Ατομική Διπλωματική Εργασία

ΑΛΓΟΡΙΘΜΟΣ ΑΝΑΓΝΩΡΙΣΗΣ ΣΦΑΛΜΑΤΩΝ ΓΙΑ ΤΟΠΟΛΟΓΙΑ ΜΑΝΧΑΤΑΝ

Γιώργος Λογγίνος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αλγόριθμος Αναγνώρισης Σφαλμάτων για Τοπολογία Μανχάταν

Γιώργος Λογγίνος

Επιβλέπουσα Καθηγήτρια
Άννα Φιλίππου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2019

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω ιδιαιτέρα την επιβλέπουσα καθηγήτρια μου, Δρ. Άννα Φιλίππου, καθώς μου εμπιστεύτηκε την ανάθεση της συγκεκριμένης εργασίας, με βοήθησε σε μεγάλο βαθμό να κατανοήσω το πρόβλημα που κλήθηκα να επιλύσω και ήταν πάντοτε πρόθυμη να βοηθήσει στην επίλυση των αποριών μου. Η καθοδήγηση της ήταν καταλυτικής σημασίας, καθώς με βοήθησε να καταλάβω τα βασικά σημεία στα οποία έπρεπε να επικεντρωθεί η έρευνα μου έτσι ώστε να είναι δυνατό να συμβάλει ουσιαστικά και στο ερευνητικό πρόγραμμα Visorsurf.

Ευχαριστώ επίσης το Δρ. Ανδρέα Πιτσιλλίδη για τις χρήσιμες συμβουλές και την πολύτιμη ανατροφοδότηση που μου έδωσε σε πολλές περιπτώσεις, η οποία συνέδραμε σε σημαντικό βαθμό στην εξέλιξη της έρευνας μου.

Επιπλέον, θα ήθελα να ευχαριστήσω το Δρ. Δημήτρη Κουζαπά ο οποίος ήταν πάντα πρόθυμος να μου λύσει οποιεσδήποτε απορίες είχα και να με συμβουλέψει σχετικά με θέματα που αφορούν την ΑΔΕ μου.

Ακόμη, νοιώθω την ανάγκη να ευχαριστήσω το μεταπτυχιακό φοιτητή κ. Κωνσταντίνο Σκίτσα καθώς ήταν ιδιαίτερα πρόθυμος να με βοηθήσει να καταλάβω τον τρόπο λειτουργίας του λογισμικού Anylogic, αλλά και να με βοηθήσει με όποιες δυσκολίες αντιμετώπισα κατά τη δημιουργία του προσομοιωτή μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου και συγκεκριμένα τους γονείς μου Νίκο και Μαρία και τα αδέλφια μου Κυριάκο και Αγγελίνα, καθώς μέσω της ηθικής τους υποστήριξης με βοήθησαν σε μεγάλο βαθμό να ολοκληρώσω την παρούσα μελέτη.

Περίληψη

Η παρουσία σφαλμάτων είναι αναπόφευκτη σε όλα τα κατανεμημένα συστήματα. Για αυτό το λόγο, πολλοί αλγόριθμοι έχουν προταθεί για την αναγνώριση και την αποφυγή τους. Εντούτοις, ο κάθε αλγόριθμος έχει διαφορετικές απαιτήσεις πλεονασμού και μπορεί να εφαρμοστεί μόνο υπό συγκεκριμένες προϋποθέσεις. Οι ιδιάζοντες περιορισμοί που προέκυψαν στα πλαίσια του ερευνητικού προγράμματος Visorsurf, έθεσαν ως επιτακτική την ανάγκη δημιουργίας ενός καινούργιου αλγόριθμου ο οποίος να μπορεί να ανταποκρίνεται στις ανάγκες του έργου.

Για αυτό το λόγο σχεδίασα ένα αλγόριθμο αναγνώρισης σφαλμάτων για τοπολογία Manhattan, ο οποίος παρουσιάζεται στην αναφορά αυτή. Ο συγκεκριμένος αλγόριθμος βασίζεται στην τεχνική ανάστροφης μετάδοσης σφάλματος και μπορεί να εντοπίσει όλα τα μόνιμα σφάλματα που παρουσιάζονται στους ελεγκτές δικτύου ενός πλέγματος. Ο αλγόριθμος, δέχεται τρεις παραμέτρους εισόδου, τη μέθοδο επιλογής NUC, τον αλγόριθμο δημιουργίας μονοπατιού και τον αλγόριθμο δρομολόγησης. Η δυνατότητα χρήσης του αλγορίθμου σε πλέγματα που υποστηρίζουν διαφορετικούς αλγόριθμους δρομολόγησης υποδεικνύει την προσαρμοστικότητα που τον διακρίνει, όπως και την προοπτική χρήσης του σε μελλοντικά έργα.

Μέσω της πειραματικής αξιολόγησης του αλγόριθμου η οποία πραγματοποιήθηκε με τη χρήση του λογισμικού Anylogic, διαπιστώθηκε η υψηλή συσχέτιση που υπάρχει μεταξύ των παραμέτρων εισόδου του αλγορίθμου, αλλά και το γεγονός ότι το κόστος που προκύπτει από τη χρησιμοποίηση του αλγορίθμου είναι πάντοτε σχετικό με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα. Έτσι, έγινε υπολογισμός των κατάλληλων τιμών εισόδου του αλγορίθμου, με τις οποίες επιτυγχάνεται ελαχιστοποίηση του κόστους χρήσης του, αναλόγως του αριθμού των σφαλμάτων που αναμένεται να εντοπιστούν στο πλέγμα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Το πρόβλημα και το κίνητρο

Οι μεταεπιφάνειες (metasurfaces) είναι λεπτές και επίπεδες τεχνητές δομές, οι οποίες με την ανακάλυψη τους έχουν οδηγήσει στη δημιουργία καινοτόμων ηλεκτρομαγνητικών και οπτικών εξαρτημάτων, των οποίων οι λειτουργίες μπορούν να χαρακτηριστούν μέχρι και αφύσικες. Παραδείγματα τέτοιων λειτουργιών είναι η δημιουργία ηλεκτρομαγνητικά αοράτων σωμάτων, η ολική απορρόφηση της ακτινοβολίας αλλά και ο χειρισμός της κατεύθυνσης των ηλεκτρομαγνητικών εκπομπών.

Εντούτοις, η δημιουργία και η χρησιμοποίηση των μεταεπιφανειών απαιτεί μεγάλο επίπεδο εξειδίκευσης. Αυτό, περιορίζει σε μεγάλο βαθμό την αξιοποίηση τους από τον ευρύτερο τομέα της μηχανικής. Για την αντιμετώπιση του συγκεκριμένου προβλήματος, το ερευνητικό πρόγραμμα VISORSURF στοχεύει στη δημιουργία μιας νέας πλατφόρμας υλικού και λογισμικού, της υπερεπιφάνειας (HyperSurface), η οποίας η ηλεκτρομαγνητική συμπεριφορά θα μπορεί να καθορίζεται προγραμματιστικά [11].

Αυτό θα επιτυγχάνεται μέσω της παρουσίας ενός δικτύου ελεγκτών (controllers) στη υπερεπιφάνεια, οι οποίοι θα είναι υπεύθυνοι για τη ρύθμιση της ηλεκτρομαγνητικής της συμπεριφοράς. Οι ελεγκτές θα είναι συνδεδεμένοι σε τοπολογία δικτύου Μανχάταν, έτσι ώστε να είναι δυνατή η μεταξύ τους επικοινωνία, η οποία επιτυγχάνεται με την ανταλλαγή πακέτων.

Σε περίπτωση που δε θα παρατηρείται κάποιο πρόβλημα σχετικό με τη λειτουργικότητα των ελεγκτών, η δρομολόγηση των πακέτων στο δίκτυο θα μπορεί να επιτυγχάνεται εύκολα με τη χρήση κάποιου απλού αλγόριθμου δρομολόγησης, όπως ο XY [10]. Εντούτοις, η παρουσία σφαλμάτων σε κάποιους από τους ελεγκτές θα είναι αναπόφευκτη και αν δε διασφαλιστεί η αποτελεσματική αναγνώριση και αποφυγή

τους, τα σφάλματα αυτά δε θα επηρεάζουν αποκλειστικά τη λειτουργικότητα ενός ελεγκτή, αλλά ολόκληρου του δικτύου και κατ' επέκταση της υπερεπιφάνειας.

Σκοπός της παρούσας εργασίας είναι η σχεδίαση ενός αλγόριθμου αναγνώρισης σφαλμάτων, ο οποίος θα μπορεί να εντοπίσει τους ελεγκτές της υπερεπιφάνειας οι οποίοι δε θα είναι λειτουργικοί λόγω της ύπαρξης κάποιου σφάλματος. Στόχος είναι ο συγκεκριμένος αλγόριθμος να αποτελέσει τη βάση για την επίτευξη επιτυχούς δρομολόγησης πακέτων στην υπερεπιφάνεια, ακόμη και υπό την παρουσία σφαλμάτων.

1.2 Η μεθοδολογία που ακολουθήθηκε

Το πρώτο βήμα που διενήργησα πριν ξεκινήσω τη σχεδίαση του αλγορίθμου, ήταν η μελέτη της σχετικής βιβλιογραφίας αλλά και των περιορισμών που υπάρχουν στο δίκτυο ελεγκτών της υπερεπιφάνειας. Μέσα από τη μελέτη μου, κατάλαβα ότι λόγω των προβλημάτων που επιφέρει στα συστήματα η παρουσία σφαλμάτων, έχουν αναπτυχθεί αρκετοί αλγόριθμοι για αναγνώριση και αποφυγή των σφαλμάτων σε NoCs (Networks on Chip), αλλά και σε πλέγματα. Εντούτοις, ο καθένας από τους υπάρχοντες αλγόριθμους έχει διαφορετικά χαρακτηριστικά και μπορεί να εφαρμοστεί μόνο σε συγκεκριμένα συστήματα λόγω των απαιτήσεων και των προϋποθέσεων που πρέπει να πληρούνται για την υλοποίηση του. Συμπέρανα λοιπόν ότι λόγω της ιδιαίτερης αρχιτεκτονικής της υπερεπιφάνειας και των περιορισμών που χαρακτηρίζουν το δίκτυο που την ελέγχει, κανένας από τους υπάρχοντες αλγόριθμους αναγνώρισης και αποφυγής σφαλμάτων δε θα μπορούσε να χρησιμοποιηθεί στο δικό μας σύστημα.

Έτσι, αποφάσισα να σχεδιάσω ένα νέο αλγόριθμο αναγνώρισης σφαλμάτων, ο οποίος να μπορεί να χρησιμοποιηθεί στο δίκτυο της υπερεπιφάνειας. Ο αλγόριθμος που σχεδίασα βασίζεται στην τεχνική ανάστροφης μετάδοσης σφάλματος (back-propagation), δηλαδή ως στόχο έχει τη μετάδοση των σφαλμάτων δικτύου προς τα πίσω, μέχρι το σφάλμα να καταλήξει στον αποστολέα ενός πακέτου. Στην περίπτωση του αλγορίθμου μου ο αποστολέας όλων των πακέτων στο δίκτυο είναι η πηγή (gateway), η οποία είναι ένας υπολογιστής τοποθετημένος εκτός του δικτύου, ο οποίος είναι υπεύθυνος για την αναγνώριση των σφαλμάτων.

Μετά το σχεδιασμό του αλγορίθμου μου, προχώρησα αποδεικνύοντας θεωρητικά την ορθότητα του αλλά και το γεγονός ότι τερματίζει πάντα. Έπειτα, προχώρησα στη μοντελοποίηση του χρησιμοποιώντας το εργαλείο Anylogic. Ακολούθως, χρησιμοποίησα το μοντέλο που έφτιαξα για να πραγματοποιήσω προσομοιώσεις, έτσι ώστε να ελέγξω πειραματικά τη λειτουργικότητα του αλγορίθμου, αλλά και για να μετρήσω το κόστος που προκύπτει από τη χρησιμοποίηση του.

1.3 Οργάνωση Κειμένου

Στο επόμενο κεφάλαιο της παρούσας αναφοράς γίνεται εκτενής αναφορά στο δίκτυο ελεγκτών που θα χρησιμοποιηθεί στα πλαίσια του προγράμματος VISORSURF, αλλά και στους περιορισμούς που χαρακτηρίζουν την υπερεπιφάνεια. Επίσης, γίνεται περιγραφή προϋπαρχουσών ιδεών σχετικά με την αναγνώριση και αποφυγή σφαλμάτων και επεξηγείται ο λόγος που ήταν αδύνατο να χρησιμοποιηθούν κάποιες συγκεκριμένες υλοποιήσεις στο δικό μας έργο.

Στο Κεφάλαιο 3 περιλαμβάνεται μια σύντομη περιγραφή του αλγορίθμου, όπως και ο ψευδοκώδικας του αλγορίθμου αλλά και το διάγραμμα καταστάσεων UML που προκύπτει από αυτόν. Ακόμη, γίνεται μια επεξήγηση των τριών παραμέτρων εισόδου του αλγορίθμου που είναι οι μέθοδοι επιλογής NUC, ο αλγόριθμος δημιουργίας μονοπατιού και ο αλγόριθμος δρομολόγησης. Επίσης, το τρίτο κεφάλαιο περιλαμβάνει και την απόδειξη ορθότητας και τερματισμού του αλγορίθμου μου, αλλά και τις περιπτώσεις κατά τις οποίες πρέπει να γίνεται η χρησιμοποίηση του.

Στο Κεφάλαιο 4, παρουσιάζεται ο προσομοιωτής που έχω δημιουργήσει έτσι ώστε να μπορέσω να μοντελοποιήσω και να εξετάσω τον αλγόριθμο μου, χρησιμοποιώντας το λογισμικό Anylogic. Γίνεται μια σύντομη επεξήγηση του τρόπου λειτουργίας του προσομοιωτή και των βασικών κλάσεων που έχουν χρησιμοποιηθεί. Ακόμη, αναφέρεται ο τρόπος με τον οποίο καθορίζονται οι διάφορες παραμέτροι προσομοίωσης από το χρήστη και επεξηγείται λεπτομερώς ο τρόπος παρουσίασης των αποτελεσμάτων της κάθε προσομοίωσης.

Στο Κεφάλαιο 5 πραγματοποιείται μια επεξήγηση των πειραμάτων που διενεργήθηκαν για να εξεταστεί η αποδοτικότητα του αλγορίθμου μου και μια εκτενής ανάλυση των αποτελεσμάτων που προέκυψαν από αυτά. Στόχος της συγκεκριμένης ανάλυσης είναι

η εύρεση των ιδανικών παραμέτρων εισόδου για να επιτυγχάνεται η βέλτιστη δυνατή επίδοση του αλγορίθμου σε κάθε περίπτωση.

Τέλος, στο Κεφάλαιο 6 γίνεται μια αξιολόγηση του αλγορίθμου βάσει των αποτελεσμάτων που έχουν προκύψει ενώ παράλληλα αναφέρονται διάφοροι περιορισμοί που προκύπτουν από τον ίδιο τον αλγόριθμο αλλά και κάποιοι περιορισμοί σχετικοί με τα πειράματα που διενεργήθηκαν για την αξιολόγηση του. Ακόμη, δίνονται και διάφορες προτροπές και κατευθύνσεις για μελλοντική εργασία.

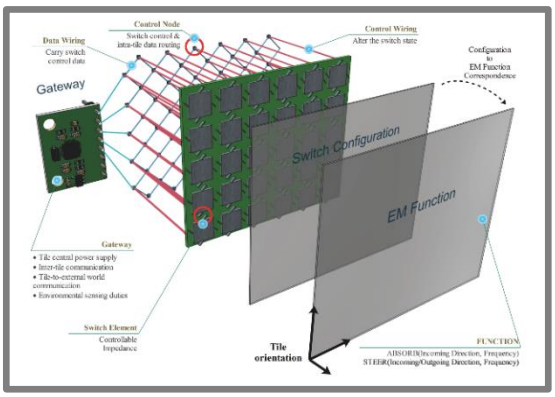
Κεφάλαιο 2

Ανασκόπηση Βιβλιογραφίας

2.1 Το project Visorsurf και οι περιορισμοί της υπερεπιφάνειας

Ο αλγόριθμος αναγνώρισης σφαλμάτων που παρουσιάζεται, έχει σχεδιαστεί έτσι ώστε να πληροί συγκεκριμένες προϋποθέσεις και περιορισμούς για να μπορεί να χρησιμοποιηθεί στα πλαίσια του ερευνητικού προγράμματος Visorsurf. Όπως αναφέρθηκε και στο Κεφάλαιο 1.1, στόχος του Visorsurf είναι η δημιουργία μιας υπερεπιφάνειας (HyperSurface), δηλαδή μιας μεταεπιφάνειας (metasurface) της οποίας η ηλεκτρομαγνητική συμπεριφορά να μπορεί να μεταβάλλεται προγραμματιστικά, με τη χρήση ενός δικτύου ελεγκτών. Ο κάθε ελεγκτής δικτύου (network controller) έχει εξαιρετικά περιορισμένη υπολογιστική ισχύ και μνήμη. Βασικές λειτουργίες των ελεγκτών είναι η ανάλυση των πακέτων που λαμβάνουν και η σωστή μεταβολή της ηλεκτρομαγνητικής συμπεριφοράς της υπερεπιφάνειας, βάσει της πληροφορίας που εμπεριέχεται στα πακέτα αυτά. Επίσης, οι ελεγκτές είναι υπεύθυνοι για τη σωστή δρομολόγηση των συγκεκριμένων πακέτων μέσα στο δίκτυο. Ο κάθε ελεγκτής διαθέτει συνολικά τέσσερις θύρες, δύο θύρες εισόδου και δύο θύρες εξόδου. Όσον αφορά τα σημεία στα οποία βρίσκονται οι τέσσερεις θύρες πάνω στον ελεγκτή, υπάρχει μια θύρα πάνω από τον ελεγκτή, μια θύρα κάτω από τον ελεγκτή, μια θύρα δεξιά από τον ελεγκτή και μια θύρα αριστερά από τον ελεγκτή. Τόσο οι θύρες εισόδου όσο και οι θύρες εξόδου είναι τοποθετημένες σε διαδοχικές θέσεις πάνω στον ελεγκτή.

Σχήμα 2.1: Το λογότυπο του ερευνητικού προγράμματος Visorsurf



Σχήμα 2.2: Η αρχιτεκτονική μιας υπερεπιφάνειας

Επιπλέον, ο κάθε ελεγκτής περιέχει συνολικά πέντε buffers. Ο ένας εξ αυτών είναι ο κεντρικός buffer επεξεργασίας, βρίσκεται στο κέντρο του ελεγκτή και είναι σε αυτόν που αποθηκεύεται το πακέτο το οποίο επεξεργάζεται ο ελεγκτής τη συγκεκριμένη χρονική στιγμή. Οι υπόλοιποι τέσσερεις buffer ουσιαστικά συνδέουν τον κεντρικό buffer επεξεργασίας με τις θύρες εισόδου και εξόδου του ελεγκτή, όπως φαίνεται στα Σχήματα 2.3, 2.4, 2.5 και 2.6. Στους buffer που είναι τοποθετημένοι μετά τις θύρες εισόδου αποθηκεύονται τα πακέτα πριν να υποστούν επεξεργασία από τον ελεγκτή και στους buffers που είναι τοποθετημένοι πριν τις θύρες εξόδου αποθηκεύονται τα πακέτα αφού υποστούν επεξεργασία από το ελεγκτή.

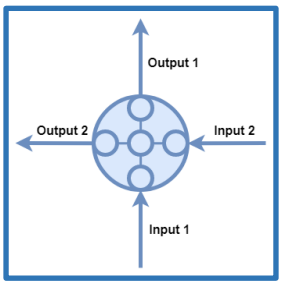
Τα πακέτα είναι δομές δεδομένων οι οποίες περιέχουν πληροφορίες χρήσιμες τόσο για τη δρομολόγηση τους στο δίκτυο, όσο και για τη μεταβολή της κατάστασης των ελεγκτών και της ηλεκτρομαγνητικής συμπεριφοράς της υπερεπιφάνειας. Τα πακέτα, αποθηκεύονται προσωρινά στους buffers των ελεγκτών κατά τη δρομολόγηση τους στο δίκτυο. Όλα τα πακέτα έχουν ως στόχο την επιτυχή δρομολόγηση προς ένα ελεγκτή που ονομάζεται προορισμός, και τη μετάδοση κάποιας πληροφορίας στο συγκεκριμένο ελεγκτή. Για να πραγματοποιηθεί με σωστό τρόπο η δρομολόγηση των πακέτων, χρησιμοποιούνται οι διάφοροι αλγόριθμοι δρομολόγησης, οι οποίοι είναι αποθηκευμένοι στους ελεγκτές.

Ένα συγκεκριμένο είδος πακέτων, τα πακέτα οδοφράγματος (roadblock packets), είναι πακέτα τα οποία χρησιμοποιούνται για την αποτροπή χρησιμοποίησης μιας θύρας εξόδου ενός ελεγκτή. Ουσιαστικά, τα πακέτα αυτά ενημερώνουν τον ελεγκτή που αποτελεί προορισμό τους ότι στο εξής δε θα μπορεί να χρησιμοποιεί μια συγκεκριμένη θύρα εξόδου του, μετατρέποντας τον σε ελεγκτή οδοφράγματος (roadblock controller). Με αυτό τον τρόπο ο συγκεκριμένος ελεγκτής αναγκάζεται να δρομολογήσει μέσω της εναπομένουσας θύρας εξόδου του όλα τα πακέτα που θα λάβει μελλοντικά.

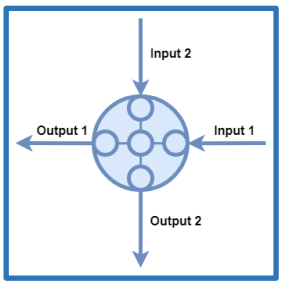
Η επικοινωνία μεταξύ των ελεγκτών είναι ασύγχρονη. Ο λόγος είναι ότι οι ελεγκτές δεν είναι δυνατό να υποστηρίξουν τη λειτουργία ρολογιού, καθώς κάτι τέτοιο θα προκαλούσε παρεμβολές, επηρεάζοντας την ηλεκτρομαγνητική λειτουργία της υπερεπιφάνειας. Συνεπώς, η διασφάλιση της λειτουργικής επικοινωνίας μεταξύ των

ελεγκτών, χωρίς την παρουσία αδιεξόδων (deadlocks) και ζωντανών αδιεξόδων (livelocks) αποτελεί μια μεγάλη πρόκληση.

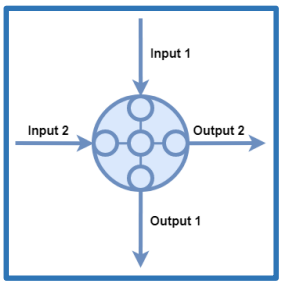
Οι ελεγκτές είναι συνδεδεμένοι μεταξύ τους με τρόπο πάνω στη υπερεπιφάνεια, έτσι ώστε να σχηματίζουν ένα πλέγμα τοπολογίας Manhattan. Ο τρόπος με τον οποίο επιτυγχάνεται αυτό είναι μέσω των τεσσάρων τύπων περιστροφής με τις οποίες μπορεί να τοποθετηθεί κάποιος ελεγκτής στο πλέγμα. Περιστρέφοντας ένα ελεγκτή κατά 90^{o}, μπορούμε να παρατηρήσουμε ότι οι θέσεις των θυρών εισόδου και εξόδου του συγκεκριμένου ελεγκτή διαφοροποιούνται κάθε φορά, ενώ αν πραγματοποιήσουμε τη συγκεκριμένη περιστροφή για τέσσερεις διαδοχικές φορές, οι θύρες εισόδου και εξόδου θα επανέλθουν στην αρχική τους θέση. Έτσι, κάθε φορά που περιστρέφεται ένας ελεγκτής κατά 90^{o}, ουσιαστικά προκύπτει ένας διαφορετικός τύπος περιστροφής. Αξιοποιώντας τα τέσσερα είδη περιστροφών που προκύπτουν, μπορούμε να συνδέσουμε τους ελεγκτές μεταξύ τους με τέτοιο τρόπο έτσι ώστε να σχηματίσουμε την τοπολογία Manhattan.



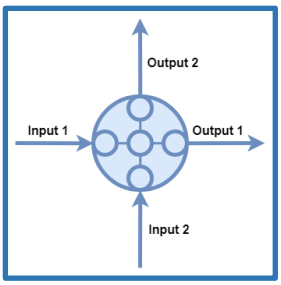
Σχήμα 2.3: Ελεγκτής τύπου περιστροφής 0



Σχήμα 2.4: Ελεγκτής τύπου περιστροφής 1



Σχήμα 2.5: Ελεγκτής τύπου περιστροφής 2



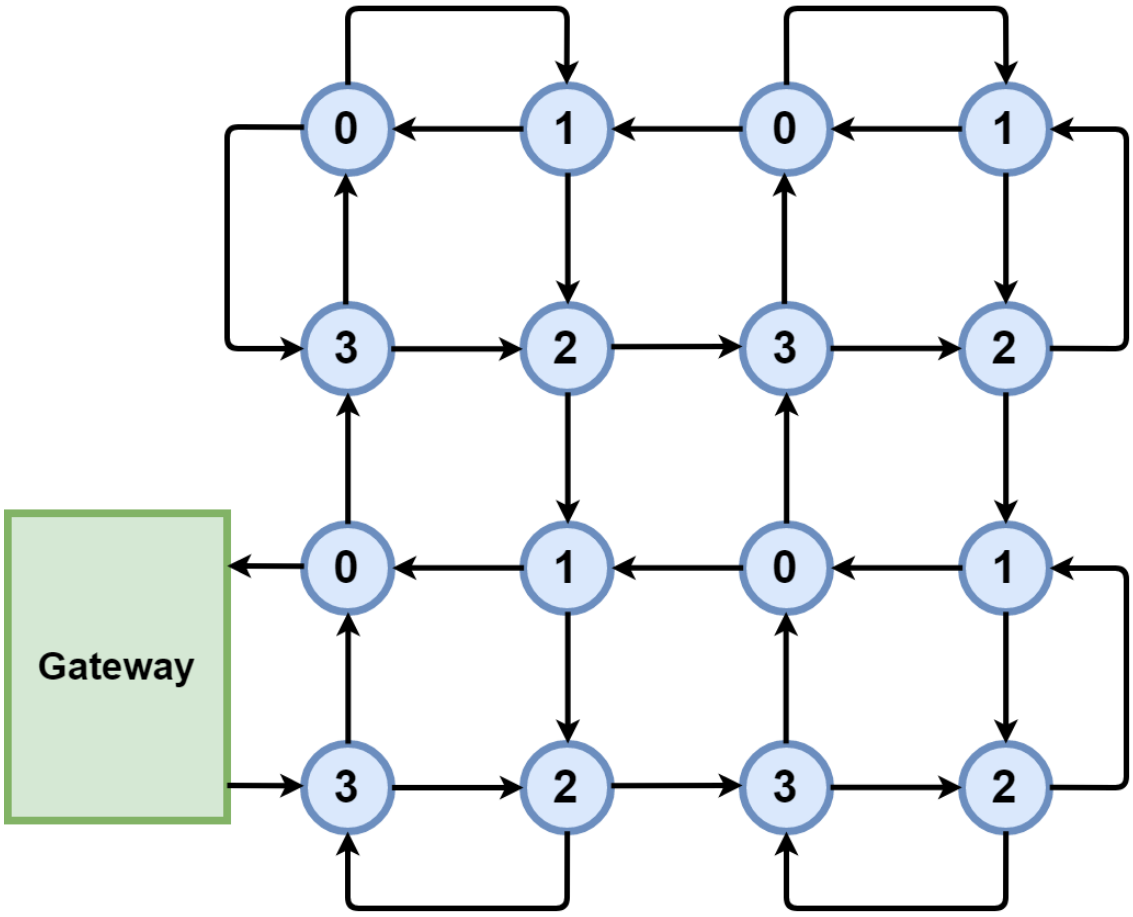
Σχήμα 2.6: Ελεγκτής τύπου περιστροφής 3

Το πλέγμα Manhattan το οποίο εξετάζουμε στην παρούσα φάση του προγράμματος Visorsurf είναι μεγέθους 24x24, δηλαδή αποτελείται συνολικά από 24 σειρές και 24 στήλες ελεγκτών, άρα συνολικά 576 ελεγκτές. Κάνοντας αναφορά στον κόμβο (x,y) του πλέγματος, εννοούμε τον κόμβο που είναι τοποθετημένος στην x-οστή σειρά και τη y-οστή στήλη του πλέγματος, από τα κάτω προς τα πάνω και από τα αριστερά προς τα δεξιά αντίστοιχα. Εντούτοις, το μέγεθος της υπερεπιφάνειας πιθανόν να είναι πολύ μεγαλύτερο από 24x24 μελλοντικά και συνεπώς είναι ιδιαίτερα σημαντικό οι αλγόριθμοι που δημιουργούνται στα πλαίσια του προγράμματος να χαρακτηρίζονται από δυνατότητα επεκτασιμότητας (scalability). Ο αλγόριθμος αναγνώρισης σφαλμάτων που έχω δημιουργήσει, μπορεί να λειτουργήσει για οποιοδήποτε μέγεθος πλέγματος

Manhattan από 2x2 και πάνω, χωρίς να υπάρχουν οποιοιδήποτε περιορισμοί όσον αφορά τον αριθμό των ελεγκτών που περιέχουν οι γραμμές και οι στήλες του πλέγματος.

Η επικοινωνία με το πλέγμα πραγματοποιείται μέσω της πηγής (gateway). Η πηγή είναι ένας υπολογιστής ο οποίος δε χαρακτηρίζεται από αυστηρούς περιορισμούς υπολογιστικής ισχύος και μνήμης όπως οι ελεγκτές και είναι σε θέση να εκτελέσει δύσκολους αλγορίθμους σε μικρό χρονικό διάστημα. Η πηγή είναι υπεύθυνη για τη μετάδοση των πακέτων στο πλέγμα με τρόπο που να διασφαλίζεται η εύρυθμη λειτουργία της υπερεπιφάνειας. Συγκεκριμένα, η πηγή είναι υπεύθυνη για τη δημιουργία και μετάδοση των κατάλληλων πακέτων έτσι ώστε να μεταβάλλεται η ηλεκτρομαγνητική συμπεριφορά της υπερεπιφάνειας βάσει των οδηγιών του χρήστη, αλλά και για την εκτέλεση αλγορίθμων για αναγνώριση (fault identification) και αποφυγή (fault tolerance) των σφαλμάτων που πιθανόν να εμφανιστούν στο πλέγμα. Η πηγή, είναι συνδεδεμένη με τη μια θύρα εισόδου του ελεγκτή που βρίσκεται στην πρώτη σειρά και στήλη του πλέγματος και μέσω αυτού μεταδίδει όλα τα πακέτα που πρέπει να αποσταλούν στο πλέγμα. Συνεπώς, η παρουσίαση κάποιου σφάλματος είτε στον ελεγκτή που είναι συνδεδεμένος με την πηγή, είτε και στους δύο ελεγκτές που βρίσκονται συνδεδεμένοι στις εξόδους του θα έχει καταστροφική επίδραση, καθώς δε θα υπάρχει πλέον δυνατότητα επικοινωνίας μεταξύ της πηγής και των υπόλοιπων ελεγκτών του πλέγματος.

Εντούτοις, σφάλματα είναι δυνατό να προκύψουν και σε οποιοδήποτε άλλο ελεγκτή του πλέγματος χωρίς να έχουν εξίσου καταστροφικές συνέπιες, καθώς δε θα εμποδίζουν την αποστολή πακέτων στο υπόλοιπο πλέγμα. Εντούτοις, για να είναι δυνατή η χρησιμοποίηση της υπερεπιφάνειας ακόμη και με την ύπαρξη των σφαλμάτων αυτών, όλα τα σφάλματα θα πρέπει να αναγνωρίζονται έτσι ώστε να αποφεύγεται η χρήση των ελεγκτών στους οποίους παρουσιάζονται. Με αυτό τον τρόπο το σύστημα θα καθίσταται ανθεκτικό σε σφάλματα.



Σχήμα 2.7: Πλέγμα τοπολογίας Manhattan 4x4 που δημιουργείται με την κατάλληλη διασύνδεση των ελεγκτών δικτύου. Στο Σχήμα διακρίνεται η σύνδεση του πλέγματος με την πηγή, αλλά και ο τύπος περιστροφής του κάθε ελεγκτή που απαρτίζει το πλέγμα.

2.2 Αλγόριθμοι αναγνώρισης και αποφυγής σφαλμάτων για κατανεμημένα συστήματα

Η δημιουργία ασύγχρονων κατανεμημένων συστημάτων τα οποία είναι ανθεκτικά σε σφάλματα απασχόλησε πολλούς ερευνητές από τα τέλη του 20ου αιώνα. Όπως αναφέρεται στο άρθρο του Gärtner [6], για τη δημιουργία ασύγχρονων κατανεμημένων συστημάτων ανθεκτικών σε σφάλματα υπάρχουν δύο βασικές διεργασίες που πρέπει να εκτελούνται, αρχικά η αναγνώριση των σφαλμάτων και έπειτα η διόρθωση τους. Ο Gärtner , μέσω του άρθρου του επικεντρώθηκε στα διάφορα είδη σφαλμάτων που είναι δυνατό να παρουσιαστούν και στις επιδράσεις που θα προκύψουν σε ένα σύστημα που είναι ανθεκτικό σε αυτά, ενώ παράλληλα καθόρισε τη γενική ορολογία που χρειαζόταν για να αναλυθεί περαιτέρω το πρόβλημα. Βάσει του συγκεκριμένου άρθρου, ένα σύστημα ανθεκτικό σε σφάλματα πρέπει να χαρακτηρίζεται τόσο από ασφάλεια (safety), όσο και από ζωτικότητα (liveliness). Επίσης, ο Gärtner τονίζει ότι σε καμιά περίπτωση δε μπορεί να δημιουργηθεί σύστημα το οποίο να είναι ανθεκτικό σε σφάλματα χωρίς να υπάρχει πλεονασμός (redundancy). Αυτό ισχύει, καθώς η ύπαρξη επιπλέον κόστους είτε σε υλικό, είτε σε λογισμικό, είτε σε χρόνο είναι αναγκαία για να μπορέσει το σύστημα να εμπλουτιστεί και να μπορεί να διαχειριστεί καταστάσεις κατά τις οποίες εμφανίστηκε κάποιο σφάλμα.

Τα σφάλματα που είναι δυνατό να προκύψουν, μπορούν να χωριστούν σε τρεις κατηγορίες. Η πρώτη κατηγορία είναι τα παροδικά σφάλματα (transient faults), τα οποία εμφανίζονται τυχαία και έχουν μικρή διάρκεια ύπαρξης. Επίσης, υπάρχουν τα διαλείποντα σφάλματα (intermittent faults), τα οποία εμφανίζονται ανά χρονικά διαστήματα και προκύπτουν από το ίδιο σημείο του συστήματος, ενώ η αντικατάσταση του κυκλώματος το οποίο τα προκαλεί, οδηγεί στην απαλοιφή τους. Τέλος, η τρίτη κατηγορία είναι τα μόνιμα σφάλματα (permanent faults) τα οποία εμφανίζονται πάντοτε, ενώ προκύπτουν και αυτά από το ίδιο σημείο του συστήματος [7].

Η ύπαρξη ενός σφάλματος μπορεί να προκύψει λόγω διάφορων φυσικών παραγόντων. Οι διακυμάνσεις στη θερμοκρασία ενός συστήματος και στην τάση του ηλεκτρικού ρεύματος που το διαπερνά πολλές φορές έχουν αρνητικές επιπτώσεις, συντείνοντας στη μείωση της διάρκειας ζωής του και στη δημιουργία σφαλμάτων. Συνεπώς, η χρήση

κατάλληλων υλικών κατασκευής είναι ιδιαίτερα σημαντική, καθώς ελαχιστοποιεί τον αριθμό των σφαλμάτων που θα προκύπτουν [7].

Πέρα από τα σφάλματα που προκύπτουν για φυσικούς λόγους, σφάλματα είναι δυνατό να προκύψουν και στο επίπεδο ζεύξης δεδομένων (data link layer), στο επίπεδο δικτύου (network layer) ή στο επίπεδο μεταφοράς (transport layer), για διαφορετικούς λόγους το καθένα [7]. Η έξυπνη αποφυγή των σφαλμάτων είναι κάτι που επιδιώκεται σε μεγάλο βαθμό, καθώς η αποδοτική επικοινωνία είναι καθοριστικής σημασίας για να υπάρχει υψηλή επίδοση σε ένα πλέγμα [3]. Έτσι, για την αντιμετώπιση των σφαλμάτων που είναι δυνατό να προκύψουν σε κάθε επίπεδο επικοινωνίας, έχουν δημιουργηθεί διάφοροι αλγόριθμοι αποφυγής σφαλμάτων (fault tolerance algorithms). Ο κάθε αλγόριθμος μπορεί να εφαρμοστεί μόνο σε συγκεκριμένα συστήματα λόγω των διαφορετικών απαιτήσεων πλεονασμού που πρέπει να ικανοποιηθούν, αλλά και λόγω των διαφορετικών περιορισμών που υπάρχουν στο κάθε σύστημα [7].

Ένας παράδειγμα αλγορίθμου αποφυγής σφαλμάτων που έχει προταθεί στο παρελθόν αποτελεί ο ARIADNE [1]. Ο ARIADNE είναι ένας αλγόριθμος επαναδιαμόρφωσης (reconfiguration) του δικτύου που διασφαλίζει την απουσία αδιεξόδων (deadlock-free). Το βασικό πλεονέκτημα του ARIADNE είναι ότι μπορεί να χρησιμοποιηθεί τόσο για αναγνώριση (identification), όσο και για αποφυγή των σφαλμάτων σε οποιαδήποτε τοπολογία δικτύου, ανεξαρτήτως αν ένα δίκτυο είναι μερικώς (partially connected) ή πλήρως συνδεδεμένο (fully connected). Εντούτοις, στο δικό μας πρόβλημα δε θα μπορούσε να χρησιμοποιηθεί ο αλγόριθμος αυτός καθώς για την υλοποίηση του απαιτείται η ύπαρξη πίνακα δρομολόγησης (routing table) σε κάθε κόμβο του δικτύου. Λόγω της εξαιρετικά περιορισμένης μνήμης που έχουν οι ελεγκτές του δικού μας πλέγματος, η αποθήκευση των πινάκων δρομολόγησης είναι κάτι το οποίο δε μπορεί να υποστηριχτεί. Ακόμη, για τη λειτουργία του ARIADNE απαιτείται η ενσωμάτωση επιπρόσθετου υλικού (hardware), όπως επιπλέον θύρες αλλά και ένα επιπλέον καλώδιο μετάδοσης για κάθε θύρα ενός ελεγκτή. Η ενσωμάτωση αυτού του επιπρόσθετου υλικού επιφέρει κινδύνους για δημιουργία αδιεξόδων στον ARIADNE, καθώς δεν υπάρχει κάποιος μηχανισμός που να εγγυάται τη δική του εύρυθμη λειτουργία.

Μια άλλη προσέγγιση που βασίζεται ακόμη περισσότερο στην προσθήκη επιπρόσθετου υλικού για την αναγνώριση των σφαλμάτων είναι η αρχιτεκτονική ANoC-TEST [8]. Η αρχιτεκτονική αυτή προνοεί την ύπαρξη περιτυλιγμάτων ελέγχου (test wrappers), τα οποία είναι ουσιαστικά υλικό τοποθετημένο γύρω από τους ελεγκτές δικτύου. Επίσης, απαιτείται η ύπαρξη ενός καναλιού διαμόρφωσης (configuration channel) αλλά και μιας επιπρόσθετης μονάδας GAC (generator-analyzer-controller / γεννήτριας-αναλυτή-ελεγκτή) η οποία είναι υπεύθυνη για τη δημιουργία διανυσμάτων ελέγχου (test vectors), την ανάλυση των αποτελεσμάτων ελέγχου αλλά και τον έλεγχο του καναλιού διαμόρφωσης. Δυστυχώς, τέτοιες προσεγγίσεις που απαιτούν επιπρόσθετο υλικό για την υλοποίηση της αναγνώρισης σφαλμάτων δε μπορούν να χρησιμοποιηθούν για την επίλυση του δικού μας προβλήματος, όχι μόνο λόγω του περιορισμένου χώρου που υπάρχει στη υπερεπιφάνεια, αλλά και επειδή θα υπάρχει μεγάλο κίνδυνος να προκύψουν παρεμβολές που θα επηρεάσουν την ηλεκτρομαγνητική της συμπεριφορά.

Μία διαφορετική ιδέα, είναι αυτή των Vitkovskiy et al. [9], οι οποίοι πρότειναν ένα δυναμικό αλγόριθμο δρομολόγησης ο οποίος αποφεύγει τα σφάλματα στις συνδέσεις μεταξύ των ελεγκτών, με τη χρήση τοπικών διαδρομών παράκαμψης (localised detouring paths). Συγκεκριμένα, όταν ο αλγόριθμος αυτός εντοπίσει σφάλμα σε κάποια σύνδεση που πρόκειται να χρησιμοποιηθεί, πραγματοποιεί μια προσαρμοστική επαναδρομολόγηση σε τοπικό επίπεδο (adaptive localised re-routing), δημιουργώντας μια εναλλακτική διαδρομή για να μπορέσει να επιτευχθεί η μετάδοση του πακέτου στον πλησιέστερο κόμβο του αρχικού μονοπατιού που είναι προσβάσιμος, έτσι ώστε να μπορέσει τελικά να καταλήξει στον τελικό του προορισμό. Εντούτοις, ο συγκεκριμένος αλγόριθμος μπορεί να χρησιμοποιηθεί υπό την προϋπόθεση ότι ο κάθε ελεγκτής μπορεί να ελέγξει κατά πόσο οι συνδέσεις του με τους γειτονικούς ελεγκτές είναι λειτουργικές ή όχι. Δυστυχώς, στο δικό μας πρόβλημα οι ελεγκτές δεν έχουν τη δυνατότητα να γνωρίζουν κατά πόσο οι εισερχόμενες προς αυτούς ή οι εξερχόμενες από αυτούς συνδέσεις είναι δυσλειτουργικές και συνεπώς ούτε αυτός ο αλγόριθμος μπορεί να αξιοποιηθεί.

