
                                deadline1=Pollap1[x]-n;
```

```
                                /* fprintf(f,"\nPeriodos %4d: Deadline tou
user 1=%d\n",n,deadline1); */
                                break;
                        }
                }
                for(x=1;x<=ekp;x++)
                {
                        if (n<=Pollap2[x])
                        {
                                deadline2=Pollap2[x]-n;
                                /* fprintf(f,"\nPeriodos %4d: Deadline tou
user 2=%d\n",n,deadline2); */
                                break;
                        }
                }

k1=deadline1-deadline2;
    k2=abs(k1);
    /* fprintf(f,"\nk=%d\n",k);                    /*testing k*/
    riza=1-p2;
    bskelos=p2*(pow(riza,k2));

    if (k1<0 && p1>bskelos)
    {   Vk[n][3]=maxx21;
       Apofasi2[n]=1;
    }
    if (k1<0 && p1<=bskelos)
    {   Vk[n][3]=maxx22;
       Apofasi2[n]=2;
    }
    if (k1>0 && p2<=bskelos)
    {   Vk[n][3]=maxx21;
       Apofasi2[n]=1;
    }
    if (k1>0 && p2>bskelos)
    {   Vk[n][3]=maxx22;
       Apofasi2[n]=2;
    }
    if (k1==0 && p1>p2)
    {   Vk[n][3]=maxx21;
       Apofasi2[n]=1;
    }
    if (k1==0 && p1<p2)
    {   Vk[n][3]=maxx22;
       Apofasi2[n]=2;
    }
        }
```

```
        if (n%t1==0 && n%t2!=0 && (n+1)%t2 == 0)
/*Otan i xroniki stigmi n einai polaplasio tou t1 kai oxi tou t2 kai to n+1 einai pollaplasio tou
t2 - pitnanes katastaseis (1,0) kai (1,1) */
        {       V[n][0]=0;
                V[n][2]=0;
                V[n][1]=(p1*(1 + V[n+1][2]))+((1-p1)*(0 + V[n+1][3]));
                max31=(p1*(1 + V[n+1][2]))+((1-p1)*(0 + V[n+1][3]));
                max32=(p2*(1 + V[n+1][3]))+((1-p2)*(0 + V[n+1][3]));
                if (max31>max32)
                {       V[n][3]= max31;
                        Apofasi[n]=1;
                }
                if (max31<max32)
                {       V[n][3]= max32;
                        Apofasi[n]=2;
                }
                if (max31==max32)
                {       V[n][3]= max31;
                        Apofasi[n]=0;
                }
        }

        if (n%t1==0 && n%t2!=0 && (n+1)%t2 == 0)
/*Otan i xroniki stigmi n einai polaplasio tou t1 kai oxi tou t2 kai to n+1 einai pollaplasio tou
t2 - pitnanes katastaseis (1,0) kai (1,1) */
        {       Vk[n][0]=0;
                Vk[n][2]=0;
                Vk[n][1]=(p1*(1 + Vk[n+1][2]))+((1-p1)*(0 + Vk[n+1][3]));
                maxx31=(p1*(1 + Vk[n+1][2]))+((1-p1)*(0 + Vk[n+1][3]));
                maxx32=(p2*(1 + Vk[n+1][3]))+((1-p2)*(0 + Vk[n+1][3]));

                for(x=1;x<=ekp;x++)
                {
                        if (n<=Pollap1[x])
                        {
                                deadline1=Pollap1[x]-n;
                                /* fprintf(f,"\nPeriodos %4d: Deadline tou
                                user 1=%d\n",n,deadline1); */
                                break;
                        }
                }
```

```
                for(x=1;x<=ekp;x++)
                {
                        if (n<=Pollap2[x])
                        {
                                deadline2=Pollap2[x]-n;
                                /* fprintf(f,"\nPeriodos %4d: Deadline tou
                                user 2=%d\n",n,deadline2); */
                                break;
                        }
                }

                k1=deadline1-deadline2;
                k2=abs(k1);
                /* fprintf(f,"\nk=%d\n",k);                    /*testing k*/
                riza=1-p2;
                bskelos=p2*(pow(riza,k2));

                if (k1<0 && p1>bskelos)
                {  Vk[n][3]=maxx31;
                   Apofasi2[n]=1;
                }
                if (k1<0 && p1<=bskelos)
                {  Vk[n][3]=maxx32;
                   Apofasi2[n]=2;
                }
                if (k1>0 && p2<=bskelos)
                {  Vk[n][3]=maxx31;
                   Apofasi2[n]=1;
                }
                if (k1>0 && p2>bskelos)
                {  Vk[n][3]=maxx32;
                   Apofasi2[n]=2;
                }
                if (k1==0 && p1>p2)
                {  Vk[n][3]=maxx31;
                   Apofasi2[n]=1;
                }
                if (k1==0 && p1<p2)
                {  Vk[n][3]=maxx32;
                   Apofasi2[n]=2;
                }
        }

if (n%t1!=0 && n%t2==0 && (n+1)%t1 != 0)
/*Otan i xroniki stigmi n einai polaplasio tou t2 kai oxi tou t1 kai to n+1 den einai pollaplasio
tou t1- pitnanes katastaseis (0,1) kai (1,1) */
        {       V[n][0]=0;
                V[n][1]=0;
```