2.3 Ανάγκη δημιουργίας νέου αλγορίθμου αναγνώρισης σφαλμάτων

Όπως αναλύθηκε στο Κεφάλαιο 2.2, λόγω των ιδιαιτεροτήτων αλλά και των περιορισμών που χαρακτηρίζουν την αρχιτεκτονική της υπερεπιφάνειας, καμία από τις υπάρχων λύσεις που αναφέρονται στη βιβλιογραφία δε θα μπορούσε να χρησιμοποιηθεί χωρίς να υπάρξουν κάποιες υποθέσεις ή αλλαγές. Έτσι, αποφάσισα να σχεδιάσω το δικό μου αλγόριθμο αναγνώρισης σφαλμάτων για τοπολογία Manhattan, ο οποίος ικανοποιεί όλους τους περιορισμούς που έχουν τεθεί στο πρόγραμμα Visorsurf. Ο αλγόριθμος που έχω σχεδιάσει βασίζεται στην τεχνική ανάστροφης μετάδοσης σφάλματος (back-propagation), καθώς ο τρόπος με τον οποίο λειτουργεί είναι η σταδιακή μετάδοση του σφάλματος από το σημείο που συμβαίνει πίσω στην πηγή, ακολουθώντας το ίδιο μονοπάτι που χρησιμοποιήθηκε και για τη μεταφορά των πακέτων προς τον μη λειτουργικό ελεγκτή. Ο αλγόριθμος αυτός δεν απαιτεί την προσθήκη υλικού στο σύστημα, αλλά ούτε και την επιπλέον χρησιμοποίηση της μνήμης των ελεγκτών για σκοπούς που αφορούν την αναγνώριση σφαλμάτων. Η δημιουργία του αλγορίθμου μου θεωρώ ότι μπορεί να θέσεις τις βάσεις και να συμβάλει σημαντικά και στη μελλοντική δημιουργία αλγόριθμου αποφυγής σφαλμάτων για το συγκεκριμένο σύστημα. Διότι βάσει των Radetzki et al. [7], για να θεωρείται ουσιαστική οποιαδήποτε ιδέα για την αποφυγή σφαλμάτων σε ένα σύστημα, πρέπει πάντα να συνδυάζεται και από μια αντίστοιχη ιδέα διάγνωσης των σφαλμάτων του εν λόγω συστήματος.

Ο αλγόριθμος μου μπορεί να πραγματοποιήσει πλήρη αναγνώριση συγκεκριμένων ειδών σφαλμάτων που μπορούν να προκύψουν στην τοπολογία Manhattan. Συγκεκριμένα, τα σφάλματα τα οποία αναγνωρίζονται πάντοτε είναι τα σφάλματα που αφορούν δυσλειτουργία των ελεγκτών και τα οποία είναι μόνιμα σφάλματα. Με τον όρο δυσλειτουργία ή μη λειτουργικότητα κάποιου ελεγκτή εννοούμε αδυναμία του ελεγκτή να λάβει ή να διαχειριστεί ένα εισερχόμενο πακέτο με αποτέλεσμα το πακέτο αυτό να μένει παγιδευμένο είτε στους buffer εισόδου και επεξεργασίας του συγκεκριμένου ελεγκτή, είτε στον buffer εξόδου των ελεγκτών που είναι συνδεδεμένοι με τις θύρες εισόδου του μη λειτουργικού ελεγκτή. Εντούτοις, ελεγκτές που έχουν απροσδιόριστη ή κακόβουλη συμπεριφορά δεν εμπίπτουν σε αυτή την κατηγορία και

η περίπτωση ύπαρξης τους δεν έχει μελετηθεί καθώς θεωρείται απομακρυσμένο, μέχρι και απίθανο να παρουσιαστούν στο δίκτυο μιας υπερεπιφάνειας.

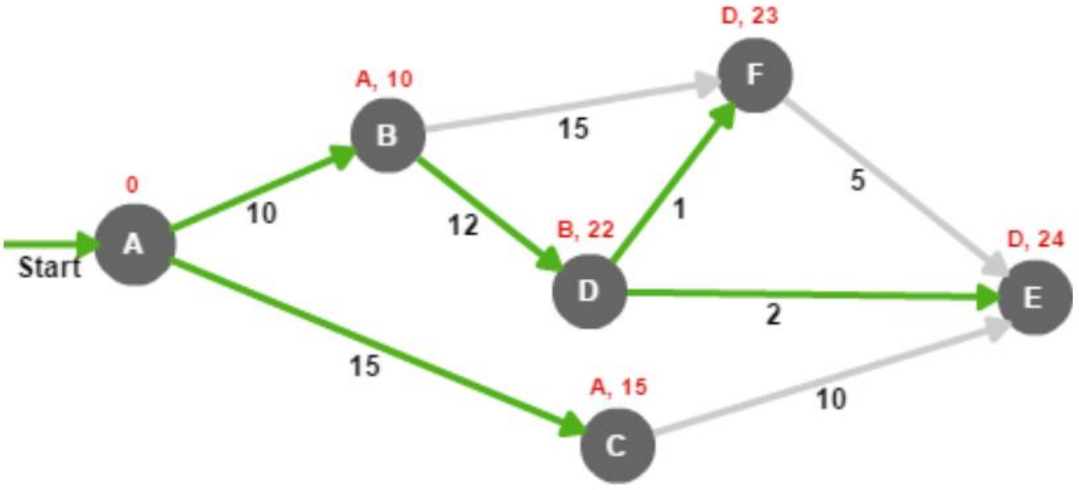
Τα μόνιμα σφάλματα που αφορούν δυσλειτουργία των ελεγκτών εντοπίζονται κάθε φορά που τρέχει ο αλγόριθμος και η αναγνώριση τους πραγματοποιείται με πλήρη ακρίβεια και ορθότητα. Επίσης, κάθε φορά που τρέχει ο αλγόριθμος είναι πιθανόν να αναγνωρίζονται και μερικά παροδικά ή διαλείποντα σφάλματα, τα οποία επίσης αφορούν δυσλειτουργία των ελεγκτών.

2.4 Αλγόριθμοι εύρεσης βραχύτερου μονοπατιού

Αναλόγως της τιμής εισόδου δύο παραμέτρων του αλγορίθμου μου, οι οποίες θα επεξηγηθούν αναλυτικά στο Κεφάλαιο 3, ο αλγόριθμος είναι δυνατό να χρησιμοποιήσει είτε τον αλγόριθμο εύρεσης βραχύτερου μονοπατιού του Dijkstra [4], είτε μια παραλλαγή του αλγορίθμου Bellman-Ford [2, 5] κατά την εκτέλεση του. Έτσι, σε αυτό το κεφάλαιο θα θυμίσω την ιδέα που πρότεινε ο Dijkstra, αλλά και την ιδέα των Bellman και Ford, όπως και το λόγο για την οποία την έχω διαφοροποιήσει για τις ανάγκες του δικού μου αλγορίθμου. Υπενθυμίζεται ότι μονοπάτι ή διαδρομή σε ένα γράφο ονομάζεται μια ακολουθία κόμβων, όπου κάθε κόμβος της ακολουθίας συνδέεται με τον επόμενό του μέσω ακμής. Ως κόστος του μονοπατιού ορίζεται το άθροισμα του βάρους των ακμών που αποτελούν το συγκεκριμένο μονοπάτι.

2.4.1 Ο Αλγόριθμος του Dijkstra

Ο αλγόριθμος που πρότεινε ο Dijkstra [4], χρησιμοποιείται για την εύρεση του μονοπατιού με το μικρότερο κόστος σε ένα γράφο με μη αρνητικά βάρη στις ακμές του. Συνεπώς, αυτό που δίνεται ως είσοδος στον αλγόριθμο είναι ένας γράφος με μη αρνητικά βάρη στις ακμές του, αλλά και ένας κόμβος αφετηρία. Ο τρόπος με τον οποίο λειτουργεί ο αλγόριθμος, είναι υπολογίζοντας το ελάχιστο κόστος που απαιτείται για τη μετάβαση από τον κόμβο αφετηρία προς κάθε άλλο κόμβο του πλέγματος, αλλά και το ποιος πρέπει να είναι ο προηγούμενος κόμβος του μονοπατιού που θα χρησιμοποιηθεί έτσι ώστε να προκύψει το συγκεκριμένο κόστος (Σχήμα 2.8).



Σχήμα 2.8: Το αποτέλεσμα εκτέλεσης του αλγόριθμου εύρεσης βραχύτερου μονοπατιού του Dijkstra σε ένα γράφο

Ακολουθώντας τη σειρά των κόμβων που διασφαλίζουν το χαμηλότερο κόστος μετάβασης για ένα κόμβο προς τα πίσω, προκύπτει η διαδρομή με το μικρότερο κόστος από τον κόμβο αφετηρία προς το συγκεκριμένο κόμβο. Έτσι, ο αλγόριθμος επιστρέφει τις διαδρομές με το μικρότερο κόστος από τον κόμβο αφετηρία προς κάθε άλλο κόμβο του γράφου. Σε περίπτωση που προς κάποιον κόμβο υπάρχουν πολλές διαδρομές με το μικρότερο κόστος, τότε επιλέγεται τυχαία μια εξ αυτών των διαδρομών.

Στην περίπτωση του δικού μας προβλήματος, ως γράφος χρησιμοποιείται το πλέγμα των ελεγκτών και ως αφετηρία ορίζεται ο ελεγκτής που είναι συνδεδεμένος με την πηγή. Χρησιμοποιώντας τον αλγόριθμο του Dijkstra προσπαθούμε να εντοπίσουμε το μονοπάτι με το μικρότερο κόστος από την πηγή προς ένα συγκεκριμένο ελεγκτή, θέτοντας τα βάρη του γραφήματος βάσει της λογικής που καθορίζει ο αλγόριθμος δημιουργίας μονοπατιού και η οποία θα επεξηγηθεί περαιτέρω στο Κεφάλαιο 3.6.

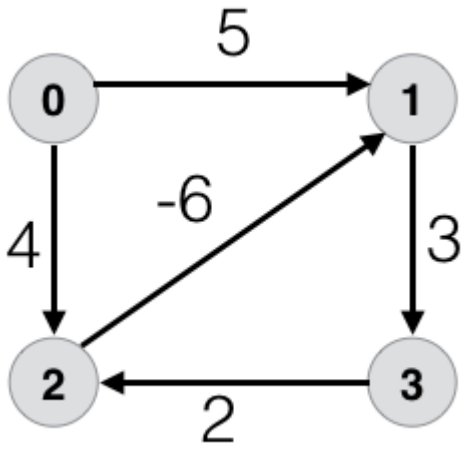
```
1   function Dijkstra(Graph, source):
2
3       create vertex set Q
4
5       for each vertex v in Graph:
6           dist[v] ← INFINITY
7           prev[v] ← UNDEFINED
8           add v to Q
9       dist[source] ← 0
10
11      while Q is not empty:
12          u ← vertex in Q with min dist[u]
13          remove u from Q
14
15          for each neighbor v of u:
16              alt ← dist[u] + length(u, v)
17              if alt < dist[v]:
18                  dist[v] ← alt
19                  prev[v] ← u
20
21      return dist[], prev[]
```

Σχήμα 2.9: Ψευδοκώδικας του αλγόριθμου εύρεσης βραχύτερου μονοπατιού του Dijkstra

2.4.2 Η παραλλαγή του Αλγορίθμου Bellman-Ford

Ο αλγόριθμος Bellman-Ford [2, 5], χρησιμοποιείται επίσης για την εύρεση του μονοπατιού με το μικρότερο κόστος σε ένα γράφο. Η βασική διαφορά του συγκεκριμένου αλγορίθμου από τον αλγόριθμο του Dijkstra, είναι ότι ο Bellman-Ford μπορεί να λειτουργήσει σωστά και για γράφους με αρνητικά βάρη στις ακμές τους. Συνεπώς, και σε αυτή την περίπτωση δίνεται ως είσοδος στον αλγόριθμο είναι ένας γράφος και ένας κόμβος αφετηρία και ο αλγόριθμος επιστρέφει τις διαδρομές με το μικρότερο κόστος από τον κόμβο αφετηρία προς κάθε άλλο κόμβο του γράφου. Επίσης, ισχύει και για τον αλγόριθμο Bellman-Ford ότι στην περίπτωση που προς κάποιον κόμβο υπάρχουν πολλές διαδρομές με το μικρότερο κόστος, τότε επιλέγεται τυχαία μια εξ αυτών των διαδρομών.

Ωστόσο, ένα βασικό μειονέκτημα που χαρακτηρίζει το συγκεκριμένο αλγόριθμο, είναι ότι παγιδεύεται στους κύκλους αρνητικού βάρους. Κύκλος σε ένα γράφο λέγεται ένα μονοπάτι το οποίο καταλήγει στον ίδιο κόμβο από τον οποίο ξεκινά. Σε περίπτωση που το άθροισμα των βαρών ενός κύκλου είναι αρνητικό, τότε ο συγκεκριμένος κύκλος είναι ένας κύκλος αρνητικού βάρους (Σχήμα 2.10). Όταν ο γράφος εισόδου περιλαμβάνει κύκλους αρνητικού βάρους, ο αλγόριθμος Bellman-Ford δε μπορεί να τερματίσει επιτυχώς την εκτέλεση του, καθώς επιδιώκοντας να ελαχιστοποιήσει το κόστος των μονοπατιών παγιδεύεται στους συγκεκριμένους κύκλους.



Σχήμα 2.10: Γράφος που του οποίου οι κόμβοι 2,1 και 3 δημιουργούν ένα κύκλο αρνητικού βάρους

Εντούτοις, για την υλοποίηση της λογικής κάποιων αλγόριθμων δημιουργίας μονοπατιού ήταν επιβεβλημένη η χρησιμοποίηση ενός αλγόριθμου εύρεσης βέλτιστου μονοπατιού ο οποίος να υποστηρίζει αρνητικά βάρη, αλλά παράλληλα να μην παγιδεύεται στους κύκλους αρνητικού βάρους. Γι' αυτό το λόγο δημιούργησα μια παραλλαγή του αλγορίθμου Bellman-Ford η οποία αναγνωρίζει και αποφεύγει τους κύκλους αρνητικού βάρους. Ο τρόπος με τον οποίο επιτυγχάνεται αυτό, είναι μέσω της απομνημόνευσης των κόμβων από τις οποίες έχει περάσει ήδη το μονοπάτι που δημιουργείται. Έτσι, κατά τη δική μου παραλλαγή, όταν ο αλγόριθμος Bellman-Ford επιχειρήσει να εισάγει σε ένα μονοπάτι ένα κόμβο που περιέχεται ήδη στο συγκεκριμένο μονοπάτι, αυτό δεν επιτρέπεται για να μη σχηματιστεί κύκλος και ο αλγόριθμος εισάγει στο μονοπάτι ένα εναλλακτικό κόμβο. Για να διασφαλιστεί η διατήρηση της βελτιστότητας του αλγορίθμου, η ακμή προς τον κόμβο που επιλέγεται κάθε φορά είναι αυτή που επιφέρει το αμέσως χαμηλότερο κόστος μετά από την ακμή ή τις ακμές που θα οδηγούσαν σε δημιουργία κύκλου.

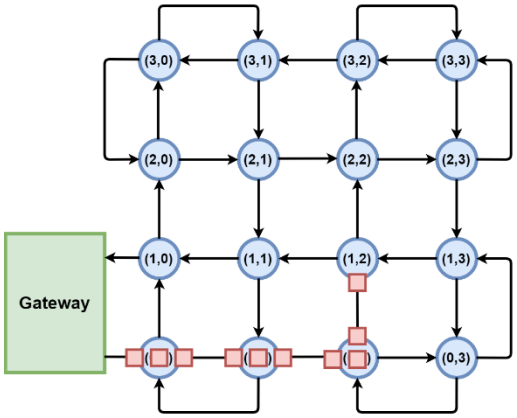
```
1   public static ArrayList<Integer> extendedBellmanFord(int[][] graph, int nodeToReturn){
2       int dist[] = new int[graph.length];
3       int previousDist[] = new int[graph.length];
4       ArrayList<Integer> prev[] = new ArrayList[graph.length];
5       for(int i=0; i<graph.length; i++) {
6           dist[i] = Integer.MAX_VALUE/2;
7           prev[i] = new ArrayList<Integer>();
8       }
9       dist[0] = 0;
10      do {
11          for(int i=0; i<graph.length; i++) {
12              previousDist[i] = dist[i];
13          }
14          for(int i=0; i<graph.length; i++) {
15              for(int j=0; j<graph.length; j++) {
16                  if((graph[i][j]!=0)&&(dist[i]+graph[i][j]<dist[j])) {
17                      if(!prev[i].contains(j)) {
18                          dist[j] = dist[i]+graph[i][j];
19                          prev[j] = new ArrayList<Integer>();
20                          prev[j].addAll(prev[i]);
21                          prev[j].add(i);
22                      }
23                  }
24              }
25          }
26      }while(isThereDistChange(dist,previousDist));
27      prev[nodeToReturn].add(nodeToReturn);
28      return prev[nodeToReturn];
29  }
30
31  public static boolean isThereDistChange(int[] currentDist, int[] previousDist) {
32      for(int i=0; i<currentDist.length; i++) {
33          if(currentDist[i]!=previousDist[i]) {
34              return true;
35          }
36      }
37      return false;
38  }
```

Σχήμα 2.11: Η παραλλαγή του αλγορίθμου Bellman-Ford που δημιούργησα η οποία αποφεύγει τους κύκλους αρνητικού βάρους, γραμμένη στη γλώσσα προγραμματισμού Java.

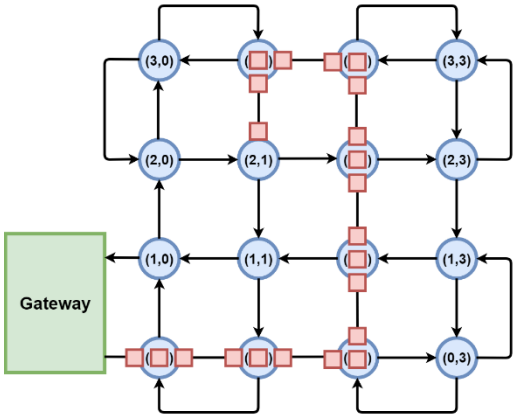
2.5 Ο αλγόριθμος δρομολόγησης XY

Η λειτουργία του αλγορίθμου μου, βασίζεται και στον αλγόριθμο δρομολόγησης ο οποίος χρησιμοποιείται στο πλέγμα και τον οποίο λαμβάνει ως παράμετρο. Ο αλγόριθμος δρομολόγησης είναι ουσιαστικά αυτός που καθορίζει τη διαδρομή που θα ακολουθήσει κάποιο πακέτο το οποίο μεταδίδεται εντός του πλέγματος, μέχρι να φτάσει στον προορισμό του. Στο πρόγραμμα Visorsurf ως βασικός αλγόριθμος δρομολόγησης για το δίκτυο της υπερεπιφάνειας καθορίστηκε ο αλγόριθμος XY [10].

Η βασική ιδέα στην οποία στηρίζεται η λειτουργία του XY, είναι ότι τα πακέτα αρχικά δρομολογούνται στο επίπεδο των σειρών του πλέγματος και ακολούθως στο επίπεδο των στηλών. Συγκεκριμένα, βάσει του XY ένα πακέτο που εισέρχεται στο πλέγμα και βρίσκεται αρχικά στον ελεγκτή (0,0) θα παραμείνει στη σειρά 0 του πλέγματος μέχρι να φτάσει ή να ξεπεράσει (λόγω του περιορισμού κατεύθυνσης της τοπολογίας Μανχάταν) τη στήλη στην οποία βρίσκεται ο προορισμός του. Μόλις γίνει αυτό, το πακέτο θα παύσει πλέον να κινείται στη σειρά 0 και θα ξεκινήσει να κινείται κατά μήκος της συγκεκριμένης στήλης, μέχρι να φτάσει ή να ξεπεράσει (λόγω του περιορισμού κατεύθυνσης της τοπολογίας Μανχάταν) τη σειρά στην οποία βρίσκεται ο προορισμός του. Σε περίπτωση που η σειρά ή η στήλη του προορισμού έχει ξεπεραστεί, το πακέτο αρχίζει να κινείται και πάλι αρχικά στο επίπεδο των σειρών και έπειτα στο επίπεδο των στηλών, μέχρι να φτάσει σε αυτόν.



Σχήμα 2.12: Δρομολόγηση πακέτων προς τον κόμβο (1,2) χρησιμοποιώντας τον αλγόριθμο δρομολόγησης XY



Σχήμα 2.13: Δρομολόγηση πακέτων προς τον κόμβο (2,1) χρησιμοποιώντας τον αλγόριθμο δρομολόγησης XY

```
if((packet.xDest==node.x) && (packet.yDest==node.y)){
    return destination_found;
}
switch (node.type) {
    case 0:
        if(packet.xDest<node.x){
            return output2;
        }else if(packet.yDest>=node.y){
            return output1;
        }else{
            return output2;
        }
    case 1:
        if(packet.xDest<node.x){
            return output1;
        }else if(packet.yDest>node.y){
            return output1;
        }else{
            return output2;
        }
    case 2:
        if(packet.xDest>node.x){
            return output2;
        }else if(packet.yDest<=node.y){
            return output1;
        }else{
            return output2;
        }
    case 3:
        if(packet.xDest>node.x){
            return output1;
        }else if(packet.yDest<node.y){
            return output1;
        }else{
            return output2;
        }
}
```

Σχήμα 2.14: Ψευδοκώδικας Αλγορίθμου XY

Κεφάλαιο 3

Ο Αλγόριθμος Αναγνώρισης Σφαλμάτων

3.1 Επεξήγηση Λογικής Αλγορίθμου

Στο κεφάλαιο αυτό περιγράφεται λεπτομερώς ο αλγόριθμος αναγνώρισης σφαλμάτων που έχω σχεδιάσει. Ο αλγόριθμος, βασίζεται στη λογική διαχωρισμού των ελεγκτών του πλέγματος σε τρία σύνολα ελεγκτών. Το πρώτο σύνολο είναι οι λειτουργικοί ελεγκτές οι οποίοι έχουν επιθυμητή συμπεριφορά η οποία αποκλείει την ύπαρξη σφάλματος (Working Nodes), το δεύτερο σύνολο είναι οι μη λειτουργικοί ελεγκτές οι οποίοι έχουν δυσλειτουργική συμπεριφορά που προκύπτει από την παρουσία κάποιου σφάλματος (Faulty Nodes), ενώ επίσης υπάρχει και το σύνολο των μη προσβάσιμων ελεγκτών (Unreachable Nodes) οι οποίοι είναι κόμβοι των οποίων η κατάσταση δε μπορεί να εξακριβωθεί, καθώς δεν υπάρχει κανένας τρόπος αποστολής πακέτων από την πηγή προς τους συγκεκριμένους ελεγκτές, λόγω των μη λειτουργικών ελεγκτών που υπάρχουν στο πλέγμα.

Αρχικά, όλοι οι ελεγκτές βρίσκονται σε ένα άλλο σύνολο ελεγκτών, το σύνολο των μη ελεγμένων ελεγκτών (Unchecked Nodes), το οποίο περιέχει ουσιαστικά τους ελεγκτές που δεν έχουν κατηγοριοποιηθεί ακόμα στα τρία σύνολα ελεγκτών που προαναφέρονται. Στόχος του αλγορίθμου, είναι με το πέρας της εκτέλεσης του όλοι οι ελεγκτές του δικτύου να έχουν αφαιρεθεί από το σύνολο των μη ελεγμένων ελεγκτών και να έχουν κατηγοριοποιηθεί με το σωστό τρόπο στα σύνολα των λειτουργικών, των μη λειτουργικών και των μη προσβάσιμων ελεγκτών, αναλόγως της κατάστασης στην οποία βρίσκονται.

Ο τρόπος με τον οποίο πραγματοποιείται η συγκεκριμένη κατηγοριοποίηση από τον αλγόριθμο , είναι μέσω της αποστολής πολλών πακέτων στο πλέγμα με παραλήπτη ένα μη ελεγμένο ελεγκτή, χρησιμοποιώντας το ίδιο μονοπάτι για την αποστολή όλων των πακέτων. Με τον όρο μονοπάτι, εννοούμε μια ακολουθία ελεγκτών όπου από τον κάθε προηγούμενο ελεγκτή της ακολουθίας υπάρχει στο πλέγμα μια εξερχόμενη σύνδεση προς τον κάθε επόμενο ελεγκτή της ακολουθίας. Τα μονοπάτια που χρησιμοποιούνται

από τον αλγόριθμο είναι άκυκλα και δεν περιέχουν κάποιο ελεγκτή που έχει κατηγοριοποιηθεί ως μη λειτουργικός. Ο λόγος είναι έτσι ώστε να υπάρχει η προοπτική για τα πακέτα που αποστέλλονται μέσω τον μονοπατιών να φτάσουν στον ελεγκτή που είναι ο προορισμός τους, για να μπορέσει να εξεταστεί η κατάσταση του συγκεκριμένου ελεγκτή.

Η βασική ιδέα είναι ότι όταν κάποιος ελεγκτής είναι δυσλειτουργικός, όταν κάποιο πακέτο φτάσει σε αυτόν δε θα μπορέσει να το λάβει ή να το διαχειριστεί και έτσι το πακέτο θα παγιδευτεί είτε στους buffer εισόδου και επεξεργασίας του συγκεκριμένου ελεγκτή, είτε στον buffer εξόδου του ελεγκτή που προηγείται στο μονοπάτι. Συνεπώς, ούτε τα υπόλοιπα πακέτα που έχουν τον ίδιο προορισμό και χρησιμοποιείται το ίδιο μονοπάτι για την αποστολή τους δε θα μπορέσουν να φτάσουν στον προορισμό τους. Συγκεκριμένα, αφού παγιδευτεί το πρώτο πακέτο, η παγίδευση και των υπόλοιπων πακέτων στο μονοπάτι θα προκληθεί αναπόφευκτα, εφόσον όλοι οι buffers του μονοπατιού θα γεμίσουν με πακέτα τα οποία θα είναι αδύνατο να φτάσουν στον τελικό τους προορισμό.

Έτσι, αν η πηγή μεταδώσει τόσα πακέτα αναγνώρισης σφαλμάτων όση είναι και η ποσότητα των buffers που περιλαμβάνονται μέσα στο μονοπάτι, σε περίπτωση που ο τελευταίος ελεγκτής του μονοπατιού περιέχει σφάλμα, όλα τα πακέτα θα παγιδευτούν σταδιακά στους buffers του μονοπατιού. Η πηγή θα μπορέσει να διαγνώσει το πρόβλημα που έχει προκύψει, καθώς ο buffer εισόδου του πρώτου ελεγκτή του πλέγματος με τον οποίο είναι συνδεδεμένη θα είναι μόνιμα γεμάτος. Συνεπώς, αν μετά τη μετάδοση όλων των πακέτων αναγνώρισης σφαλμάτων η πηγή αντιμετωπίσει μια τέτοια κατάσταση θα καταλήξει στο ασφαλές συμπέρασμα ότι ο τελευταίος ελεγκτής του συγκεκριμένου μονοπατιού, ο οποίος αποτελεί τον προορισμό των πακέτων, είναι μη λειτουργικός, ενώ όλοι οι υπόλοιποι ελεγκτές του μονοπατιού είναι λειτουργικοί.

Βεβαίως, είναι επίσης υπαρκτό και το ενδεχόμενο να υπάρχει κάποιος άλλος προηγούμενος ελεγκτής στο μονοπάτι που να περιέχει σφάλμα. Σε μια τέτοια περίπτωση, η παγίδευση των πακέτων στο πλέγμα θα αρχίσει να γίνεται πολύ πιο νωρίς. Συνεπώς, η πηγή θα διαπιστώσει την μόνιμη πληρότητα του buffer εισόδου του πρώτου ελεγκτή του πλέγματος πολύ πιο νωρίς και ενώ δεν έχει προλάβει να

μεταδώσει όλα τα πακέτα αναγνώρισης σφαλμάτων που θα έπρεπε στο πλέγμα. Έτσι, βάσει του αριθμού των πακέτων αναγνώρισης σφαλμάτων που έχει προλάβει να μεταδώσει και γνωρίζοντας ότι ο κάθε ελεγκτής χρησιμοποιεί τρείς από τους buffers του για τη μετάδοση των πακέτων, η πηγή θα μπορεί να υπολογίσει σε ποιον ακριβώς κόμβο του πλέγματος έχει εμφανιστεί το σφάλμα και να κατηγοριοποιήσει το συγκεκριμένο ελεγκτή του πλέγματος ως μη λειτουργικό και τους ελεγκτές που προηγούνται από αυτόν στο μονοπάτι ως λειτουργικούς.

Ο διαχωρισμός που κάνει η πηγή κατά πόσο η πληρότητα του buffer εισόδου του πρώτου ελεγκτή του πλέγματος οφείλεται σε αναμονή για εκκένωση του buffer επεξεργασίας ή σε παγίδευση πακέτου επιτυγχάνεται είναι μέσω των Gateway Time-outs. Συγκεκριμένα, μετά την αποστολή του κάθε πακέτου η πηγή ξεκινά να μετρά το χρόνο που έχει περάσει από την εν λόγω αποστολή. Όταν ο χρόνος αυτός ξεπεράσει το μέγιστο χρόνο παραμονής ενός πακέτου μέσα σε ένα ελεγκτή και ο buffer εισόδου του ελεγκτή που είναι συνδεδεμένος με την πηγή δεν έχει ακόμη ελευθερωθεί, τότε πραγματοποιείται Gateway Time-out και η πηγή αντιλαμβάνεται ότι έχει προκύψει παγίδευση των πακέτων στο πλέγμα λόγω κάποιου σφάλματος.

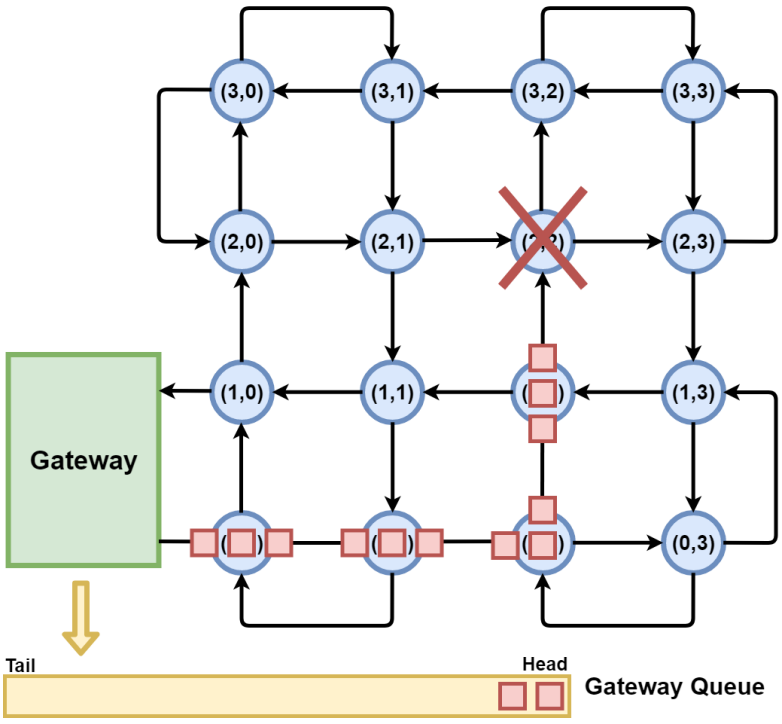
Συνεπώς, στην περίπτωση που όλα τα πακέτα αναγνώρισης σφαλμάτων αποστάλθηκαν επιτυχώς και δεν έχει προκύψει Gateway Time-out ακόμη και μετά την αποστολή του τελευταίου πακέτου, γίνεται αντιληπτό από την πηγή ότι όλοι οι κόμβοι του μονοπατιού είναι λειτουργικοί.

Πριν να πραγματοποιηθεί η μετάδοση των πακέτων μετάδοσης σφαλμάτων στο πλέγμα, η πηγή τα δημιουργεί και τα εισάγει σε μια ουρά που ονομάζεται Gateway Queue. Κάθε φορά που ο πρώτος ελεγκτής του πλέγματος είναι σε θέση να δεχτεί το κάποιο πακέτο, η πηγή αφαιρεί το επόμενο πακέτο που υπάρχει στην Gateway Queue και του το αποστέλλει. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να αδειάσει η Gateway Queue.

Βεβαίως, για να είναι δυνατή η εφαρμογή της διαδικασίας που έχει περιγράφει πιο πάνω, θα πρέπει πρώτα να σχηματιστεί το μονοπάτι που θα ακολουθούν τα πακέτα αναγνώρισης σφαλμάτων. Για να γίνει αυτό, πριν από τα πακέτα αναγνώρισης σφαλμάτων θα πρέπει πρώτα να αποστέλλονται τα κατάλληλα πακέτα οδοφράγματος

τα οποία να καθορίζουν τις εξόδους των ελεγκτών που δεν πρέπει να χρησιμοποιούνται, έτσι ώστε να μην παρεκκλίνουν τα πακέτα από το επιθυμητό μονοπάτι.

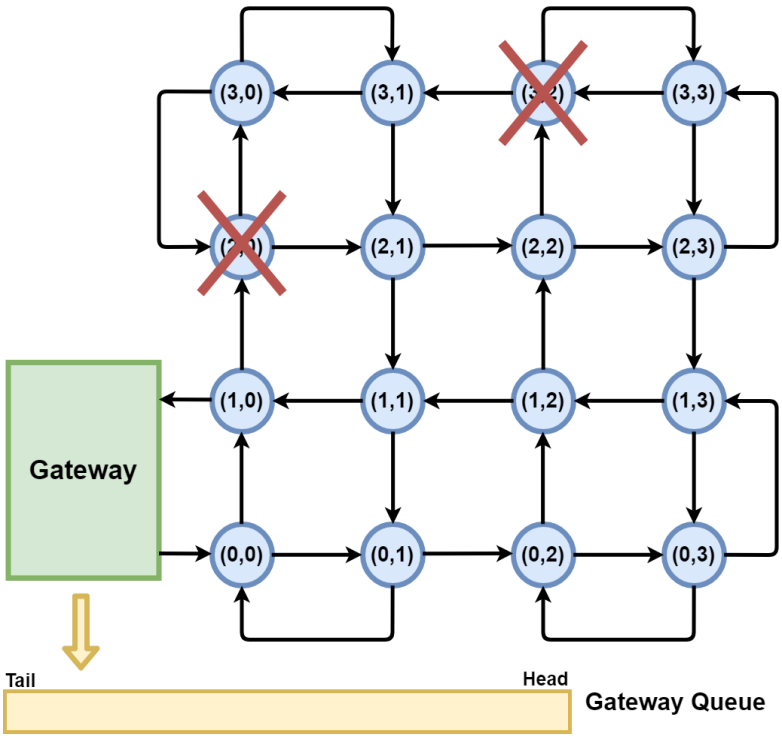
Συνεπώς, αρχικά πραγματοποιείται η εισχώρηση των πακέτων οδοφράγματος στο Gateway Queue τα οποία έχουν στόχο το σχηματισμό του επιθυμητού μονοπατιού και ακολούθως η εισχώρηση των πακέτων αναγνώρισης σφαλμάτων τα οποία έχουν στόχο την εξέταση της λειτουργικότητας του μονοπατιού. Για να διασφαλιστεί ότι ούτε τα πακέτα οδοφράγματος θα παρεκκλίνουν από το επιθυμητό μονοπάτι, πρώτα γίνεται η ταξινόμηση τους με βάση τη θέση που έχει ο προορισμός τους μέσα στο μονοπάτι και έπειτα ακολουθεί η εισχώρηση τους στο Gateway Queue. Συγκεκριμένα, πρώτο στο Gateway Queue τοποθετείται το πακέτο οδοφράγματος του οποίου ο προορισμός είναι πιο κοντά στην αρχή του μονοπατιού και τελευταίο το πακέτο οδοφράγματος του οποίου ο προορισμός είναι πιο κοντά στο τέλος του μονοπατιού. Με αυτό τον τρόπο διασφαλίζεται ότι η σταδιακή διαμόρφωση του μονοπατιού από την πηγή μέχρι τον τελευταίο κόμβο του μονοπατιού θα γίνει με τη σωστή σειρά, χωρίς υπάρξει κάποια παρέκκλιση πακέτου, καθώς όλα τα πακέτα θα βρίσκονται ανά πάσα στιγμή στο κομμάτι του μονοπατιού που έχει ήδη διαμορφωθεί.



Σχήμα 3.1: Αναγνώριση σφάλματος στον κόμβο (2,2) μέσω Gateway Time-out

Η ύπαρξη των πακέτων οδοφράγματος είναι βεβαίως κάτι που πρέπει να λαμβάνεται υπόψη και κατά την παρουσίαση ενός Gateway Time-out, έτσι ώστε να υπολογίζεται σωστά ο ελεγκτής στον οποίο εμφανίζεται το σφάλμα. Συγκεκριμένα, για να υπολογίσει τον αριθμό των πακέτων που έχουν παγιδευτεί στο πλέγμα, η πηγή αφαιρεί από το συνολικό αριθμό των πακέτων που έχουν αποσταλεί, τα πακέτα οδοφράγματος που έχουν ήδη φτάσει στον προορισμό τους. Αφού υπολογιστεί ο αριθμός των παγιδευμένων πακέτων και εφόσον είναι γνωστό ότι ο κάθε ελεγκτής χρησιμοποιεί τρεις από τους buffers του για τη μετάδοση των πακέτων, η πηγή μπορεί εύκολα μέσω μιας απλής διαίρεσης να συμπεράνει σε ποιο κόμβο του μονοπατιού παρατηρείται το σφάλμα.

Αφού λοιπόν εξεταστεί η λειτουργικότητα των ελεγκτών ενός μονοπατιού, πραγματοποιείται καθαρισμός του πλέγματος έτσι ώστε να αφαιρεθούν τα παγιδευμένα πακέτα και να αρθούν τα οδοφράγματα που είχαν τοποθετηθεί. Ακολούθως, εξετάζεται η λειτουργικότητα νέων μονοπατιών των οποίων ο τελευταίος ελεγκτής είναι μη ελεγμένος, μέχρι να αδειάσει εντελώς το σύνολο των μη ελεγμένων ελεγκτών.



Σχήμα 3.2: Αναγνώριση των ελεγκτών (3,0) και (3,1) ως μη προσβάσιμων ελεγκτών, καθώς ο αλγόριθμος δημιουργίας μονοπατιού, γνωρίζοντας ότι οι κόμβοι (2,0) και (3,2) είναι μη λειτουργικοί, αδυνατεί να δημιουργήσει μονοπάτι προς αυτούς.

Εντούτοις, υπάρχει και η πιθανότητα να επιλεγεί κάποιος ελεγκτής προς τον οποίο δεν είναι δυνατό να δημιουργηθεί κάποιο μονοπάτι από την πηγή, λόγω ύπαρξης μη λειτουργικών ελεγκτών σε όλα τα πιθανά μονοπάτια που θα μπορούσαν να χρησιμοποιηθούν. Σε μια τέτοια περίπτωση, ο αλγόριθμος δημιουργίας μονοπατιού ο οποίος είναι υπεύθυνος για τη δημιουργία των μονοπατιών προς τους μη ελεγμένους ελεγκτές αδυνατεί να καθορίσει κάποιο μονοπάτι και η πηγή κατηγοριοποιεί το συγκεκριμένο ελεγκτή ως μη προσβάσιμο.

Όταν όμως υπάρχει πρόσβαση προς ένα μη ελεγμένο ελεγκτή, αυτή το πιο πιθανόν είναι να μπορεί να επιτευχθεί μέσω πολλών διαφορετικών μονοπατιών. Στόχος του αλγόριθμου δημιουργίας μονοπατιού είναι να εντοπίσει το βέλτιστο από αυτά τα μονοπάτια, δηλαδή το μονοπάτι που αν ακολουθηθεί, θα βοηθήσει στη γρηγορότερη κατηγοριοποίηση των μη ελεγμένων ελεγκτών από τον αλγόριθμο. Βεβαίως, υπάρχουν πολλοί διαφορετικοί αλγόριθμοι που μπορούν να καθορίσουν κάποιο μονοπάτι βάσει κριτηρίων και για αυτό το λόγο ο αλγόριθμος δημιουργίας μονοπατιού δίνεται ως παράμετρος στον αλγόριθμο μου. Ακόμη δύο παραμέτροι που λαμβάνει ο αλγόριθμος μου είναι οι μεθόδοι επιλογής NUC, οι οποίες καθορίζουν με πια σειρά οι μη-ελεγμένοι ελεγκτές θα επιλεχθούν ως προορισμοί και ο αλγόριθμος δρομολόγησης που χρησιμοποιείται από το συγκεκριμένο πλέγμα, έτσι ώστε να μπορέσει να υπολογιστεί σε ποιους ελεγκτές θα πρέπει να αποστέλλονται πακέτα οδοφράγματος για να διασφαλίζεται η σωστή δημιουργία των μονοπατιών.

Περισσότερες λεπτομέριες σχετικά με τις παραμέτρους και λαμβάνει ο αλγόριθμος και τις προυποθέσεις που πρέπει να πληρούν, δίνονται στα επόμενα υποκεφάλαια. Επίσης, παρουσιάζεται ο ψευδοκώδικας και η λεκτική περιγραφή του αλγορίθμου, το διάγραμμα καταστάσεων UML που προβάλει τις διάφορες καταστάσεις στις οποίες μπορεί να βρεθεί ο αλγόριθμος κατά την εκτέλεση του, αλλά και η απόδειξη ορθότητας και πληρότητας του αλγορίθμου.

3.2 Ψευδοκώδικας Αλγορίθμου

```
1   void faultIdentification(Set<Node> allNodes, int gridSize);
2
3   void faultIdentification(allNodes, gridSize){
4       read(nucSelectionMethod);
5       read(pathCreationAlgorithm);
6       read(routingAlgorithm);
7
8       Set<Node> uncheckedNodes = allNodes;
9       Set<Node> faultyNodes = empty;
10      Set<Node> workingNodes = empty;
11      Set<Node> unreachableNodes = empty;
12      Queue<Node> gatewayQueue = empty;
13
14      while(!uncheckedNodes.isEmpty()){
15          Node nodeUnderCheck = uncheckedNodes.getNUC(nucSelectionMethod);
16          List path = createPath(nodeUnderCheck, pathCreationAlgorithm);
17          if(path == undefined){
18              uncheckedNodes.remove(nodeUnderCheck);
19              unreachableNodes.add(nodeUnderCheck);
20          }else{
21              List<Node> roadblockNodes = calculateRoadblockNodes(routingAlgorithm);
22              for(int i:1 to roadblockNodes.size()){
23                  gatewayQueue.add(roadblockPacket(roadblockNodes.get(i)));
24              }
25              for(int i:1 to path.length*BUFFERS_PER_NODE){
26                  gatewayQueue.add(faultIdentificationPacket(nodeUnderCheck));
27              }
28              while((!gatewayQueue.isEmpty()) && (!gatewayTimeout())){
29                  if(FIRST_NODE.readyToReceive()){
30                      sendPacket(gatewayQueue.dequeue());
31                  }
32              }
33              if((gatewayQueue.isEmpty()) && (!gatewayTimeout())){
34                  uncheckedNodes.removeAll(path);
35                  workingNodes.addAll(path);
36                  flushGrid();
37              }else{
38                  int stackedPackets = packetsSent - roadblockPacketsArrived;
39                  int faultyNodeIndex = (stackedPackets/BUFFERS_PER_NODE) - 1;
40                  uncheckedNodes.remove(path.get(faultyNodeIndex));
41                  faultyNodes.add(path.get(faultyNodeIndex));
42                  for(int i:0 to faultyNodeIndex-1){
43                      uncheckedNodes.remove(path.get(i));
44                      workingNodes.add(path.get(i));
45                  }
46                  gatewayQueue = empty;
47                  flushGrid();
48              }
49          }
50      }
51  }
```

Σχήμα 3.3: Ψευδοκώδικας Αλγορίθμου

3.3 Λεκτική Περιγραφή Αλγορίθμου

1. Διάβασε τη μέθοδο επιλογής NUC, τον αλγόριθμο δημιουργίας μονοπατιού και τον αλγόριθμο δρομολόγησης που δίνονται από το χρήστη. *(Γραμμές 4-6)*
2. Δημιούργησε ένα σύνολο ελεγκτών, το Unchecked Nodes (UNC), το οποίο αρχικά περιέχει όλους τους ελεγκτές του πλέγματος. *(Γραμμή 8)*
3. Δημιούργησε τρία ακόμη σύνολα ελεγκτών, το Faulty Nodes (FN), το Unreachable Nodes (UNR) και το Working Nodes (WN), τα οποία είναι αρχικά άδεια. *(Γραμμές 9-11)*
4. Δημιούργησε μια ουρά ελεγκτών, την Gateway Queue (GQ), η οποία είναι αρχικά άδεια. *(Γραμμή 12)*
5. Ενόσω το UNC δεν είναι άδειο: *(Γραμμή 14)*
    a. Βάσει της μεθόδου επιλογής NUC, επέλεξε ένα κόμβο από το UNC και όρισε τον ως Node Under Check (NUC). *(Γραμμή 15)*
    b. Δημιούργησε ένα μονοπάτι από την πηγή στον NUC, χρησιμοποιώντας τον αλγόριθμο δημιουργίας μονοπατιού. Αν ο αλγόριθμος δημιουργίας μονοπατιού αδυνατεί να δημιουργήσει κάποιο μονοπάτι, αφαίρεσε τον NUC από το UNC και εισήγαγε τον στο UNR. Ακολούθως, επέστρεψε στην αρχή του βήματος 5. *(Γραμμές 16-19)*
    c. Υπολόγισε σε ποιους ελεγκτές θα πρέπει να σταλούν πακέτα οδοφράγματος έτσι ώστε να δημιουργηθεί το μονοπάτι, σύμφωνα με τον αλγόριθμο δρομολόγησης. Εισήγαγε τα πακέτα οδοφράγματος που πρέπει να σταλούν στους ελεγκτές αυτούς στη GQ, αρχίζοντας με το πακέτο που έχει ως προορισμό τον κόμβο που βρίσκεται πιο κοντά στην αρχή του μονοπατιού και τελειώνοντας με τον κόμβο που βρίσκεται πιο μακριά από την αρχή του μονοπατιού. *(Γραμμές 21-24)*
    d. Εισήγαγε (μήκος μονοπατιού * buffers ανά κόμβο) πακέτα αναγνώρισης σφαλμάτων με προορισμό τον NUC στη GQ. *(Γραμμές 25-27)*
    e. Ενόσω η GQ δεν είναι άδεια και δεν έχει παρουσιαστεί Gateway Time-out (GT) στην πηγή, έλεγξε κατά πόσο ο πρώτος ελεγκτής του πλέγματος είναι έτοιμος να παραλάβει το επόμενο πακέτο. Αν ο πρώτος ελεγκτής του πλέγματος είναι έτοιμος να παραλάβει το επόμενο πακέτο,

ανάκτησε το πρώτο πακέτο από την GQ και στείλε το σε αυτόν. *(Γραμμές 28-32)*

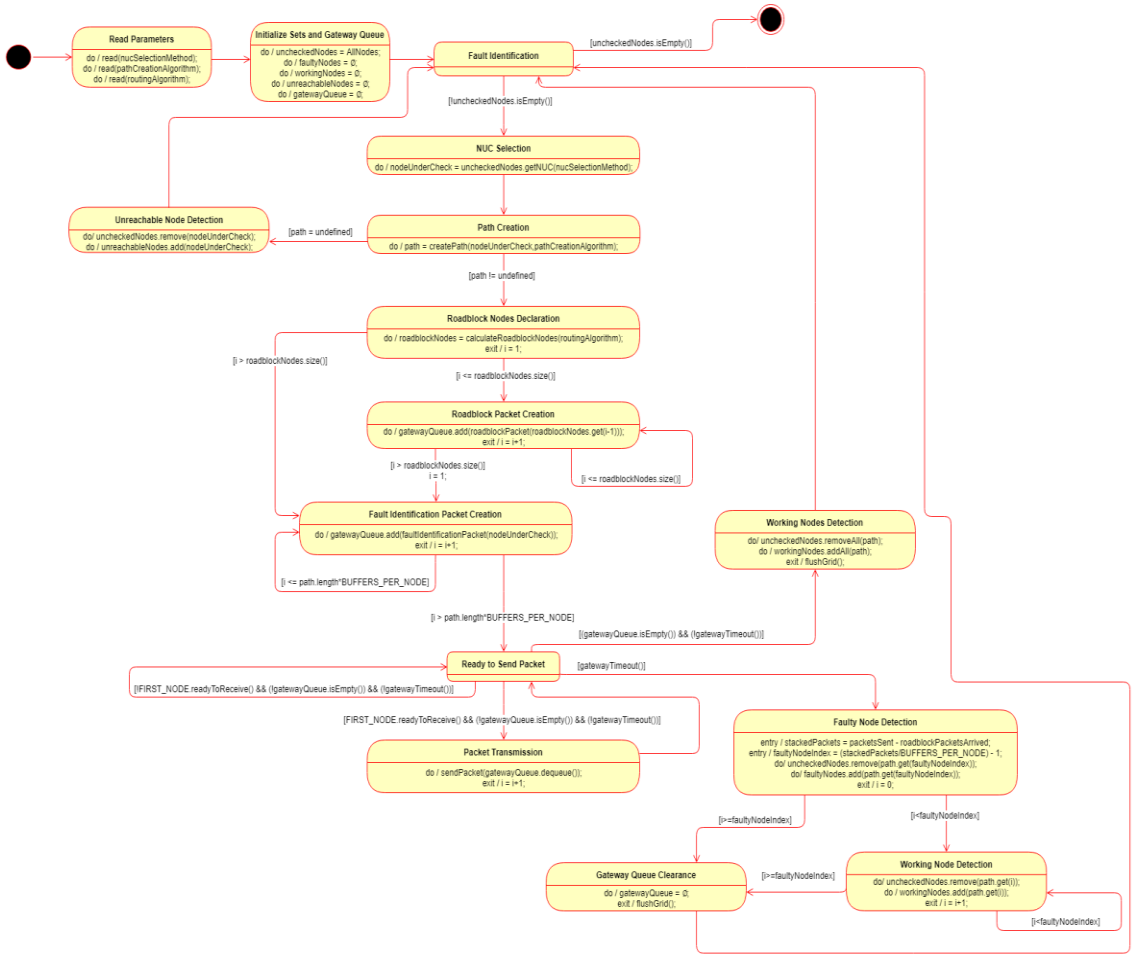
f. Εάν η GQ είναι άδεια και δεν παρουσιάστηκε GT στην πηγή, αφαίρεσε όλους τους ελεγκτές του μονοπατιού από το UNC και εισήγαγε τους στο WN. Ακολούθως, κάνε flush το πλέγμα και επέστρεψε στην αρχή του βήματος 5. *(Γραμμές 33-36)*

g. Εάν παρουσιάστηκε GT στην πηγή, υπολόγισε τον αριθμό ων πακέτων που έχουν παγιδευτεί στο πλέγμα (SP, SP = Ποσότητα όλων των πακέτων που έχουν σταλεί – Ποσότητα των πακέτων οδοφράγματος που έχουν φτάσει στον προορισμό τους). Ακολούθως, υπολόγισε ποιος είναι ο ελαττωματικός ελεγκτής (Δείκτης του ελαττωματικού κόμβου στο μονοπάτι = (SP/buffers ανά κόμβο)-1). Αφαίρεσε τον ελαττωματικό κόμβο από το UNC και εισήγαγε τον στο FN. Επιπρόσθετα, αφαίρεσε όλους τους ελεγκτές του μονοπατιού που προηγούνται του ελαττωματικού κόμβου από το UNC και εισήγαγε τους στο WN. Έπειτα, άδειασε την GQ, κάνε flush το πλέγμα και επέστρεψε στην αρχή του βήματος 5. *(Γραμμές 38-47)*

6. Το σύνολο FN περιλαμβάνει όλους τους μη λειτουργικούς ελεγκτές, το σύνολο UNR περιλαμβάνει όλους τους μη προσβάσιμους ελεγκτές και το σύνολο WN περιλαμβάνει όλους τους λειτουργικούς ελεγκτές του πλέγματος.

3.4 Διάγραμμα Καταστάσεων UML

Το διάγραμμα καταστάσεων UML προβάλει τις διάφορες καταστάσεις στις οποίες μπορεί να βρεθεί ο αλγόριθμος κατά την εκτέλεση του, αλλά και τις ενέργειες οι οποίες προκαλούν τη μεταβολή της κατάστασης στην οποία βρίσκεται κάθε στιγμή ο αλγόριθμος. Όπως διακρίνεται και από το διάγραμμα, ο τερματισμός του αλγορίθμου επιτυγχάνεται μόνο μετά το άδειασμα του συνόλου των μη ελεγμένων ελεγκτών.



Σχήμα 3.4: Διάγραμμα Καταστάσεων UML

3.5 Μέθοδοι Επιλογής NUC

Μια από τις παραμέτρους που δέχεται ως είσοδο ο αλγόριθμος μου είναι η μέθοδος επιλογής NUC. Σε κάθε επανάληψη του βρόγχου του αλγορίθμου (γραμμές 14-50), επιλέγεται ένας μη ελεγμένος ελεγκτής και τίθεται ως Node Under Check (NUC), μέχρι το τέλος της συγκεκριμένης επανάληψης. Ο NUC, είναι ο ελεγκτής που θα χρησιμοποιηθεί ως προορισμός για τα πακέτα αναγνώρισης σφαλμάτων που θα αποσταλούν από τον αλγόριθμο. Η μέθοδος επιλογής NUC είναι αυτή που καθορίζει τη σειρά με την οποία οι μη ελεγμένοι ελεγκτές θα οριστούν ως NUC. Συνολικά δημιούργησα είκοσι μεθόδους επιλογής NUC, τις οποίες χρησιμοποίησα κατά τη διεκπεραίωση των προσομοιώσεων μου.

Οι μέθοδοι επιλογής NUC 9 μέχρι 16, προβαίνουν σε υπολογισμό είτε της Ευκλείδειας απόστασης, είτε της απόστασης Μανχάταν μεταξύ του κάθε κόμβου του πλέγματος και ενός καθορισμένου σημείου στο πλέγμα. Ακολούθως, οι συγκεκριμένες μέθοδοι επιλογής NUC ταξινομούν τους ελεγκτές με αύξουσα ή φθίνουσα σειρά βάσει των αποστάσεων που έχουν υπολογιστεί και καθορίζουν με αυτό τον τρόπο τη σειρά με την οποία οι κόμβοι θα τεθούν ως NUC.

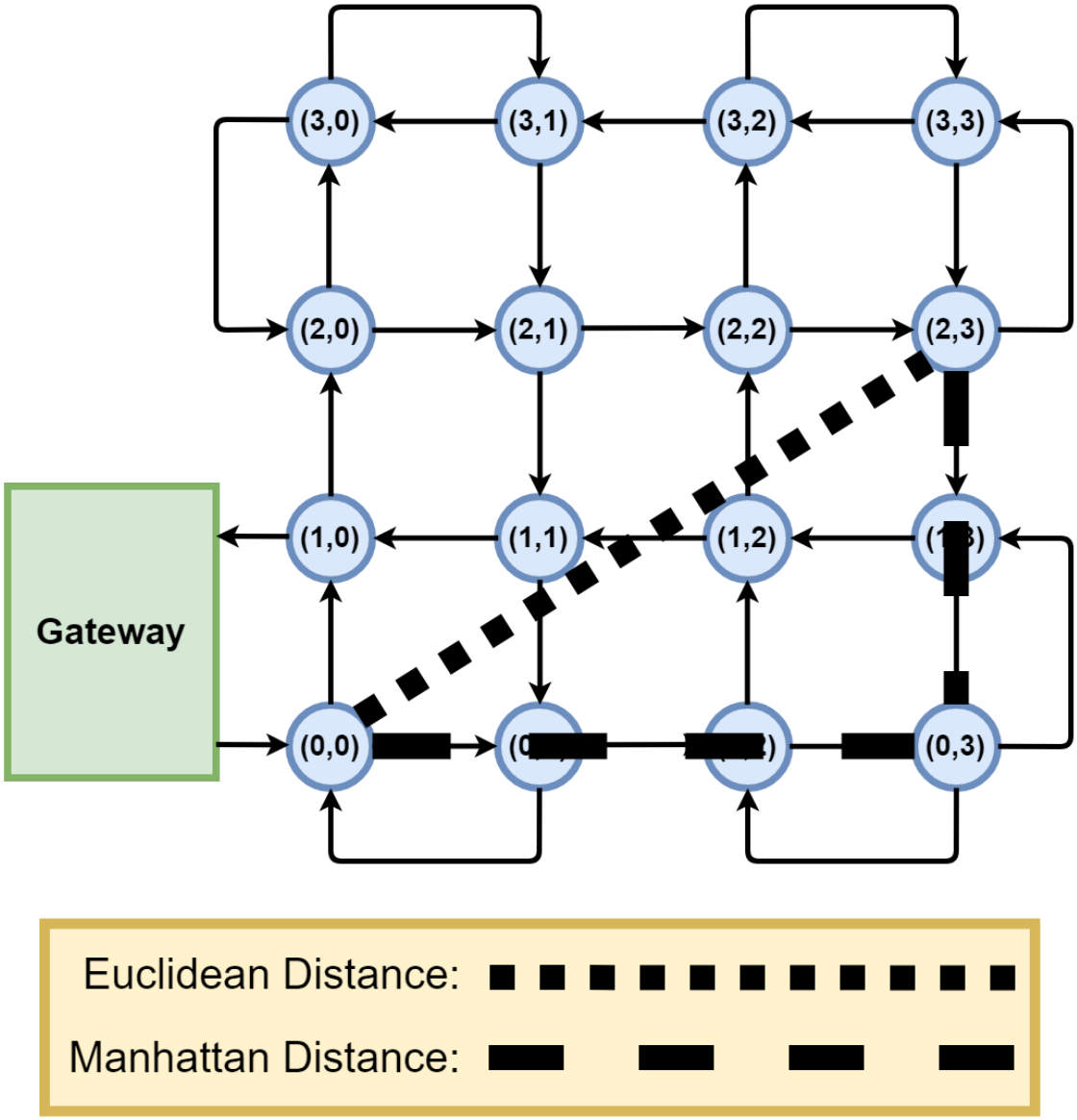
Η Ευκλείδεια απόσταση που χρησιμοποιείται από κάποιες εξ αυτών των μεθόδων, ουσιαστικά υπολογίζει το πόσο απέχουν οι καρτεσιανές συντεταγμένες δύο σημείων στο πλέγμα. Δηλαδή, μετρά το μήκος ενός ιδεατού ευθύγραμμου τμήματος το οποίο αν υπήρχε θα ένωνε τα συγκεκριμένα σημεία. Η απόσταση Μανχάταν είναι μια διαφορετική μετρική, κατά την οποία η απόσταση μεταξύ δύο σημείων είναι ίση με το άθροισμα των απόλυτων διαφορών των καρτεσιανών τους συντεταγμένων. Στην πράξη, υπολογίζοντας την απόσταση Μανχάταν μεταξύ δύο σημείων του πλέγματος, αυτό που μας επιστρέφεται είναι ο ελάχιστος αριθμός συνδέσεων (χωρίς να λαμβάνεται υπόψη η κατεύθυνση μετάδοσης των συνδέσεων) που χρειάζεται να χρησιμοποιηθούν για να ενωθούν τα συγκεκριμένα σημεία του πλέγματος.

Εάν τα σημεία του πλέγματος c1 και c2 έχουν συντεταγμένες (x1,y1) και (x2,y2) αντίστοιχα, η Ευκλείδεια απόσταση μεταξύ των σημείων δίνεται από τον τύπο:

$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$

και η απόσταση Manhattan μεταξύ των σημείων δίνεται από τον τύπο:

d = $|x1 - x2| + |y1 - y2|$



Σχήμα 3.6: Η γραφική απεικόνιση της Ευκλείδειας απόστασης και της απόστασης Manhattan μεταξύ των ελεγκτών (0,0) και (2,3) σε ένα πλέγμα μεγέθους 4x4.

Στον πίνακα 3.1 παρουσιάζονται αναλυτικά οι είκοσι μεθόδου επιλογής NUC που έχω δημιουργήσει.

Μέθοδος Επιλογής NUC	Περιγραφή
1	Η επιλογή ξεκινά από την κάτω-αριστερή γωνία του πλέγματος και συνεχίζεται από τα αριστερά προς τα δεξιά για κάθε σειρά ελεγκτών.
2	Η επιλογή ξεκινά από την κάτω-δεξιά γωνία του πλέγματος και συνεχίζεται από τα δεξιά προς τα αριστερά για κάθε σειρά ελεγκτών.
3	Η επιλογή ξεκινά από την πάνω-αριστερή γωνία του πλέγματος και συνεχίζεται από τα αριστερά προς τα δεξιά για κάθε σειρά ελεγκτών.
4	Η επιλογή ξεκινά από την πάνω-δεξιά γωνία του πλέγματος και συνεχίζεται από τα δεξιά προς τα αριστερά για κάθε σειρά ελεγκτών.
5	Η επιλογή ξεκινά από την αριστερή-κάτω γωνία του πλέγματος και συνεχίζεται από τα κάτω προς τα πάνω για κάθε στήλη ελεγκτών.
6	Η επιλογή ξεκινά από την αριστερή-πάνω γωνία του πλέγματος και συνεχίζεται από τα πάνω προς τα κάτω για κάθε στήλη ελεγκτών.
7	Η επιλογή ξεκινά από την δεξιά-κάτω γωνία του πλέγματος και συνεχίζεται από τα κάτω προς τα πάνω για κάθε στήλη ελεγκτών.
8	Η επιλογή ξεκινά από την δεξιά-πάνω γωνία του πλέγματος και συνεχίζεται από τα πάνω προς τα κάτω για κάθε στήλη ελεγκτών.
9	Επιλέγει τους ελεγκτές βάσει της Ευκλείδειας απόστασης που έχουν από το κέντρο του πλέγματος, με αύξουσα σειρά.
10	Επιλέγει τους ελεγκτές βάσει της απόστασης Μανχάταν που έχουν από το κέντρο του πλέγματος, με αύξουσα σειρά.
11	Επιλέγει τους ελεγκτές βάσει της Ευκλείδειας απόστασης που έχουν από το κέντρο του πλέγματος, με φθίνουσα σειρά.
12	Επιλέγει τους ελεγκτές βάσει της απόστασης Μανχάταν που έχουν από το κέντρο του πλέγματος, με φθίνουσα σειρά.
13	Επιλέγει τους ελεγκτές βάσει της Ευκλείδειας απόστασης που έχουν από την πηγή, με αύξουσα σειρά.

14	Επιλέγει τους ελεγκτές βάσει της Ευκλείδειας απόστασης που έχουν από την πηγή, με φθίνουσα σειρά.
15	Επιλέγει τους ελεγκτές βάσει της απόστασης Μανχάταν που έχουν από την πηγή, με αύξουσα σειρά.
16	Επιλέγει τους ελεγκτές βάσει της απόστασης Μανχάταν που έχουν από την πηγή, με φθίνουσα σειρά.
17	Αρχικά, υπολογίζει το ελάχιστο κόστος που απαιτείται για τη μετάδοση από την πηγή σε κάθε κόμβο του πλέγματος, χρησιμοποιώντας τον αλγόριθμο εύρεσης βραχύτερου μονοπατιού του Dijkstra [4]. Ακολούθως, επιλέγει τους ελεγκτές βάσει του ελάχιστου κόστους που έχει υπολογιστεί, με αύξουσα σειρά.
18	Αρχικά, υπολογίζει το ελάχιστο κόστος που απαιτείται για τη μετάδοση από την πηγή σε κάθε κόμβο του πλέγματος, χρησιμοποιώντας τον αλγόριθμο εύρεσης βραχύτερου μονοπατιού του Dijkstra. Ακολούθως, επιλέγει τους ελεγκτές βάσει του ελάχιστου κόστους που έχει υπολογιστεί, με φθίνουσα σειρά.
19	Τυχαία επιλογή ελεγκτών.
20	Η μέθοδος επιλογής που χρησιμοποιείται για τη λειτουργία του Αλγόριθμου Δημιουργίας Μονοπατιού 5. Αν το σύνολο FN είναι άδειο, τότε θέτει ως NUC τον τελευταίο κόμβο του προκαθορισμένου μονοπατιού. Διαφορετικά, θέτει ως NUC τον πρώτο κόμβο του προκαθορισμένου μονοπατιού που ανήκει στο σύνολο UNC.

Πίνακας 3.1: Μέθοδοι επιλογής NUC

3.6 Αλγόριθμοι Δημιουργίας Μονοπατιού

Μια ακόμη παράμετρος που λαμβάνεται υπόψη από τον αλγόριθμο αναγνώρισης σφαλμάτων, είναι ο αλγόριθμος δημιουργίας μονοπατιού. Ο αλγόριθμος δημιουργίας μονοπατιού είναι υπεύθυνος για να δημιουργήσει το επόμενο μονοπάτι του οποίου οι ελεγκτές θα ελέγχουν κατά πόσο είναι λειτουργικοί ή μη λειτουργικοί. Τα μονοπάτια που δημιουργούνται από κάποιο αλγόριθμο δημιουργίας μονοπατιού πρέπει πάντοτε να πληρούν τις εξής προδιαγραφές:

- Να αποτελούνται αποκλειστικά από λειτουργικούς και μη ελεγμένους ελεγκτές

- Να περιέχουν τουλάχιστον ένα μη ελεγμένο κόμβο, τον NUC

- Να μην περιέχουν κύκλους

- Να είναι πεπερασμένα

Οποιοσδήποτε αλγόριθμος δημιουργεί μονοπάτια ελεγκτών που πληρούν τις πιο πάνω προδιαγραφές, τερματίζει πάντα και αδυνατεί να καθορίσει κάποιο μονοπάτι αν και μόνο αν δεν υπάρχει κανένα μονοπάτι προς τον NUC που να αποτελείται αποκλειστικά από λειτουργικούς και μη ελεγμένους ελεγκτές, θεωρείται ένας έγκυρος αλγόριθμος δημιουργίας μονοπατιού. Συνολικά δημιούργησα πέντε αλγόριθμους δημιουργίας μονοπατιού τους οποίους χρησιμοποίησα για τη διεκπεραίωση των προσομοιώσεων μου. Οι πέντε αυτοί αλγόριθμοι βασίζονται είτε στον αλγόριθμο εύρεσης βραχύτερου μονοπατιού του Dijkstra [4], είτε σε μια παραλλαγή του αλγορίθμου Bellman-Ford [2, 5] που αποφεύγει τους κύκλους αρνητικού βάρους. Ακολουθεί η παρουσίαση των εν λόγω αλγορίθμων:

Αλγόριθμος 1:

Αλγόριθμος του Dijkstra που λαμβάνει υπόψη μόνο τους λειτουργικούς και τους μη ελεγμένους ελεγκτές. Όλες οι ακμές του πλέγματος θεωρείται ότι έχουν βάρη ίσα με 1.

Αλγόριθμος 2:

Αλγόριθμος του Dijkstra που λαμβάνει υπόψη μόνο τους λειτουργικούς και τους μη ελεγμένους ελεγκτές. Οι ακμές προς τους μη ελεγμένους ελεγκτές θεωρείται ότι έχουν

βάρη ίσα με 1 και οι ακμές προς τους λειτουργικούς ελεγκτές θεωρείται ότι έχουν βάρη ίσα με |Ελεγκτές πλέγματος|+1.

Αλγόριθμος 3:

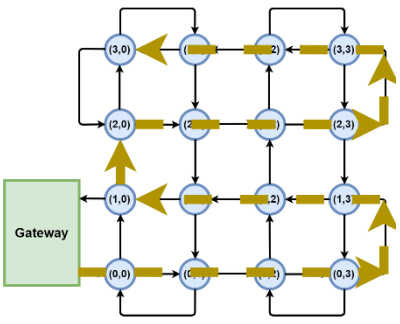
Έκδοση του Bellman-Ford που αποφεύγει τους κύκλους αρνητικού βάρους. Λαμβάνει υπόψη μόνο τους λειτουργικούς και τους μη ελεγμένους ελεγκτές. Οι ακμές προς τους μη ελεγμένους ελεγκτές θεωρείται ότι έχουν βάρη ίσα με -1 και οι ακμές προς τους λειτουργικούς ελεγκτές θεωρείται ότι έχουν βάρη ίσα με |Ελεγκτές πλέγματος|+1.

Αλγόριθμος 4:

Έκδοση του Bellman-Ford που αποφεύγει τους κύκλους αρνητικού βάρους. Λαμβάνει υπόψη μόνο τους λειτουργικούς και τους μη ελεγμένους ελεγκτές. Οι ακμές προς τους μη ελεγμένους ελεγκτές θεωρείται ότι έχουν βάρη ίσα με -|Ελεγκτές πλέγματος|-1 και οι ακμές προς τους λειτουργικούς ελεγκτές θεωρείται ότι έχουν βάρη ίσα με 1.

Αλγόριθμος 5:

Ο αλγόριθμος δημιουργεί ένα προκαθορισμένο μονοπάτι που περιέχει όλους τους ελεγκτές του πλέγματος. Εάν ο NUC είναι ο τελευταίος ελεγκτής του μονοπατιού, ο αλγόριθμος επιστρέφει το προκαθορισμένο μονοπάτι. Διαφορετικά, χρησιμοποιείται ο αλγόριθμος 1 για να δημιουργήσει ένα μονοπάτι προς τον NUC. Αν το συγκεκριμένο μονοπάτι προς τον NUC δεν περιέχει κάποιο κοινό κόμβο με το κομμάτι του προκαθορισμένου μονοπατιού που έπεται του NUC, τα δύο αυτά μονοπάτια συνενώνονται και ο αλγόριθμος επιστρέφει τη συνένωση τους. Διαφορετικά, ο αλγόριθμος επιστρέφει το μονοπάτι προς τον NUC που δημιουργήθηκε από τον αλγόριθμο 1.



Σχήμα 3.7: Το προκαθορισμένο μονοπάτι που καθορίζεται από τον αλγόριθμο δημιουργίας μονοπατιού 5

Ο αλγόριθμος 1 αποτέλεσε την πρωταρχική μου ιδέα για τη δημιουργία ενός σύντομου σε μήκους μονοπατιού προς τον NUC. Ακολούθως, παρατήρησα ότι πολλά από τα μονοπάτια που πρόκυπταν περιείχαν μεγάλο αριθμό λειτουργικών ελεγκτών και πιο μικρό αριθμό μη ελεγμένων ελεγκτών. Κάτι τέτοιο δεν είναι επιθυμητό καθώς μέσω της κάθε δημιουργίας μονοπατιού ο στόχος είναι να επιτυγχάνεται ο έλεγχος όσων περισσότερων μη ελεγμένων ελεγκτών γίνεται, έτσι ώστε ο αλγόριθμος να αποκτά γρήγορα επιπλέον γνώση. Έτσι, αποφάσισα να αποτρέψω την περίληψη λειτουργικών ελεγκτών μέσα στα μονοπάτια που δημιουργούνται και έτσι δημιούργησα τον αλγόριθμο 2, ο οποίος αποθαρρύνει την περίληψη λειτουργικών ελεγκτών, προσδίδοντας της επιπλέον κόστος. Έπειτα, σκέφτηκα ότι πέρα από το να αποθαρρύνω την περίληψη λειτουργικών ελεγκτών στα μονοπάτια, θα μπορούσα να βρω κάποιο τρόπο για να ενθαρρύνω την περίληψη μη ελεγμένων ελεγκτών σε αυτά. Με αυτή την σκέψη δημιούργησα τον αλγόριθμο 3. Ακολούθως, συμπέρανα ότι αυτό που κάνει ο αλγόριθμος 3 βάσει των βαρών που θέτει, είναι να αποθαρρύνει έντονα την περίληψη λειτουργικών ελεγκτών στα μονοπάτια και να ενθαρρύνει ελαφρώς την περίληψη μη ελεγμένων ελεγκτών σε αυτά. Έτσι, αποφάσισα να δημιουργήσω και την αντίστροφη προσέγγιση, δηλαδή ένα αλγόριθμο που να αποθαρρύνει ελαφρώς την περίληψη λειτουργικών ελεγκτών στα μονοπάτια και να ενθαρρύνει έντονα την περίληψη μη ελεγμένων ελεγκτών σε αυτά. Συνεπώς, κατέληξα στον αλγόριθμο 4. Ο αλγόριθμος 5 αποτελεί μια εναλλακτική και πολύ ιδιάζουσα προσέγγιση η οποία βασίζεται στη δημιουργία ενός προκαθορισμένου μονοπατιού που περιλαμβάνει όλους τους ελεγκτές του πλέγματος και χρησιμοποιείται καθ’ όλη τη διάρκεια εκτέλεσης του αλγορίθμου αναγνώρισης σφαλμάτων.

3.7 Αλγόριθμοι Δρομολόγησης

Η τρίτη παράμετρος που λαμβάνει ο αλγόριθμος αναγνώρισης σφαλμάτων είναι ο αλγόριθμος που χρησιμοποιείται για τη δρομολόγηση των πακέτων στο πλέγμα. Αναλόγως του αλγόριθμου δρομολόγησης που χρησιμοποιείται, ο αλγόριθμος αναγνώρισης σφαλμάτων ορίζει κάθε φορά διαφορετικούς ελεγκτές ως ελεγκτές οδοφράγματος, έτσι ώστε να σχηματίζεται με το σωστό τρόπο κάθε φορά το μονοπάτι προς τον NUC. Συγκεκριμένα, ο κάθε αλγόριθμος δρομολόγησης έχει διαφορετική πολυπλοκότητα και χαρακτηριστικά και υλοποιείται μέσω των αποφάσεων που λαμβάνουν οι ελεγκτές του πλέγματος, οι οποίοι αποτελούν ένα κατανεμημένο σύστημα. Συνεπώς, όταν δεν είμαστε σίγουροι αν κάποιος ελεγκτής θα στείλει το πακέτο στην επιθυμητή έξοδο έτσι ώστε να δημιουργηθεί το μονοπάτι που θέλουμε, πρέπει να μετατρέπουμε το συγκεκριμένο κόμβο σε κόμβο οδοφράγματος για να το διασφαλίσουμε.

Στις προσομοιώσεις που διενήργησα χρησιμοποίησα συνολικά δύο αλγόριθμους δρομολόγησης, το XY ο οποίος έχει προταθεί από τους Zhang et al. [10] (Σχήμα 2.14) και το Linear Motion ο οποίος είναι ένας πιο απλός αλγόριθμος δρομολόγησης που δημιούργησα εγώ (Σχήμα 3.12). Ο XY βασίζεται σε δύο κύριες παραμέτρους για να καθορίσει την έξοδο του κόμβου που θα επιλεγεί κάθε φορά, στις συντεταγμένες του συγκεκριμένου κόμβου στο πλέγμα και στις συντεταγμένες του κόμβου προορισμού του πακέτου. Μια τρίτη παράμετρος που χρησιμοποιείται από το XY είναι ο τύπος περιστροφής του συγκεκριμένου κόμβου, ο οποίος όμως μπορεί εύκολα να υπολογιστεί εάν γνωρίζουμε τις συντεταγμένες του κόμβου στο πλέγμα. Συνεπώς, ο XY είναι δυνατό να λάβει διαφορετική απόφαση σχετικά την έξοδο που θα επιλεγεί για δύο πακέτα που θα εισέλθουν σε ένα κόμβο από την ίδια είσοδο, αναλόγως των συντεταγμένων του κόμβου προορισμού των πακέτων αυτών.

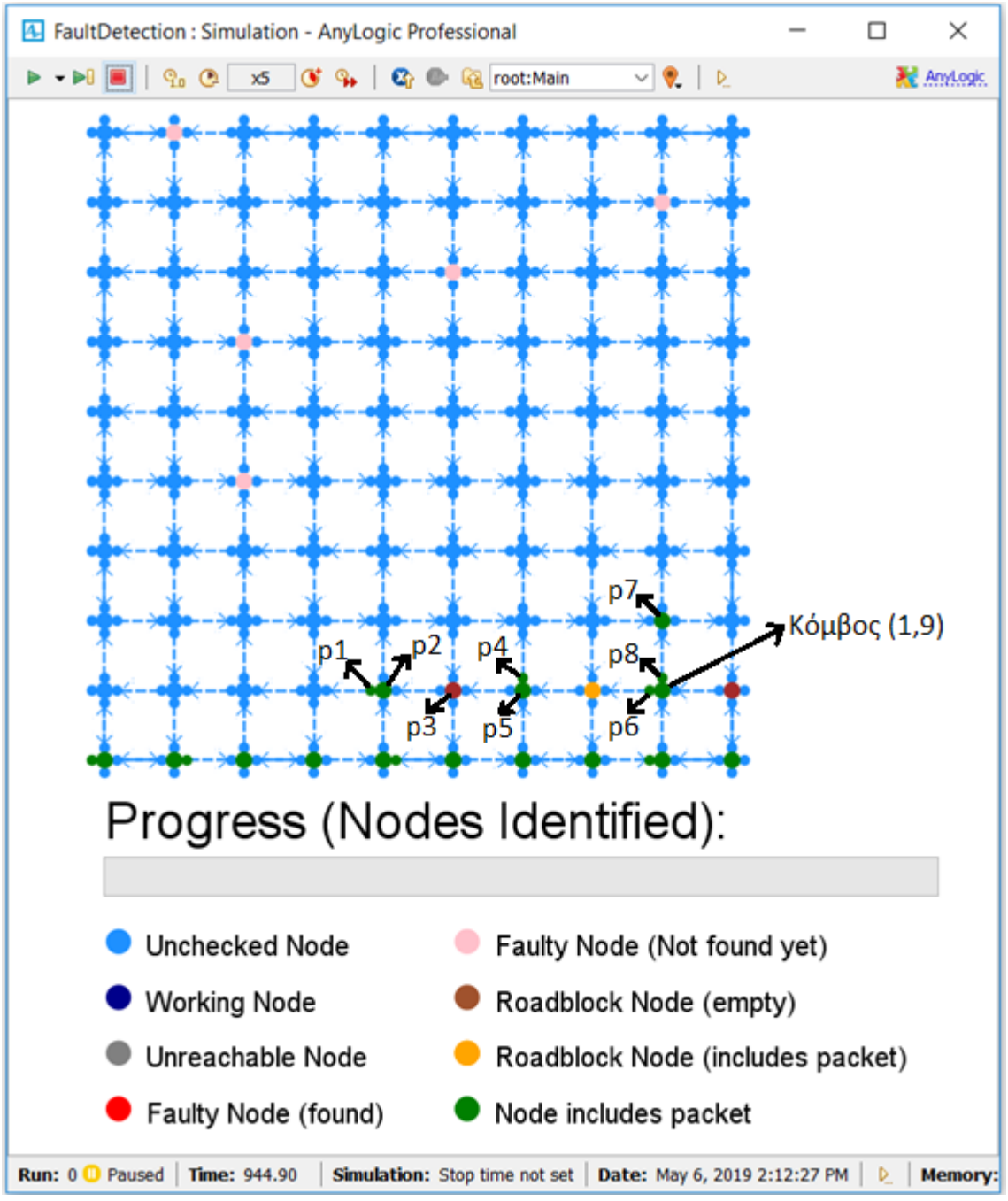
Αντιθέτως, ο Linear Motion βασίζεται αποκλειστικά σε μια παράμετρο για να καθορίσει την έξοδο του κόμβου που θα επιλεγεί κάθε φορά και αυτή είναι η είσοδος του κόμβου που χρησιμοποιήθηκε για την εισδοχή του πακέτου στον κόμβο. Συνεπώς, αν χρησιμοποιείται ο Linear Motion και εισέλθουν δύο πακέτα από την ίδια είσοδο σε ένα κόμβο, τότε θα γνωρίζουμε ότι τα δύο πακέτα ακολούθως θα εξέλθουν από την ίδια

έξοδο του κόμβου, ανεξαρτήτως των συντεταγμένων των ελεγκτών προορισμού των συγκεκριμένων πακέτων.

Το συγκεκριμένο γεγονός αποτελεί μια ένδειξη ότι ίσως η χρησιμοποίηση του Linear Motion να επιφέρει μικρότερο κόστος κατά την εκτέλεση του αλγορίθμου αναγνώρισης σφαλμάτων, καθώς στην περίπτωση χρησιμοποίησης του XY, η σωστή δρομολόγηση των πακέτων μπορεί να επιτευχθεί μόνο όταν προηγηθεί μετατροπή όλων των ελεγκτών του μονοπατιού που θέλουμε να σχηματίσουμε σε ελεγκτές οδοφράγματος. Αντιθέτως, λόγω των ντετερμινιστικών αποφάσεων δρομολόγησης που λαμβάνουν οι ελεγκτές όταν χρησιμοποιείται ο Linear Motion, δεν απαιτείται κάθε φορά η μετατροπή όλων των ελεγκτών του μονοπατιού ως ελεγκτών οδοφράγματος, παρά μόνο αυτών που ορίζουν κάποια έξοδο που δε συνάδει με την κατεύθυνση δημιουργίας του μονοπατιού.

Σε περίπτωση που επιχειρήσουμε να χρησιμοποιήσουμε το XY, χωρίς να θέσουμε όλους τους ελεγκτές του μονοπατιού ως ελεγκτές οδοφράγματος, τότε θα παρατηρηθεί το πρόβλημα που παρουσιάζεται στο Σχήμα 3.8. Στο συγκεκριμένο στιγμιότυπο μιας προσομοίωσης κατά την οποία χρησιμοποιείται ο αλγόριθμος XY, ο ελεγκτής (1,9) δεν έχει καθοριστεί ως ελεγκτής οδοφράγματος. Συνεπώς, η δρομολόγηση των πακέτων p1, p2, p3, p4, p5 και p6 γίνεται με το σωστό τρόπο, προς τα αριστερά, βάσει της κατεύθυνσης που καθορίζει το επιθυμητό μονοπάτι. Εντούτοις, μερικά άλλα πακέτα, όπως το p7 και το p8 έχουν παρεκκλίνει από το μονοπάτι και ακολουθούν μια διαφορετική διαδρομή προς τον κόμβο προορισμού τους. Αυτό συμβαίνει διότι ενώ ο προορισμός των πακέτων p1, p2, p3, p4, p5 και p6 είναι αριστερότερα από τον κόμβο (1,9) και δε χρειάζεται να τοποθετηθεί κάποιο οδόφραγμα για να τα διατηρήσει στο επιθυμητό μονοπάτι, κάτι τέτοιο δεν ισχύει για τα πακέτα p7 και p8. Τα πακέτα αυτά έχουν προορισμό στην ίδια στήλη ή δεξιότερα από τον κόμβο (1,9) και έτσι ο ελεγκτής χρησιμοποιώντας τον XY τα δρομολογεί προς τα πάνω, εκτός του επιθυμητού μονοπατιού. Άρα, ο μόνος τρόπος για διασφαλιστεί η διατήρηση των πακέτων στο επιθυμητό μονοπάτι καθώς χρησιμοποιείται ο XY, είναι με τη μετατροπή όλων των ελεγκτών σε ελεγκτές οδοφράγματος. Σε περίπτωση που χρησιμοποιούσαμε το Linear Motion το φαινόμενο που παρουσιάζεται στο Σχήμα 3.8 θα είχε αποφευχθεί, λόγω της

ανεξαρτησίας του αλγόριθμου δρομολόγησης από τον προορισμό του πακέτου. Συγκεκριμένα, εφόσον το p1 δρομολογηθεί σωστά από τον κόμβο (1,9), τότε γνωρίζουμε ότι το ίδιο θα συμβεί και για όλα τα επόμενα πακέτα, αφού όλα τα πακέτα εισέρχονται στον κόμβο (1,9) χρησιμοποιώντας την ίδια είσοδο.



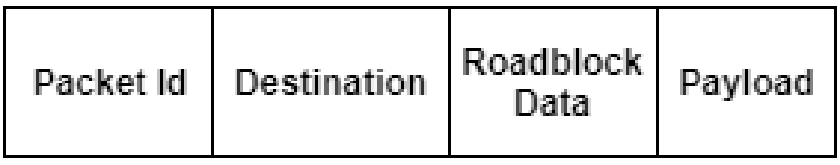
Σχήμα 3.8: Αναπαράσταση του προβλήματος που προκύπτει λόγω διαφορετικών προορισμών πακέτων όταν χρησιμοποιείται ο XY χωρίς να καθορίζονται όλοι οι ελεγκτές ως ελεγκτές οδοφράγματος.

Εντούτοις, ο αλγόριθμος δρομολόγησης XY είναι ο βασικός αλγόριθμος που χρησιμοποιείται στο ερευνητικό πρόγραμμα Visorsurf και η ενσωμάτωση του στη

υπερεπιφάνεια έχει ήδη ολοκληρωθεί. Αντιθέτως, η υπερεπιφάνεια δεν υποστηρίζει στο παρών στάδιο το Linear Motion και η ενσωμάτωση του θεωρείται δύσκολη λόγω των περιορισμών που υπάρχουν. Έτσι, αποφάσισα να δημιουργήσω μια μικρή παραλλαγή του XY, στην οποία απαλείφεται το πρόβλημα της εξάρτησης της επιλογής εξόδου των ελεγκτών από τις συντεταγμένες του κόμβου προορισμού.

Στην τρίτη εναλλακτική επιλογή αλγόριθμου δρομολόγησης έδωσα το όνομα XY με εκτενή πακέτα (Σχήμα 3.11), καθώς κατά τη χρησιμοποίηση του εν λόγω αλγόριθμου δρομολόγησης, τα πακέτα οδοφράγματος χρειάζεται να έχουν μεγαλύτερο μέγεθος από το κανονικό. Συγκεκριμένα, ενώ όλα τα πακέτα που μεταδίδονται στην υπερεπιφάνεια έχουν τη δομή που φαίνεται στο Σχήμα 3.9 κατά τη χρήση των υπόλοιπων αλγόριθμων δρομολόγησης, κατά τη χρησιμοποίηση του XY με εκτενή πακέτα χρησιμοποιείται μια εκτενής δομή πακέτου για τα πακέτα οδοφράγματος, η οποία παρουσιάζεται στο Σχήμα 3.10. Στη δομή αυτή, πέρα του βασικού προορισμού που περιλαμβάνεται σε όλα τα πακέτα, περιλαμβάνεται και ένα δεύτερο ζευγάρι συντεταγμένων κόμβου του πλέγματος το οποίο ονομάζεται κρυφός προορισμός.

Ο XY με εκτενή πακέτα λειτουργεί θέτοντας ως κόμβο προορισμού όλων των πακέτων τον τελευταίο κόμβο του μονοπατιού. Εφόσον λοιπόν όλα τα πακέτα έχουν τον ίδιο προορισμό, όταν δύο πακέτα εισέλθουν από την ίδια είσοδο σε ένα κόμβο, τότε γνωρίζουμε ότι τα πακέτα αυτά θα εξέλθουν από την ίδια έξοδο του κόμβου, εφόσον οι δύο παράμετροι που καθορίζουν την έξοδο που είναι οι συντεταγμένες του κόμβου προορισμού των πακέτων και οι συντεταγμένες του κόμβου στο πλέγμα θα



Σχήμα 3.9: Η δομή των πακέτων που μεταδίδονται στην υπερπιφάνεια

Σχήμα 3.10: Η εκτενής δομή των πακέτων οδοφράγματος που μεταδίδονται στην υπερεπιφάνεια όταν χρησιμοποιείται ο αλγόριθμος δρομολόγησης XY με εκτενή πακέτα

παραμείνουν σταθερές. Συνεπώς, με αυτό τον τρόπο αίρεται η εξάρτηση της επιλογής εξόδου των ελεγκτών από τις συντεταγμένες του κόμβου προορισμού.