```
                V[n][2]=(p2*(1 + V[n+1][0]))+((1-p2)*(0 + V[n+1][2]));
                max41=(p1*(1 + V[n+1][2]))+((1-p1)*(0 + V[n+1][3]));
                max42=(p2*(1 + V[n+1][1]))+((1-p2)*(0 + V[n+1][3]));
                if (max41>max42)
                {       V[n][3]= max41;
                        Apofasi[n]=1;
                }
                if (max41<max42)
                {       V[n][3]= max42;
                        Apofasi[n]=2;
                }
                if (max41==max42)
                {       V[n][3]= max41;
                        Apofasi[n]=0;
                }
        }

 if (n%t1!=0 && n%t2==0 && (n+1)%t1 != 0)
/*Otan i xroniki stigmi n einai polaplasio tou t2 kai oxi tou t1 kai to n+1 den einai pollaplasio
tou t1- pitnanes katastaseis (0,1) kai (1,1) */
        {       Vk[n][0]=0;
                Vk[n][1]=0;
                Vk[n][2]=(p2*(1 + Vk[n+1][0]))+((1-p2)*(0 + Vk[n+1][2]));
                maxx41=(p1*(1 + Vk[n+1][2]))+((1-p1)*(0 + Vk[n+1][3]));
                maxx42=(p2*(1 + Vk[n+1][1]))+((1-p2)*(0 + Vk[n+1][3]));

                for(x=1;x<=ekp;x++)
                {
                        if (n<=Pollap1[x])
                        {
                                deadline1=Pollap1[x]-n;
                                /* fprintf(f,"\nPeriodos %4d: Deadline tou
                                user 1=%d\n",n,deadline1); */
                                break;
                        }
                }

                for(x=1;x<=ekp;x++)
                {
                        if (n<=Pollap2[x])
                        {
                                deadline2=Pollap2[x]-n;
                                /* fprintf(f,"\nPeriodos %4d: Deadline tou
                                user 2=%d\n",n,deadline2); */
                                break;
                        }
                }
```

```
k1=deadline1-deadline2;
k2=abs(k1);
/* fprintf(f,"\nk=%d\n",k);                    /*testing k*/
riza=1-p2;
bskelos=p2*(pow(riza,k2));

if (k1<0 && p1>bskelos)
{  Vk[n][3]=maxx41;
   Apofasi2[n]=1;
}
if (k1<0 && p1<=bskelos)
{  Vk[n][3]=maxx42;
   Apofasi2[n]=2;
}
if (k1>0 && p2<=bskelos)
{  Vk[n][3]=maxx41;
   Apofasi2[n]=1;
}
if (k1>0 && p2>bskelos)
{  Vk[n][3]=maxx42;
   Apofasi2[n]=2;
}
if (k1==0 && p1>p2)
{  Vk[n][3]=maxx41;
   Apofasi2[n]=1;
}
if (k1==0 && p1<p2)
{  Vk[n][3]=maxx42;
   Apofasi2[n]=2;
}
}

    if (n%t1!=0 && n%t2==0 && (n+1)%t1 == 0)
/*Otan i xroniki stigmi n einai polaplasio tou t2 kai oxi tou t1 kai to n+1 einai pollaplasio tou t1- pitnanes katastaseis (0,1) kai (1,1)*/
    {       V[n][0]=0;
            V[n][1]=0;
            V[n][2]=(p2*(1 + V[n+1][1]))+((1-p2)*(0 + V[n+1][3]));
            max51=(p1*(1 + V[n+1][3]))+((1-p1)*(0 + V[n+1][3]));
            max52=(p2*(1 + V[n+1][1]))+((1-p2)*(0 + V[n+1][3]));
            if (max51>max52)
            {       V[n][3]= max51;
                    Apofasi[n]=1;
            }
```

```
                if (max51<max52)
                {       V[n][3]= max52;
                        Apofasi[n]=2;
                }
                if (max51==max52)
                {       V[n][3]= max51;
                        Apofasi[n]=0;
                }
}

        if (n%t1!=0 && n%t2==0 && (n+1)%t1 == 0)
/*Otan i xroniki stigmi n einai polaplasio tou t2 kai oxi tou t1 kai to n+1 einai pollaplasio tou
t1- pitnanes katastaseis (0,1) kai (1,1)*/
{       Vk[n][0]=0;
                Vk[n][1]=0;
                Vk[n][2]=(p2*(1 + Vk[n+1][1]))+((1-p2)*(0 + Vk[n+1][3]));
                maxx51=(p1*(1 + Vk[n+1][3]))+((1-p1)*(0 + Vk[n+1][3]));
                maxx52=(p2*(1 + Vk[n+1][1]))+((1-p2)*(0 + Vk[n+1][3]));