Εντούτοις, μπορεί να ισχύει για όλα τα πακέτα αναγνώρισης σφαλμάτων ότι έχουν ως προορισμό τον τελευταίο κόμβο του μονοπατιού, όμως αυτό είναι κάτι που δεν ισχύει για τα πακέτα οδοφράγματος που έχουν ως προορισμό τους ελεγκτές οδοφράγματος. Για να λειτουργήσει λοιπόν ο αλγόριθμος XY με εκτενή πακέτα, μέσα στα πακέτα οδοφράγματος χρειάζεται να εισαχθεί και ένα δεύτερο ζευγάρι συντεταγμένων, δηλαδή ο κρυφός προορισμός. Όταν κάποιος ελεγκτής λαμβάνει ένα πακέτο οδοφράγματος, χρησιμοποιεί αρχικά τον κρυφό προορισμό για να ελέγξει αν ο συγκεκριμένος ελεγκτής είναι ο πραγματικός προορισμός του πακέτου. Σε περίπτωση που ο συγκεκριμένος ελεγκτής δεν είναι ο πραγματικός προορισμός του πακέτου, τότε δρομολογεί κανονικά το πακέτο βάσει του φανερού του προορισμού, που είναι πάντοτε ο τελευταίος ελεγκτής του μονοπατιού. Συνεπώς, κατά τη χρησιμοποίηση του αλγόριθμου δρομολόγησης XY με εκτενή πακέτα, δεν απαιτείται κάθε φορά η μετατροπή όλων των ελεγκτών του μονοπατιού ως ελεγκτών οδοφράγματος όπως συμβαίνει όταν χρησιμοποιείται ο κανονικός XY, παρά μόνο των ελεγκτών που ορίζουν κάποια έξοδο που δε συνάδει με την κατεύθυνση δημιουργίας του μονοπατιού. Αντιθέτως, το αυξημένο μέγεθος που έχουν τα πακέτα οδοφράγματος κατά τη χρησιμοποίηση του XY με εκτενή πακέτα αποτελεί ένα σοβαρό μειονέκτημα καθώς τέτοια πακέτα θα προκαλούν μεγαλύτερες καθυστερήσεις μετάδοσης στο πλέγμα. Επίσης, είναι πολύ πιθανόν η ύπαρξη τέτοιων πακέτων να μην μπορεί να υποστηριχτεί από τη υπερεπιφάνεια και συνεπώς ο συγκεκριμένος αλγόριθμος να μην είναι δυνατό να χρησιμοποιηθεί στην πράξη.

```
if((packet.xHidden==node.x) && (packet.yHidden==node.y)){
    return destination_found;
}
switch (node.type) {
    case 0:
        if(packet.xDest<node.x){
            return output2;
        }else if(packet.yDest>=node.y){
            return output1;
        }else{
            return output2;
        }
    case 1:
        if(packet.xDest<node.x){
            return output1;
        }else if(packet.yDest>node.y){
            return output1;
        }else{
            return output2;
        }
    case 2:
        if(packet.xDest>node.x){
            return output2;
        }else if(packet.yDest<=node.y){
            return output1;
        }else{
            return output2;
        }
    case 3:
        if(packet.xDest>node.x){
            return output1;
        }else if(packet.yDest<node.y){
            return output1;
        }else{
            return output2;
        }
}
```

Σχήμα 3.11: Ψευδοκώδικας αλγορίθμου XY με εκτενή πακέτα

```
if((packet.xDest==node.x) && (packet.yDest==node.y)){
    return destination_found;
}
if(node.input_used == input1){
    return output1;
}
if(node.input_used == input2){
    return output2;
}
```

Σχήμα 3.12: Ψευδοκώδικας αλγορίθμου Linear Motion

3.8 Απόδειξη Ορθότητας και Τερματισμού Αλγορίθμου

Στόχος του αυτού του υποκεφαλαίου, είναι να αποδειχτεί ότι ο αλγόριθμος που έχω σχεδιάσει είναι ορθός και ότι καταλήγει πάντα σε σωστή κατηγοριοποίηση των ελεγκτών βάσει της λειτουργικότητας τους. Συγκεκριμένα, μετά το τέλος της εκτέλεσης του αλγορίθμου πρέπει πάντοτε το σύνολο των λειτουργικών ελεγκτών να περιέχει όλους και μόνο τους ελεγκτές του πλέγματος που έχουν την επιθυμητή συμπεριφορά και δεν επηρεάζονται από την παρουσία κάποιου σφάλματος, το σύνολο των μη λειτουργικών ελεγκτών να περιέχει όλους και μόνο τους ελεγκτές οι οποίοι έχουν δυσλειτουργική συμπεριφορά που προκύπτει από την παρουσία κάποιου σφάλματος και το σύνολο των μη προσβάσιμων ελεγκτών να περιέχει όλους και μόνο τους ελεγκτές προς τους οποίους δεν μπορεί να δημιουργηθεί κανένα μονοπάτι από την πηγή που να μην περιέχει κάποιο μη λειτουργικό ελεγκτή. Η διασφάλιση του γεγονός ότι ο αλγόριθμος μου προβαίνει σε σωστή κατηγοριοποίηση των ελεγκτών και καταλήγει πάντοτε σε σωστά αποτελέσματα γίνεται μέσω της απόδειξης του θεωρήματος 1.

Εξίσου σημαντικό είναι βέβαια να διασφαλιστεί και ότι ο αλγόριθμος μου τερματίζει πάντα, ασχέτως των διαφορετικών καταστάσεων στις οποίες μπορεί να βρεθεί κατά την εκτέλεση του. Αυτό διασφαλίζεται μέσω της απόδειξης του θεωρήματος 2.

Για τις αποδείξεις των δύο θεωρημάτων κρίθηκε αναγκαία η δημιουργία τριών λημμάτων, οι αποδείξεις των οποίων περιλαμβάνονται στο τέλος του υποκεφαλαίου.

Θεώρημα 1: Μετά από οποιοδήποτε αριθμό επαναλήψεων του βρόγχου του αλγορίθμου (γραμμές 14-50), όπου UNC το σύνολο των Unchecked Nodes, UNR το σύνολο των Unreachable Nodes, FN το σύνολο των Faulty Nodes, WN το σύνολο των Working Nodes, θα ισχύει ότι $UNR \cap FN = \emptyset$, $FN \cap WN = \emptyset$, $UNR \cap WN = \emptyset$ και $UNC \cap (UNR \cup FN \cup WN) = \emptyset$ αλλά και επίσης ότι:

- $\forall x \in UNR$ ο x είναι όντως ένας μη προσβάσιμος ελεγκτής
- $\forall x \in WN$ ο x είναι όντως ένας λειτουργικός ελεγκτής
- $\forall x \in FN$ ο x είναι όντως ένας μη λειτουργικός ελεγκτής

Θεώρημα 2: Ο αλγόριθμος τερματίζει πάντα και όταν τερματίσει θα ισχύει ότι UNR∪FW∪WN=A, όπου UNR το σύνολο των Unreachable Nodes, FN το σύνολο των Faulty Nodes, WN το σύνολο των Working Nodes και A το σύνολο όλων των ελεγκτών.

Λήμμα 1: Οι αλγόριθμοι δημιουργίας μονοπατιού τερματίζουν πάντα και αδυνατούν να καθορίσουν κάποιο μονοπάτι αν και μόνο αν δεν υπάρχει κανένα μονοπάτι προς τον NUC το οποίο αποτελείται αποκλειστικά από ελεγκτές x, όπου $x \in UNC \cup WN$. Επίσης, για κάθε μονοπάτι (path) που επιστρέφεται από τους αλγόριθμους δημιουργίας μονοπατιού ισχύουν οι εξής ιδιότητες:

- $\forall x ((x \in path) \rightarrow (x \in UNC \cup WN))$, όπου x ελεγκτής
- $\forall i \forall j ((i \neq j) \rightarrow (path[i] \neq path[j]))$, όπου i,j φυσικοί αριθμοί μικρότεροι ή ίσοι με το μέγεθος του μονοπατιού και path[k] ο k-οστός ελεγκτής του μονοπατιού

Λήμμα 2: Έστω path το μονοπάτι αποστολής που έχει οριστεί από τον αλγόριθμο δημιουργίας μονοπατιού. Όταν κατά την αποστολή πακέτων το Gateway Queue (GQ) αδειάσει χωρίς να έχει προκύψει Gateway Time-out (GT), τότε γνωρίζουμε ότι $\forall x \in$ path, ο x είναι λειτουργικός ελεγκτής.

Λήμμα 3: Έστω path το μονοπάτι αποστολής που έχει οριστεί από τον αλγόριθμο δημιουργίας μονοπατιού. Όταν κατά την αποστολή των πακέτων προκύψει Gateway Time-out (GT), τότε γνωρίζουμε ότι:

- $\exists f \in$ path, όπου ο f είναι μη λειτουργικός ελεγκτής και k είναι η θέση του f μέσα στο path=$n_1 n_2 \ldots n_{k-1} f n_{k+1} \ldots n_m$, όπου k = ⌈(Packets sent – Roadblock packets arrived + 1) / Buffers per node⌉
- $\forall i<k$ ο n_i είναι ένας λειτουργικός ελεγκτής, όπου ο i είναι φυσικός αριθμός

3.8.2 Απόδειξη Θεωρήματος 1

Η απόδειξη του θεωρήματος θα γίνει με επαγωγή ως προς τον αριθμό των επαναλήψεων του βρόγχου του αλγορίθμου (γραμμές 14 μέχρι 50).

Βάση Επαγωγής: Iteration = 0

Γνωρίζουμε ότι πριν να εκτελεστεί καμιά επανάληψη του βρόγχου του αλγορίθμου (γραμμές 14 -50), ισχύει ότι UNC=A (γραμμή 8) και UNR= Ø, FN= Ø, WN= Ø (γραμμές 9-11). Συνεπώς, όντως ισχύει ότι UNR∩FN = Ø, FN∩WN = Ø, UNR∩WN = Ø και UNC∩(UNR∪FN∪WN) =Ø και εφόσον τα σύνολα UNR,FN και WN είναι κενά, άρα δεν περιέχουν κανένα κόμβο, είναι τετριμμένο ότι:

- ∀ x∈ UNR ο x είναι όντως ένας μη προσβάσιμος ελεγκτής
- ∀ x∈ WN ο x είναι όντως ένας λειτουργικός ελεγκτής
- ∀ x∈ FN ο x είναι όντως ένας μη λειτουργικός ελεγκτής

Επαγωγική Υπόθεση: Iteration = k

Θέτουμε ως UNC_k το σύνολο των Unchecked Nodes, UNR_k το σύνολο των Unreachable Nodes, FN_k το σύνολο των Faulty Nodes και WN_k το σύνολο των Working Nodes μετά την k-οστή επανάληψη του βρόγχου του αλγορίθμου (γραμμές 14-50). Υποθέτουμε ότι ισχύει $UNR_k∩FN_k$ = Ø, $FN_k∩WN_k$ = Ø, $UNR_k∩WN_k$ = Ø και $UNC_k∩(UNR_k∪FN_k∪WN_k)$ =Ø αλλά και επίσης ότι:

- ∀ x∈ UNR_k ο x είναι όντως ένας μη προσβάσιμος ελεγκτής
- ∀ x∈ WN_k ο x είναι όντως ένας λειτουργικός ελεγκτής
- ∀ x∈ FN_k ο x είναι όντως ένας μη λειτουργικός ελεγκτής

Επαγωγικό Βήμα: Iteration = k+1

Θέτουμε ως UNC_{k+1} το σύνολο των Unchecked Nodes, UNR_{k+1} το σύνολο των Unreachable Nodes, FN_{k+1} το σύνολο των Faulty Nodes και WN_{k+1} το σύνολο των Working Nodes μετά την k+1-οστή επανάληψη του βρόγχου του αλγορίθμου (γραμμές

14-50). Θέλουμε να αποδείξουμε ότι ισχύει $UNR_{k+1} \cap FN_{k+1} = \emptyset$, $FN_{k+1} \cap WN_{k+1} = \emptyset$, $UNR_{k+1} \cap WN_{k+1} = \emptyset$ και $UNC_{k+1} \cap (UNR_{k+1} \cup FN_{k+1} \cup WN_{k+1}) = \emptyset$ αλλά και επίσης ότι:

- $\forall x \in UNR_{k+1}$ ο x είναι όντως ένας μη προσβάσιμος ελεγκτής
- $\forall x \in WN_{k+1}$ ο x είναι όντως ένας λειτουργικός ελεγκτής
- $\forall x \in FN_{k+1}$ ο x είναι όντως ένας μη λειτουργικός ελεγκτής

Από την επαγωγική υπόθεση γνωρίζουμε ότι οι πιο πάνω ιδιότητες ισχύουν για τα σύνολα των Unchecked Nodes, Unreachable Nodes, Faulty Nodes και Working Nodes μετά το τέλος της k-οστής επανάληψης του βρόγχου του αλγορίθμου. Αυτό που θα κάνουμε για να αποδείξουμε ότι οι ιδιότητες ισχύουν και μετά το τέλος της k+1-οστής επανάληψης είναι να δούμε τι συμβαίνει κατά την εκτέλεση της k+1-οστής επανάληψης του βρόγχου του αλγορίθμου.

Στην αρχή της k+1-οστής επανάληψης του βρόγχου του αλγορίθμου (γραμμές 14-50), γίνεται ο ορισμός κάποιου κόμβου των UNC ως Node Under Check(NUC) , δηλαδή $NUC \in UNC$ (γραμμή 15), βάσει της σειράς επιλογής των NUC που δόθηκε από το χρήστη. Ακολούθως, στη γραμμή 16, καθορίζεται το μονοπάτι που θα χρησιμοποιηθεί για την αποστολή πακέτων στον NUC (path), βάσει του αλγόριθμου δημιουργίας μονοπατιού που δόθηκε από το χρήστη. Στη συνέχεια, στη γραμμή 17, ελέγχεται κατά πόσο η προσπάθεια καθορισμού μονοπατιού από τον αλγόριθμο δημιουργίας μονοπατιού ήταν ανεπιτυχής. Εδώ προκύπτουν δύο υποπεριπτώσεις:

1. Η προσπάθεια καθορισμού μονοπατιού από τον αλγόριθμο δημιουργίας μονοπατιού ήταν ανεπιτυχής

Σε αυτή την υποπερίπτωση, γνωρίζουμε ότι δεν υπάρχει κανένα μονοπάτι προς τον NUC το οποίο αποτελείται αποκλειστικά από ελεγκτές x, όπου $x \in UNC_k \cup WN_k$ (Λήμμα 1). Συνεπώς, θα είναι αληθής η συνθήκη της γραμμής 17 και ο NUC θα αφαιρεθεί από το σύνολο των μη ελεγμένων ελεγκτών (γραμμή 18) και θα προστεθεί στο σύνολο των μη προσβάσιμων ελεγκτών (γραμμή 19). Έπειτα, θα γίνει επιστροφή στην αρχή του βρόγχου (γραμμή 14), χωρίς να διαφοροποιηθούν περαιτέρω τα σύνολα των ελεγκτών. Άρα, συνολικά σε αυτή την υποπερίπτωση ισχύει ότι:

$UNC_{k+1} = UNC_k - \{NUC\}$

$UNR_{k+1} = UNR_k \cup \{NUC\}$

$WN_{k+1} = WN_k$

$FN_{k+1} = FN_k$

Συνεπώς, ισχύει ότι $UNR_{k+1} \cap FN_{k+1} = \emptyset$, $FN_{k+1} \cap WN_{k+1} = \emptyset$ και $UNR_{k+1} \cap WN_{k+1} = \emptyset$, καθώς γνωρίζουμε ότι αρχικά ισχύει ότι $UNR_k \cap FN_k = \emptyset$, $FN_k \cap WN_k = \emptyset$ και $UNR_k \cap WN_k = \emptyset$ (επαγωγική υπόθεση) και τον NUC έχει εισαχθεί αποκλειστικά στο σύνολο UNR_{k+1}. Επιπρόσθετα, ισχύει ότι $UNC_{k+1} \cap (UNR_{k+1} \cup FN_{k+1} \cup WN_{k+1}) = \emptyset$ καθώς γνωρίζουμε ότι αρχικά ισχύει ότι $UNR_k \cap (UN_k \cup FN_k \cup WN_k) = \emptyset$ (επαγωγική υπόθεση) και ο ελεγκτής (NUC) που εισάχθηκε στο σύνολο $UNR_{k+1} \cup FN_{k+1} \cup WN_{k+1}$ έχει αφαιρεθεί από το σύνολο UNC_{k+1} και δεν έχει εισαχθεί ξανά σε αυτό. Ακόμη, τον NUC σωστά έχει εισαχθεί στο σύνολο UNR_{k+1}, καθώς είναι όντως ένας μη προσβάσιμος ελεγκτής εφόσον δεν υπάρχει κανένα μονοπάτι που να φτάνει σε αυτόν το οποίο να περιλαμβάνει μόνο ελεγκτές που είναι λειτουργικοί ($x \in WN_k$) ή μη ελεγμένοι και άρα δυνητικά λειτουργικοί ($x \in UNC_k$). Συνεπώς ισχύει και ότι:

- $\forall\ x \in UNR_{k+1}$ ο x είναι όντως ένας μη προσβάσιμος ελεγκτής βάσει της επαγωγικής υπόθεσης εφόσον $UNR_{k+1} = UNR_k \cup \{NUC\}$, και αφού ο NUC είναι όντως ένας μη προσβάσιμος ελεγκτής.
- $\forall\ x \in WN_{k+1}$ ο x είναι όντως ένας λειτουργικός ελεγκτής βάσει της επαγωγικής υπόθεσης εφόσον $WN_{k+1} = WN_k$.
- $\forall\ x \in FN_{k+1}$ ο x είναι όντως ένας μη λειτουργικός ελεγκτής βάσει της επαγωγικής υπόθεσης εφόσον $FN_{k+1} = FN_k$.

2. Η προσπάθεια καθορισμού μονοπατιού από τον αλγόριθμο δημιουργίας μονοπατιού ήταν επιτυχής

Σε αυτή την υποπερίπτωση, θα είναι ψευδής η συνθήκη της γραμμής 17 και έτσι θα ακολουθήσει υπολογισμός των ελεγκτών στους οποίους πρέπει να αποσταλούν πακέτα οδοφράγματος για να σχηματιστεί επιτυχώς το μονοπάτι από την πηγή προς τον NUC, βάσει του αλγόριθμου δρομολόγησης που επιλέχθηκε από το χρήστη (γραμμή 21).

Έπειτα, τα πακέτα οδοφράγματος που πρέπει να αποσταλούν για τον καθορισμό των ελεγκτών αυτών ως ελεγκτών οδοφράγματος, θα εισαχθούν στο Gateway Queue (GQ). Συγκεκριμένα, θα αποσταλεί ένα πακέτο οδοφράγματος σε κάθε κόμβο ο οποίος πρέπει να καθοριστεί ως ελεγκτής οδοφράγματος (γραμμές 22-24). Στη συνέχεια, θα εισαχθούν στο GQ και τα πακέτα αναγνώρισης σφαλμάτων που πρέπει αποσταλούν προς τον NUC. Συγκεκριμένα, θα αποσταλούν τόσα πακέτα αναγνώρισης σφαλμάτων προς τον NUC όσο είναι το μήκος του μονοπατιού προς τον NUC επί τον αριθμό των buffer που περιέχει ο κάθε ελεγκτής. (γραμμές 25-27). Ακολούθως, θα ξεκινήσει η αποστολή των πακέτων από το GQ στο πλέγμα. Συγκεκριμένα, ενόσω το GQ δεν είναι άδειο και δεν έχει παρουσιαστεί Gateway Time-out (GT), όποτε ο πρώτος ελεγκτής του πλέγματος είναι έτοιμος να λάβει το επόμενο πακέτο η πηγή θα αφαιρεί το πρώτο πακέτο που βρίσκεται στο GQ και θα το αποστέλλει σε αυτόν (γραμμές 28-32). Από τη γραμμή 28 μπορούμε να συμπεράνουμε ότι η αποστολή των πακέτων από το GQ στο πλέγμα μπορεί να σταματήσει για δύο λόγους. Είτε επειδή το GQ θα έχει αδειάσει, είτε επειδή θα έχει προκύψει GT. Στη γραμμή 33 ελέγχεται κατά πόσο η αποστολή πακέτων έχει σταματήσει λόγω του ότι το GQ έχει αδειάσει, χωρίς να προκύψει GT. Συνεπώς εδώ έχουμε δύο ακόμη υποπεριπτώσεις:

2.1. Η αποστολή πακέτων έχει σταματήσει καθώς το GQ έχει αδειάσει, χωρίς να προκύψει GT

Σε μια τέτοια περίπτωση γνωρίζουμε ότι δεν υπάρχει κανένας μη λειτουργικός ελεγκτής στο μονοπάτι που χρησιμοποιήθηκε για την αποστολή πακέτων στον NUC (Λήμμα 2). Συνεπώς, όλοι οι ελεγκτές του μονοπατιού θα αφαιρεθούν από το σύνολο των μη ελεγμένων ελεγκτών (γραμμή 34) και θα προστεθούν στο σύνολο των λειτουργικών ελεγκτών (γραμμή 35). Έπειτα, θα πραγματοποιηθεί καθαρισμός του πλέγματος (γραμμή 36) και θα γίνει επιστροφή στην αρχή του βρόγχου (γραμμή 14), χωρίς να διαφοροποιηθούν περαιτέρω τα σύνολα των ελεγκτών. Άρα, συνολικά σε αυτή την υποπερίπτωση ισχύει ότι:

$UNC_{k+1} = UNC_k - \{x\}, \forall x \in$ path to NUC

$UNR_{k+1} = UNR_k$

$WN_{k+1} = WN_k \cup \{x\}, \forall x \in$ path to NUC

$FN_{k+1} = FN_k$

Συνεπώς, ισχύει ότι $UNR_{k+1} \cap FN_{k+1} = \emptyset$, $FN_{k+1} \cap WN_{k+1} = \emptyset$ και $UNR_{k+1} \cap WN_{k+1} = \emptyset$, καθώς γνωρίζουμε ότι αρχικά ισχύει ότι $UNR_k \cap FN_k = \emptyset$, $FN_k \cap WN_k = \emptyset$ και $UNR_k \cap WN_k = \emptyset$ (επαγωγική υπόθεση) και οι ελεγκτές x ($\forall x \in$ path to NUC) έχουν εισαχθεί αποκλειστικά στο σύνολο WN_{k+1}. Επιπρόσθετα, ισχύει ότι $UNC_{k+1} \cap (UNR_{k+1} \cup FN_{k+1} \cup WN_{k+1}) = \emptyset$ καθώς γνωρίζουμε ότι αρχικά ισχύει ότι $UNR_k \cap (UN_k \cup FN_k \cup WN_k) = \emptyset$ (επαγωγική υπόθεση) και οι ελεγκτές x ($\forall x \in$ path to NUC) που εισήχθησαν στο σύνολο $UNR_{k+1} \cup FN_{k+1} \cup WN_{k+1}$ έχουν αφαιρεθεί από το σύνολο UNC_{k+1} και δεν έχουν εισαχθεί ξανά σε αυτό. Ακόμη, οι ελεγκτές x ($\forall x \in$ path to NUC) σωστά έχουν εισαχθεί στο σύνολο WN_{k+1}, καθώς είναι όντως λειτουργικοί ελεγκτές εφόσον το GQ έχει αδειάσει χωρίς να προκύψει GT (Λήμμα 2). Συνεπώς ισχύει και ότι:

- $\forall x \in UNR_{k+1}$ ο x είναι όντως ένας μη προσβάσιμος ελεγκτής βάσει της επαγωγικής υπόθεσης εφόσον $UNR_{k+1}=UNR_k$.
- $\forall x \in WN_{k+1}$ ο x είναι όντως ένας λειτουργικός ελεγκτής βάσει της επαγωγικής υπόθεσης εφόσον $WN_{k+1}= WN_k \cup \{x\}$ ($\forall x \in$ path to NUC), και αφού οι x είναι όντως λειτουργικοί ελεγκτές.
- $\forall x \in FN_{k+1}$ ο x είναι όντως ένας μη λειτουργικός ελεγκτής βάσει της επαγωγικής υπόθεσης εφόσον $FN_{k+1}=FN_k$.

2.2. Η αποστολή πακέτων έχει σταματήσει καθώς έχει προκύψει GT

Σε μια τέτοια περίπτωση γνωρίζουμε ότι υπάρχει σίγουρα ένας μη λειτουργικός ελεγκτής στο μονοπάτι που χρησιμοποιήθηκε για την αποστολή πακέτων στον NUC (Λήμμα 3). Έτσι, αρχικά θα υπολογιστεί ο αριθμός των πακέτων που έχουν παγιδευτεί στο πλέγμα (γραμμή 38) και έπειτα θα χρησιμοποιηθεί ο αριθμός αυτός για να υπολογιστεί ποιος ελεγκτής του μονοπατιού είναι μη λειτουργικός (γραμμή 39, Λήμμα 3). Ακολούθως, ο συγκεκριμένος ελεγκτής (f) θα αφαιρεθεί από το σύνολο των μη ελεγμένων ελεγκτών (γραμμή 40) και θα προστεθεί στο σύνολο των μη λειτουργικών ελεγκτών (γραμμή 41).Επίσης, γνωρίζουμε ότι όλοι οι ελεγκτές του μονοπατιού που προηγούνται του μη λειτουργικού κόμβου που έχουμε εντοπίσει, είναι λειτουργικοί

(Λήμμα 3). Έτσι, οι συγκεκριμένοι ελεγκτές θα αφαιρεθούν από το σύνολο των μη ελεγμένων ελεγκτών και θα προστεθούν στο σύνολο των λειτουργικών ελεγκτών (γραμμές 42-45). Έπειτα, θα πραγματοποιηθεί άδειασμα του GQ (γραμμή 46), καθαρισμός του πλέγματος (γραμμή 47) και θα γίνει επιστροφή στην αρχή του βρόγχου (γραμμή 14), χωρίς να διαφοροποιηθούν περαιτέρω τα σύνολα των ελεγκτών. Άρα, συνολικά σε αυτή την υποπερίπτωση ισχύει ότι:

$UNC_{k+1} = UNC_k - \{f, x\}$, $\forall x \in$ path to NUC and precedes f

$UNR_{k+1} = UNR_k$

$WN_{k+1} = WN_k \cup \{x\}$, $\forall x \in$ path to NUC and precedes f

$FN_{k+1} = FN_k \cup \{f\}$

Συνεπώς, ισχύει ότι $UNR_{k+1} \cap FN_{k+1} = \emptyset$, $FN_{k+1} \cap WN_{k+1} = \emptyset$ και $UNR_{k+1} \cap WN_{k+1} = \emptyset$, καθώς γνωρίζουμε ότι αρχικά ισχύει ότι $UNR_k \cap FN_k = \emptyset$, $FN_k \cap WN_k = \emptyset$ και $UNR_k \cap WN_k = \emptyset$ (επαγωγική υπόθεση) και ο ελεγκτής f έχει εισαχθεί αποκλειστικά στο σύνολο FN, ενώ οι ελεγκτές x ($\forall x \in$ path to NUC and precedes f) έχουν εισαχθεί αποκλειστικά στο σύνολο WN_{k+1}. Επιπρόσθετα, ισχύει ότι $UNC_{k+1} \cap (UNR_{k+1} \cup FN_{k+1} \cup WN_{k+1}) = \emptyset$ καθώς γνωρίζουμε ότι αρχικά ισχύει ότι $UNR_k \cap (UN_k \cup FN_k \cup WN_k) = \emptyset$ (επαγωγική υπόθεση) και οι ελεγκτές f και x ($\forall x \in$ path to NUC and precedes f) που εισήχθησαν στο σύνολο $UNR_{k+1} \cup FN_{k+1} \cup WN_{k+1}$ έχουν αφαιρεθεί από το σύνολο UNC_{k+1} και δεν έχουν εισαχθεί ξανά σε αυτό. Ακόμη, ο f σωστά έχει εισαχθεί στο σύνολο FN_{k+1}, καθώς είναι όντως ένας μη λειτουργικός ελεγκτής (Λήμμα 3) και οι ελεγκτές x ($\forall x \in$ path to NUC and precedes f) σωστά έχουν εισαχθεί στο σύνολο WN_{k+1}, καθώς όντως είναι λειτουργικοί ελεγκτές εφόσον έχει προκύψει GT λόγω του f (Λήμμα 3). Συνεπώς ισχύει και ότι:

- $\forall x \in UNR_{k+1}$ ο x είναι όντως ένας μη προσβάσιμος ελεγκτής βάσει της επαγωγικής υπόθεσης εφόσον $UNR_{k+1}=UNR_k$.
- $\forall x \in WN_{k+1}$ ο x είναι όντως ένας λειτουργικός ελεγκτής βάσει της επαγωγικής υπόθεσης εφόσον $WN_{k+1}= WN_k \cup \{y\}$ ($\forall y \in$ path to NUC and precedes f), και αφού οι y είναι όντως λειτουργικοί ελεγκτές.
- $\forall x \in FN_{k+1}$ ο x είναι όντως ένας μη λειτουργικός ελεγκτής βάσει της επαγωγικής υπόθεσης εφόσον $FN_{k+1}=FN_k \cup \{f\}$, και αφού ο f είναι όντως ένας μη λειτουργικός

ελεγκτής.

Αυτό ολοκληρώνει την απόδειξη του θεωρήματος 1.

3.8.3 Απόδειξη Θεωρήματος 2

Έστω UNC το σύνολο των Unchecked Nodes πριν την επανάληψη του βρόγχου του αλγορίθμου (γραμμές 14-50) και UNC' το σύνολο των Unchecked Nodes μετά την επανάληψη του βρόγχου του αλγορίθμου. Για να τερματίσει ο αλγόριθμος πρέπει κάποια στιγμή η συνθήκη του βρόγχου του αλγορίθμου (γραμμή 14) να μην ικανοποιείται, δηλαδή το σύνολο των UNC να γίνει κάποια στιγμή κενό. Συνεπώς, για να αποδείξουμε ότι ο αλγόριθμος τερματίζει αρκεί να αποδείξουμε ότι σε κάθε επανάληψη του βρόγχου του αλγορίθμου (γραμμές 14-50) ισχύει ότι $\exists x$ ($(x \in UNC) \wedge (x \notin UNC')$), όπου ο x είναι ελεγκτής.

Αρχικά, είναι ξεκάθαρο ότι σε κανένα σημείο του βρόγχου του αλγορίθμου (γραμμές 14-50) δεν πραγματοποιείται πρόσθεση κόμβου ή ελεγκτών στο σύνολο UNC και συνεπώς αφαίρεση οποιουδήποτε κόμβου ή ελεγκτών από το σύνολο UNC μας εγγυάται ότι ο ελεγκτής ή οι ελεγκτές αυτοί είναι αδύνατο υπάρξουν στο σύνολο UNC'.

Ακολούθως, θα υποθέσουμε ότι για κάποια επανάληψη του αλγόριθμου ισχύει $\neg\exists x$ $((x \in UNC) \wedge (x \notin UNC'))$ όπου ο x είναι ελεγκτής, για να φτάσουμε σε αντίφαση. Στη συγκεκριμένη επανάληψη του βρόγχου η συνθήκη της γραμμής 17 δεν μπορεί να ικανοποιείται καθώς στη γραμμή 18 που είναι εντός του πεδίου του βρόγχου πραγματοποιείται αφαίρεση κόμβου από το σύνολο UNC. Είμαστε σίγουροι ότι ο ελεγκτής αυτός αρχικά περιλαμβάνεται στο σύνολο UNC, καθώς πρόκειται για τον NUC. Επίσης, ούτε η συνθήκη της γραμμής 33 είναι δυνατό να ικανοποιείται καθώς σε μια τέτοια περίπτωση θα πραγματοποιηθούν αφαιρέσεις ελεγκτών από το σύνολο UNC στη γραμμή 34 και είμαστε σίγουροι ότι τουλάχιστον ένας από τους ελεγκτές που θα αφαιρεθούν, ο NUC, αρχικά περιλαμβάνεται στο σύνολο UNC. Συνεπώς, συμπεραίνουμε ότι κατά τη συγκεκριμένη επανάληψη του αλγορίθμου θα εκτελεστεί σίγουρα η γραμμή 40 κατά την οποία αφαιρείται από το σύνολο UNC ο ελεγκτής που εντοπίστηκε στη γραμμή 39 ότι είναι μη λειτουργικός. Εφόσον από το Λήμμα 1

γνωρίζουμε ότι στο μονοπάτι προς τον NUC περιλαμβάνονται μόνο ελεγκτές x όπου x∈UNCUWN και από το Θεώρημα 1 γνωρίζουμε ότι μετά από οποιοδήποτε αριθμό επαναλήψεων του βρόγχου του αλγορίθμου (γραμμές 14-50) ισχύει ότι ∀y∈ WN ο y είναι όντως ένας λειτουργικός ελεγκτής, συμπεραίνουμε ότι ο ελεγκτής που εντοπίστηκε στη γραμμή 39 και αφαιρέθηκε από το UNC στη γραμμή 40, περιλαμβανόταν στο UNC πριν από την επανάληψη. Άρα, αναπόφευκτα για τη συγκεκριμένη επανάληψη ισχύει ότι ∃x ((x∈UC) ∧ (x∉UC')). Συνεπώς, καταλήγουμε σε αντίφαση, και ο αρχικός μας συλλογισμός ότι σε κάθε επανάληψη του βρόγχου του αλγορίθμου (γραμμές 14-50) ισχύει ∃x ((x∈UC) ∧ (x∉UC')) όπου ο x είναι ελεγκτής, είναι ορθός.

Άρα, αποδείχθηκε ότι ο αλγόριθμος τερματίζει πάντα.

Απομένει να αποδείξουμε ότι όταν ο αλγόριθμος τερματίσει θα ισχύει ότι UNRUFNUWN=A όπου A το σύνολο όλων των ελεγκτών. Γνωρίζουμε ότι αρχικά ισχύει UNR= Ø, FN= Ø, WN= Ø (γραμμές 9-11) και ότι το σύνολο UNC περιέχει αρχικά όλους τους ελεγκτές του πλέγματος, δηλαδή UNC=A (γραμμή 8). Συνεπώς, γνωρίζοντας ότι απαραίτητη συνθήκη για να τερματίσει ο αλγόριθμος είναι τελικά να ισχύει ότι UNC= Ø (γραμμή 14), αν αποδείξουμε ότι όποτε ένας ελεγκτής ή ένα υποσύνολο ελεγκτών αφαιρείται από το σύνολο UNC κατά την εκτέλεση του αλγορίθμου τότε ο συγκεκριμένος ελεγκτής ή το συγκεκριμένο υποσύνολο ελεγκτών θα εισαχθεί σε ένα εκ των συνόλων UNR,FN ή WN στο μέλλον, τότε θα δείξουμε ότι όντως στο τέλος θα ισχύει ότι UNRUFNUWN=A. Μέσα στον αλγόριθμο αφαιρέσεις ελεγκτών ή συνόλων ελεγκτών από το σύνολο UNC πραγματοποιούνται στα εξής σημεία:

- Στη γραμμή 18 και ο ελεγκτής που αφαιρείται εισάγεται αμέσως μετά στο σύνολο UNR στη γραμμή 19.
- Στη γραμμή 34 και το σύνολο ελεγκτών που αφαιρείται εισάγεται αμέσως μετά στο σύνολο WN στη γραμμή 35.
- Στη γραμμή 40 και ο ελεγκτής που αφαιρείται εισάγεται αμέσως μετά στο σύνολο FN στη γραμμή 41.
- Στη γραμμή 43 και ο ελεγκτής που αφαιρείται εισάγεται αμέσως μετά στο σύνολο WN στη γραμμή 44.

Συνεπώς, παρατηρούμε ότι όντως όποτε ένας ελεγκτής ή ένα σύνολο ελεγκτών αφαιρείται από το σύνολο UNC κατά την εκτέλεση του αλγορίθμου τότε ο συγκεκριμένος ελεγκτής ή το συγκεκριμένο υποσύνολο ελεγκτών θα εισαχθεί σε ένα εκ των συνόλων UNR,FN ή WN στο μέλλον.

Άρα, αυτό ολοκληρώνει την απόδειξη του θεωρήματος 2 αφού φάνηκε ότι όντως ο αλγόριθμος τερματίζει πάντα και όταν τερματίσει θα ισχύει ότι UNRUFNUWN=A.

3.8.4 Απόδειξη Λήμματος 1

Η απόδειξη της ορθότητας του λήμματος 1 προκύπτει τετριμμένα από τις προδιαγραφές που έχουν τεθεί για τους αλγόριθμους δημιουργίας μονοπατιού. Συγκεκριμένα, έχει καθοριστεί ότι για να θεωρείται κάποιος αλγόριθμος δημιουργίας μονοπατιού έγκυρος, θα πρέπει να τερματίζει πάντα και να αδυνατεί να καθορίσει κάποιο μονοπάτι αν και μόνο αν δεν υπάρχει κανένα μονοπάτι προς τον NUC που να αποτελείται αποκλειστικά από λειτουργικούς και μη ελεγμένους ελεγκτές.

Ακόμη, υπάρχουν δύο επιπρόσθετες προδιαγραφές που καθορίζουν ότι τα μονοπάτια που επιστρέφονται δεν πρέπει να περιέχουν ποτέ κύκλους και ότι πρέπει να αποτελούνται αποκλειστικά από λειτουργικούς και μη ελεγμένους ελεγκτές.

Συνεπώς, οι προδιαγραφές αυτές ολοκληρώνουν την απόδειξη του λήμματος 1.

3.8.5 Απόδειξη Λήμματος 2

Κατά την εκτέλεση του βρόγχου του αλγορίθμου (γραμμές 14-50), εισάγονται αρχικά στο Gateway Queue όσα πακέτα οδοφράγματος χρειάζεται να αποσταλούν για τη δημιουργία του μονοπατιού προς τον NUC (γραμμές 22-24). Ακολούθως, στο Gateway Queue εισάγονται και τόσα πακέτα αναγνώρισης σφαλμάτων όσος είναι και ο αριθμός των ελεγκτών που περιλαμβάνονται στο μονοπάτι προς τον NUC πολλαπλασιασμένος με τον αριθμό των buffers που θα χρησιμοποιηθούν από κάθε ελεγκτή (γραμμές 25-27). Έπειτα, όταν ξεκινήσει η διαδικασία αποστολής των πακέτων (γραμμές 28-32), το Gateway Queue σταδιακά αποστείλει όλα τα πακέτα που περιέχει για να δρομολογηθούν μέσα στο πλέγμα. Καθώς τα πακέτα δρομολογούνται μέσα στο

πλέγμα, τα πακέτα οδοφράγματος χρησιμοποιούνται για την τοποθέτηση οδοφραγμάτων έτσι ώστε να προκύψει το επιθυμητό μονοπάτι προς τον NUC και ακολούθως καταστρέφονται. Εντούτοις, τα πακέτα αναγνώρισης σφαλμάτων δε καταστρέφονται και παραμένουν μέσα στο μονοπάτι μέχρι να φτάσουν στον τελευταίο ελεγκτή του μονοπατιού που είναι ο προορισμός τους, ο NUC. Το καθένα από αυτά τα πακέτα για να φτάσει μέχρι τον NUC, πρέπει να περάσει από όλους τους ελεγκτές του μονοπατιού και κατ' επέκταση και από τους buffers εισόδου, επεξεργασίας και εξόδου των συγκεκριμένων ελεγκτών.

Έτσι, αφού ο αριθμός των πακέτων αναγνώρισης σφαλμάτων που αποστέλλονται είναι ίσος με τον αριθμό των buffers που χρησιμοποιούνται στο μονοπάτι, σε περίπτωση που το μονοπάτι περιλαμβάνει κάποιο μη λειτουργικό κόμβο που εμποδίζει τη διέλευση των πακέτων, αναπόφευκτα το μονοπάτι θα γεμίσει από παγιδευμένα πακέτα. Εάν το μονοπάτι γεμίσει από παγιδευμένα πακέτα πριν αποσταλούν στο πλέγμα όλα τα πακέτα που περιέχει το Gateway Queue, τότε θα προκύψει Gateway Time-out.

Συνεπώς, ο μόνος τρόπος να μην προκύψει Gateway Time-out είναι αν και μόνο αν δεν περιλαμβάνεται κάποιος μη λειτουργικός κόμβος στο μονοπάτι για να εμποδίσει τη διέλευση των πακέτων, και έτσι αυτά καταφέρουν με επιτυχία να φτάσουν στον NUC, επιτρέποντας παράλληλα το άδειασμα του Gateway Queue. Εφόσον αυτή είναι η μοναδική περίπτωση αδειάσματος του Gateway Queue χωρίς να προκύψει Gateway Time-out και για να υλοποιηθεί επιβάλλεται όλοι οι ελεγκτές του μονοπατιού να είναι λειτουργικοί, η απόδειξη του λήμματος 2 έχει ολοκληρωθεί.

3.8.6 Απόδειξη Λήμματος 3

Κατά την εκτέλεση του βρόγχου του αλγορίθμου (γραμμές 14-50), εισάγονται αρχικά στο Gateway Queue όσα πακέτα οδοφράγματος χρειάζεται να αποσταλούν για τη δημιουργία του μονοπατιού προς τον NUC (γραμμές 22-24). Ακολούθως, στο Gateway Queue εισάγονται και τόσα πακέτα αναγνώρισης σφαλμάτων όσος είναι και ο αριθμός των ελεγκτών που περιλαμβάνονται στο μονοπάτι προς τον NUC πολλαπλασιασμένος με τον αριθμό των buffers που θα χρησιμοποιηθούν από κάθε ελεγκτή (γραμμές 25-27). Έπειτα, όταν ξεκινήσει η διαδικασία αποστολής των πακέτων (γραμμές 28-32), το

Gateway Queue σταδιακά αποστείλει όλα τα πακέτα που περιέχει για να δρομολογηθούν μέσα στο πλέγμα. Καθώς τα πακέτα δρομολογούνται μέσα στο πλέγμα, τα πακέτα οδοφράγματος χρησιμοποιούνται για την τοποθέτηση οδοφραγμάτων έτσι ώστε να προκύψει το επιθυμητό μονοπάτι προς τον NUC και ακολούθως καταστρέφονται. Εντούτοις, τα πακέτα αναγνώρισης σφαλμάτων δε καταστρέφονται και παραμένουν μέσα στο μονοπάτι μέχρι να φτάσουν στον τελευταίο ελεγκτή του μονοπατιού που είναι ο προορισμός τους, ο NUC. Το καθένα από αυτά τα πακέτα για να φτάσει μέχρι τον NUC, πρέπει να περάσει από όλους τους ελεγκτές του μονοπατιού και κατ' επέκταση και από τους buffers εισόδου, επεξεργασίας και εξόδου των συγκεκριμένων ελεγκτών.