                for(x=1;x<=ekp;x++)
                {
                        if (n<=Pollap1[x])
                        {
                                deadline1=Pollap1[x]-n;
                                /* fprintf(f,"\nPeriodos %4d: Deadline tou
user 1=%d\n",n,deadline1); */
                                break;
                        }
                }
                for(x=1;x<=ekp;x++)
                {
                        if (n<=Pollap2[x])
                        {
                                deadline2=Pollap2[x]-n;
                                /* fprintf(f,"\nPeriodos %4d: Deadline tou
user 2=%d\n",n,deadline2); */
                                break;
                        }
                }
    k1=deadline1-deadline2;
    k2=abs(k1);
    /* fprintf(f,"\nk=%d\n",k);                    /*testing k*/
    riza=1-p2;
    bskelos=p2*(pow(riza,k2));
```

```
    if (k1<0 && p1>bskelos)
    {   Vk[n][3]=maxx51;
        Apofasi2[n]=1;
    }
    if (k1<0 && p1<=bskelos)
    {   Vk[n][3]=maxx52;
        Apofasi2[n]=2;
    }
    if (k1>0 && p2<=bskelos)
    {   Vk[n][3]=maxx51;
        Apofasi2[n]=1;
    }
    if (k1>0 && p2>bskelos)
    {   Vk[n][3]=maxx52;
        Apofasi2[n]=2;
    }
    if (k1==0 && p1>p2)
    {   Vk[n][3]=maxx51;
        Apofasi2[n]=1;
    }
    if (k1==0 && p1<p2)
    {   Vk[n][3]=maxx52;
        Apofasi2[n]=2;
    }
}
        if (n%t1!=0 && n%t2!=0)
        {
                for(r=0;r<sit;r++)
                {
                        if(r==0 && (n+1)%t1 != 0 && (n+1)%t2 != 0)
                        {       V[n][r]= V[n+1][r]+ 0;
                        }
                        else if (r==0 && (n+1)%t1 == 0 && (n+1)%t2 != 0 )
                        {       V[n][r]= V[n+1][r+1] + 0 ;
                        }
                        else if (r==0 && (n+1)%t1 != 0 && (n+1)%t2 == 0 )
                        {       V[n][r]= V[n+1][r+2] + 0 ;
                        }
                        else if (r==0 && (n+1)%t1 == 0 && (n+1)%t2 == 0 )
                        {       V[n][r]= V[n+1][r+3] + 0 ;
                        }

                        else if (r==1 && (n+1)%t1 != 0 && (n+1)%t2 != 0)
                        {       V[n][r]= (p1*(1 + V[n+1][r-1]))+((1-p1)*(0 + V[n+1][r]));
                        }
                        else if (r==1 && (n+1)%t1 == 0 && (n+1)%t2 != 0)
                        {       V[n][r]= (p1*(1 + V[n+1][r]))+((1-p1)*(0 + V[n+1][r]));
                        }
```

```
else if (r==1 && (n+1)%t1 != 0 && (n+1)%t2 == 0)
{       V[n][r]= (p1*(1 + V[n+1][r+1]))+((1-p1)*(0 + V[n+1][r+2]));
}
else if (r==1 && (n+1)%t1 == 0 && (n+1)%t2 == 0)
{       V[n][r]= (p1*(1 + V[n+1][r+2]))+((1-p1)*(0 + V[n+1][r+2]));
}


else if (r==2 && (n+1)%t1 != 0 && (n+1)%t2 != 0)
{       V[n][r]= (p2*(1 + V[n+1][r-2]))+((1-p2)*(0 + V[n+1][r]));
}
else if (r==2 && (n+1)%t1 == 0 && (n+1)%t2 != 0)
{       V[n][r]= (p2*(1 + V[n+1][r-1]))+((1-p2)*(0 + V[n+1][r+1]));
}
else if (r==2 && (n+1)%t1 != 0 && (n+1)%t2 == 0)
{       V[n][r]= (p2*(1 + V[n+1][r]))+((1-p2)*(0 + V[n+1][r]));
}
else if (r==2 && (n+1)%t1 == 0 && (n+1)%t2 == 0)
{       V[n][r]= (p2*(1 + V[n+1][r+1]))+((1-p2)*(0 + V[n+1][r+1]));
}


else if (r==3 && (n+1)%t1 != 0 && (n+1)%t2 != 0)
{
        max61=(p1*(1 + V[n+1][r-1]))+((1-p1)*(0 + V[n+1][r]));
        max62=(p2*(1 + V[n+1][r-2]))+((1-p2)*(0 + V[n+1][r]));
        if (max61>max62)
        {       V[n][r]= max61;
                Apofasi[n]=1;
        }
        if (max61<max62)
        {       V[n][r]= max62;
                Apofasi[n]=2;
        }
         if (max61==max62)
        {       V[n][r]= max61;
                Apofasi[n]=0;
        }
}

else if (r==3 && (n+1)%t1 == 0 && (n+1)%t2 != 0)
{
        max71=(p1*(1 + V[n+1][r]))+((1-p1)*(0 + V[n+1][r]));
        max72=(p2*(1 + V[n+1][r-2]))+((1-p2)*(0 + V[n+1][r]));
        if (max71>max72)
        {       V[n][r]= max71;
                Apofasi[n]=1;
        }
```

```
                    if (max71<max72)
                    {       V[n][r]= max72;
                            Apofasi[n]=2;
                    }
                    if (max71==max72)
                    {       V[n][r]= max71;
                            Apofasi[n]=0;
                    }
            }

            else if (r==3 && (n+1)%t1 != 0 && (n+1)%t2 == 0)
            {
                    max81=(p1*(1 + V[n+1][r-1]))+((1-p1)*(0 + V[n+1][r]));
                    max82=(p2*(1 + V[n+1][r]))+((1-p2)*(0 + V[n+1][r]));
                    if (max81>max82)
                    {       V[n][r]= max81;
                            Apofasi[n]=1;
                    }
                    if (max81<max82)
                    {       V[n][r]= max82;
                            Apofasi[n]=2;
                    }
                    if (max81==max82)
                    {       V[n][r]= max81;
                            Apofasi[n]=0;
                    }
            }

            else if (r==3 && (n+1)%t1 == 0 && (n+1)%t2 == 0)
            {
                    max91=(p1*(1 + V[n+1][r]))+((1-p1)*(0 + V[n+1][r]));
                    max92=(p2*(1 + V[n+1][r]))+((1-p2)*(0 + V[n+1][r]));
                    if (max91>max92)
                    {       V[n][r]= max91;
                            Apofasi[n]=1;
                    }
                    if (max91<max92)
                    {       V[n][r]= max92;
                            Apofasi[n]=2;
                    }
                    if (max91==max92)
                    {       V[n][r]= max91;
                            Apofasi[n]=0;
                    }
            }
        }
    }
```

```
if (n%t1!=0 && n%t2!=0)
{
        for(r=0;r<sit;r++)
        {
                if(r==0 && (n+1)%t1 != 0 && (n+1)%t2 != 0)
                {       Vk[n][r]= Vk[n+1][r]+ 0;
                }
                else if (r==0 && (n+1)%t1 == 0 && (n+1)%t2 != 0 )
                {       Vk[n][r]= Vk[n+1][r+1] + 0 ;
                }
                else if (r==0 && (n+1)%t1 != 0 && (n+1)%t2 == 0 )
                {       Vk[n][r]= Vk[n+1][r+2] + 0 ;
                }
                else if (r==0 && (n+1)%t1 == 0 && (n+1)%t2 == 0 )
                {       Vk[n][r]= Vk[n+1][r+3] + 0 ;
                }


                else if (r==1 && (n+1)%t1 != 0 && (n+1)%t2 != 0)
                {       Vk[n][r]= (p1*(1 + Vk[n+1][r-1]))+((1-p1)*(0 + Vk[n+1][r]));
                }
                else if (r==1 && (n+1)%t1 == 0 && (n+1)%t2 != 0)
                {       Vk[n][r]= (p1*(1 + Vk[n+1][r]))+((1-p1)*(0 + Vk[n+1][r]));
                }
                else if (r==1 && (n+1)%t1 != 0 && (n+1)%t2 == 0)
                {     Vk[n][r]= (p1*(1 + Vk[n+1][r+1]))+((1-p1)*(0 + Vk[n+1][r+2]));
                }
                else if (r==1 && (n+1)%t1 == 0 && (n+1)%t2 == 0)
                {     Vk[n][r]= (p1*(1 + Vk[n+1][r+2]))+((1-p1)*(0 + Vk[n+1][r+2]));
                }

                else if (r==2 && (n+1)%t1 != 0 && (n+1)%t2 != 0)
                {       Vk[n][r]= (p2*(1 + Vk[n+1][r-2]))+((1-p2)*(0 + Vk[n+1][r]));
                }
                else if (r==2 && (n+1)%t1 == 0 && (n+1)%t2 != 0)
                {      Vk[n][r]= (p2*(1 + Vk[n+1][r-1]))+((1-p2)*(0 + Vk[n+1][r+1]));
                }
                else if (r==2 && (n+1)%t1 != 0 && (n+1)%t2 == 0)
                {       Vk[n][r]= (p2*(1 + Vk[n+1][r]))+((1-p2)*(0 + Vk[n+1][r]));
                }
                else if (r==2 && (n+1)%t1 == 0 && (n+1)%t2 == 0)
                {     Vk[n][r]= (p2*(1 + Vk[n+1][r+1]))+((1-p2)*(0 + Vk[n+1][r+1]));
                }
```