Έτσι, αφού ο αριθμός των πακέτων αναγνώρισης σφαλμάτων που αποστέλλονται είναι ίσος με τον αριθμό των buffers που χρησιμοποιούνται στο μονοπάτι, σε περίπτωση που το μονοπάτι περιλαμβάνει κάποιο μη λειτουργικό κόμβο που εμποδίζει τη διέλευση των πακέτων, αναπόφευκτα το μονοπάτι θα γεμίσει από παγιδευμένα πακέτα. Εάν το μονοπάτι γεμίσει από παγιδευμένα πακέτα πριν αποσταλούν στο πλέγμα όλα τα πακέτα που περιέχει το Gateway Queue, τότε θα προκύψει Gateway Time-out. Έτσι, σε περίπτωση που προέκυψε Gateway Time-out είναι βέβαιο ότι αυτό έχει προκληθεί από την παρουσία ενός μη λειτουργικού κόμβου στο μονοπάτι.

Για να εντοπιστεί η θέση σου συγκεκριμένου μη λειτουργικού κόμβου μέσα στο μονοπάτι, θα πρέπει αρχικά να υπολογιστεί ο αριθμός των πακέτων που έχουν παγιδευτεί μέσα στο πλέγμα. Αυτό γίνεται αφαιρώντας τον αριθμό των πακέτων οδοφράγματος που έχουν φτάσει στον προορισμό τους και κατ' επέκταση έχουν καταστραφεί, από το συνολικό αριθμό των πακέτων που έχουν αποσταλεί στο πλέγμα. Ακολούθως, θα πρέπει ο αριθμός των πακέτων που έχουν παγιδευτεί στο μονοπάτι να διαιρεθεί με τον αριθμό των buffers του κάθε ελεγκτή που περιλαμβάνονται στο μονοπάτι και συνεπώς περιέχουν κάποιο παγιδευμένο πακέτο. Αυτό θα μας δώσει τον αριθμό των ελεγκτών μέσα στους οποίους υπάρχουν παγιδευμένα πακέτα, άρα προσθέτοντας ακόμη ένα παγιδευμένο πακέτο στο πλέγμα προκύπτει και η θέση του μη λειτουργικού ελεγκτή που έχει προκαλέσει την παγίδευση των πακέτων στο μονοπάτι. Γνωρίζοντας τη θέση του ελεγκτή που αποτρέπει τη διέλευση των πακέτων

μέσα στο μονοπάτι, μπορούμε να καταλήξουμε στο συμπέρασμα ότι όλοι οι προηγούμενοι ελεγκτές του μονοπατιού είναι λειτουργικοί. Αυτό ισχύει, καθώς σε περίπτωση που κάποιος από τους ελεγκτές αυτούς ήταν μη λειτουργικός, η διέλευση των πακέτων στο μονοπάτι θα είχε αρχίσει να εμποδίζεται νωρίτερα και συνεπώς ο αριθμός των πακέτων που είχαν παγιδευτεί στο μονοπάτι θα ήταν μικρότερος.

Συνεπώς, αυτό ολοκληρώνει την απόδειξη του λήμματος 3.

3.9 Περιπτώσεις Χρησιμοποίησης Αλγορίθμου

Ο αλγόριθμος αναγνώρισης σφαλμάτων που σχεδίασα θα ήταν καλό να χρησιμοποιείται σε τρεις περιπτώσεις. Η πρώτη περίπτωση είναι μόλις ξεκινά η λειτουργία του συστήματος, καθώς ενδέχεται να υπάρχουν ήδη δυσλειτουργικοί ελεγκτές οι οποίοι πρέπει να εντοπίζονται πριν προτού ξεκινήσει η μετάδοση πακέτων για άλλους σκοπούς.

Η δεύτερη περίπτωση είναι όταν με κάποιο τρόπο αντιληφθούμε με κάποιο τρόπο ότι έχει προκύψει ένα καινούργιο σφάλμα στο πλέγμα, το οποίο δεν έχει αναγνωριστεί. Η διάγνωση της παρουσίας ενός καινούργιου σφάλματος μπορεί να επιτυγχάνεται με δύο τρόπους. Ο πρώτος τρόπος είναι άμεσα, με τη χρήση πακέτων επιβεβαίωσης (acknowledgment packets). Το συγκεκριμένο είδος πακέτων δεν μεταδίδεται από την πηγή και έχει ως προορισμό κάποιο ελεγκτή, αλλά το αντίθετο. Μεταδίδεται από κάποιον ελεγκτή που έχει λάβει ένα πακέτο και έχει στόχο την ενημέρωση της πηγής για τη λήψη του συγκεκριμένου πακέτου. Αν το πακέτο επιβεβαίωσης δε φτάσει στην πηγή μέσα στο αναμενόμενο χρονικό διάστημα, τότε θα γνωρίζουμε ότι κάποιο σφάλμα αποτρέπει τη δρομολόγηση είτε του αρχικού πακέτου που αποστάλθηκε από την πηγή, είτε του πακέτου επιβεβαίωσης και έτσι θα πρέπει να πραγματοποιηθεί αναγνώριση του εν λόγω σφάλματος. Ένας εναλλακτικός τρόπος διάγνωσης προκύπτει και πάλι από την τεχνική ανάστροφης μετάδοσης σφάλματος. Όταν η πηγή αδυνατεί να στείλει κάποιο πακέτο στον ελεγκτή με τον οποίο συνδέεται καθώς είναι συνεχώς γεμάτος, αυτό σημαίνει ότι κάπου στο πλέγμα έχει προκύψει σφάλμα το οποίο οδηγεί σε παγίδευση των πακέτων στους buffers των ελεγκτών, κάτι που έχει έμμεσα επηρεάσει και τον κόμβο ο οποίος είναι συνδεδεμένος με την πηγή. Μειονέκτημα της συγκεκριμένης μεθόδου είναι ότι προκύπτει ένα μεγαλύτερο χρονικό διάστημα από τη στιγμή που προκύπτει ένα σφάλμα μέχρι να γίνει αντιληπτό ότι υπάρχει, ενώ παροδικά ή διαλείποντα σφάλματα μπορεί να μην τύχουν διάγνωσης. Συνεπώς, όταν με ένα από τους δύο αυτούς τρόπους διαγνωσθεί η παρουσία ενός καινούργιου σφάλματος στο πλέγμα, θα πρέπει να επαναξιολογηθεί την κατάσταση όλων των ελεγκτών του πλέγματος μέσω του αλγόριθμου αναγνώρισης σφαλμάτων.

Η τρίτη περίπτωση στην οποία θεωρώ ότι θα ήταν καλό να εκτελείται ο αλγόριθμος μου είναι ανά τακτά χρονικά διαστήματα, χωρίς να υπάρχει κάποια συγκεκριμένη απόδειξη για την παρουσία καινούργιου σφάλματος. Αυτό το προτείνω καθώς ένα σφάλμα μπορεί να έχει εμφανιστεί, αλλά λόγω του ότι ελεγκτής που το παρουσιάζει δεν έχει χρησιμοποιηθεί ακόμη, να μην έχει γίνει αντιληπτή η ύπαρξη του. Έτσι, ίσως να ήταν καλύτερο να διατεθεί λίγος χρόνος για την έγκαιρη ανακάλυψη του σφάλματος, καθώς το κόστος που θα προκληθεί από το ίδιο το σφάλμα και από μια ενδεχόμενη επαναδρομολόγηση πακέτων, ίσως να είναι πολύ μεγαλύτερο. Ακόμη, σε περίπτωση εντοπισμού κάποιου παροδικού σφάλματος από τον αλγόριθμο, μια επαναληπτική εκτέλεση θα μπορέσει να ενημερώσει την πηγή ότι το εν λόγω σφάλμα δεν υπάρχει πλέον και συνεπώς η δρομολόγηση των πακέτων θα μπορεί να γίνεται χρησιμοποιώντας και το συγκεκριμένο ελεγκτή.

Κεφάλαιο 4

Προσομοιωτής στην Anylogic

4.1 Τι είναι η Anylogic και τι μπορεί να κάνει

Για τη μοντελοποίηση και την προσομοίωση του αλγορίθμου μου χρησιμοποίησα το λογισμικό Anylogic. Η Anylogic είναι ένα εργαλείο μοντελοποίησης και προσομοίωσης που αναπτύχθηκε από την εταιρεία The AnyLogic Company. Η Anylogic υποστηρίζει διάφορες μεθοδολογίες προσομοίωσης, όπως προσομοιώσεις βασισμένες σε πράκτορες (agent-based),προσομοιώσεις διακριτών γεγονότων (discrete events) και προσομοιώσεις δυναμικών του συστήματος (system dynamics). Επίσης, η Anylogic αποτελεί ένα ανεξάρτητο πλατφόρμας (cross-platform) λογισμικό, καθώς μπορεί να χρησιμοποιηθεί σε υπολογιστές που τρέχουν διαφορετικά λειτουργικά συστήματα (Windows, macOS, Linux). Επιπρόσθετα, η Anylogic είναι ένα λογισμικό που χρησιμοποιείται για την προσομοίωση ενός μεγάλου φάσματος εφαρμογών καθώς παρέχει τη δυνατότητα για πραγματοποίηση προσομοιώσεων μεγάλης ακρίβειας, ενώ παράλληλα το γραφικό περιβάλλον της εφαρμογής βοηθά στην ευκολότερη κατανόηση του τρόπου λειτουργίας των αλγορίθμων που χρησιμοποιούνται, αλλά και στην ευκολότερη αποσφαλμάτωση του συστήματος σε περίπτωση που παρατηρηθεί κάποια ανεπιθύμητη συμπεριφορά. Συνεπώς, η Anylogic μπορεί να χρησιμοποιηθεί για την πραγματοποίηση πολύπλοκων προσομοιώσεων που αφορούν θέματα όπως είναι η σωστή διαχείριση των ουρών αναμονής σε ένα αεροδρόμιο ή μια τράπεζα ή όπως είναι η ανάπτυξη ενός οικοσυστήματος.

Ακόμη ένας λόγος για τον οποίο επέλεξα την Anylogic ως το κατάλληλο λογισμικό για τη μοντελοποίηση και την προσομοίωση του αλγορίθμου μου είναι ότι υποστηρίζει τη γλώσσα προγραμματισμού Java. Η Java, είναι η γλώσσα προγραμματισμού την οποία χρησιμοποίησα περισσότερο κατά τη διάρκεια των σπουδών μου, αλλά και κατά τη διάρκεια της ενασχόλησης μου στο ερευνητικό πρόγραμμα Visorsurf. Συνεπώς, είχα την ευκαιρία να εξοικειωθώ σε μεγάλο βαθμό με τη συγκεκριμένη γλώσσα προγραμματισμού, αλλά και με τις βασικές αρχές της αντικειμενοστρέφειας που είναι άρρηκτα συνδεδεμένες μαζί της. Επιπρόσθετα, στα αρχικά στάδια της ανάπτυξης του

αλγορίθμου, είχα αναπτύξει ένα πιο απλό προσομοιωτή χρησιμοποιώντας αποκλειστικά τη γλώσσα προγραμματισμού Java και το ολοκληρωμένο περιβάλλον ανάπτυξης Eclipse, για να ελέγχω την ορθότητα των ιδεών μου. Συνεπώς, γνώριζα ήδη ότι αρκετά κομμάτια λογισμικού που είχαν χρησιμοποιηθεί και στον αρχικό προσομοιωτή ήταν πλήρως λειτουργικά και ελεγμένα. Παραδείγματα τέτοιων κομματιών λογισμικού είναι ο κώδικας που χρησιμοποιείται για τη δημιουργίας των ακμών του πλέγματος και η υλοποίηση του αλγόριθμου εύρεσης βραχύτερου μονοπατιού του Dijkstra. Έτσι, ήταν ιδιαίτερα βοηθητικό για εμένα ότι μπόρεσα να μεταφέρω τα συγκεκριμένα κομμάτια λογισμικού και στον νέο μου προσομοιωτή εφόσον ήμουν βέβαιος για την ορθότητα τους.

4.2 Κύριες Κλάσεις

Κατά τη δημιουργία του προσομοιωτή στην Anylogic έθεσα ως βασικό στόχο να γίνει μια όσο το δυνατό πιο ρεαλιστική μοντελοποίηση του συστήματος. Γι' αυτό ακριβώς το λόγο επέλεξα να χρησιμοποιήσω τον αντικειμενοστρεφή προγραμματισμό (object-oriented programming), έτσι ώστε να μπορέσω να προσομοιώσω τις διάφορες οντότητες του συστήματος ως αντικείμενα. Συνεπώς, δημιούργησα μια κλάση για κάθε διαφορετικό είδος οντότητας που ήθελα να προσομοιώσω. Συγκεκριμένα, δημιούργησα την κλάση Node η οποία αντιπροσωπεύει τους ελεγκτές του πλέγματος, την κλάση Edge η οποία αντιπροσωπεύει τις συνδέσεις μεταξύ των ελεγκτών, την κλάση Packet η οποία αντιπροσωπεύει τα πακέτα που μεταδίδονται μέσα στο πλέγμα και την κλάση Gateway η οποία αντιπροσωπεύει την πηγή. Βεβαίως, υπάρχουν και μερικές επιπλέον βοηθητικές κλάσεις όπως οι κλάσεις ShortestPath, WeightedEdge και διάφορες υλοποιήσεις της διεπαφής (interface) Comparator. Όλες αυτές οι κλάσεις περιέχουν κομμάτια λογισμικού που θα εκτελούνται από την πηγή και θα μπορούσαν εναλλακτικά να αποτελούσαν μέρος της κλάσης Gateway. Εντούτοις, προτίμησα να κάνω το διαχωρισμό αυτό, έτσι ώστε ο κώδικας μου να είναι πιο ευανάγνωστος και αρθρωτός αλλά και για να διασφαλίσω ότι θα μπορούν να εφαρμοστούν με εύκολο τρόπο μελλοντικές αναβαθμίσεις και αλλαγές.

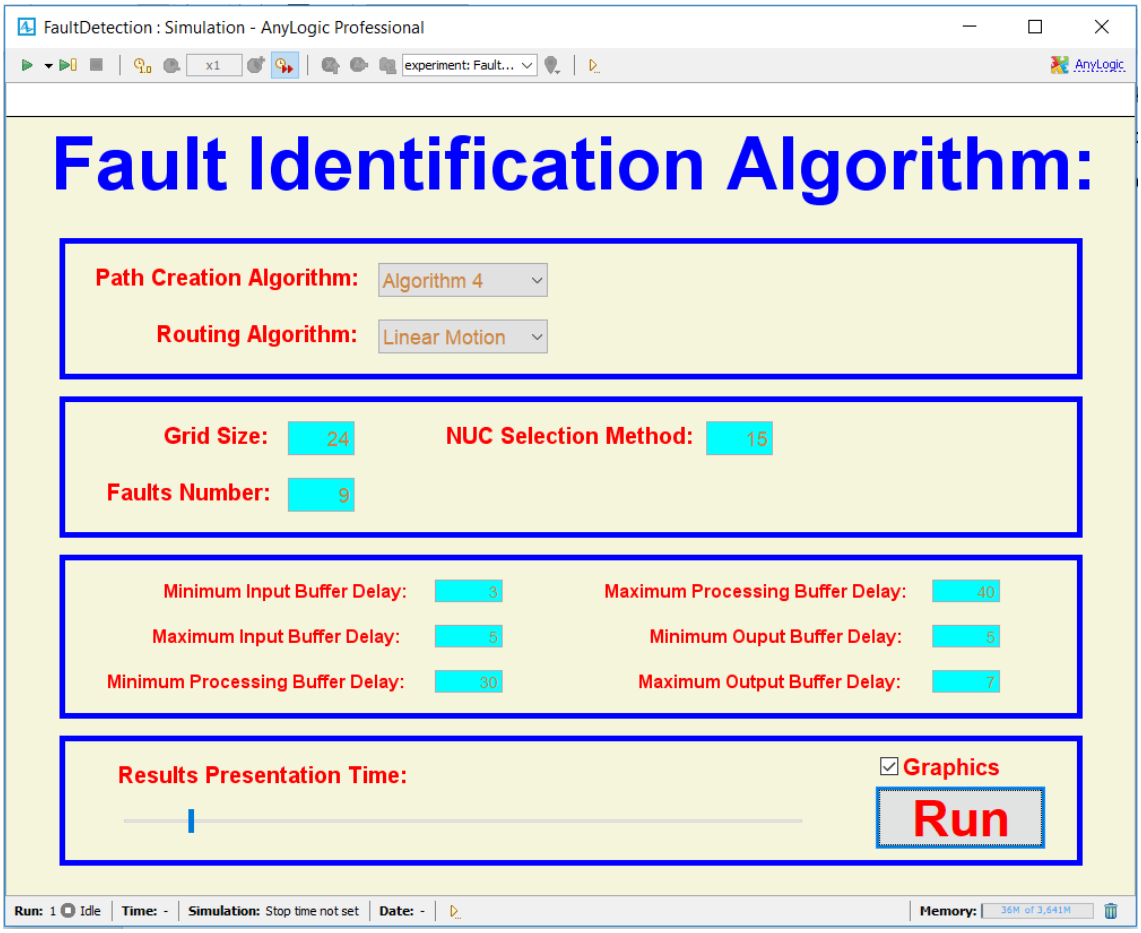
4.3 Παράμετροι που Καθορίζονται από το Χρήστη

Ο προσομοιωτής παρέχει στο χρήστη τη δυνατότητα να καθορίσει τις διάφορες παραμέτρους της προσομοίωσης μέσω της γραφικής διεπαφής χρήστη (graphical user interface). Συγκεκριμένα, ο χρήστης μπορεί να επιλέξει τις εισόδους που θα δοθούν στον αλγόριθμο αναγνώρισης σφαλμάτων, όπως τον αλγόριθμο δρομολόγησης, τη μέθοδο επιλογής NUC και τον αλγόριθμο δημιουργίας μονοπατιού που θα χρησιμοποιηθούν. Επιπλέον, ο χρήστης μπορεί να καθορίσει κι άλλες παραμέτρους, που δε σχετίζονται άμεσα με τον αλγόριθμο αναγνώρισης σφαλμάτων αλλά με την ίδια την προσομοίωση, όπως είναι το μέγεθος του πλέγματος στο οποίο θα τρέξει ο αλγόριθμος, τον αριθμό των ελαττωματικών ελεγκτών που θα υπάρχουν στο πλέγμα και τις καθυστερήσεις που θα προκύπτουν από τη μετάδοση των πακέτων μεταξύ των διάφορων buffers.

Συγκεκριμένα, θα παρατηρούνται τρία είδη καθυστερήσεων κατά τη μετάδοση των πακέτων στο πλέγμα. Το πρώτο είδος είναι η καθυστέρηση στον buffer εισόδου, η οποία θα προκύπτει λόγω του χρόνου μετάδοσης του πακέτου (transmission delay) από τον buffer εισόδου στον buffer επεξεργασίας ενός κόμβου. Το δεύτερο είδος είναι η καθυστέρηση στον buffer επεξεργασίας που θα προκύπτει λόγω του χρόνου επεξεργασίας (transmission delay) και λόγω του χρόνου που θα χρειαστεί για την ενδεχόμενη μετάδοση του πακέτου (transmission delay) από τον buffer επεξεργασίας στον buffer εξόδου ενός κόμβου. Τέλος, το τρίτος είδος είναι η καθυστέρηση στον buffer εξόδου η οποία περιλαμβάνει τον χρόνου μετάδοσης του πακέτου (transmission delay) από τον buffer εξόδου ενός κόμβου στον buffer εισόδου του επόμενου κόμβου. Πριν την πραγματοποίηση της προσομοίωσης, ο χρήστης θα καθορίζει τον ελάχιστο αλλά και το μέγιστο χρόνο που θα είναι δυνατό να προκύπτει για το κάθε ένα από τα συγκεκριμένα είδη καθυστερήσεων.

Ακόμη, ο χρήστης μπορεί να επιλέξει κατά πόσο θα ήθελε να του εμφανίζεται μια γραφική απεικόνιση της προσομοίωσης. Η εμφάνιση της γραφικής απεικόνισης επιτρέπει στο χρήστη να βλέπει την κατάσταση όλων των ελεγκτών κατά τη διάρκεια της προσομοίωσης, καθώς και που βρίσκεται το κάθε πακέτο μέσα στο πλέγμα ανά πάσα στιγμή. Με τη χρήση ενός ολισθητή (slider), ο χρήστης θα μπορεί να καθορίσει

για πόσο μεγάλο χρονικό διάστημα θα βλέπει την τελική κατάσταση του πλέγματος μετά την ολοκλήρωση της προσομοίωσης, με τους μη λειτουργικούς ελεγκτές να είναι πλέον επισημασμένοι. Εντούτοις, η εμφάνιση της γραφικής απεικόνισης προκαλεί μεγάλη αύξηση στο χρόνο εκτέλεσης του αλγορίθμου και θα ήταν καλό να αποφεύγεται όταν πρωταρχικός στόχος του χρήστη είναι η άμεση εξαγωγή των αποτελεσμάτων της προσομοίωσης.



Σχήμα 4.1: Η γραφική διεπαφή χρήστη η οποία χρησιμοποιείται για τον καθορισμό των παραμέτρων της προσομοίωσης

4.4 Τρόπος Λειτουργίας Προσομοιωτή

Μέσω της Anylogic μπορεί κάποιος να δημιουργήσει διάφορα συμβάντα (events). Ένα συμβάν αποτελείται από μια ακολουθία εντελών και μπορεί είτε να εκτελείται μια φορά σε μια προκαθορισμένη χρονική στιγμή, είτε να είναι επαναλαμβανόμενο. Σε περίπτωση που ένα συμβάν είναι επαναλαμβανόμενο, τότε μπορεί να εκτελείται είτε μετά από ένα προκαθορισμένο χρονικό διάστημα, είτε όταν ικανοποιηθεί μια συνθήκη. Η διαχείριση της ροής ελέγχου του προσομοιωτή μου βασίζεται σε μεγάλο βαθμό στη χρήση των συμβάντων.

Συγκεκριμένα, υπάρχουν δύο συμβάντα τα οποία εκτελούνται μόνο μια φορά, κατά το πρώτο δευτερόλεπτο της προσομοίωσης, το Initialization και το ShowEdges. Το Initialization περιέχει εντολές οι οποίες διαβάζουν τις τιμές εισόδου που καταχώρησε ο χρήστης στην αρχική σελίδα της διεπαφής και παράλληλα δημιουργεί τη νέα σελίδα της διεπαφής η οποία θα ενημερώνει το χρήστη για την πρόοδο της προσομοίωσης καθώς θα πραγματοποιείται. Παράλληλα, κατά τη διάρκεια εκτέλεσης του συγκεκριμένου συμβάντος, ο προσομοιωτής τους μη λειτουργικούς ελεγκτές του πλέγματος. Ο αριθμός των μη λειτουργικών ελεγκτών καθορίζεται από το χρήστη, αλλά το ποιοι ακριβώς ελεγκτές του πλέγματος θα είναι μη λειτουργικοί, καθορίζεται τυχαία από τον προσομοιωτή. Το ShowEdges είναι υπεύθυνο για την παρουσίαση των συνδέσεων μεταξύ των ελεγκτών του πλέγματος στη διεπαφή, σε περίπτωση που ο χρήστης ζήτησε να βλέπει τη γραφική απεικόνιση του πλέγματος κατά τη διάρκεια της προσομοίωσης. Ο λόγος που αμφότερα τα συμβάντα εκτελούνται μόνο μια φορά στην αρχή, είναι διότι τόσο οι τιμές εισόδου του προσομοιωτή, όσο και οι συνδέσεις μεταξύ των ελεγκτών του πλέγματος, δεν πρόκειται να διαφοροποιηθούν κατά την πραγματοποίηση της προσομοίωσης.

Εντούτοις, υπάρχουν και αρκετά συμβάντα τα οποία είναι επαναλαμβανόμενα. Τα συγκεκριμένα συμβάντα είναι καθοριστικής σημασίας καθώς μέσω αυτών προσομοιώνονται οι λειτουργίες, οι υπολογισμοί και οι μεταβολές που πραγματοποιούνται σε κάθε κομμάτι (component) του συστήματος, είναι αυτό είναι η πηγή, είτε κάποιος ελεγκτής της υπερεπιφάνειας, είτε κάποιος buffer ενός κόμβου. Όλα αυτά τα συμβάντα, ξεκινούν να εκτελούνται από το πρώτο δευτερόλεπτο της

προσομοίωσης και επαναλαμβάνονται ανά μια μονάδα χρόνου. Παρακάτω θα αναλυθεί η σειρά με την οποία εκτελούνται τα συγκεκριμένα συμβάντα μέσα σε κάθε μονάδα χρόνου, αλλά και το τι ακριβώς πραγματοποιούν οι εντολές που περιλαμβάνονται στο καθένα από αυτά.

Πρώτο χρονικά εκτελείται το συμβάν ShowNodes. Το συμβάν αυτό, είναι υπεύθυνο για την παρουσίαση των ελεγκτών του πλέγματος αλλά και των buffers των ελεγκτών αυτών στη διεπαφή, σε περίπτωση που ο χρήστης ζήτησε να βλέπει τη γραφική απεικόνιση του πλέγματος κατά τη διάρκεια της προσομοίωσης. Ο λόγος που το ShowNodes είναι ένα επαναλαμβανόμενο συμβάν, σε αντίθεση με το ShowEdges, είναι καθώς η κατάσταση του κάθε κόμβου και του κάθε buffer είναι πιθανόν να αλλάξει σε κάθε μονάδα χρόνου και συνεπώς πρέπει ανά μονάδα χρόνου να υπολογίζεται εκ νέου το χρώμα με το οποίο εμφανίζεται ο κάθε ελεγκτής και ο κάθε buffer στη γραφική απεικόνιση.

Ακολούθως, εκτελείται το συμβάν CalculateTimeChanges. Οι εντολές του συγκεκριμένου συμβάντος αποσκοπούν στη μεταβολή της τιμής του ρολογιού της πηγής, έτσι ώστε η πηγή να γνωρίζει την ακριβή χρονική στιγμή στην οποία βρίσκεται η προσομοίωση.

Μετά από αυτό το συμβάν, εκτελούνται με τυχαία σειρά άλλα 4 συμβάντα τα οποία είναι υπεύθυνα για την προσομοίωση των λειτουργιών που διεκπεραιώνονται ανά μονάδα χρόνου στα διάφορα κομμάτια του συστήματος. Ο λόγος που τα 4 αυτά συμβάντα εκτελούνται με τυχαία σειρά, είναι διότι το σύστημα μας είναι ασύγχρονο και έτσι δεν είναι βέβαιο με ποια σειρά θα ολοκληρώσουν τα διάφορα κομμάτια τις λειτουργίες και τους υπολογισμούς τους. Τα συγκεκριμένα 4 συμβάντα είναι το CalculateGatewayChanges, το CalculateInputChanges, το CalculateNodeChanges και το CalculateOutputChanges.

Το CalculateGatewayChanges περιέχει εντολές για τη διεκπεραίωση των υπολογισμών της πηγής ανά μονάδα χρόνου. Στην πηγή τρέχει ο αλγόριθμος αναγνώρισης σφαλμάτων, οπότε όλοι οι υπολογισμοί του αλγορίθμου πραγματοποιούνται μέσα στο CalculateGatewayChanges και στις μεθόδους που καλούνται από αυτό. Επίσης, το CalculateGatewayChanges είναι υπεύθυνο για την εμφάνιση των αποτελεσμάτων μόλις

ολοκληρωθεί η εκτέλεση του αλγορίθμου, αλλά και για τον τερματισμό της προσομοίωσης.

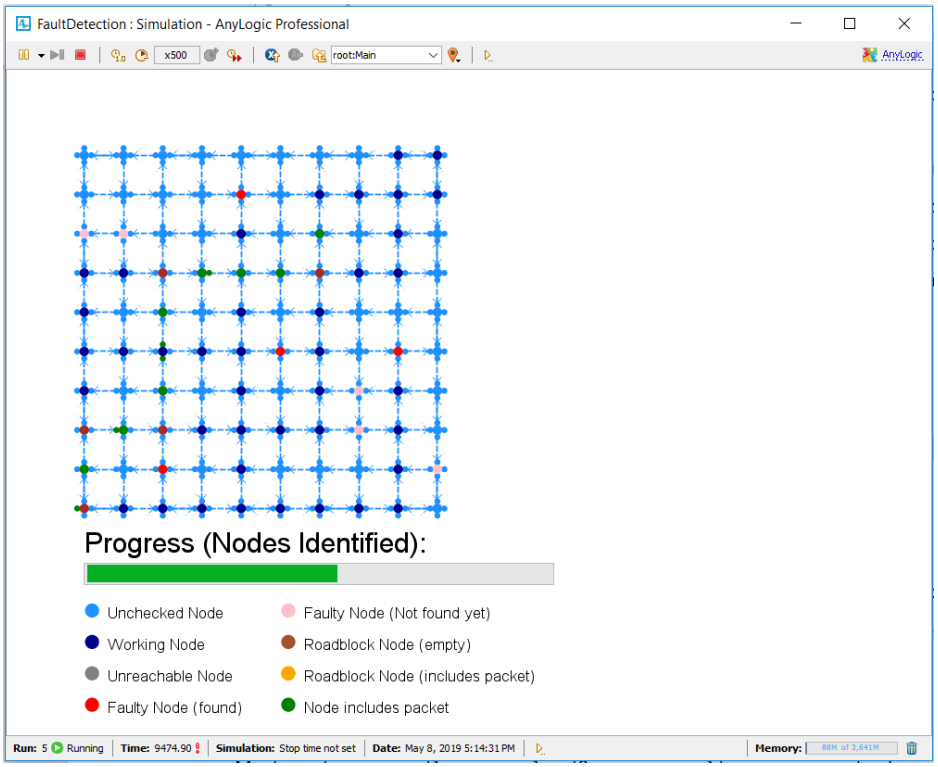
Οι εντολές των CalculateNodeChanges, CalculateOutputChanges και CalculateInputChanges εκτελούνται για κάθε κόμβο του πλέγματος. Τα συμβάντα αυτά είναι υπεύθυνα για τη μετάδοση των πακέτων από τον buffer εισόδου του κόμβου στον buffer επεξεργασίας, από τον buffer επεξεργασίας του κόμβου στον buffer εξόδου και από τον buffer εξόδου του κόμβου στον buffer εισόδου του επόμενου κόμβου, αντιστοίχως. Το CalculateOutputChanges είναι επίσης υπεύθυνο για το διάβασμα και την επεξεργασία του πακέτου από τον κόμβο, αλλά και για τον καθορισμού της εξόδου που θα χρησιμοποιηθεί για το κάθε πακέτο, βάσει του αλγόριθμου δρομολόγησης ο οποίος ουσιαστικά υλοποιείται μέσω της εκτέλεσης του συγκεκριμένου συμβάντος. Η μετάδοση των πακέτων βεβαίως δεν πραγματοποιείται κάθε φορά που εκτελούνται τα συγκεκριμένα συμβάντα, δηλαδή σε κάθε μονάδα χρόνου της προσομοίωσης. Για να διασφαλίσω ότι το μοντέλο μου είναι όσο πιο ρεαλιστικό γίνεται, κάθε φορά που εισέρχεται ένα πακέτο σε κάποιο buffer, υπολογίζεται το χρονικό διάστημα κατά το οποίο το πακέτο θα παραμείνει μέσα στο συγκεκριμένο buffer. Ο χρόνος αυτός είναι διαφορετικός για κάθε ξεχωριστή μετάδοση πακέτου και καθορίζεται τυχαία από τα διαστήματα των καθυστερήσεων που όρισε ο χρήστης, αναλόγως του είδους του buffer στο οποίο βρίσκεται το πακέτο. Η τυχαία επιλογή της τιμής από τα διαστήματα που όρισε ο χρήστης γίνεται χρησιμοποιώντας κανονική κατανομή, με μέση τιμή (maxDelay+maxDelay)/2 και διακύμανση maxDelay-minDelay, όπου minDelay και maxDelay η ελάχιστη και μέγιστη τιμή που όρισε ο χρήστης για το κάθε είδος καθυστέρησης αντιστοίχως.

4.5 Εμφάνιση Αποτελεσμάτων Προσομοίωσης

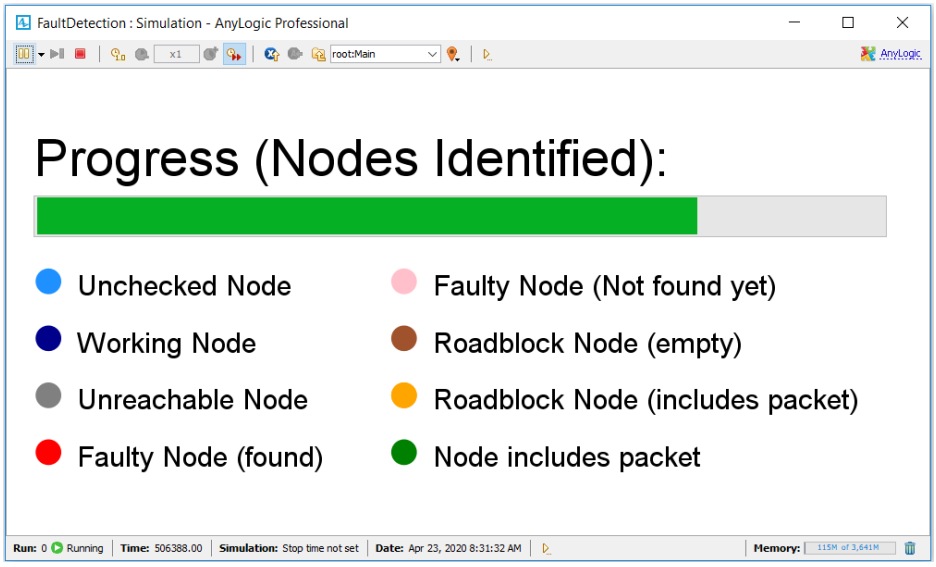
Η ενημέρωση του χρήστη γίνεται τόσο με τη χρήση της γραφικής απεικόνισης κατά τη διάρκεια της προσομοίωσης, όσο και με διάφορα μηνύματα στην οθόνη κατά την ολοκλήρωση της προσομοίωσης. Παράλληλα, ο προσομοιωτής αποθηκεύει σε συγκεκριμένο αρχείο στον υπολογιστή το ιστορικό και αποτελέσματα όλων των προσομοιώσεων.

Σε περίπτωση που ο χρήστης επιλέξει να βλέπει τη γραφική απεικόνιση κατά τη διάρκεια της προσομοίωσης , ο προσομοιωτής σχηματίζει το πλέγμα βάσει του μεγέθους του πλέγματος και του αριθμού των μη λειτουργικών ελεγκτών που έχουν καθοριστεί από το χρήστη. Στην απεικόνιση του πλέγματος φαίνονται οι ελεγκτές, οι συνδέσεις μεταξύ των ελεγκτών, καθώς και οι buffers εισόδου και εξόδου του κάθε κόμβου. Κατά τη διάρκεια της προσομοίωσης η μετάδοση πακέτων στο πλέγμα γίνεται αντιληπτή από το χρήστη, καθώς το χρώμα του κάθε κόμβου αλλά και των buffer του μεταβάλλεται ανάλογα με την κατάσταση στην οποία βρίσκονται. Όσον αφορά τους ελεγκτές, το χρώμα τους εξαρτάται από το σύνολο ελεγκτών στο οποίο ανήκουν (μη ελεγμένοι, λειτουργικοί, μη λειτουργικοί, μη προσβάσιμοι), από την παρουσία κάποιου πακέτου μέσα σε αυτούς αλλά και από το κατά πόσο είναι ή όχι ελεγκτές οδοφράγματος. Όταν κάποιος ελεγκτής είναι μη ελεγμένος τότε αυτός υπό κανονικές συνθήκες θα έχει είτε γαλάζιο, είτε ροζ χρώμα. Γαλάζιοι είναι όσοι ελεγκτές είναι λειτουργικοί αλλά δεν έχουν αναγνωριστεί ακόμη, ενώ ροζ είναι όσοι ελεγκτές είναι μη λειτουργικοί αλλά δεν έχουν αναγνωριστεί ακόμη. Όταν ένας ελεγκτής αναγνωρίζεται, το χρώμα του μεταβάλλεται σε μπλε εάν πρόκειται για λειτουργικό κόμβο, σε κόκκινο εάν πρόκειται για μη λειτουργικό κόμβο και σε γκρίζο εάν πρόκειται για μη προσβάσιμο κόμβο. Επίσης, όταν κάποιος ελεγκτής περιέχει πακέτο το χρώμα του μεταβάλλεται σε πράσινο, ενώ όταν μετατρέπεται σε κόμβο οδοφράγματος το χρώμα του μεταβάλλεται σε καφέ. Τέλος, όταν κάποιος ελεγκτής οδοφράγματος περιέχει πακέτο, απεικονίζεται με κίτρινο χρώμα. Όσον αφορά τους buffers των ελεγκτών, αυτοί εμφανίζονται με γαλάζιο χρώμα, το οποίο αλλάζει σε πράσινο όταν κάποιος buffer περιέχει πακέτο. Η συγκεκριμένη ανάλυση των χρωμάτων παρουσιάζεται σε υπόμνημα και κατά τη διάρκεια της προσομοίωσης. Ακόμη, υπάρχει και μια μπάρα προόδου (progress bar) η οποία ενημερώνει το χρήστη για την πρόοδο της προσομοίωσης και συγκεκριμένα για

το ποσοστό ελεγκτών που απομένει να αναγνωριστούν για να ολοκληρωθεί η προσομοίωση. Συνεπώς, η γραφική απεικόνιση επιτρέπει στο χρήστη να έχει πλήρη και συνεχή αντίληψη για την κατάσταση των ελεγκτών του πλέγματος, αλλά και για την πρόοδο που έχει υπάρξει σχετικά με την αναγνώριση σφαλμάτων, καθ' όλη τη διάρκεια της προσομοίωσης.

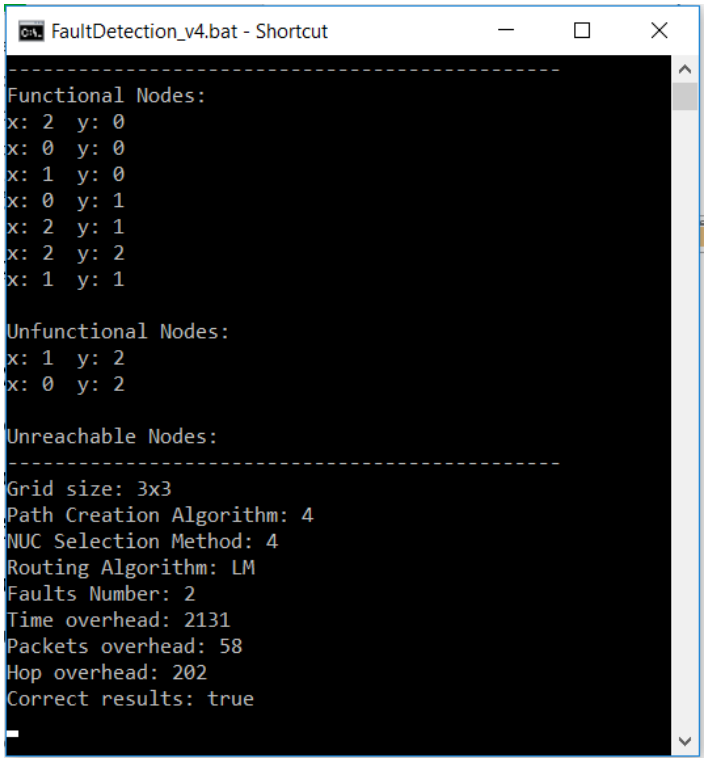


Σχήμα 4.2: Η γραφική απεικόνιση, η οποία παρουσιάζεται στο χρήστη κατά τη διάρκεια εκτέλεσης του αλγορίθμου για ένα πλέγμα μεγέθους 10x10



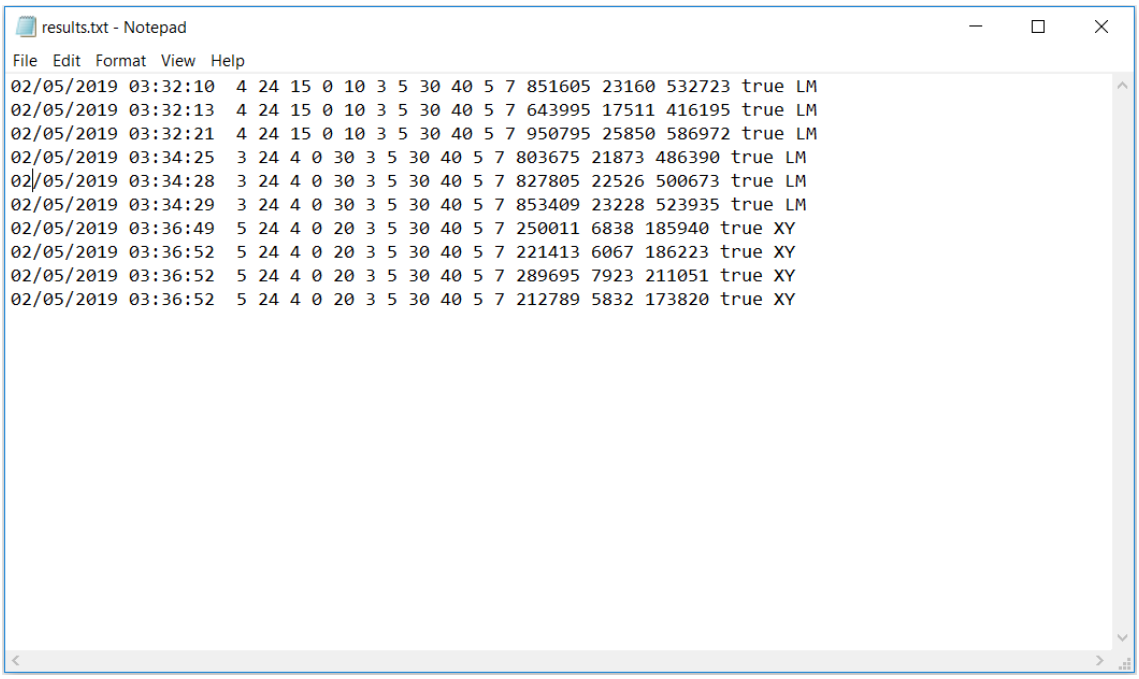
Σχήμα 4.3: Η ενημέρωση για την πρόοδο του αλγορίθμου που παρουσιάζεται σε κάποιο χρήστη που δεν έχει επιλέξει την εμφάνιση της γραφικής απεικόνισης

Μετά το πέρας της εκτέλεσης του αλγορίθμου, τα αποτελέσματα που προέκυψαν από την αναγνώριση που έγινε τυπώνονται στην οθόνη. Συγκεκριμένα, τυπώνονται τα τρία σύνολα ελεγκτών στα οποία έχει διαχωρίσει ο αλγόριθμος του ελεγκτές του πλέγματος. Δηλαδή, κάτω από τις αντίστοιχες επικεφαλίδες τυπώνονται οι ελεγκτές οι οποίοι περιέχονται στα σύνολα των λειτουργικών, των μη λειτουργικών και των μη προσβάσιμων ελεγκτών. Ακόμη, στην οθόνη τυπώνονται και μερικές επιπλέον πληροφορίες που αφορούν την προσομοίωση που μόλις έχει ολοκληρωθεί, όπως το μέγεθος του πλέγματος, ο αριθμός των μη λειτουργικών ελεγκτών που υπήρχαν σε αυτό και οι αλγόριθμοι δρομολόγησης και δημιουργίας μονοπατιού όπως και η μέθοδος επιλογής NUC που χρησιμοποιήθηκαν. Επίσης, τυπώνονται και κάποιες πληροφορίες που αφορούν το κόστος της συγκεκριμένης εκτέλεσης του αλγορίθμου και συγκεκριμένα το χρονικό κόστος (time overhead), το κόστος σε πακέτα (packet overhead) και το κόστος σε μεταδόσεις (hop overhead) που υπήρξε. Τέλος, ο προσομοιωτής επιβεβαιώνει αν τα αποτελέσματα που εξήχθησαν από τον αλγόριθμο αναγνώρισης σφαλμάτων είναι έγκυρα, κάτι που ισχύει σε όλες τις περιπτώσεις.