```
else if (r==3 && (n+1)%t1 != 0 && (n+1)%t2 != 0)
{
        maxx61=(p1*(1 + Vk[n+1][r-1]))+((1-p1)*(0 + Vk[n+1][r]));
        maxx62=(p2*(1 + Vk[n+1][r-2]))+((1-p2)*(0 + Vk[n+1][r]));

        for(x=1;x<=ekp;x++)
        {
                if (n<=Pollap1[x])
                {
                        deadline1=Pollap1[x]-n;
                        /* fprintf(f,"\nPeriodos %4d: Deadline tou
                        user 1=%d\n",n,deadline1); */
                        break;
                }
        }

        for(x=1;x<=ekp;x++)
        {
                if (n<=Pollap2[x])
                {
                        deadline2=Pollap2[x]-n;
                        /* fprintf(f,"\nPeriodos %4d: Deadline tou
                        user 2=%d\n",n,deadline2); */
                        break;
                }
        }
        k1=deadline1-deadline2;
        k2=abs(k1);
        /* fprintf(f,"\nk=%d\n",k);                    /*testing k*/
        riza=1-p2;
        bskelos=p2*(pow(riza,k2));

        if (k1<0 && p1>bskelos)
        {   Vk[n][3]=maxx61;
            Apofasi2[n]=1;
        }
        if (k1<0 && p1<=bskelos)
        {   Vk[n][3]=maxx62;
            Apofasi2[n]=2;
        }
        if (k1>0 && p2<=bskelos)
        {   Vk[n][3]=maxx61;
            Apofasi2[n]=1;
        }
        if (k1>0 && p2>bskelos)
        {   Vk[n][3]=maxx62;
            Apofasi2[n]=2;
        }
```

```
            if (k1==0 && p1>p2)
            {   Vk[n][3]=maxx61;
                Apofasi2[n]=1;
            }
            if (k1==0 && p1<p2)
            {   Vk[n][3]=maxx62;
                Apofasi2[n]=2;
            }
}

else if (r==3 && (n+1)%t1 == 0 && (n+1)%t2 != 0)
{
        maxx71=(p1*(1 + Vk[n+1][r]))+((1-p1)*(0 + Vk[n+1][r]));
        maxx72=(p2*(1 + Vk[n+1][r-2]))+((1-p2)*(0 + Vk[n+1][r]));

        for(x=1;x<=ekp;x++)
        {
                if (n<=Pollap1[x])
                {
                        deadline1=Pollap1[x]-n;
                        /* fprintf(f,"\nPeriodos %4d: Deadline tou
                        user 1=%d\n",n,deadline1); */
                        break;
                }
        }
        for(x=1;x<=ekp;x++)
        {
                if (n<=Pollap2[x])
                {
                        deadline2=Pollap2[x]-n;
                        /* fprintf(f,"\nPeriodos %4d: Deadline tou
                        user 2=%d\n",n,deadline2); */
                        break;
                }
        }
        k1=deadline1-deadline2;
        k2=abs(k1);
        /* fprintf(f,"\nk=%d\n",k);                    /*testing k*/
        riza=1-p2;
        bskelos=p2*(pow(riza,k2));

        if (k1<0 && p1>bskelos)
        {   Vk[n][3]=maxx71;
            Apofasi2[n]=1;
        }
        if (k1<0 && p1<=bskelos)
        {   Vk[n][3]=maxx72;
            Apofasi2[n]=2;
```

```
            }
            if (k1>0 && p2<=bskelos)
            {   Vk[n][3]=maxx71;
                Apofasi2[n]=1;
            }
            if (k1>0 && p2>bskelos)
            {   Vk[n][3]=maxx72;
                Apofasi2[n]=2;
            }
            if (k1==0 && p1>p2)
            {   Vk[n][3]=maxx71;
                Apofasi2[n]=1;
            }
            if (k1==0 && p1<p2)
            {   Vk[n][3]=maxx72;
                Apofasi2[n]=2;
            }
}


else if (r==3 && (n+1)%t1 != 0 && (n+1)%t2 == 0)
{
        maxx81=(p1*(1 + Vk[n+1][r-1]))+((1-p1)*(0 + Vk[n+1][r]));
        maxx82=(p2*(1 + Vk[n+1][r]))+((1-p2)*(0 + Vk[n+1][r]));

        for(x=1;x<=ekp;x++)
        {
                if (n<=Pollap1[x])
                {
                        deadline1=Pollap1[x]-n;
                        /* fprintf(f,"\nPeriodos %4d: Deadline tou
                        user 1=%d\n",n,deadline1); */
                        break;
                }
        }

        for(x=1;x<=ekp;x++)
        {
                 if (n<=Pollap2[x])
                {
                        deadline2=Pollap2[x]-n;
                        /* fprintf(f,"\nPeriodos %4d: Deadline tou
                        user 2=%d\n",n,deadline2); */
                        break;
                }
        }
        k1=deadline1-deadline2;
        k2=abs(k1);
```