Σχήμα 4.4: Τα αποτελέσματα και οι πληροφορίες που εμφανίζονται στην οθόνη μετά την ολοκλήρωση της εκτέλεσης του αλγορίθμου

Επιπλέον, στο αρχείο results.txt αποθηκεύονται διάφορες πληροφορίες σχετικά με την κάθε προσομοίωση που πραγματοποιείται. Συγκεκριμένα, για κάθε προσομοίωση που ολοκληρώνεται, δημιουργείται μια νέα γραμμή στο αρχείο στην οποία αναγράφεται η ημερομηνία και η ώρα τερματισμού της προσομοίωσης όπως και οι παράμετροι εισόδου της συγκεκριμένης προσομοίωσης. Δηλαδή, για κάθε προσομοίωση τυπώνεται ο αλγόριθμος δρομολόγησης, ο αλγόριθμος δημιουργίας μονοπατιού, η μέθοδος επιλογής NUC, το μέγεθος πλέγματος και ο αριθμός των μη λειτουργικών ελεγκτών που υπήρχαν σε αυτό, όπως επίσης και τα διάφορα διαστήματα καθυστερήσεων, όπως αυτά ορίστηκαν από το χρήστη. Ακόμη, στο αρχείο τυπώνονται και οι πληροφορίες που είναι σχετικές με το κόστος της εκτέλεσης του αλγορίθμου κατά τη συγκεκριμένη προσομοίωση και συγκεκριμένα το χρονικό κόστος (time overhead), το κόστος σε πακέτα (packet overhead) και το κόστος σε μεταδόσεις (hop overhead) που υπήρξε. Τέλος, για την κάθε προσομοίωση τυπώνεται στο αρχείο και η επιβεβαίωση ότι τα αποτελέσματα που εξήχθησαν από τον αλγόριθμο αναγνώρισης σφαλμάτων είναι έγκυρα.



Σχήμα 4.5: Το αρχείο results.txt το οποίο περιέχει πληροφορίες για τις τελευταίες προσομοιώσεις που διενεργήθηκαν

4.6 Άλλα εργαλεία που χρησιμοποίησα

Πέρα από την Anylogic χρησιμοποίησα και διάφορα άλλα εργαλεία λογισμικού τόσο για την πραγματοποίηση των προσομοιώσεων μου, όσο και για την ανάλυση των αποτελεσμάτων που προέκυψαν και τη δημιουργία αυτής της αναφοράς.

Συγκεκριμένα, χρησιμοποίησα σε πολύ μεγάλο βαθμό το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) Eclipse Oxygen. Χρησιμοποιώντας το IDE αυτό και τη γλώσσα προγραμματισμού Java, είχα δημιουργήσει αρχικά ένα πιο απλό προσομοιωτή που μου επέτρεπε να ελέγχω την ορθότητα των ιδεών μου. Επιπρόσθετα, το χρησιμοποίησα για να υλοποιήσω αλγόριθμους στη γλώσσα προγραμματισμού Java, όπως για παράδειγμα την παραλλαγή του αλγόριθμου Bellman-Ford, προτού τους ενσωματώσω στον προσομοιωτή μου στην Anylogic. Ακόμη, χρησιμοποίησα το Eclipse για να δημιουργήσω ένα πρόγραμμα το οποίο είναι υπεύθυνο να διαβάζει και να αναλύει τα αποτελέσματα των προσομοιώσεων που έγιναν από τον προσομοιωτή στην Anylogic.

Για τη συγγραφή του ψευδοκώδικα χρησιμοποίησα τον επεξεργαστή κειμένου και πηγαίου κώδικα (text editor and source code editor) Notepad ++, ενώ για τη δημιουργία των διαφόρων διαγραμμάτων χρησιμοποίησα το διαδικτυακό λογισμικό δημιουργίας διαγραμμάτων (online diagram software) draw.io.

Επιπλέον, για την επεξεργασία των δεδομένων και τη δημιουργία των γραφικών παραστάσεων που αναπαριστούν τα αποτελέσματα των προσομοιώσεων χρησιμοποίησα το πρόγραμμα λογιστικών φύλλων Microsoft Excel και για τη συγγραφή της παρούσας αναφοράς το πρόγραμμα επεξεργασίας κειμένου Microsoft Word.

Κεφάλαιο 5

Αποτελέσματα Προσομοιώσεων

5.1 Τρόπος Συλλογής και Ανάλυσης Αποτελεσμάτων

Μόλις ολοκλήρωσα τη δημιουργία του μοντέλου στον προσομοιωτή, έθεσα ως βασικό στόχο τη συλλογή αρκετών δεδομένων τα οποία να οδηγούν στην εξαγωγή ασφαλών συμπερασμάτων για τον αλγόριθμο μου. Συνεπώς, υπήρξε η ανάγκη για πραγματοποίηση μεγάλου αριθμού προσομοιώσεων, έτσι ώστε να υπάρχει ικανοποιητικό δείγμα για να γίνει η ανάλυση.

Για να διασφαλίσω την εύκολη διεξαγωγή όλων αυτών των προσομοιώσεων, δημιούργησα μια εναλλακτική έκδοση του προσομοιωτή η οποία επιτρέπει τη δημιουργία πολλών διαδοχικών προσομοιώσεων για ένα εύρος τιμών που καθορίζεται για τις παραμέτρους εισόδου. Συγκεκριμένα, στη συγκεκριμένη έκδοση του προσομοιωτή ο χρήστης δεν καθορίζει απλά μια τιμή, αλλά ένα εύρος τιμών για τη σειρά επιλογής NUC και τον αλγόριθμο δημιουργίας μονοπατιού που θα χρησιμοποιηθούν, το μέγεθος του πλέγματος αλλά και τον αριθμό των μη λειτουργικών ελεγκτών που θα υπάρχουν σε αυτό. Ο προσομοιωτής θα δημιουργήσει μια εκτέλεση του αλγορίθμου για κάθε πιθανό συνδυασμό τιμών που δύναται να πάρουν αυτές οι τέσσερεις παράμετροι εισόδου, βάσει του εύρους των τιμών που καθόρισε ο χρήστης. Ακόμη, ο χρήστης έχει την ευκαιρία να ζητήσει να γίνουν περισσότερες από μια εκτελέσεις για κάθε πιθανό συνδυασμό τιμών εισόδου που μπορεί να προκύψει, σε περίπτωση που το επιθυμεί.

Μόλις ολοκληρωθούν οι προσομοιώσεις, το αρχείο results.txt θα περιλαμβάνει τις παραμέτρους εισόδου τους, όπως και τα διάφορα είδη κόστους (overhead) του αλγορίθμου που προέκυψαν κατά την εκτέλεση της κάθε προσομοίωσης. Για την ανάλυση των δεδομένων που περιέχονται στο αρχείο αυτό, δημιούργησα ένα αναλυτή αποτελεσμάτων στη Java, χρησιμοποιώντας το ολοκληρωμένο περιβάλλον ανάπτυξης Eclipse Oxygen.

Ο συγκεκριμένος αναλυτής αποτελεσμάτων έχει δύο λειτουργίες. Η πρώτη λειτουργία είναι η αφαίρεση των αποκλινόντων δεδομένων (outliers). Σε κάποιες προσομοιώσεις λόγω του ότι ο ελεγκτής που είναι συνδεδεμένος με την πηγή, ή ελεγκτές συνδεδεμένοι σε αυτόν είναι μη λειτουργικοί, η συντριπτική πλειοψηφία των ελεγκτών του πλέγματος αναγνωρίζονται ως μη προσβάσιμοι καθώς δεν μπορεί να δημιουργηθεί κάποιο μονοπάτι προς αυτούς. Ο καθορισμός ενός κόμβου ως μη προσβάσιμου έχει μηδενικό κόστος (overhead) εφόσον είναι μια διαδικασία που γίνεται μέσω του αλγορίθμου αναγνώρισης που τρέχει στην πηγή, χωρίς να χρειάζεται να πραγματοποιηθεί οποιαδήποτε αποστολή πακέτου στο πλέγμα. Συνεπώς, ο χρόνος διεκπεραίωσης (time overhead) των συγκεκριμένων προσομοιώσεων είναι κατά πολύ μικρότερος από το μέσο χρόνο διεκπεραίωσης οποιασδήποτε άλλης προσομοίωσης όπου δεν υπάρχει αυτό το εξαιρετικά μεγάλο ποσοστό μη προσβάσιμων ελεγκτών. Έτσι, για να μην επηρεαστεί ο μέσος χρόνος διεκπεραίωσης των προσομοιώσεων μόνο για συγκεκριμένους συνδυασμούς παραμέτρων εισόδων όπου έτυχε να υπάρξει αυτός ο μεγάλος αριθμός μη προσβάσιμων ελεγκτών, επέλεξα να διαγράψω εντελώς τις συγκεκριμένες αποκλίνουσες προσομοιώσεις από το δείγμα μου. Συγκεκριμένα, αφαίρεσα από το δείγμα μου όλες τις προσομοιώσεις όπου ο χρόνος διεκπεραίωσης του αλγορίθμου ήταν μικρότερος από 50000 μονάδες χρόνου.

Η δεύτερη λειτουργία του αναλυτή αποτελεσμάτων είναι το διάβασμα των αποτελεσμάτων των προσομοιώσεων και ο υπολογισμός του μέσου χρονικού κόστους (average time overhead), του μέσου κόστους σε πακέτα (average packet overhead) και του μέσου κόστους σε μεταδόσεις (average hop overhead) μεταξύ των προσομοιώσεων οι οποίες έχουν κοινές παραμέτρους εισόδου. Τα κόστη αυτά τυπώνονται στην οθόνη για κάθε συνδυασμό τιμών των παραμέτρων εισόδου.

Για να μπορούν να προκύψουν χρήσιμα αποτελέσματα και ασφαλή συμπεράσματα από τα πειράματα που πραγματοποίησα, φρόντισα κάθε φορά να μεταβάλλεται μόνο ο αριθμός των σφαλμάτων που υπάρχουν στο πλέγμα και ακόμη μια παράμετρος εισόδου του αλγορίθμου, δηλαδή είτε ο αλγόριθμος δρομολόγησης, είτε ο αλγόριθμος δημιουργίας μονοπατιού, είτε η μέθοδος επιλογής NUC. Συνεπώς, η συγκεκριμένη παράμετρος του αλγορίθμου που μεταβάλλεται όπως και ο αριθμός των σφαλμάτων

στο πλέγμα αποτελούν τις ανεξάρτητες μεταβλητές στα πειράματα μου, ενώ τα μέσα κόστη αποτελούν τις εξαρτημένες μεταβλητές.

Ο λόγος που διατηρούσα κάθε φορά σταθερές τις υπόλοιπες παραμέτρους, είναι για να διασφαλίσω ότι δε θα επηρεαστούν τα αποτελέσματα μου από τυχόν μεταβολή τους, θέτοντας τις κατ' αυτό τρόπο ως ελεγχόμενες μεταβλητές. Η μόνη μεταβλητή η οποία είναι μη ελεγχόμενη στα πειράματα που πραγματοποίησα είναι η τοποθεσία των μη λειτουργικών ελεγκτών στο πλέγμα. Ο λόγος που η συγκεκριμένη μεταβλητή είναι μη ελεγχόμενη είναι καθώς για k μη λειτουργικούς ελεγκτές στο πλέγμα σε ένα πλέγμα με n ελεγκτές, χρειάζεται να πραγματοποιηθούν $\binom{n}{\mathrm{k}}$ προσομοιώσεις για να ληφθούν υπόψη όλες οι περιπτώσεις. Αναλογιζόμενοι ότι το n στη δική μας περίπτωση είναι ίσο με 576 εφόσον μελετάμε πλέγμα μεγέθους 24x24, για κάθε αριθμό σφαλμάτων θα έπρεπε να πραγματοποιηθούν $\binom{576}{\mathrm{k}}$ προσομοιώσεις, κάτι που είναι αδύνατο με την υπολογιστική ισχύ που έχουμε σήμερα. Έτσι, η επιλογή της τοποθεσίας των μη λειτουργικών ελεγκτών στο πλέγμα γίνεται τυχαία και η μείωση της επίδρασης της συγκεκριμένης μη ελεγχόμενης μεταβλητής επιτυγχάνεται μέσω της αφαίρεσης των αποκλινόντων δεδομένων.

Εφόσον ο αναλυτής αποτελεσμάτων εξάγει τα τελικά αποτελέσματα, αυτά μεταφέρονται στην Microsoft Excel για να αναλυθούν περαιτέρω και για να εντοπιστούν τυχόν σχέσεις μέσω εξαρτημένων και ανεξάρτητων μεταβλητών, μέσω των γραφικών παραστάσεων που δημιουργούνται. Κύριος στόχος της ανάλυσης που πραγματοποιείται είναι ο εντοπισμός των τιμών των παραμέτρων εισόδου του αλγορίθμου, οι οποίες οδηγούν στην ταχύτερη και αποδοτικότερη εκτέλεση του.

Η αξιολόγηση των διάφορων παραμέτρων γίνεται βάσει του κόστους που προκύπτει κάθε φορά. Το χρονικό κόστος υποδεικνύει το χρόνο που χρειάζεται ο αλγόριθμος αναγνώρισης σφαλμάτων για να ολοκληρώσει την εκτέλεση του. Μεγαλύτερο χρονικό κόστος σημαίνει ότι η αναγνώριση των σφαλμάτων πραγματοποιείται με πιο αργό ρυθμό και θα χρειαστεί μεγαλύτερο χρονικό διάστημα για να ολοκληρωθεί. Το κόστος σε πακέτα είναι μια εναλλακτική μετρική η οποία υποδεικνύει τον αριθμό των πακέτων που χρειάζεται να μεταδοθούν από την πηγή στο πλέγμα μέχρι να ολοκληρωθεί η εκτέλεση του αλγορίθμου. Επίσης, υπάρχει και το κόστος σε μεταβάσεις. Μια

μετάβαση (hop), ορίζεται ως η μετάδοση ενός πακέτου από ένα ελεγκτή σε ένα γειτονικό του ελεγκτή. Έτσι, το κόστος σε μεταβάσεις μετρά το συνολικό αριθμό των μεταβάσεων που πραγματοποιούνται στο πλέγμα κατά την εκτέλεση του αλγόριθμου αναγνώρισης σφαλμάτων. Η συγκεκριμένη μετρική είναι ιδιαίτερα χρήσιμη, καθώς για να μπορέσουν να επεξεργαστούν και να δρομολογήσουν τα πακέτα οι ελεγκτές χρειάζεται να καταναλώσουν περισσότερη ενέργεια. Συνεπώς, όσο μικρότερος είναι ο αριθμός των μεταβάσεων που πραγματοποιείται, τότε μικρότερη είναι και η ενέργεια που απαιτείται από το σύστημα για την ολοκλήρωση της εκτέλεσης του αλγορίθμου.

5.2 Αποτελέσματα για Μεθόδους Επιλογής NUC

5.2.1 Περιγραφή Πειραμάτων

Για να μελετήσω τον τρόπο με τον οποίο οι μέθοδοι επιλογής NUC επηρεάζουν την επίδοση του αλγορίθμου αναγνώρισης σφαλμάτων, διενήργησα συνολικά έξι πειράματα, χρησιμοποιώντας διαφορετικό συνδυασμό αλγόριθμου δρομολόγησης και αλγόριθμου δημιουργίας μονοπατιού κάθε φορά. Όλα τα πειράματα πραγματοποιήθηκαν για μέγεθος πλέγματος 24x24 με σταθερά όρια καθυστερήσεων, ενώ ο αριθμός των σφαλμάτων στο πλέγμα σε κάθε πείραμα διακυμάνθηκε από τα 0 μέχρι τα 100 σφάλματα. Για λόγους ταχύτερης ολοκλήρωσης των προσομοιώσεων, τα πειράματα πραγματοποιήθηκαν για όλους τους αριθμούς σφαλμάτων του συγκεκριμένου διαστήματος [0,100] που είναι πολλαπλάσια του 5.

Κατά την εκτέλεση των πειραμάτων 1 μέχρι 5, για κάθε μέθοδο επιλογής NUC πραγματοποιήθηκαν 100 εκτελέσεις του αλγορίθμου για όλους τους αριθμούς σφαλμάτων. Από τα αποτελέσματα των εκτελέσεων αυτών αφαιρέθηκαν όπως και σε κάθε άλλη περίπτωση τα αποκλίνοντα δεδομένα. Έπειτα, για κάθε συνδυασμό μεθόδου επιλογής NUC και αριθμού σφαλμάτων υπολογίστηκε ο μέσος όρος του χρονικού κόστους, του κόστους σε πακέτα και του κόστους σε μεταδόσεις που προέκυψε από τις διάφορες εκτελέσεις του αλγορίθμου. Οι συγκεκριμένοι μέσοι όροι κόστους ανά αριθμό σφαλμάτων φαίνονται στις γραφικές παραστάσεις 5.13, 5.14 και 5.15 για το πείραμα 4, ενώ οι εν λόγω γραφικές παραστάσεις των υπόλοιπων πειραμάτων μπορούν να βρεθούν στο παράρτημα Α της παρούσας αναφοράς

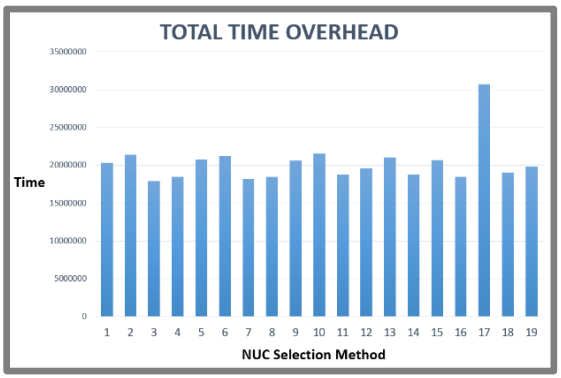
Ακολούθως, για κάθε μέθοδο επιλογής NUC υπολογίστηκε το άθροισμα των μέσων όρων κόστους για όλους τους αριθμούς σφαλμάτων. Έτσι, σε κάθε πείραμα προέκυψε ένα συνολικό κόστος ανά μέθοδο επιλογής NUC. Οι γραφικές παραστάσεις που παρουσιάζουν τα συγκεκριμένα συνολικά κόστη για κάθε μέθοδο επιλογής NUC είναι οι 5.1, 5.2, 5.3 για το πείραμα 1, οι 5.4, 5.5, 5.6 για το πείραμα 2, οι 5.7, 5.8, 5.9 για το πείραμα 3, οι 5.10, 5.11, 5.12 για το πείραμα 4 και οι 5.16, 5.17 και 5.18 για το πείραμα 5.

Η διαφορά του πειράματος 5 από τα υπόλοιπα πειράματα προκύπτει από το γεγονός ότι στο συγκεκριμένο πείραμα χρησιμοποιήθηκε ο αλγόριθμος δημιουργίας μονοπατιού 5. Ο συγκεκριμένος αλγόριθμος δημιουργίας μονοπατιού, λόγω της ιδιαιτερότητας που τον χαρακτηρίζει, μπορεί να λειτουργήσει αποκλειστικά σε συνδυασμό με τη μέθοδο επιλογής NUC 20, η οποία καθορίζει τη δημιουργία ενός προκαθορισμένου μονοπατιού. Συνεπώς, εφόσον στο συγκεκριμένο πείραμα η μέθοδος επιλογής NUC δεν ήταν δυνατό να μεταβληθεί, αποφάσισα να τρέξω τις εκτελέσεις για δύο διαφορετικούς αλγόριθμους δρομολόγησης, το XY και το Linear Motion. Οι μέσοι όροι κόστους των εκτελέσεων που διενεργήθηκαν ανά αριθμό σφαλμάτων φαίνονται στις γραφικές παραστάσεις 5.19, 5.20 και 5.21.

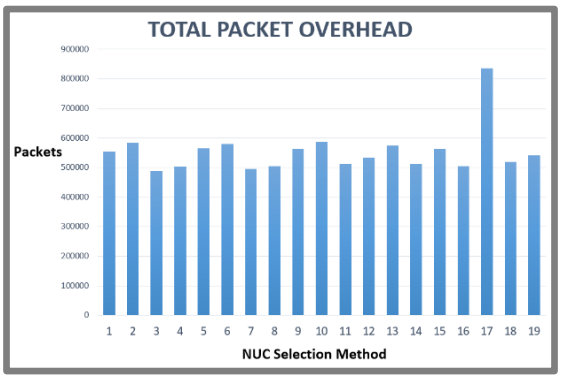
5.2.2 Αποτελέσματα Πειραμάτων

Πείραμα 1	
Αλγόριθμος Δημιουργίας Μονοπατιού	Αλγόριθμος 1
Αλγόριθμος Δρομολόγησης	Linear Motion

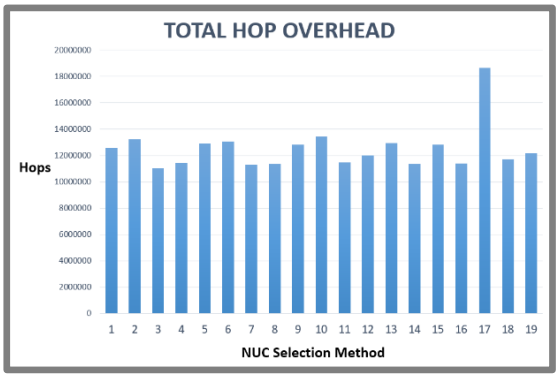
Πίνακας 5.1: Παράμετροι πειράματος 1



Σχήμα 5.1: Το συνολικό χρονικό κόστος για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 1



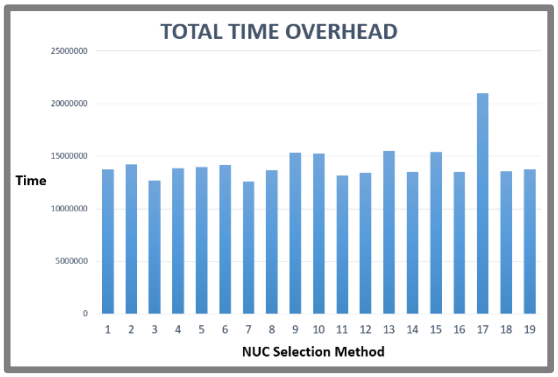
Σχήμα 5.2: Το συνολικό κόστος σε πακέτα για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 1



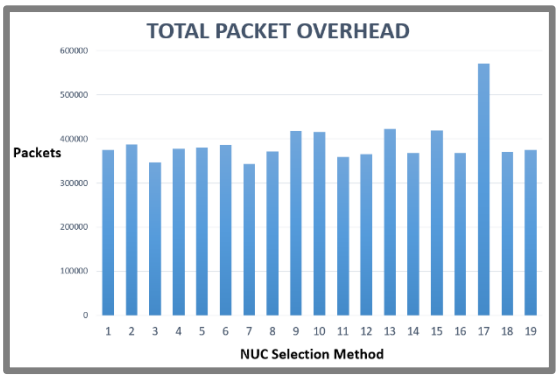
Σχήμα 5.3: Το συνολικό κόστος σε μεταδόσεις για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 1

Πείραμα 2	
Αλγόριθμος Δημιουργίας Μονοπατιού	Αλγόριθμος 2
Αλγόριθμος Δρομολόγησης	Linear Motion

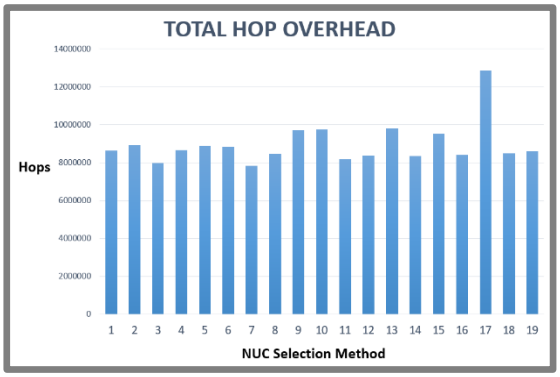
Πίνακας 5.2: Παράμετροι πειράματος 2



Σχήμα 5.4: Το συνολικό χρονικό κόστος για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 2



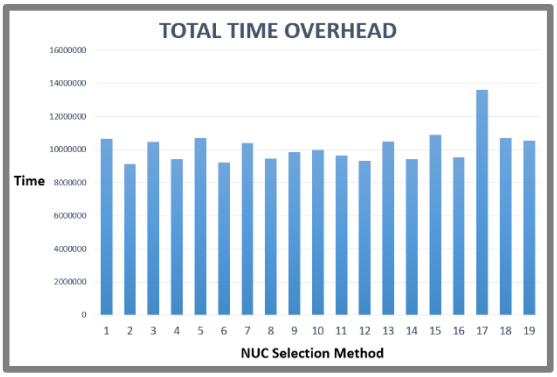
Σχήμα 5.5: Το συνολικό κόστος σε πακέτα για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 2



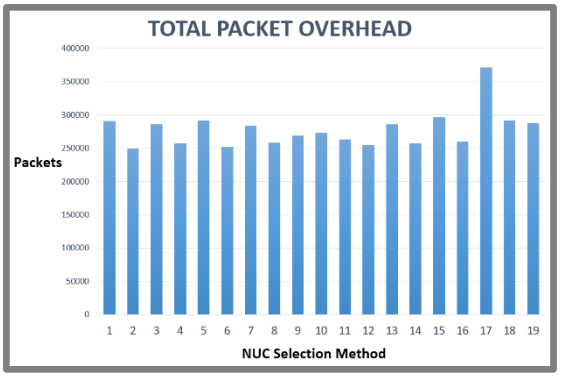
Σχήμα 5.6: Το συνολικό κόστος σε μεταδόσεις για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 2

Πείραμα 3	
Αλγόριθμος Δημιουργίας Μονοπατιού	Αλγόριθμος 3
Αλγόριθμος Δρομολόγησης	Linear Motion

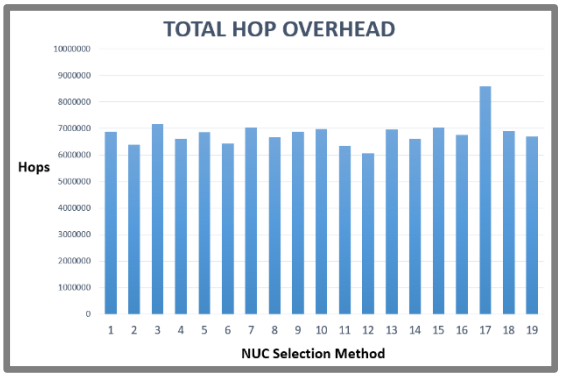
Πίνακας 5.3: Παράμετροι πειράματος 3



Σχήμα 5.7: Το συνολικό χρονικό κόστος για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 3



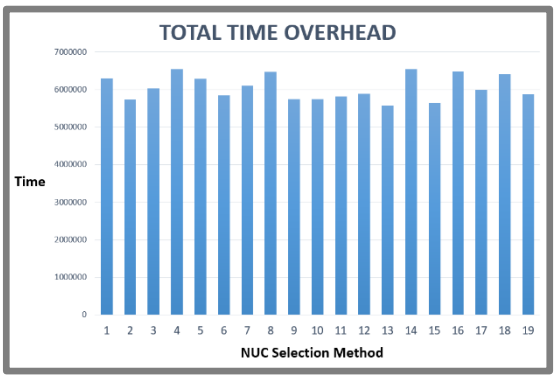
Σχήμα 5.8: Το συνολικό κόστος σε πακέτα για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 3



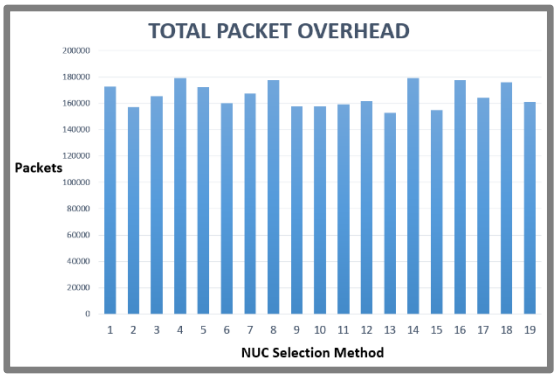
Σχήμα 5.9: Το συνολικό κόστος σε μεταδόσεις για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 3

Πείραμα 4	
Αλγόριθμος Δημιουργίας Μονοπατιού	Αλγόριθμος 4
Αλγόριθμος Δρομολόγησης	Linear Motion

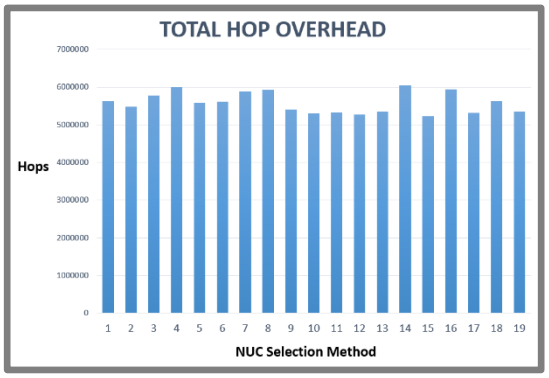
Πίνακας 5.4: Παράμετροι πειράματος 4



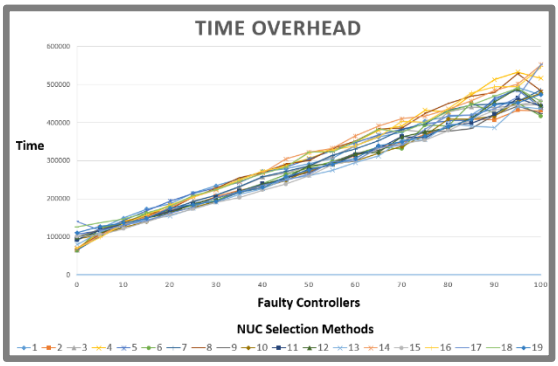
Σχήμα 5.10: Το συνολικό χρονικό κόστος για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 4



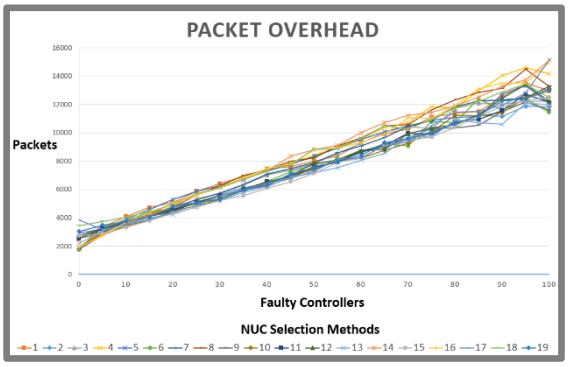
Σχήμα 5.11: Το συνολικό κόστος σε πακέτα για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 4



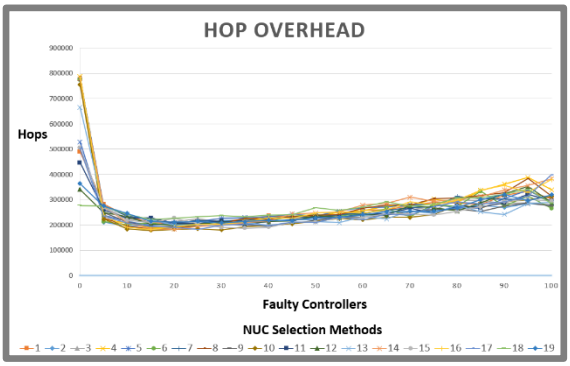
Σχήμα 5.12: Το συνολικό κόστος σε μεταδόσεις για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 4



Σχήμα 5.13: Το χρονικό κόστος σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 4



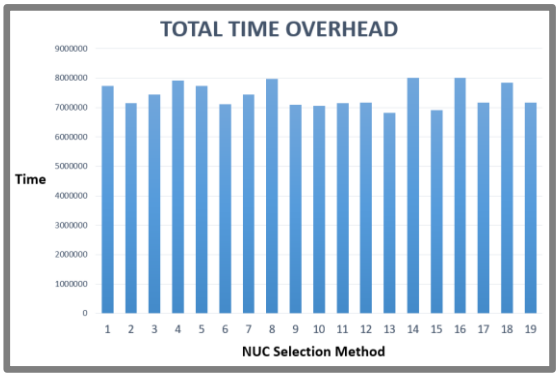
Σχήμα 5.14: Το κόστος σε πακέτα σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 4



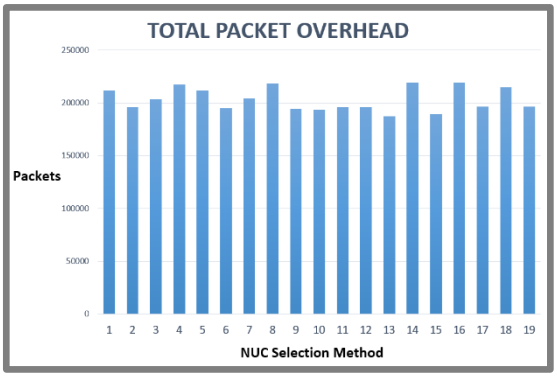
Σχήμα 5.15: Το κόστος σε μεταδόσεις σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 4

Πείραμα 5	
Αλγόριθμος Δημιουργίας Μονοπατιού	Αλγόριθμος 4
Αλγόριθμος Δρομολόγησης	XY

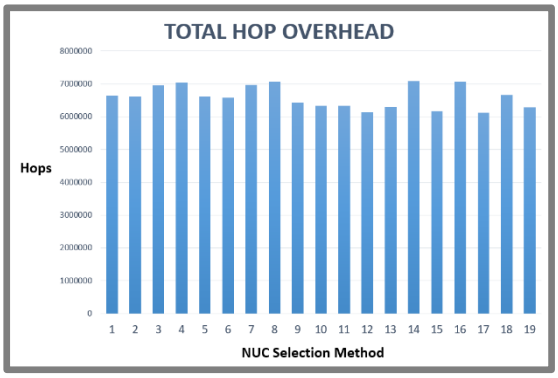
Πίνακας 5.5: Παράμετροι πειράματος 5



Σχήμα 5.16: Το συνολικό χρονικό κόστος για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 5



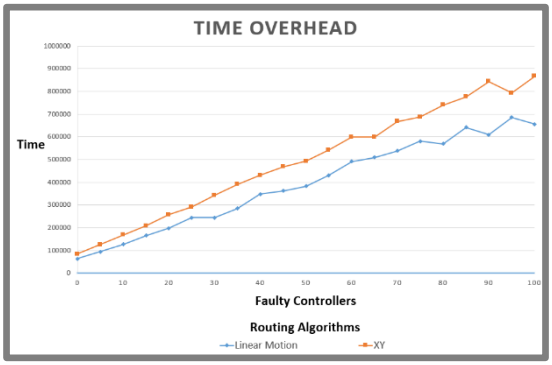
Σχήμα 5.17: Το συνολικό κόστος σε πακέτα για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 5



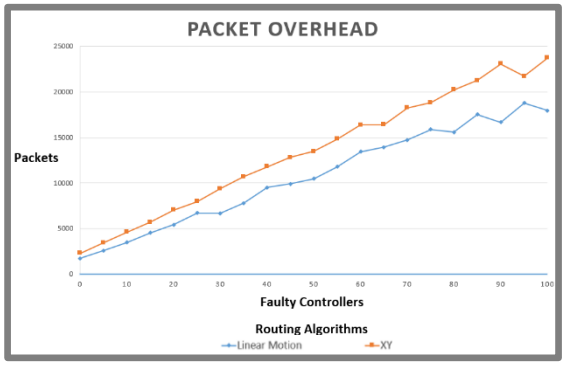
Σχήμα 5.18: Το συνολικό κόστος σε μεταδόσεις για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 5

Πείραμα 6	
Αλγόριθμος Δημιουργίας Μονοπατιού	Αλγόριθμος 5
Μέθοδος Επιλογής NUC	20
Αλγόριθμοι Δρομολόγησης	XY & Linear Motion

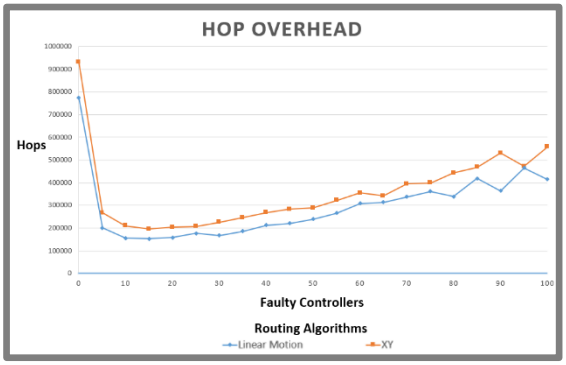
Πίνακας 5.6: Παράμετροι πειράματος 6



Σχήμα 5.19: Το χρονικό κόστος σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε αλγόριθμο δρομολόγησης, όπως προέκυψε κατά την εκτέλεση του πειράματος 6



Σχήμα 5.20: Το κόστος σε πακέτα σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε αλγόριθμο δρομολόγησης, όπως προέκυψε κατά την εκτέλεση του πειράματος 6



Σχήμα 5.21: Το κόστος σε μεταδόσεις σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε αλγόριθμο δρομολόγησης, όπως προέκυψε κατά την εκτέλεση του πειράματος 6

5.2.3 Ανάλυση Αποτελεσμάτων

Η αρχική παρατήρηση που προκύπτει από τη μελέτη των γραφικών παραστάσεων συνολικού κόστους του κάθε πειράματος είναι ότι το χρονικό κόστος και το κόστος σε πακέτα είναι πάντοτε ανάλογα και αλληλεξαρτώμενα για κάθε μέθοδο επιλογής NUC. Συγκεκριμένα, αν για κάποιο συνδυασμό αλγόριθμων δρομολόγησης και δημιουργίας μονοπατιού μια μέθοδος επιλογής NUC προκαλεί μικρότερο χρονικό κόστος από μια άλλη, τότε γνωρίζουμε σίγουρα ότι χρησιμοποιώντας τη συγκεκριμένη μέθοδο επιλογής NUC θα προκύψει σίγουρα και μικρότερη αποστολή πακέτων μέχρι να ολοκληρωθεί ο αλγόριθμος και το αντίστροφο. Επίσης, φαίνεται να υπάρχει και μια συσχέτιση του κόστους σε μεταδόσεις με τα άλλα δύο είδη κόστους για κάθε μέθοδο επιλογής NUC, ωστόσο αυτή δεν είναι εξίσου ανάλογη και απόλυτη. Συγκεκριμένα, αν για κάποιο συνδυασμό αλγόριθμων δρομολόγησης και δημιουργίας μονοπατιού μια μέθοδος επιλογής NUC προκαλεί μικρότερο κόστος σε μεταδόσεις από μια άλλη, τότε γνωρίζουμε ότι χρησιμοποιώντας τη συγκεκριμένη μέθοδο επιλογής NUC υπάρχει πολύ μεγάλη πιθανότητα να προκύψει και μικρότερο χρονικό κόστος αλλά και μικρότερο κόστος σε πακέτα μετά την ολοκλήρωση της εκτέλεσης του αλγορίθμου και το αντίστροφο. Εντούτοις, τόσο από τη μορφή των γραφικών παραστάσεων όσο και μέσω κάποιων συγκεκριμένων αντιπαραδειγμάτων μπορούμε να καταλάβουμε ότι η συγκεκριμένη συσχέτιση μεταξύ του κόστους σε μεταδόσεις και των υπόλοιπων ειδών κόστους δεν ισχύει πάντα. Αυτό μπορεί να γίνει εύκολα αντιληπτό μέσα από τις γραφικές παραστάσεις του πειράματος 3, όπου οι γραφικές παραστάσεις 5.7 και 5.8 που αναπαριστούν το χρονικό κόστος και το κόστος σε πακέτα των μεθόδων επιλογής NUC αντίστοιχα έχουν πανομοιότυπη μορφή, ενώ η γραφική παράσταση 5.9 που αναπαριστά το κόστος σε μεταδόσεις των μεθόδων επιλογής NUC, έχει ελαφρώς διαφοροποιημένη μορφή. Ένα αντιπαράδειγμα μέσω του οποίου φαίνεται πιο ξεκάθαρα η συγκεκριμένη διαφοροποίηση είναι το κόστος για τις μεθόδους επιλογής NUC 12 και 13 στο πείραμα 4. Συγκεκριμένα, αν και η χρησιμοποίηση της μεθόδου επιλογής NUC 13 προκαλεί μικρότερο χρονικό κόστος και κόστος σε πακέτα κατά την εκτέλεση του αλγορίθμου απ’ ότι η μέθοδος επιλογής NUC 12, εντούτοις φαίνεται ότι ο αριθμός των μεταβάσεων που πραγματοποιούνται από τη μέθοδο επιλογής NUC 13 είναι μεγαλύτερος. Τέτοια αντιπαραδείγματα μπορούν να βρεθούν σε όλα τα

πειράματα, ωστόσο αποτελούν μειοψηφία καθώς στη γενική περίπτωση υπάρχει μια συσχέτιση μεταξύ όλων των ειδών κόστους.

Έτσι, ζητούμενο ήταν να εντοπιστεί μια μέθοδος επιλογής NUC η οποία είναι η βέλτιστη και η οποία προκαλεί το μικρότερο κόστος ανεξαρτήτως των υπόλοιπων παραμέτρων. Για να διερευνήσουμε κατά πόσο υπάρχει μια τέτοια βέλτιστη μέθοδος επιλογής NUC, θα εστιάσουμε στα αποτελέσματα των πειραμάτων 1,2,3 και 4, κοινή παράμετρο των οποίων αποτελεί η χρησιμοποίηση του Linear Motion ως αλγόριθμου δρομολόγησης. Αφού λοιπόν ο αλγόριθμος δρομολόγησης είναι κοινός, η διαφορά μεταξύ των συγκεκριμένων πειραμάτων προκύπτει από τη χρησιμοποίηση διαφορετικού αλγόριθμου δημιουργίας μονοπατιού στο καθένα από αυτά. Όπως φαίνεται από τις γραφικές παραστάσεις συνολικού κόστους των πειραμάτων αυτών, η διαφοροποίηση του αλγόριθμου δημιουργίας μονοπατιού επηρεάζει σε πολύ μεγάλο βαθμό το κόστος που προκύπτει από την κάθε μέθοδο επιλογής NUC. Συγκεκριμένα, για κάθε αλγόριθμο δημιουργίας μονοπατιού, είναι διαφορετική η μέθοδος επιλογής NUC που προκαλεί το μικρότερο συνολικό κόστος. Στους πιο κάτω πίνακες, παρουσιάζονται για κάθε αλγόριθμο δημιουργίας μονοπατιού οι μέθοδοι επιλογής NUC που έχει αποδειχτεί πειραματικά ότι επιφέρουν το μικρότερο συνολικό κόστος, για τα τρία είδη κόστους που μελετούμε. Υπενθυμίζεται ότι τα αποτελέσματα αυτά προέκυψαν κατά τη χρησιμοποίηση του αλγόριθμου δρομολόγησης Linear Motion.