```
            /* fprintf(f,"\nk=%d\n",k);                    /*testing k*/
            riza=1-p2;
            bskelos=p2*(pow(riza,k2));

            if (k1<0 && p1>bskelos)
            {   Vk[n][3]=maxx81;
                Apofasi2[n]=1;
            }
            if (k1<0 && p1<=bskelos)
            {   Vk[n][3]=maxx82;
                Apofasi2[n]=2;
            }
            if (k1>0 && p2<=bskelos)
            {   Vk[n][3]=maxx81;
                Apofasi2[n]=1;
            }
            if (k1>0 && p2>bskelos)
            {   Vk[n][3]=maxx82;
                Apofasi2[n]=2;
            }
            if (k1==0 && p1>p2)
            {   Vk[n][3]=maxx81;
                Apofasi2[n]=1;
            }
            if (k1==0 && p1<p2)
            {   Vk[n][3]=maxx82;
                Apofasi2[n]=2;
            }
        }


    else if (r==3 && (n+1)%t1 == 0 && (n+1)%t2 == 0)
    {
            maxx91=(p1*(1 + Vk[n+1][r]))+((1-p1)*(0 + Vk[n+1][r]));
            maxx92=(p2*(1 + Vk[n+1][r]))+((1-p2)*(0 + Vk[n+1][r]));



            for(x=1;x<=ekp;x++)
            {
                    if (n<=Pollap1[x])
                    {
                            deadline1=Pollap1[x]-n;
                            /* fprintf(f,"\nPeriodos %4d: Deadline tou
                            user 1=%d\n",n,deadline1); */
                            break;
                    }
            }
```

```
for(x=1;x<=ekp;x++)
{
        if (n<=Pollap2[x])
        {
                deadline2=Pollap2[x]-n;
                /* fprintf(f,"\nPeriodos %4d: Deadline tou
                user 2=%d\n",n,deadline2); */
                break;
        }
}

k1=deadline1-deadline2;
k2=abs(k1);
/* fprintf(f,"\nk=%d\n",k);                    /*testing k*/
riza=1-p2;
bskelos=p2*(pow(riza,k2));


if (k1<0 && p1>bskelos)
{   Vk[n][3]=maxx91;
    Apofasi2[n]=1;
}
if (k1<0 && p1<=bskelos)
{   Vk[n][3]=maxx92;
    Apofasi2[n]=2;
}
if (k1>0 && p2<=bskelos)
{   Vk[n][3]=maxx91;
    Apofasi2[n]=1;
}
if (k1>0 && p2>bskelos)
{   Vk[n][3]=maxx92;
    Apofasi2[n]=2;
}
if (k1==0 && p1>p2)
{   Vk[n][3]=maxx91;
    Apofasi2[n]=1;
}
if (k1==0 && p1<p2)
{   Vk[n][3]=maxx92;
    Apofasi2[n]=2;
}
        }
    }
}
```

```
fprintf(f,"\n\n");
for(n=ekp;n>=0;n--)                          /* array test print */
{       fprintf(f,"n=%d",n);
        for(r=0;r<sit;r++)
        {
                fprintf(f,"%9.3f",V[n][r]);
                if(r==3)
                {       fprintf(f,"\n");
                }
        }
}
fprintf(f,"\n\n");

for(n=ekp;n>=0;n--)                          /* array test print */
{       fprintf(f,"n=%d",n);
        for(r=0;r<sit;r++)
        {
                fprintf(f,"%9.3f",Vk[n][r]);
                if(r==3)
                {       fprintf(f,"\n");
                }
        }
}

 fprintf(f,"\n\n");
 for(n=ekp;n>=0;n--)                          /* array test print */
 {
        fprintf(f,"Apofasi 1 kai 2 stin periodo %d : %d     %d\n",n,Apofasi[n],Apofasi2[n]);
 }

 fprintf(f,"\n\nPollaplasia periodou tou user 1:\n");

 for(n=1;n<=(ekp/t1);n++)                     /* array test print */
 {
        fprintf(f,"%d\n",Pollap1[n]);
 }
 fprintf(f,"\n\nPollaplasia periodou tou user 2:\n");

 for(n=1;n<=(ekp/t2);n++)                     /* array test print */
 {
         fprintf(f,"%d\n",Pollap2[n]);
 }

 fprintf(f,"\n\n");
```

```
 for(n=0;n<=ekp;n++)
  {
/*Elegxos an h beltisth politikh den allazei sta shmeia pou den einai pollaplasia ths mias h ths allhs periodou.*/
        if (n%t1!=0 && n%t2!=0 && Apofasi[n]==Apofasi[n-1])
        {
fprintf(f,"Periodos %4d: H beltisth politikh den allakse ayth tin periodo pou den einai pollaplasio ths mias h ths allhs periodou\n",n);
        }

        if (n%t1!=0 && n%t2!=0 && Apofasi[n]==0)
        {
fprintf(f,"Periodos %4d: H beltisth politikh den allakse ayth tin periodo pou den einai pollaplasio ths mias h ths allhs periodou\n",n);
        }

        if (n%t1!=0 && n%t2!=0 && Apofasi[n-1]==0)
        {
fprintf(f,"Periodos %4d: H beltisth politikh den allakse ayth tin periodo pou den einai pollaplasio ths mias h ths allhs periodou\n",n);
        }

if (n%t1!=0 && n%t2!=0 && Apofasi[n]!=Apofasi[n-1] && Apofasi [n]!=0 && Apofasi [n-1]!=0)
        {
fprintf(f,"\nPeriodos %d: H beltisth politikh allakse tin periodo %d enw ayti den einai pollaplasio ths mias h ths allhs periodou\n",n,n);
                test1=1;
        }
 }
  if (test1!=1)
   {    fprintf(f,"\nH beltisth politikh den allakse stis periodous pou den einai pollaplasia ths mias h ths allhs periodou\n");
   }

for(n=0;n< ekp;n++)
{
/*An o user me th megaluterh pi0anothta epituxias exei kai mikrotero deadline, exei proteraiothta.*/
for(x=1;x<=ekp;x++)
        {
                if (n<Pollap1[x])
                {
                        dl1=Pollap1[x]-n;
                        if (dl1==0)
                        {
                                dl1=t1;
                        }
```

```
/* fprintf(f,"\nPeriodos %4d: Deadline tou user 1=%d\n",n,dl1); */
                        break;
                }
        }
        for(x=1;x<=ekp;x++)
        {
                if (n<Pollap2[x])
                {
                        dl2=Pollap2[x]-n;
                        if (dl2==0)
                        {
                        dl2=t2;
                        }
                         /* fprintf(f,"\nPeriodos %4d: Deadline tou user 2=%d\n",n,dl2); */
                        break;
                }
}
        if (p1>p2 && dl1<=dl2 && Apofasi[n]==1)
        {       fprintf(f,"\nPeriodos %4d: Epeidi o user 1 (megalyteri pi8anothta epityxias)
exei mikrotero i iso deadline exei proteraiothta.\n",n);
        }

        if (p1>p2 && dl1<=dl2 && Apofasi[n]==0)
       {        fprintf(f,"\nPeriodos %4d: Epeidi o user 1 (megalyteri pi8anothta epityxias)
exei mikrotero i iso deadline exei proteraiothta.\n",n);
        }

        if (p1>p2 && dl1<=dl2 && Apofasi[n]==2)
        {       fprintf(f,"\nPeriodos %d: Enw o user 1 (megalyteri pi8anothta epityxias) exei
mikrotero i iso deadline DEN exei proteraiothta.\n",n);
                test2=1;
        }
        if (p2>p1 && dl2<=dl1 && Apofasi[n]==2)
        {       fprintf(f,"\nPeriodos %4d: Epeidi o user 2 (megalyteri pi8anothta epityxias) xei
mikrotero i iso deadline exei proteraiothta.\n\n",n);
        }

        if (p2>p1 && dl2<=dl1 && Apofasi[n]==0)
        {       fprintf(f,"\nPeriodos %4d: Epeidi o user 2 (megalyteri pi8anothta epityxias)
exei mikrotero i iso deadline exei proteraiothta.\n\n",n);
        }

        if (p2>p1 && dl2<=dl1 && Apofasi[n]==1)
        {       fprintf(f,"\nPeriodos %d: Enw o user 2 (megalyteri pi8anothta epityxias) exei
mikrotero i iso deadline DEN exei proteraiothta.\n",n);
                test3=1;
        }
  }
```

```
/* fprintf(f,"\ntest2=%d\n",test2);
   fprintf(f,"\ntest3=%d\n",test3); */

 if (test2!=1 && p1>p2)
  {     fprintf(f,"\nOtan o user 1 (megalyteri pi8anothta epityxias) eixe mikrotero i iso
deadline eixe kai proteraiothta.\n\n");
  }

 if (test3!=1 && p2>p1)
  {     fprintf(f,"\nOtan o user 2 (megalyteri pi8anothta epityxias) eixe mikrotero i iso
deadline eixe kai proteraiothta.\n\n");
  }

 if (p1==p2)
  {     fprintf(f,"\nOi pithanotites epityxias kai gia tous 2 users einai ises\n\n");
  }

 fprintf(f,"\nTo  V(1,1) ti xroniki stigmi 0 einai: %1.3f\n",V[0][3]);
 fprintf(f,"\nTo Vk(1,1) ti xroniki stigmi 0 einai: %1.3f\n\n",Vk[0][3]);
/*}*/

fclose(f);