Αλγόριθμος Δημιουργίας Μονοπατιού 1		
Είδος Κόστους	**Μέθοδος Επιλογής NUC που προκαλεί το μικρότερο συνολικό κόστος**	**Γραφική Παράσταση από την οποία προκύπτει το εν λόγω αποτέλεσμα**
Χρονικό Κόστος	3	5.1
Κόστος σε πακέτα	3	5.2
Κόστος σε μεταδόσεις	3	5.3

Πίνακας 5.7: Μέθοδοι επιλογής NUC που προκαλούν το μικρότερο συνολικό κόστος, κατά τη χρησιμοποίηση του αλγόριθμου δημιουργίας μονοπατιού 1 και του αλγόριθμου δρομολόγησης Linear Motion

Αλγόριθμος Δημιουργίας Μονοπατιού 2		
Είδος Κόστους	**Μέθοδος Επιλογής NUC που προκαλεί το μικρότερο συνολικό κόστος**	**Γραφική Παράσταση από την οποία προκύπτει το εν λόγω αποτέλεσμα**
Χρονικό Κόστος	7	5.4
Κόστος σε πακέτα	7	5.5
Κόστος σε μεταδόσεις	7	5.6

Πίνακας 5.8: Μέθοδοι επιλογής NUC που προκαλούν το μικρότερο συνολικό κόστος, κατά τη χρησιμοποίηση του αλγόριθμου δημιουργίας μονοπατιού 2 και του αλγόριθμου δρομολόγησης Linear Motion

Αλγόριθμος Δημιουργίας Μονοπατιού 3		
Είδος Κόστους	**Μέθοδος Επιλογής NUC που προκαλεί το μικρότερο συνολικό κόστος**	**Γραφική Παράσταση από την οποία προκύπτει το εν λόγω αποτέλεσμα**
Χρονικό Κόστος	2	5.7
Κόστος σε πακέτα	2	5.8
Κόστος σε μεταδόσεις	12	5.9

Πίνακας 5.9: Μέθοδοι επιλογής NUC που προκαλούν το μικρότερο συνολικό κόστος, κατά τη χρησιμοποίηση του αλγόριθμου δημιουργίας μονοπατιού 3 και του αλγόριθμου δρομολόγησης Linear Motion

Αλγόριθμος Δημιουργίας Μονοπατιού 4		
Είδος Κόστους	**Μέθοδος Επιλογής NUC που προκαλεί το μικρότερο συνολικό κόστος**	**Γραφική Παράσταση από την οποία προκύπτει το εν λόγω αποτέλεσμα**
Χρονικό Κόστος	13	5.10
Κόστος σε πακέτα	13	5.11
Κόστος σε μεταδόσεις	15	5.12

Πίνακας 5.10: Μέθοδοι επιλογής NUC που προκαλούν το μικρότερο συνολικό κόστος, κατά τη χρησιμοποίηση του αλγόριθμου δημιουργίας μονοπατιού 4 και του αλγόριθμου δρομολόγησης Linear Motion

Από τους πιο πάνω πίνακες γίνεται προφανές ότι δεν υπάρχει κάποια μέθοδος επιλογής NUC η οποία να μπορεί να θεωρηθεί ως βέλτιστη, καθώς αναλόγως του ποιος αλγόριθμος δημιουργίας μονοπατιού χρησιμοποιείται είναι διαφορετική η μέθοδος επιλογής NUC που προκαλεί το μικρότερο συνολικό κόστος. Ακόμη, από τις γραφικές παραστάσεις 5.13, 5.14 και 5.15 μπορούμε να διαπιστώσουμε ότι η βελτιστότητα των μεθόδων επιλογής NUC εξαρτάται και από τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα. Ακόμη και αν διατηρήσουμε τους αλγόριθμους δρομολόγησης και

δημιουργίας μονοπατιού σταθερούς, για διαφορετικό αριθμό σφαλμάτων είναι διαφορετικές οι μέθοδοι επιλογής NUC που επιφέρουν το μικρότερο δυνατό κόστος.

Μέθοδος επιλογής NUC που προκαλεί το μικρότερο κόστος ανά αριθμό σφαλμάτων																				
	Ποσότητα Σφαλμάτων																			
Είδος Κόστους	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**
Χρονικό Κόστος	6	16	15	15	13	15	15	15	15	15	15	13	13	13	6	15	9	9	13	2	6
Κόστος σε πακέτα	6	16	15	15	13	15	15	15	15	15	15	13	13	13	6	15	9	9	13	2	6
Κόστος σε μεταδόσεις	18	2	10	10	14	17	10	15	15	10	17	13	10	13	10	15	15	13	13	13	6

Πίνακας 5.11: Παρουσίαση των μεθόδων επιλογής NUC που επιφέρουν το μικρότερο κόστος ανά αριθμό σφαλμάτων, όταν χρησιμοποιείται ο αλγόριθμο δημιουργίας μονοπατιού 4 και ο αλγόριθμος δρομολόγησης Linear Motion. Τα αποτελέσματα εξάχθηκαν από τις γραφικές παραστάσεις 5.13, 5.14 και 5.15.

Μέθοδος επιλογής NUC που προκαλεί το μικρότερο κόστος ανά αριθμό σφαλμάτων																				
	Ποσότητα Σφαλμάτων																			
Είδος Κόστους	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**
Χρονικό Κόστος	3	7	15	15	15	13	15	13	15	13	13	13	13	13	13	6	15	15	13	13	15
Κόστος σε πακέτα	2	7	15	15	15	13	15	13	15	13	13	13	13	13	13	6	15	15	13	13	15
Κόστος σε μεταδόσεις	12	6	10	10	10	10	10	10	10	10	9	13	13	13	13	6	10	9	13	13	15

Πίνακας 5.12: Παρουσίαση των μεθόδων επιλογής NUC που επιφέρουν το μικρότερο κόστος ανά αριθμό σφαλμάτων, όταν χρησιμοποιείται ο αλγόριθμο δημιουργίας μονοπατιού 4 και ο αλγόριθμος δρομολόγησης XY. Τα αποτελέσματα εξάχθηκαν από τις γραφικές παραστάσεις Α.10, Α.11 και Α.12 του Παραρτήματος Α.

Αυτό επιβεβαιώνεται και από τους πίνακες 5.11 και 5.12, όπου γίνεται ξεκάθαρο ότι ακόμα και κατά τη χρήση των ίδιων αλγόριθμων δημιουργίας μονοπατιού και δρομολόγησης, δεν υπάρχει κάποια μέθοδος επιλογής NUC που να επιφέρει πάντοτε το μικρότερο δυνατό κόστος, αλλά πάντοτε το κόστος αυτό εξαρτάται και από τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα. Παρόμοια αποτελέσματα με αυτά των πινάκων 5.11 και 5.12 για διαφορετικούς συνδυασμούς αλγορίθμων δρομολόγησης και δημιουργίας μονοπατιού μπορούν να βρεθούν στα παραρτήματα ουσιαστικά επιβεβαιώνουν και ενισχύουν το συμπέρασμα που έχουμε εξάγει.

Έχοντας διαπιστώσει ότι η αποδοτικότητα των μεθόδων επιλογής NUC εξαρτάται σε μεγάλο βαθμό από τους αλγόριθμους δημιουργίας μονοπατιού που χρησιμοποιούνται

και τον αριθμό των σφαλμάτων που υπάρχει στο πλέγμα, απομένει να εξετάσουμε κατά πόσο επηρεάζονται στον ίδιο βαθμό και από τη διαφοροποίηση των αλγορίθμων δρομολόγησης. Αρχικά, από τις γραφικές παραστάσεις 5.19, 5.20 και 5.21 μπορούμε εύκολα να διαπιστώσουμε ότι χρησιμοποιώντας των ίδιο αλγόριθμο δημιουργίας μονοπατιού και μέθοδο επιλογής NUC, προκύπτει διαφορετικό κόστος για κάθε αριθμό σφαλμάτων αναλόγως του αλγόριθμου δρομολόγησης που χρησιμοποιείται. Συνεπώς, είναι βέβαιο ότι η διαφοροποίηση του αλγόριθμου δρομολόγησης επηρεάζει το κόστος που προκύπτει όταν χρησιμοποιείται μια συγκεκριμένη μέθοδος επιλογής NUC. Για να αξιολογήσουμε κατά πόσο αυτή η διαφοροποίηση του κόστους είναι ανάλογη για όλες τις μεθόδους επιλογή NUC, θα συγκρίνουμε το περιεχόμενο των πινάκων 5.11 και 5.12. Βάσει της σύγκρισης παρατηρούμε ότι τα περιεχόμενα των δύο πινάκων είναι κοινά κατά μόλις 35%. Αυτό σημαίνει ότι η διαφοροποίηση του αλγόριθμου δρομολόγησης δεν επηρεάζει με τον ίδιο τρόπο το κόστος που προκύπτει από τη χρησιμοποίηση της κάθε μεθόδου επιλογής NUC. Αυτό σημαίνει ότι διατηρώντας τον αλγόριθμο δημιουργίας μονοπατιού σταθερό, αν μια μέθοδος επιλογής NUC μας παρέχει χαμηλότερο κόστος όταν χρησιμοποιούμε ένα συγκεκριμένο αλγόριθμο δρομολόγησης, δε σημαίνει κατ’ ανάγκη ότι αν αλλάξουμε τον αλγόριθμο δρομολόγησης που χρησιμοποιούμε η συγκεκριμένη μέθοδος επιλογής NUC θα παραμείνει η βέλτιστη.

Συνοψίζοντας, από την ανάλυση στην οποία έχουμε προβεί διαπιστώσαμε ότι οι μέθοδοι επιλογής NUC είναι ένας πολύ ασταθής παράγοντας, καθώς ο τρόπος με τον οποίο επηρεάζει την επίδοση του αλγορίθμου αναγνώρισης σφαλμάτων εξαρτάται τόσο από τον συνδυασμό των αλγορίθμων δρομολόγησης και δημιουργίας μονοπατιού που χρησιμοποιούνται, όσο και από τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα. Συνεπώς, δεν υπάρχει κάποια μέθοδος επιλογής NUC που μπορεί να θεωρηθεί ως βέλτιστη, αλλά αναλόγως των υπόλοιπων παραμέτρων πρέπει να επιλέγεται και η μέθοδος επιλογής NUC η οποία θα επιφέρει το ελάχιστο δυνατό κόστος και κατ’ επέκταση την καλύτερη δυνατή επίδοση για τον αλγόριθμο αναγνώρισης σφαλμάτων.

5.3 Αποτελέσματα για Αλγόριθμους Δημιουργίας Μονοπατιού

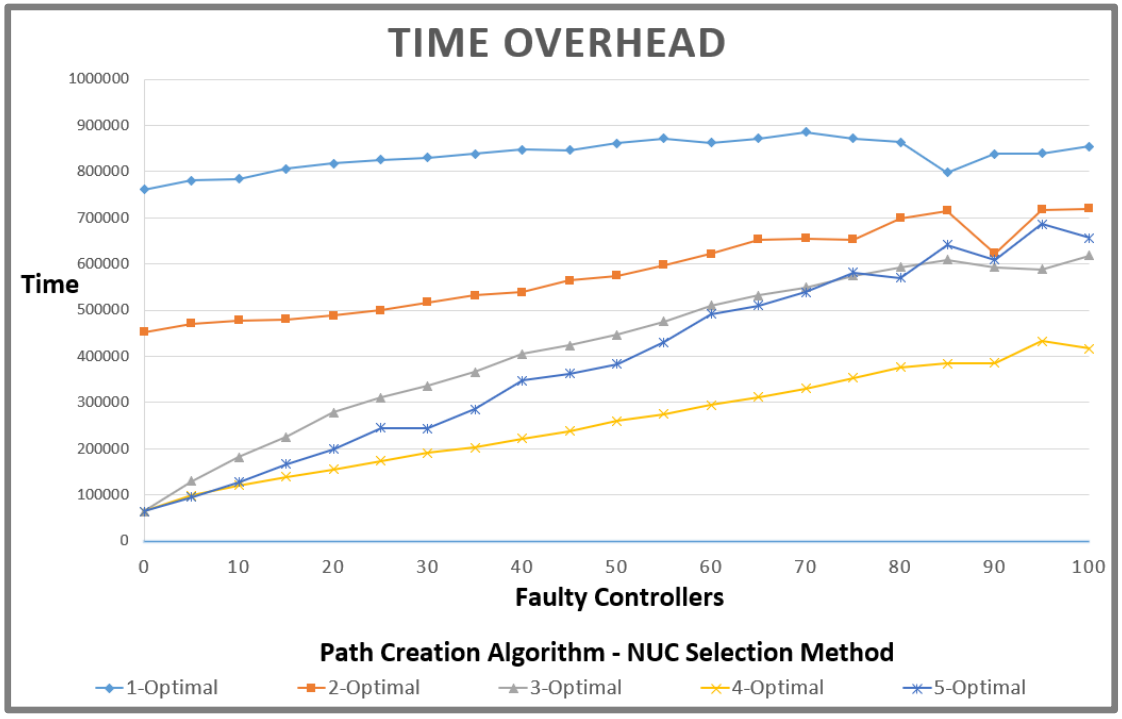
5.3.1 Περιγραφή Πειραμάτων

Για να μελετήσω το πως επηρεάζει η χρησιμοποίηση των διαφορετικών αλγορίθμων δημιουργίας μονοπατιού την επίδοση του αλγορίθμου αναγνώρισης σφαλμάτων, χρησιμοποίησα τα αποτελέσματα των πειραμάτων 1,2,3,4 και 6 του Κεφαλαίου 5.2, δημιουργώντας γραφικές παραστάσεις βάσει του αλγορίθμου δημιουργίας μονοπατιού που χρησιμοποιήθηκε σε κάθε πείραμα. Υπενθυμίζεται ότι σε όλα τα εν λόγω πειράματα ο αλγόριθμος δρομολόγησης που χρησιμοποιήθηκε ήταν ο Linear Motion.

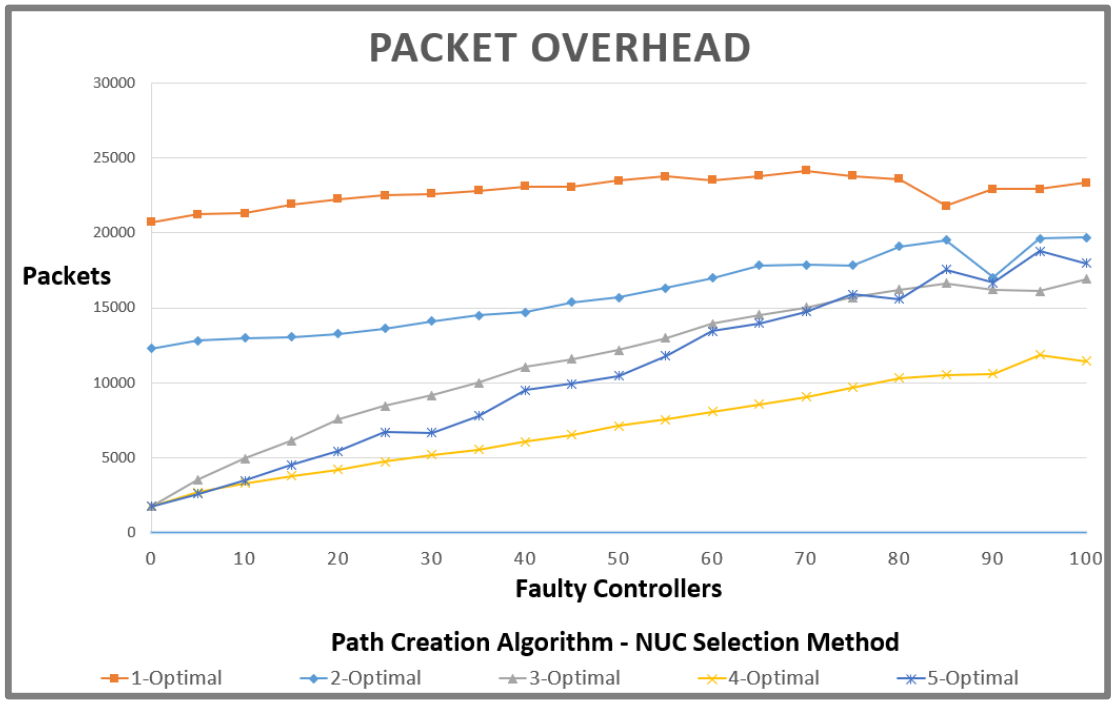
Εφόσον από το Κεφάλαιο 5.2 εξάχθηκε το συμπέρασμα ότι η κάθε μέθοδος επιλογής NUC επηρεάζει διαφορετικά την επίδοση που προκύπτει κατά τη χρησιμοποίηση των διάφορων αλγορίθμων δημιουργίας μονοπατιού, αποφάσισα για κάθε συνδυασμό αλγορίθμου δημιουργίας μονοπατιού και αριθμού σφαλμάτων στο πλέγμα να λάβω υπόψη τη μέθοδο επιλογής NUC που προκαλεί το μικρότερο κόστος. Κάτι τέτοιο ουσιαστικά μου επιτρέπει να συγκρίνω τις βέλτιστες επιδόσεις που μπορούν να επιτευχθούν από τον κάθε αλγόριθμο δημιουργίας μονοπατιού.

Οι μέσοι όροι κόστους των εκτελέσεων που πραγματοποιήθηκαν για κάθε συνδυασμό αλγορίθμου επιλογής μονοπατιού και αριθμού σφαλμάτων φαίνονται στις γραφικές παραστάσεις 5.22, 5.23 και 5.25. Οι γραφικές παραστάσεις που παρουσιάζουν το άθροισμα των μέσων όρων κόστους για όλους τους αριθμούς σφαλμάτων ανά αλγόριθμο επιλογής μονοπατιού είναι οι 5.25, 5.26 και 5.27.

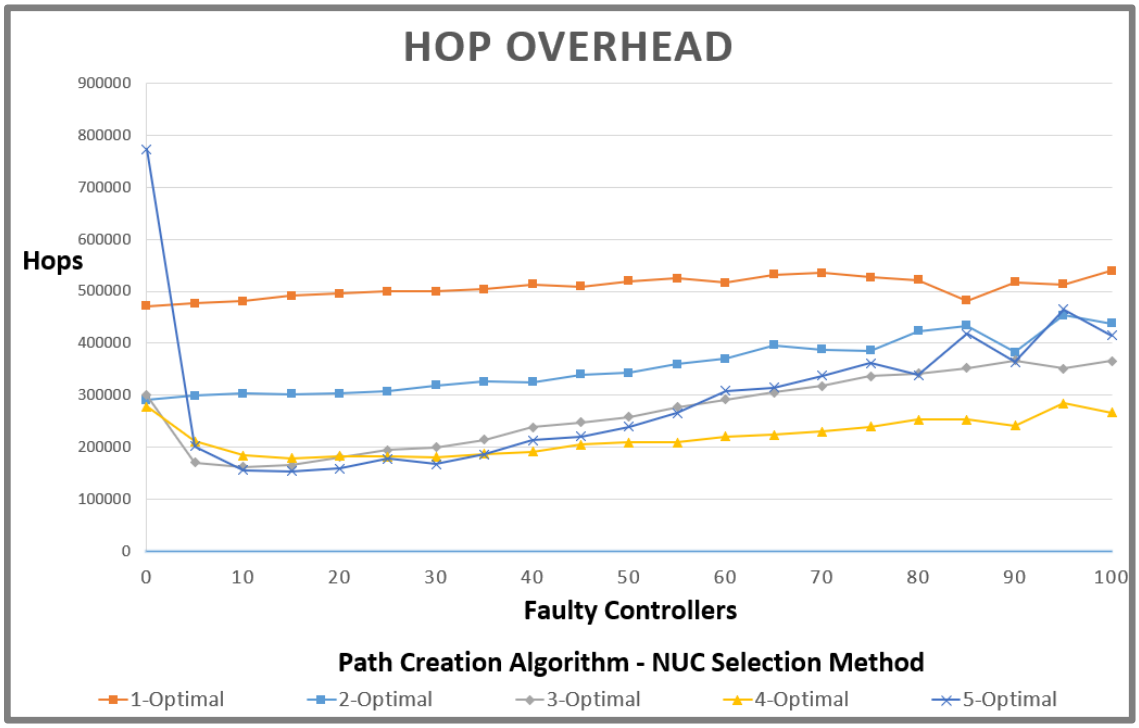
5.3.2 Αποτελέσματα Πειραμάτων



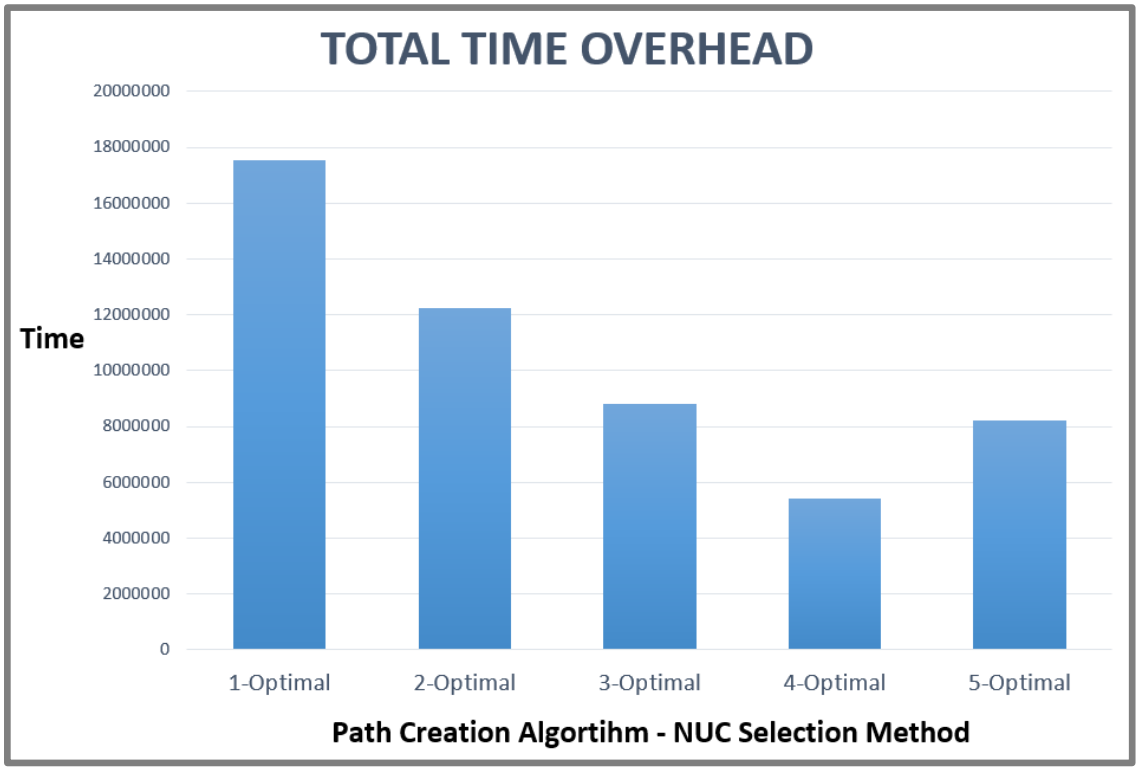
Σχήμα 5.22: Το χρονικό κόστος σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε αλγόριθμο δημιουργίας μονοπατιού. Χρησιμοποιήθηκαν οι μέθοδοι επιλογής NUC που προκαλούν το μικρότερο χρονικό κόστος για τον κάθε συνδυασμό αριθμού σφαλμάτων και αλγορίθμου δημιουργίας μονοπατιού, βάσει των αποτελεσμάτων που προέκυψαν από τα πειράματα 1,2,3,4 και 6 του Κεφαλαίου 5.2.



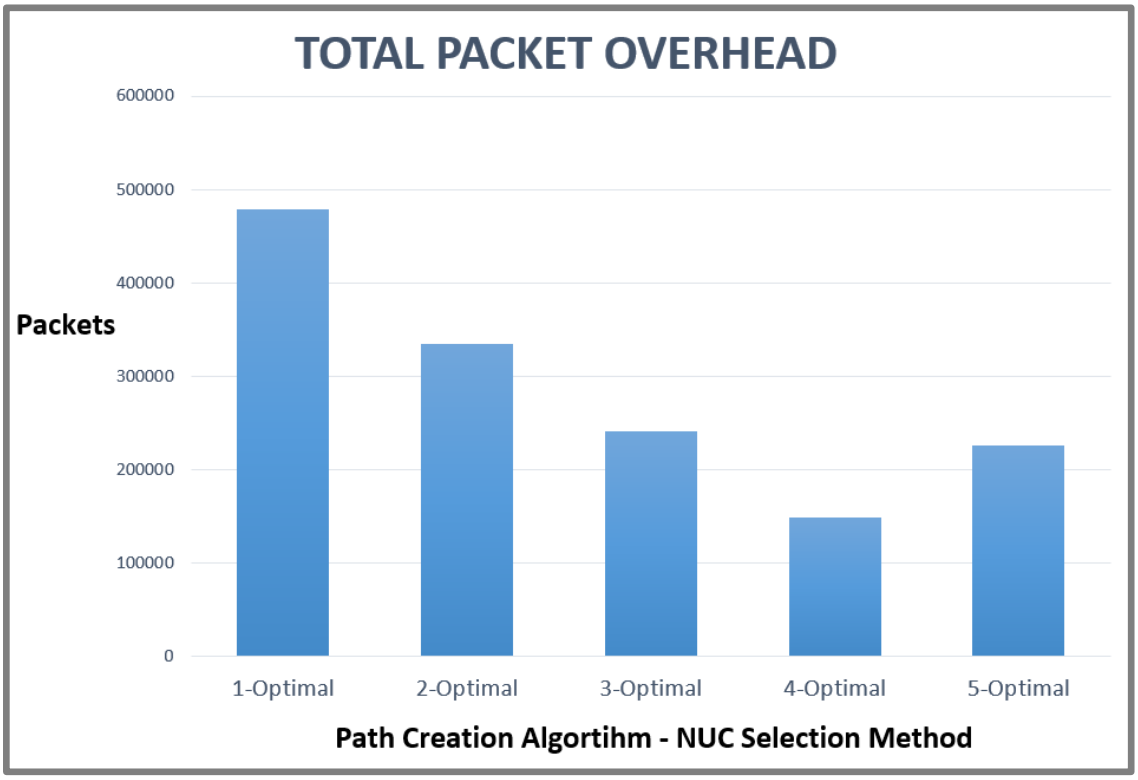
Σχήμα 5.23: Το κόστος σε πακέτα σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε αλγόριθμο δημιουργίας μονοπατιού. Χρησιμοποιήθηκαν οι μέθοδοι επιλογής NUC που προκαλούν το μικρότερο κόστος σε πακέτα για τον κάθε συνδυασμό αριθμού σφαλμάτων και αλγορίθμου δημιουργίας μονοπατιού, βάσει των αποτελεσμάτων που προέκυψαν από τα πειράματα 1,2,3,4 και 6 του Κεφαλαίου 5.2.



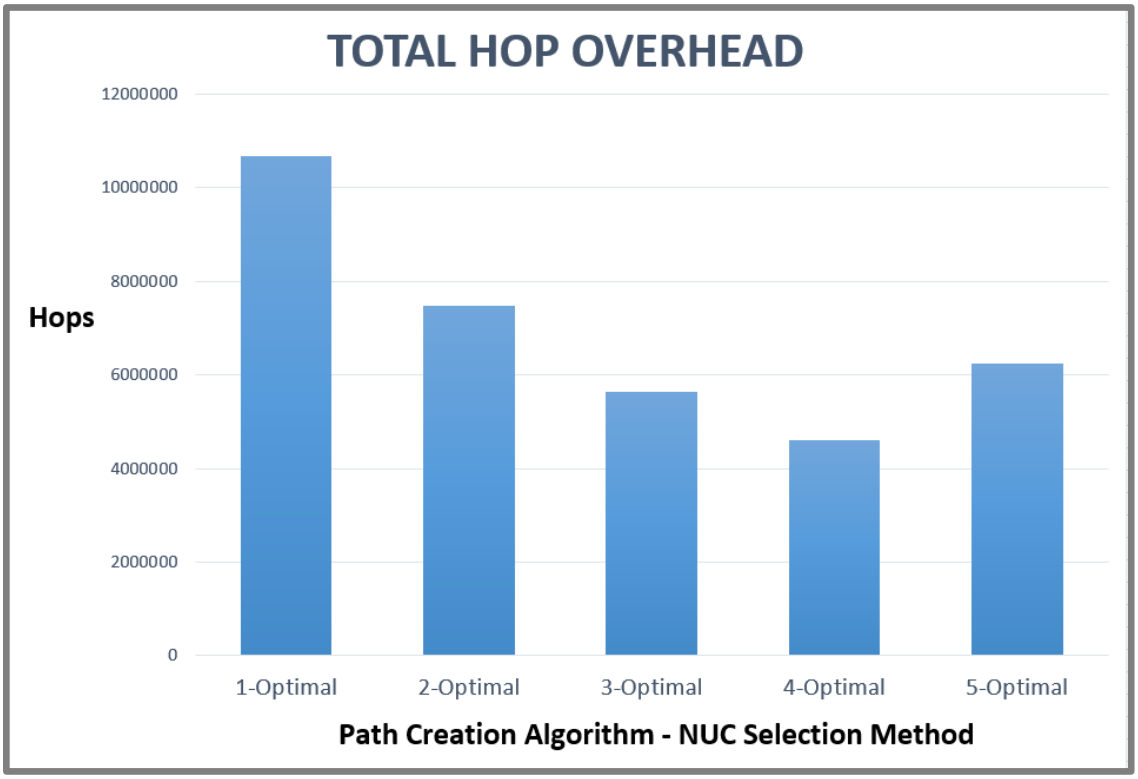
Σχήμα 5.24: Το κόστος σε μεταδόσεις σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε αλγόριθμο δημιουργίας μονοπατιού. Χρησιμοποιήθηκαν οι μέθοδοι επιλογής NUC που προκαλούν το μικρότερο κόστος σε μεταδόσεις για τον κάθε συνδυασμό αριθμού σφαλμάτων και αλγορίθμου δημιουργίας μονοπατιού, βάσει των αποτελεσμάτων που προέκυψαν από τα πειράματα 1,2,3,4 και 6 του Κεφαλαίου 5.2.



Σχήμα 5.25: Το συνολικό χρονικό κόστος για τον κάθε αλγόριθμο δημιουργίας μονοπατιού, όταν χρησιμοποιούνται οι μέθοδοι επιλογής NUC που προκαλούν το μικρότερο χρονικό κόστος αναλόγως του αριθμού των σφαλμάτων που υπάρχουν στο πλέγμα, βάσει των αποτελεσμάτων που προέκυψαν από τα πειράματα 1,2,3,4 και 6 του Κεφαλαίου 5.2.



Σχήμα 5.26: Το συνολικό κόστος σε πακέτα για τον κάθε αλγόριθμο δημιουργίας μονοπατιού, όταν χρησιμοποιούνται οι μέθοδοι επιλογής NUC που προκαλούν το μικρότερο κόστος σε πακέτα αναλόγως του αριθμού των σφαλμάτων που υπάρχουν στο πλέγμα, βάσει των αποτελεσμάτων που προέκυψαν από τα πειράματα 1,2,3,4 και 6 του Κεφαλαίου 5.2.



Σχήμα 5.27: Το συνολικό κόστος σε μεταδόσεις για τον κάθε αλγόριθμο δημιουργίας μονοπατιού, όταν χρησιμοποιούνται οι μέθοδοι επιλογής NUC που προκαλούν το μικρότερο κόστος σε μεταδόσεις αναλόγως του αριθμού των σφαλμάτων που υπάρχουν στο πλέγμα, βάσει των αποτελεσμάτων που προέκυψαν από τα πειράματα 1,2,3,4 και 6 του Κεφαλαίου 5.2.

5.3.3 Ανάλυση Αποτελεσμάτων

Όπως προέκυψε και για τις γραφικές παραστάσεις που αφορούσαν τις μεθόδους επιλογής NUC, παρατηρούμε ότι και στις γραφικές παραστάσεις των αλγορίθμων δημιουργίας μονοπατιού υπάρχει απόλυτη αναλογία που μεταξύ του χρονικού κόστους και του κόστους σε πακέτα που προκύπτει σε κάθε περίπτωση. Ακόμη, φαίνεται και σε αυτή την περίπτωση να υπάρχει μια συσχέτιση μεταξύ του κόστους σε μεταδόσεις με τα άλλα δύο είδη κόστους για κάθε αλγόριθμο δημιουργίας μονοπατιού, όμως ούτε σε αυτή την περίπτωση η συγκεκριμένη συσχέτιση δεν είναι ανάλογη και απόλυτη. Αυτό καθίσταται προφανές λόγω της διαφοροποίησης που παρατηρείται στη μορφή των γραφικών παραστάσεων που αναφέρονται στο κόστος σε μεταδόσεις. Ακόμη, μπορεί να επιβεβαιωθεί εύκολα, καθώς ενώ για στο διάστημα από 10 μέχρι 35 σφαλμάτων η χρήση του αλγορίθμου δημιουργίας μονοπατιού 4 προκαλεί το μικρότερο δυνατό χρονικό κόστος, εντούτοις είναι η χρήση του αλγορίθμου δημιουργίας μονοπατιού 5 που προκαλεί το μικρότερο δυνατό κόστος σε μεταδόσεις.

Το γενικότερο συμπέρασμα που προκύπτει από όλες τις γραφικές παραστάσεις του κεφαλαίου αυτού, είναι ότι ο αλγόριθμος επιλογής μονοπατιού 4 είναι αυτός που επιφέρει πάντοτε το μικρότερο δυνατό κόστος όταν ο αριθμός των σφαλμάτων που υπάρχουν στο πλέγμα ξεπεράσει ένα όριο. Συγκεκριμένα, από τις γραφικές παραστάσεις 5.22, 5.23 και 5.24 συμπεραίνουμε ότι ο αλγόριθμος επιλογής μονοπατιού 4 είναι ο αποδοτικότερος σε σχέση με το χρόνο εκτέλεσης και τον αριθμό των πακέτων που στέλνονται όταν υπάρχουν 8 ή περισσότερα σφάλματα στο πλέγμα, ενώ είναι και ο αποδοτικότερος σε σχέση με τον αριθμό τον μεταβάσεων που πραγματοποιούνται όταν υπάρχουν 36 ή περισσότερα σφάλματα στο πλέγμα. Επιπλέον, η ευρύτερη βελτιστότητα του συγκεκριμένου αλγόριθμου δημιουργίας μονοπατιού αποδεικνύεται και από το γεγονός ότι προκαλεί το μικρότερο συνολικό κόστος και για τις τρεις κατηγορίας κόστους που εξετάζονται.

Οι μόνες περιπτώσεις κατά τις οποίες είναι καλύτερο να χρησιμοποιείται ένας εναλλακτικός αλγόριθμος δημιουργίας μονοπατιού, είναι όταν ο αριθμός των σφαλμάτων που υπάρχουν στο πλέγμα είναι μειωμένος. Συγκεκριμένα, σε περίπτωση που υπάρχουν 7 ή λιγότερα σφάλματα στο πλέγμα, ο αλγόριθμος δημιουργίας

μονοπατιού του οποίου η χρησιμοποίηση συνεπάγεται τον ταχύτερο τερματισμό της διαδικασίας αναγνώρισης σφαλμάτων και τη μικρότερη παραγωγή πακέτων είναι ο αλγόριθμος 5, κάτι το οποίο προκύπτει από τις γραφικές παραστάσεις 5.22 και 5.23. Όσον αφορά τη μικρότερη δυνατή πραγματοποίηση μεταβάσεων, βάσει της γραφικής παράστασης 5.24 αυτή επιτυγχάνεται από τον αλγόριθμο δημιουργίας μονοπατιού 5 όταν υπάρχουν από 10 μέχρι 35 σφάλματα στο πλέγμα και από τον αλγόριθμο δημιουργίας μονοπατιού 3 όταν υπάρχουν από 3 μέχρι 9 σφάλματα στο πλέγμα. Σε περίπτωση που στο πλέγμα υπάρχουν 2 ή λιγότερα σφάλματα ο στόχος μας είναι να επιτύχουμε τη μικρότερη δυνατή πραγματοποίηση μεταβάσεων, τότε είναι καλύτερο να προτιμούμε και πάλι τον αλγόριθμο 4.

Συνοψίζοντας, σε αυτό το κεφάλαιο καταφέραμε μέσω της επιλογής της πιο κατάλληλης μεθόδου επιλογής NUC για την κάθε περίπτωση να συγκρίνουμε τα καλύτερα αποτελέσματα που μπορούν να προκύψουν όταν χρησιμοποιείται ο κάθε αλγόριθμος δημιουργίας μονοπατιού. Γενικό συμπέρασμα είναι ότι για μεγάλους αριθμούς σφαλμάτων στο πλέγμα η χρησιμοποίηση του αλγορίθμου δημιουργίας μονοπατιού 4 είναι η βέλτιστη από όλες τις απόψεις. Εάν ο αριθμός των σφαλμάτων στο πλέγμα είναι μικρός, τότε θα πρέπει να δούμε ποιος είναι ο βασικός μας στόχος για να επιλέξουμε τον αλγόριθμο δημιουργίας μονοπατιού που θα χρησιμοποιήσουμε. Αν πιο σημαντικό για εμάς είναι ο ταχύτερος τερματισμός της διαδικασίας αναγνώρισης σφαλμάτων τότε θα επιλέξουμε τη χρήση του αλγόριθμου δημιουργίας μονοπατιού 4, ενώ αν προτιμούμε η αναγνώριση σφαλμάτων να γίνει με τρόπο που να πραγματοποιείται ο μικρότερος αριθμός μεταβάσεων στο πλέγμα, τότε θα προτιμήσουμε ένα εκ’ των αλγορίθμων δημιουργίας μονοπατιού 3,4 ή 5, αναλόγως της εκτίμησης που έχουμε για τον αριθμό των σφαλμάτων που θέλουμε να αναγνωρίσουμε.

5.4 Αποτελέσματα για Αλγόριθμους Δρομολόγησης

5.4.1 Περιγραφή Πειραμάτων

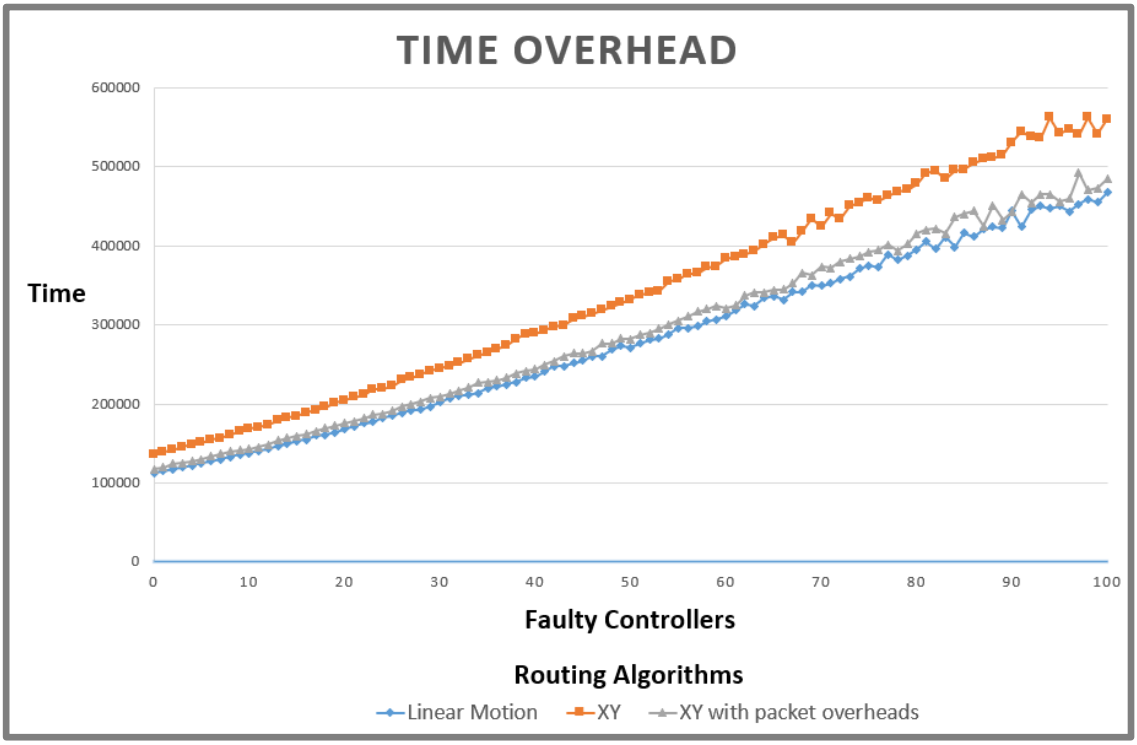
Όπως διαπιστώσαμε στο Κεφάλαιο 5.3, ο αλγόριθμος δημιουργίας μονοπατιού ο οποίος επιφέρει τη μεγαλύτερη βελτίωση στην επίδοση του αλγορίθμου αναγνώρισης σφαλμάτων στις περισσότερες περιπτώσεις είναι ο αλγόριθμος 4. Έτσι, κατά το πείραμα που διενήργησα για να συγκρίνω τους τρεις αλγόριθμους δρομολόγησης, το Linear Motion, το XY και το XY με εκτενή πακέτα, επέλεξα να εξετάσω μόνο τη διαφορά αποδοτικότητας που προκύπτει μόνο όταν χρησιμοποιείται ο αλγόριθμος δημιουργίας μονοπατιού 4.

Εφόσον φάνηκε ότι η αποδοτικότητα που προκαλούν κάθε φορά οι μέθοδοι επιλογής NUC δεν είναι σταθερή και επηρεάζεται από τον αλγόριθμο δρομολόγησης που χρησιμοποιείται, υπήρξε η ανάγκη περιορισμού της επίδρασης που προκαλεί ο συγκεκριμένος αστάθμητος παράγοντας. Έτσι, επέλεξα για το πείραμα που διενήργησα να χρησιμοποιήσω τη μέθοδο επιλογής NUC 19, η οποία είναι ουσιαστικά η τυχαία σειρά επιλογής των ελεγκτών NUC.