system("pause");
}

int lcm(int a,int b)
{
        int n;
for(n=1;;n++)
        {
                if(n%a == 0 && n%b == 0)
                return n;
        }
 }
```

Βιβλιογραφία

[1] Abdelzaher T., K. Arzen, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, A. Cervin, J. Lehoczky, A.K. Mok and L. Sha, (2004) "Real Time Scheduling Theory: A Historical Perspective," *Real Time Systems*, 28, 101-155.

[2] Audsley N. and A. Burns, (1990) "Real-Time System Scheduling," Department of Computer Science, University of York, UK.

[3] Bambos N. and A. Dua, (2007) "Downlink Wireless Packet Scheduling with Deadlines," *IEEE Tranactions on Mobile Computing*, 6 (12).

[4] Baruah S.K., (1995) "Fairness in Periodic Real-Time Scheduling," *Proceedings of Real Time Systems Symposium*, Pisa, Italy, 200-209.

[5] Budin L., M. Golub and D. Jakobovic, (2000) "Genetic Algorithms in Real-Time Imprecise Computing," *Journal of Computing and Information Technology*, 8.

[6] Burchard A., J. Liebeherr, Y. Oh and S.H. Son, (1995) "New Strategies for Assigning Real-Time Tasks to Multiprocessor Systems," *IEEE Transactions on Computers*, 44 (12), 1429-1442.

[7] Buttazzo G.C., (2003) "Rate monotonic vs. EDF: Judgement Day," *Lecture Notes in Computer Science*, 2855, 67-83.

[8] Buttazzo G.C. and M. Spuri, (1994) "Efficient Aperiodic Service under Earliest Deadline Scheduling," *Proceedings of the 74th IEEE Real-Time Systems Symposium (RTSS'94)*, San Juan Puerto Rico, 2-11.

[9] Cottet F., J. Delacroix, C. Kaiser and Z. Mammeri, (2002) "Scheduling in real time systems," *John Wiley & Sons*.

[10] Chung J.Y., W.S. Liu and W.K. Shih, (1991) "Algorithms for Scheduling Imprecise Computations to Minimize Total Error," *SIAM Journal on Computing*, 20 (3), 537-552.

[11] Davari S. and L. Sha, (1992) "Sources of Unbounded Priority Inversion in Real-Time Systems and a Comparative Study of Possible Solutions," *Operating Systems Review*, 26 (2), 110-120.

[12] Ding V., J.P. Lehoczky and L. Sha, (1987) "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average case Behavior," *Tech Report, Department of Statistics, Canergie-Mellon.*

[13] Goodenough J.B. and L. Sha, (1989) "A Review of Analytic Real-Time Scheduling Theory and its Application to Ada," *Ada: The Design Choice*, ed. A. Alvarez, Cambridge University Press, Madrid, 137-148.

[14] Ho K.I.J, J.Y.T. Leung and W.D. Wei, (1997) "Scheduling Imprecise Computation Tasks with 0/1 Constraints," *Discrete Applied Mathematics*, 78, 117-132.

[15] Karatza H.D. and G. Stavrinidis, (2008) "Performance Evaluation of Gang Scheduling in Distributed Real-Time Systems with Possible Software Faults," *Proceedings of the 2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems-Spects 2008*, Edinburgh, Scotland, UK, 1-7.

[16] Katcher D.I., S.S. Sathaye and J.K. Strosnider, (1995) "Fixed Priority Scheduling with Limited Priority Levels," *IEEE Transactions on Computers*, 44 (9), 1140-1144.

[17] Krauskopf T., K.J. Lin, W.S. Liu and S. Natarajan, (1987) "Concord: A system of imprecise computations," *Proceedings of the 1987 IEEE COMPSAC*, Japan.

[18] Layland J.W. and C.L. Liu, (1973) "Scheduling Algorithms For Multiprogramming In A Hard Real-Time Environment," *The Association of Computing Machinery*, 20 (1), 46-61.

[19] Lehoczky J.P. and L. Sha, (1986) "Performance of Real-Time Bus Scheduling Algorithms," *ACM Performance Evaluation Review*, 14 (1).

[20] Lin K.J., J.W.S. Liu and S. Natarajan, (1987) "Imprecise Results: Utilizing Partial Computations in Real-Time Systems," *Proceedings of the 8th Real-Time Systems Symposium*, 210-217.

[21] Lin K.J., J.W.S. Liu and S. Natarajan, (1987) "Scheduling Real-Time, Periodic Jobs Using Imprecise Results," *Proceedings of the 8th Real-Time Systems Symposium*, 252-260.

[22] Lortz V.B. and K.G. Shin, (1995) "Semaphore Queue Priority Assignment for Real-Time Multiprocessor Synchronization," *IEEE Transactions on Software Engineering*, 21 (10), 834-843.

[23] Miller F.W., (1992) "The Performance of a Mixed Priority Real-Time Scheduling Algorithm," *Operating Systems Review*, 26 (4), 5-13.

[24] Parnas D.L. and J. Xu, (1993) "On Satisfying Timing Constraints in Hard-Real-Time Systems," *IEEE Transactions on Software Engineering*, 19 (1), 70-84.

[25] Rajkumar R., S.S. Sathaye and L. Sha, (1994) "Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems," *Proceedings of the IEEE*, 82 (1), 68-82.

[26] Αναστασόπουλος Ι., (2009) "Κατανεμημένα Συστήματα Πραγματικού Χρόνου", Τμήμα Πληροφορικής, Α.Π.Θ.