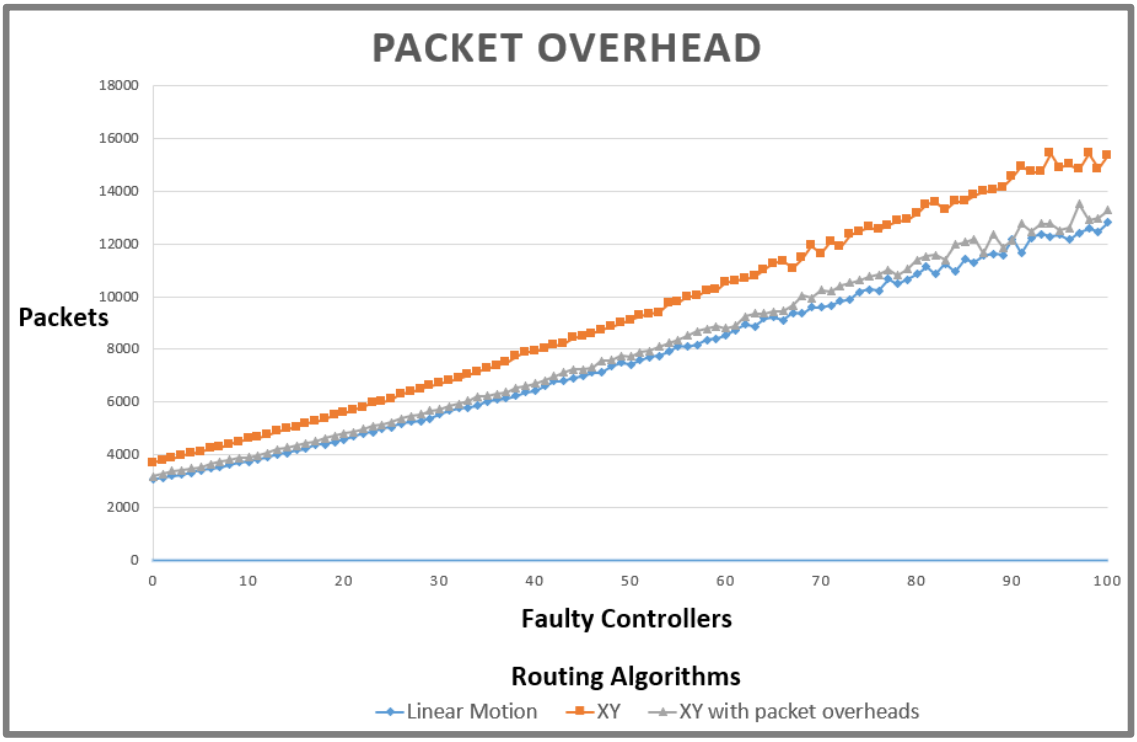
Όπως σε όλα μου τα πειράματα, έτσι και σε αυτή την περίπτωση χρησιμοποίησα μέγεθος πλέγματος 24x24 με σταθερά όρια καθυστερήσεων, ενώ ο αριθμός των σφαλμάτων στο πλέγμα σε κάθε πείραμα διακυμάνθηκε από τα 0 μέχρι τα 100 σφάλματα. Για κάθε αλγόριθμο δρομολόγησης πραγματοποιήθηκαν 100 εκτελέσεις του αλγορίθμου για όλους τους αριθμούς σφαλμάτων. Από τα αποτελέσματα των εκτελέσεων αυτών αφαιρέθηκαν και σε αυτή την περίπτωση τα αποκλίνοντα δεδομένα. Έπειτα, για κάθε συνδυασμό αλγόριθμου δρομολόγησης και αριθμού σφαλμάτων υπολογίστηκε ο μέσος όρος του χρονικού κόστους, του κόστους σε πακέτα και του κόστους σε μεταδόσεις που προέκυψε από τις διάφορες εκτελέσεις του αλγορίθμου. Οι συγκεκριμένοι μέσοι όροι κόστους ανά αριθμό σφαλμάτων φαίνονται στις γραφικές παραστάσεις 5.28, 5.29 και 5.30.

Ακολούθως, για κάθε αλγόριθμο δρομολόγησης υπολογίστηκε το άθροισμα των μέσων όρων κόστους για όλους τους αριθμούς σφαλμάτων. Έτσι, προέκυψε ένα συνολικό κόστος ανά αλγόριθμο δρομολόγησης. Οι γραφικές παραστάσεις που παρουσιάζουν τα συγκεκριμένα συνολικά κόστη για κάθε αλγόριθμο δρομολόγησης είναι οι 5.31, 5.32 και 5.33.

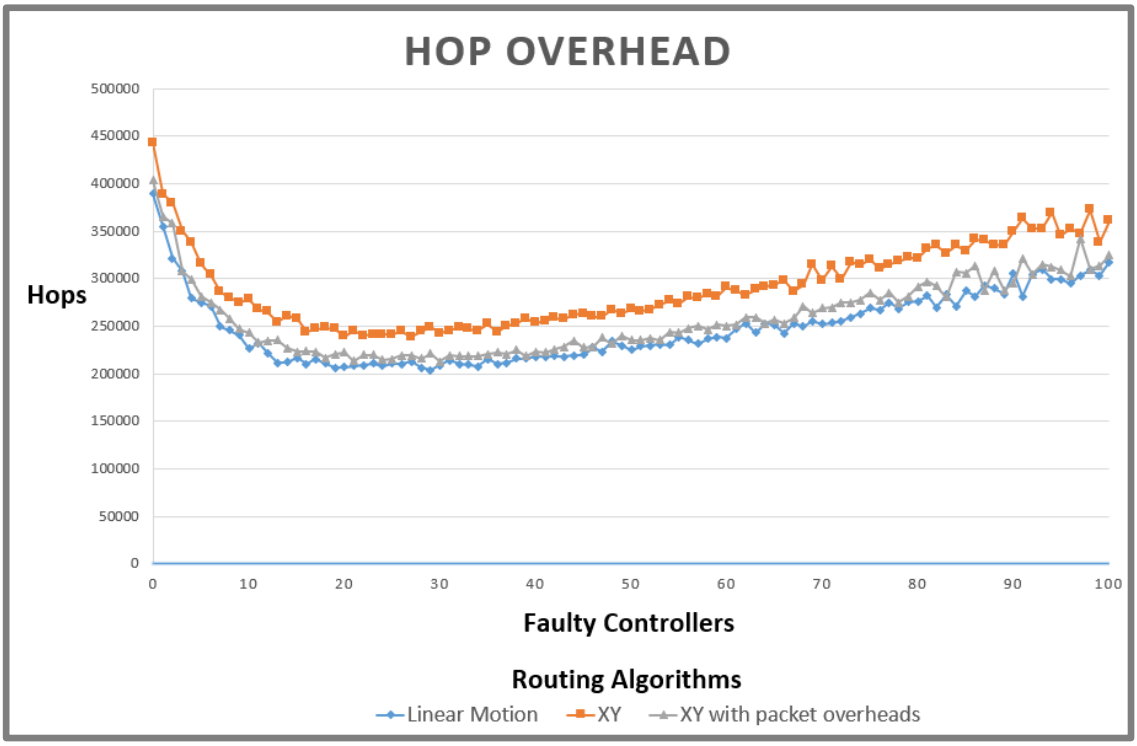
5.4.2 Αποτελέσματα πειραμάτων



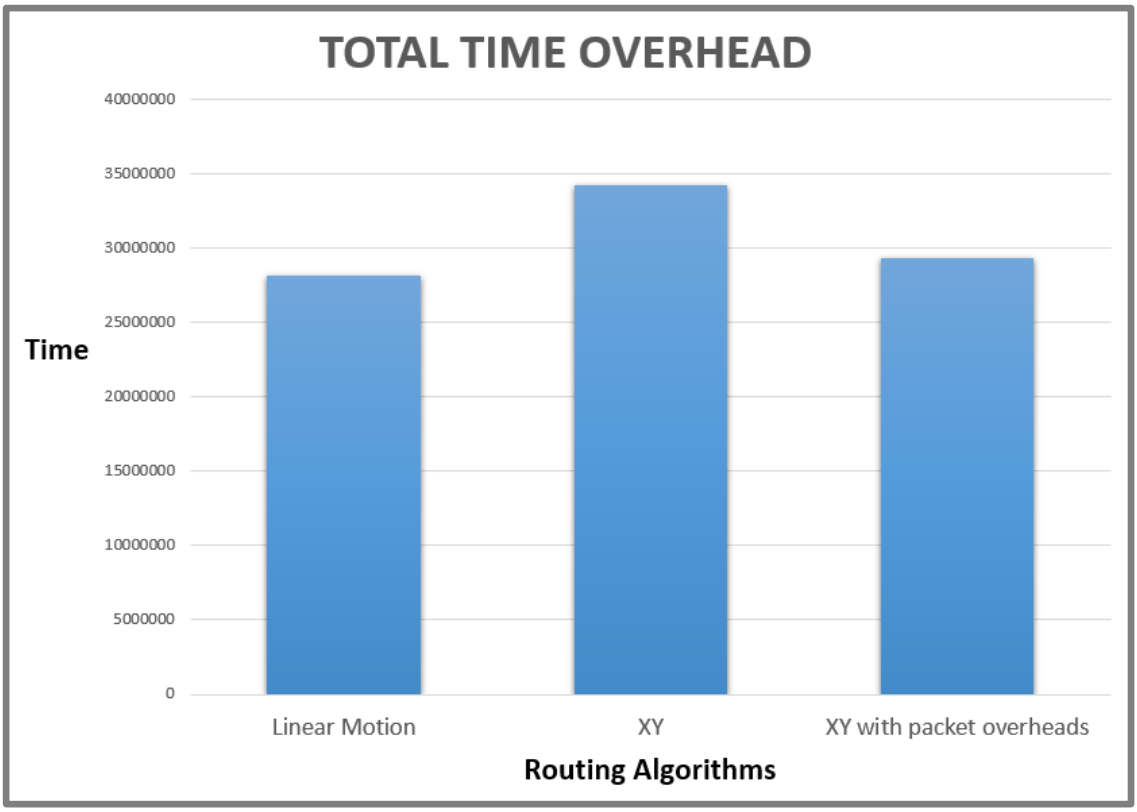
Σχήμα 5.28: Το χρονικό κόστος σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε αλγόριθμο δρομολόγησης. Χρησιμοποιήθηκε η μέθοδος επιλογής NUC 19 και ο αλγόριθμος δημιουργίας μονοπατιού 4.



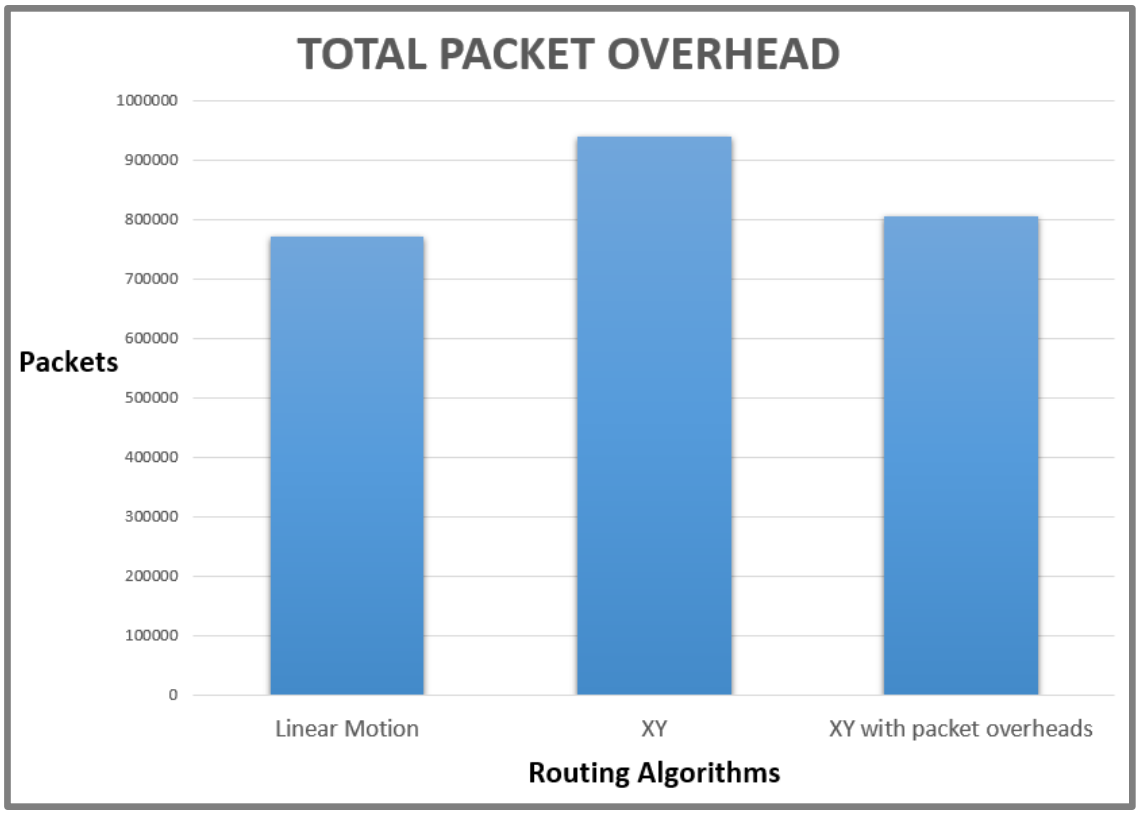
Σχήμα 5.29: Το κόστος σε πακέτα σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε αλγόριθμο δρομολόγησης. Χρησιμοποιήθηκε η μέθοδος επιλογής NUC 19 και ο αλγόριθμος δημιουργίας μονοπατιού 4.



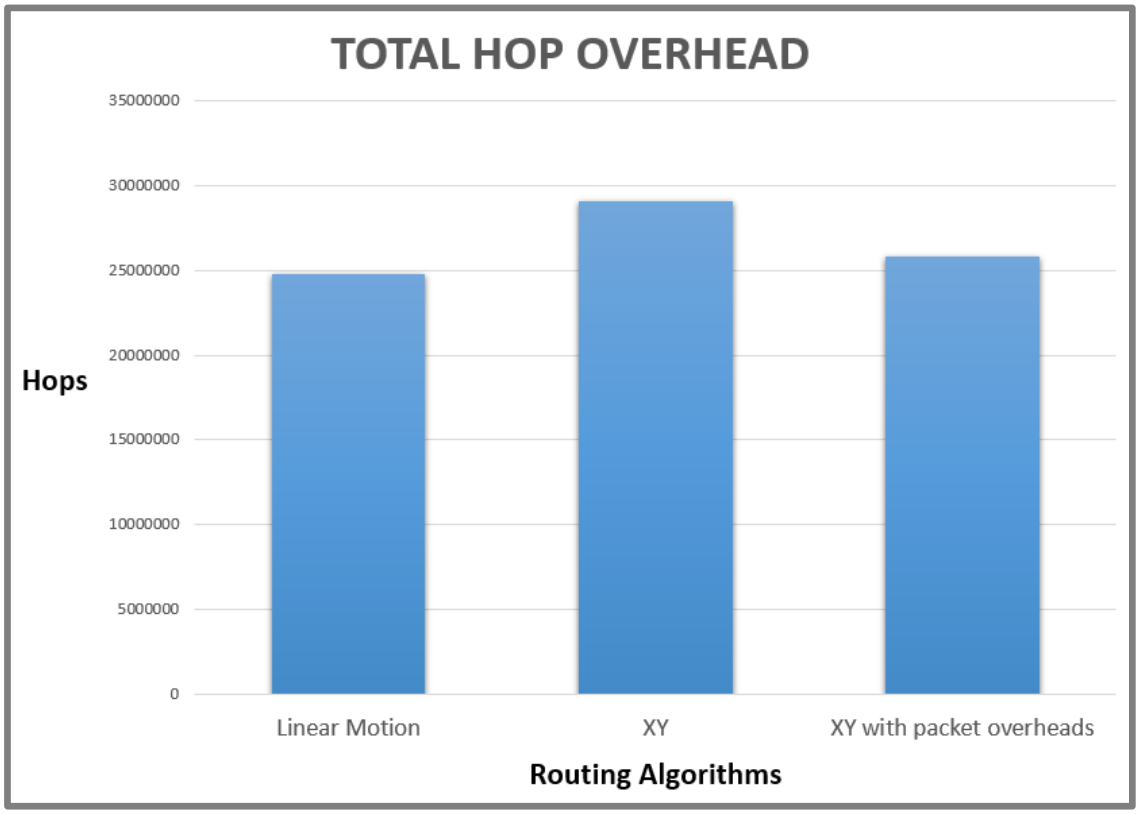
Σχήμα 5.30: Το κόστος σε μεταδόσεις σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε αλγόριθμο δρομολόγησης. Χρησιμοποιήθηκε η μέθοδος επιλογής NUC 19 και ο αλγόριθμος δημιουργίας μονοπατιού 4.



Σχήμα 5.31: Το συνολικό χρονικό κόστος για τον κάθε αλγόριθμο δρομολόγησης. Χρησιμοποιήθηκε η μέθοδος επιλογής NUC 19 και ο αλγόριθμος δημιουργίας μονοπατιού 4.



Σχήμα 5.32: Το συνολικό κόστος σε πακέτα για τον κάθε αλγόριθμο δρομολόγησης. Χρησιμοποιήθηκε η μέθοδος επιλογής NUC 19 και ο αλγόριθμος δημιουργίας μονοπατιού 4.



Σχήμα 5.33: Το συνολικό κόστος σε μεταδόσεις για τον κάθε αλγόριθμο δρομολόγησης. Χρησιμοποιήθηκε η μέθοδος επιλογής NUC 19 και ο αλγόριθμος δημιουργίας μονοπατιού 4.

5.4.3 Ανάλυση Αποτελεσμάτων

Το βασικό συμπέρασμα που προκύπτει από τις πιο πάνω γραφικές παραστάσεις είναι ότι όταν χρησιμοποιείται ο Linear Motion ως αλγόριθμος δρομολόγησης προκύπτει πάντοτε χαμηλότερο κόστος. Αυτό ισχύει για όλα τα είδη κόστους και ισχύει τόσο σε συνολικό επίπεδο (γραφικές παραστάσεις 5.31, 5.32 και 5.33), όσο και για κάθε συγκεκριμένο αριθμό σφαλμάτων που μπορεί να παρουσιαστούν στο πλέγμα (γραφικές παραστάσεις 5.28, 5.29 και 5.30). Συνεπώς, η χρησιμοποίηση του Linear Motion πρέπει να προτιμάται έναντι του XY και του XY με εκτενή πακέτα, καθώς ελαχιστοποιεί το κόστος και ταυτόχρονα αυξάνει την επίδοση του αλγόριθμου αναγνώρισης σφαλμάτων.

Το γεγονός ότι ο Linear Motion επιφέρει καλύτερα αποτελέσματα από το XY σε κάθε περίπτωση είναι κάτι το αναμενόμενο. Αυτό ισχύει, καθώς όταν χρησιμοποιείται ο XY απαιτείται κάθε φορά η μετατροπή όλων των ελεγκτών ενός μονοπατιού που ως ελεγκτών οδοφράγματος, κάτι που δεν ισχύει όταν χρησιμοποιείται ο Linear Motion. Εφόσον λοιπόν πρέπει να μετατρέπονται περισσότεροι κόμβοι σε ελεγκτές οδοφράγματος, συμπεραίνουμε ότι πρέπει να αποστέλλονται και περισσότερα πακέτα οδοφράγματος και έτσι είναι φυσιολογικό να προκύπτει και μεγαλύτερο κόστος.

Επίσης, παρατηρούμε ότι και ο αλγόριθμος XY με εκτενή πακέτα προκαλεί σε κάθε περίπτωση μικρότερο κόστος από το XY, κάτι που είναι επίσης αναμενόμενο καθώς σχεδίασα το συγκεκριμένο αλγόριθμο με σκοπό να άρω το πρόβλημα της μετατροπής όλων των ελεγκτών ενός μονοπατιού ως ελεγκτές οδοφράγματος. Εντούτοις, αν και παρατηρούμε ότι η συγκεκριμένη παραλλαγή βελτιώνει κατά πολύ την επίδοση του XY, ούτε αυτός ο αλγόριθμος επιφέρει τόσο καλά αποτελέσματα όσο ο Linear Motion. Μια υπόθεση για το λόγο που ισχύει αυτό είναι ότι κατά τη χρησιμοποίηση του XY με εκτενή πακέτα υπάρχει και πάλι αυξημένη ανάγκη για δημιουργία ελεγκτών roadblock σε σχέση με όταν χρησιμοποιείται ο Linear Motion. Ο XY με εκτενή πακέτα δε χρησιμοποιεί κάποια συγκεκριμένη λογική για να καθορίσει το κατά πόσο κάποιος ελεγκτής θα μετατραπεί σε κόμβο οδοφράγματος και έτσι αυτό εξαρτάται πάντοτε από τον κοινό προορισμό των πακέτων. Συνεπώς, εφόσον ο κοινός προορισμός των πακέτων είναι διαφορετικός κάθε φορά, κατά τη δρομολόγηση με το XY με εκτενή πακέτα ο κάθε

ελεγκτής του μονοπατιού έχει πιθανότητα 50% να μετατραπεί σε κόμβο οδοφράγματος. Στην περίπτωση του Linear Motion, τοποθετείται οδόφραγμα σε κάποιο κόμβο του πλέγματος μόνο όταν πρέπει να διαφοροποιηθεί η κατεύθυνση μετάδοσης των πακέτων από το συγκεκριμένο κόμβο. Μια εύλογη υπόθεση είναι ότι οι κόμβοι στους οποίους πραγματοποιείται η διαφοροποίησης της κατεύθυνσης μετάδοσης των πακέτων αποτελούν μειοψηφία και έτσι όταν χρησιμοποιείται ο Linear Motion η πιθανότητα που έχει κάποιος ελεγκτής να μετατραπεί σε κόμβο οδοφράγματος είναι μικρότερη από 50%. Αυτό θα δικαιολογούσε και το μικρότερο κόστος που προκύπτει όταν χρησιμοποιείται ο Linear Motion σε σχέση με όταν χρησιμοποιείται ο XY με εκτενή πακέτα.

Συνοψίζοντας, το ευρύτερο συμπέρασμα που προκύπτει από το υποκεφάλαιο αυτό, είναι ότι ο αλγόριθμος δρομολόγησης που επιφέρει το μικρότερο δυνατό κόστος και παράλληλα βοηθά στη βελτιστοποίηση της επίδοσης του αλγόριθμου αναγνώρισης σφαλμάτων είναι ο Linear Motion, κάτι που ισχύει ανεξαρτήτως των υπόλοιπων παραμέτρων. Μια εναλλακτική και επίσης αρκετά αποδοτική λύση θα μπορούσε να αποτελέσει ο XY με εκτενή πακέτα, ενώ ο XY προκαλεί σε κάθε περίπτωση το μεγαλύτερο κόστος λόγω της μετατροπής όλων των ελεγκτών σε ελεγκτές οδοφράγματος που πραγματοποιείται κατά τη χρήση του.

5.5 Σύνοψη Αποτελεσμάτων

Το γενικό συμπέρασμα που έχει προκύψει από την ανάλυση των αποτελεσμάτων είναι ότι οι διαφορετικοί συνδυασμοί τιμών εισόδου των παραμέτρων του αλγορίθμου επηρεάζουν σε μεγάλο βαθμό το κόστος που προκύπτει από τη χρησιμοποίηση του. Επίσης, έχει διαφανεί ότι αναλόγως του αριθμού των σφαλμάτων που υπάρχουν στο πλέγμα, είναι διαφορετικές και οι τιμές των παραμέτρων εισόδου που πρέπει να χρησιμοποιούνται έτσι ώστε να βελτιστοποιείται η απόδοση του αλγορίθμου και παράλληλα να ελαχιστοποιείται το κόστος το οποίο προκύπτει.

Συγκεκριμένα, από τα αποτελέσματα του κεφαλαίου 5.4 έχει διαφανεί ότι σε κάθε περίπτωση ο αλγόριθμος δρομολόγησης που προκαλεί το μικρότερο κόστος και που θα ήταν καλύτερο να τον προτιμούμε σε κάθε περίπτωση είναι ο Linear Motion. Επίσης, στο κεφάλαιο 5.3 καταλήξαμε στο συμπέρασμα ότι όταν επιδιώκουμε ελαχιστοποίηση του χρονικού κόστους και του κόστους σε πακέτα, είναι καλύτερο να χρησιμοποιείται ο αλγόριθμος επιλογής μονοπατιού 5 όταν υπάρχουν 7 ή λιγότερα σφάλματα στο πλέγμα και ο αλγόριθμος δημιουργίας μονοπατιού 4 όταν ο αριθμός των σφαλμάτων είναι μεγαλύτερος. Στην περίπτωση που επιδιώκουμε ελαχιστοποίηση των μεταβάσεων που πραγματοποιούνται, τότε καλό θα ήταν να χρησιμοποιείται ο αλγόριθμος δημιουργίας μονοπατιού 4 όταν υπάρχουν λιγότερα από 3 σφάλματα στο πλέγμα, ο αλγόριθμος δημιουργίας μονοπατιού 3 όταν υπάρχουν από 3 μέχρι 9 σφάλματα στο πλέγμα, ο αλγόριθμος δημιουργίας μονοπατιού 5 όταν υπάρχουν από 10 μέχρι 35 σφάλματα και ξανά ο αλγόριθμος δημιουργίας μονοπατιού 4 όταν ο αριθμός των σφαλμάτων ξεπερνά τα 35. Τέλος, στο κεφάλαιο 2 είχαμε συμπεράνει ότι οι μέθοδοι επιλογής NUC αποτελούν ένα αστάθμητο παράγοντα που η αποδοτικότητα που προσδίδει εξαρτάται όχι μόνο από τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα αλλά και από τις τιμές και των υπόλοιπων παραμέτρων εισόδου.

Ο πίνακας 5.13 παρουσιάζει τους προτεινόμενους συνδυασμούς τιμών των παραμέτρων εισόδου, οι οποίες επιφέρουν το μικρότερο χρονικό κόστος και το μικρότερο κόστος σε πακέτα αναλόγως του αριθμού των σφαλμάτων που υπάρχουν στο πλέγμα. Επίσης, ο πίνακας 5.14 παρουσιάζει τους αντίστοιχους προτεινόμενους συνδυασμούς τιμών των παραμέτρων εισόδου, οι οποίες επιφέρουν το μικρότερο

κόστος σε μεταβάλειες και πάλι αναλόγως του αριθμού των σφαλμάτων που υπάρχουν στο πλέγμα.

Συνδυασμός τιμών των παραμέτρων που προκαλεί το μικρότερο χρονικό κόστος και το μικρότερο κόστος σε πακέτα ανά αριθμό σφαλμάτων																					
	Ποσότητα Σφαλμάτων																				
Παράμετρος Εισόδου	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**
Αλγόριθμος Δρομολόγησης	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM
Αλγόριθμος Δημιουργίας Μονοπατιού	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Μέθοδος Επιλογής NUC	20	20	15	15	13	15	15	15	15	15	15	13	13	13	6	15	9	9	13	2	6

Πίνακας 5.13: Ο συνδυασμός τιμών των παραμέτρων εισόδου που προκαλεί το μικρότερο χρονικό κόστος και το μικρότερο κόστος σε πακέτα αναλόγως του αριθμού των σφαλμάτων που υπάρχουν στο πλέγμα. Για τη δημιουργία του πίνακα χρησιμοποιήθηκαν δεδομένα από τις γραφικές παραστάσεις 5.13, 5.14, 5.22, 5.23, 5.28 και 5.29.

Συνδυασμός τιμών των παραμέτρων που προκαλεί το μικρότερο κόστος σε μεταβάσεις ανά αριθμό σφαλμάτων																					
	Ποσότητα Σφαλμάτων																				
Παράμετρος Εισόδου	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**
Αλγόριθμος Δρομολόγησης	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM	LM
Αλγόριθμος Δημιουργίας Μονοπατιού	4	3	5	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4
Μέθοδος Επιλογής NUC	18	6	20	20	20	20	20	20	15	10	17	13	10	13	10	15	15	13	13	13	6

Πίνακας 5.14: Ο συνδυασμός τιμών των παραμέτρων εισόδου που προκαλεί το μικρότερο κόστος σε μεταβάσεις αναλόγως του αριθμού των σφαλμάτων που υπάρχουν στο πλέγμα. Για τη δημιουργία του πίνακα χρησιμοποιήθηκαν δεδομένα από τις γραφικές παραστάσεις 5.14, 5.24, 5.30 και Α.19.

Κεφάλαιο 6

Συμπεράσματα

6.1 Γενικά Συμπεράσματα

Για την επιτυχή δημιουργία μιας λειτουργικής υπερεπιφάνειας, είναι απαραίτητη προϋπόθεση να υπάρχει πάντοτε δυνατότητα επικοινωνίας με όλους τους λειτουργικούς και προσβάσιμους ελεγκτές που βρίσκονται τοποθετημένοι σε αυτή. Το πιο θεμελιώδες βήμα για την επίτευξη αυτής της επικοινωνίας, είναι η αναγνώριση των σφαλμάτων τα οποία ενδέχεται να την αποτρέπουν. Ο αλγόριθμος που παρουσιάζεται σε αυτή την αναφορά επιτυγχάνει την πλήρη και ορθή αναγνώριση όλων των μόνιμων σφαλμάτων τα οποία θα παρουσιάζονται στους ελεγκτές της υπερεπιφάνειας.

Επίσης, ο αλγόριθμος ικανοποιεί όλες τις απαιτήσεις και τους περιορισμούς που έχουν τεθεί στα πλαίσια του ερευνητικού προγράμματος Visorsurf, καθώς μπορεί να χρησιμοποιηθεί σε τοπολογία Manhattan και χαρακτηρίζεται από δυνατότητα επεκτασιμότητας (scalability). Ακόμη, η υλοποίηση του δεν απαιτεί την προσθήκη επιπλέον υλικού, αλλά ούτε και τη δέσμευση μνήμης από τους ελεγκτές ή τη διεκπεραίωση περίπλοκων υπολογισμών από αυτούς. Ακόμη, ο αλγόριθμος μπορεί να λειτουργήσει σε συνδυασμό με τις υπάρχουσες τεχνολογίες που χρησιμοποιούνται στην υπερεπιφάνεια, όπως ο αλγόριθμος δρομολόγησης XY. Όλα αυτά υποδεικνύουν ότι ο προτεινόμενος αλγόριθμος πληροί όλες τις απαραίτητες προϋποθέσεις για να μπορέσει να εφαρμοστεί και σε πρακτικό επίπεδο στα πλαίσια του project Visorsurf. Παράλληλα, το γεγονός ότι ο αλγόριθμος μπορεί να υποστηριχτεί και από δίκτυα Manhattan που χρησιμοποιούν οποιοδήποτε άλλο αλγόριθμο δρομολόγησης, υποδεικνύει την προσαρμοστικότητα του και την προοπτική χρήσης του σε μελλοντικά έργα.

Όσον αφορά την ορθότητα του αλγορίθμου, αυτή έχει αποδειχθεί τόσο θεωρητικά μέσω της απόδειξης του Κεφαλαίου 3.8, όσο και εμπειρικά μέσω των προσομοιώσεων που έχουν πραγματοποιηθεί. Συγκεκριμένα, έχει αποδειχθεί ότι ο αλγόριθμος επιτυγχάνει το στόχο του και μπορεί να κάνει πλήρη αναγνώριση σφαλμάτων,

ανακαλύπτοντας κατά πόσο ο κάθε ελεγκτής του πλέγματος είναι λειτουργικός, μη λειτουργικός ή μη προσβάσιμος από την πηγή.

Επιπλέον, η τεχνική ανάστροφης μετάδοσης σφάλματος που χρησιμοποιείται από τον αλγόριθμο έχει διαφανεί ότι συντείνει στον πολύ πιο γρήγορο τερματισμό του και παράλληλα επιφέρει πολύ πιο μικρό κόστος σε σχέση με άλλες προσεγγίσεις που δοκιμάστηκαν παλαιότερα, όπως η αναγνώριση σφαλμάτων μέσω πακέτων επιβεβαίωσης. Το κύριο μειονέκτημα της συγκεκριμένης προσέγγισης είναι ότι δε μπορεί να υποστηρίξει την ταυτόχρονη μετάδοση πολλαπλών πακέτων αναγνώρισης σφαλμάτων σε διαφορετικά μονοπάτια. Αυτό ισχύει, καθώς αν γινόταν ταυτόχρονη χρήση πολλών μονοπατιών, η ασύγχρονη φύση του δικτύου μπορεί να οδηγούσε σε παγίδευση πολλών πακέτων μέσα σε αυτό και συνεπώς σε λανθασμένη συσχέτιση σφαλμάτων. Συνεπώς, για να πραγματοποιηθεί αναγνώριση σφαλμάτων με τη χρήση πακέτων επιβεβαίωσης θα πρέπει να υπάρχει κάθε στιγμή μόνο ένα πακέτο στο δίκτυο, κάτι που καθιστά τη συγκεκριμένη διαδικασία ιδιαίτερα χρονοβόρα. Επίσης, η προσπάθεια που έγινε για να χρησιμοποιηθεί η συγκεκριμένη μέθοδος, ανέδειξε ότι χωρίς τη χρησιμοποίηση των οδοφραγμάτων δύσκολα μπορεί να επιτευχθεί πλήρης αναγνώριση των σφαλμάτων. Συγκεκριμένα, ένας αλγόριθμος που δημιουργήθηκε και αφορούσε αναγνώριση σφαλμάτων χρησιμοποιώντας πακέτα επιβεβαίωσης αλλά όχι οδοφράγματα, μπορούσε να εντοπίσει μόνο το 90-95% των σφαλμάτων που παρουσιάζονταν σε ελεγκτές.

Επιστρέφοντας στον αλγόριθμου μου, αυτό που διαπιστώθηκε μέσω της πειραματικής ανάλυσης είναι ότι υπάρχει υψηλή συσχέτιση μεταξύ των παραμέτρων εισόδου και ο τρόπος με τον οποίο επηρεάζει η μια την άλλη είναι αρκετές φορές απρόβλεπτος. Εντούτοις, φάνηκε ότι αναλόγως του αριθμού των σφαλμάτων που υπάρχουν στο πλέγμα, υπάρχει πάντοτε κάποιος συνδυασμός παραμέτρων εισόδου που επιφέρει καλύτερα αποτελέσματα για τον αλγόριθμο. Έτσι, βάσει του αριθμού των σφαλμάτων που εντοπίστηκαν κατά την προηγούμενη εκτέλεση του αλγορίθμου ή γενικότερα του αριθμού των σφαλμάτων που αναμένεται να εντοπιστούν στο πλέγμα, πρέπει να επιλέγονται πάντοτε και οι κατάλληλες παράμετροι εισόδου για τον αλγόριθμο.

6.2 Περιορισμοί

Ο αλγόριθμος που έχω σχεδιάσει διακρίνεται και από ορισμένους περιορισμούς. Ο σημαντικότερος από αυτούς τους περιορισμούς, είναι ότι γίνεται εστίαση σε ένα συγκεκριμένο είδος σφαλμάτων, στα σφάλματα που αφορούν δυσλειτουργία των ελεγκτών. Εντούτοις, σε πρακτικό επίπεδο είναι δυνατό να παρουσιαστούν και διαφορετικές μορφές σφαλμάτων στο πλέγμα και μια εξίσου σημαντική κατηγορία σφαλμάτων είναι και τα σφάλματα που είναι δυνατό να προκύπτουν από συνδέσεις μεταξύ ελεγκτών οι οποίες είναι δυσλειτουργικές. Λόγω της χρησιμοποίησης των οδοφραγμάτων, ο αλγόριθμος μου χαρακτηρίζεται από μεγάλη ευελιξία στην εύρεσης και δρομολόγηση πακέτων μέσω εναλλακτικών μονοπατιών. Συνεπώς, θεωρώ ότι στα πλαίσια μιας μελλοντικής εργασίας θα μπορούσε να επεκταθεί η υπάρχουσα μορφή του αλγορίθμου, έτσι ώστε να είναι δυνατό με τον ίδιο αλγόριθμο να πραγματοποιείται αναγνώριση των σφαλμάτων που υπάρχουν τόσο στους ελεγκτές του πλέγματος, όσο και στις συνδέσεις μεταξύ αυτών.

Ένας άλλος περιορισμός αφορά το είδος των σφαλμάτων που αναγνωρίζονται από τον αλγόριθμο μου. Ο αλγόριθμος μου εστιάζεται στον εντοπισμό των μόνιμων σφαλμάτων τα οποία αναμένεται να αποτελέσουν και την πιο μεγάλη απειλή για την εύρυθμη λειτουργία της υπερεπιφάνειας. Εντούτοις, θα ήταν χρήσιμος να υπάρχει κάποιος επιπλέον μηχανισμός ή αλγόριθμος για τον εντοπισμό των παροδικών αλλά και των διαλειπόντων σφαλμάτων του συστήματος, αλλά και κάποιος τρόπος να πραγματοποιείται διαχωρισμός μεταξύ των συγκεκριμένων ειδών σφαλμάτων που ενδέχεται να προκύψουν στο πλέγμα. Κάτι τέτοιο ίσως να μπορεί να επιτευχθεί μέσω επαναλαμβανόμενων εκτελέσεων του δικού μου αλγόριθμου, ωστόσο αυτό ίσως επιφέρει μεγάλο κόστος.

Πέρα από τους περιορισμούς που αφορούν τις δυνατότητες που παρέχει ο αλγόριθμος μου, υπάρχουν και κάποιοι άλλοι περιορισμοί που σχετίζονται με τα πειράματα τα οποία έχω διεξάγει. Ένας βασικός περιορισμός ο οποίος αναλύεται εκτενώς στο Κεφάλαιο 5.1, σχετίζεται με τον καθορισμό της τοποθεσίας των σφαλμάτων στο πλέγμα ως μη ελεγχόμενης μεταβλητής κατά τη διεξαγωγή των πειραμάτων. Η ύπαρξη τοποθετήσεων σφαλμάτων κατά τις οποίες ο αριθμός των μη προσβάσιμων ελεγκτών

ήταν μεγάλος ή υπήρχαν πολλοί μη λειτουργικοί ελεγκτές κοντά στην πηγή, πιθανόν να επηρέασαν σε κάποιο βαθμό τα αποτελέσματα των προσομοιώσεων. Εντούτοις, η αφαίρεση των αποκλινόντων δεδομένων θεωρώ ότι βοήθησε σε μεγάλο βαθμό στην άμβλυνση του εν λόγω προβλήματος.

Μια ακόμη παραχώρηση που έγινε κατά την πραγματοποίηση των πειραμάτων ήταν το ότι δε λήφθηκε υπόψη ο χρόνος που χρειάζεται η πηγή για να ολοκληρώσει του διάφορους υπολογισμούς που πραγματοποιούνται κατά την εκτέλεση του αλγορίθμου αναγνώρισης σφαλμάτων. Η πηγή είναι ένας υπολογιστής που διακρίνεται από μεγάλη υπολογιστική ισχύ και είναι σε θέση να ολοκληρώσει απλούς υπολογισμούς σε πολύ μικρό χρονικό διάστημα. Συνεπώς, δεν έλαβα υπόψη τις καθυστερήσεις που θα προκύπτουν από την πηγή και επέλεξα να εστιάσω κάθε φορά στις καθυστερήσεις που θα προκύπτουν από τη μετάδοση των πακέτων στο πλέγμα, όπου ήταν και η ουσία του προβλήματος που κλήθηκα να επιλύσω.

Ακόμη ένα πρόβλημα για εμένα ήταν και το γεγονός ότι ακόμη δεν έχουν καθοριστεί οι ακριβείς καθυστερήσεις που θα προκύπτουν στο πλέγμα κατά τη μετάδοση των πακέτων μεταξύ των διάφορων ελεγκτών και buffers. Έτσι, για να διασφαλίσω την ορθότητα των αποτελεσμάτων μου, έθεσα κάποια προκαθορισμένα όρια καθυστερήσεων σε μονάδες χρόνου τα οποία ήταν κοινά για όλα τα πειράματα που διενήργησα, καθιστώντας τη συγκριμένη μεταβλητή ως ελεγχόμενη. Εντούτοις, εφόσον δεν ήταν εφικτό να λάβω υπόψη μου τα πραγματικά όρια καθυστερήσεων πριν διενεργήσω τα πειράματα μου, θα χρειαστεί να διενεργηθούν νέα πειράματα με τα πραγματικά δεδομένα για να υπολογιστεί με ακριβή τρόπο η χρονική καθυστέρηση που θα προκύπτει από την εκτέλεση του αλγορίθμου μου.

6.3 Προτάσεις για Επέκταση Ιδέας και Μελλοντική Εργασία

Πέρα από την επανεκτέλεση των πειραμάτων με τα σωστά χρονικά δεδομένα για να διαφανεί ο χρόνος εκτέλεσης του αλγορίθμου μου, υπάρχουν και πολλές επιπρόσθετες ιδέες που μπορούν να εφαρμοστούν για να επεκταθεί και να αξιοποιηθεί περαιτέρω ο αλγόριθμος, αλλά και για να αναλυθούν περαιτέρω κάποια από τα αποτελέσματα που έχουν προκύψει μέσω των πειραμάτων μου.

Κάτι το οποίο θεωρώ ότι πρέπει να αξιολογηθεί, είναι το κατά πόσο θα πρέπει ο αλγόριθμος να εκτελείται και σε κάποιες επιπλέον περιπτώσεις, πέρα από αυτές που αναφέρονται στο Κεφάλαιο 3.9. Πιο συχνή εκτέλεση του αλγορίθμου είναι δυνατό να βοηθήσει στον έγκαιρο εντοπισμό των σφαλμάτων, πριν αυτά δημιουργήσουν κάποια δυσλειτουργία στη υπερεπιφάνεια. Επιπλέον, μέσω συχνότερων εκτελέσεων ίσως είναι δυνατός και ο εντοπισμός των παροδικών και των διαλειπόντων σφαλμάτων. Εντούτοις, πιο συχνή εκτέλεση του αλγορίθμου σίγουρα θα επιφέρει και περισσότερο κόστος στο σύστημα και από τη στιγμή που δεν υπάρχει κάποια ακριβής πρόβλεψη του χρόνου εκτέλεσης του αλγορίθμου στην πράξη, είναι αδύνατο να γίνει κάποια περαιτέρω προτροπή σχετικά με τη συχνότητα εκτέλεσης του αλγορίθμου στο παρών στάδιο.

Μια άλλη ιδέα για μελλοντική μελέτη, είναι η πιο λεπτομερής εξέταση των αποτελεσμάτων που αφορούν τους αλγόριθμους δρομολόγησης Linear Motion και XY με εκτενή πακέτα. Όπως φάνηκε πειραματικά, η χρησιμοποίηση του Linear Motion επιφέρει μικρότερο κόστος από τη χρησιμοποίηση του XY με εκτενή πακέτα. Στο Κεφάλαιο 5.4, δόθηκε μια λογική υπόθεση για το λόγο που επιφέρει αυτή τη διαφορά απόδοσης μεταξύ των αλγορίθμων. Εντούτοις, η συγκεκριμένη υπόθεση δε διερευνήθηκε περαιτέρω και μπορεί σίγουρα να αποτελέσει την αφετηρία για τη διενέργεια επιπρόσθετων πειραμάτων ή μιας μαθηματικής απόδειξης για να την επαληθεύσουν.

Όσον αφορά την επέκταση του αλγορίθμου, θεωρώ ότι προτεραιότητα πρέπει να αποτελέσει η προσθήκη της δυνατότητας αναγνώρισης σφαλμάτων που προκύπτουν στις συνδέσεις μεταξύ των ελεγκτών. Το συγκεκριμένο είδος αναμένεται να

παρουσιάζεται σε εκτενή βαθμό στο πλέγμα και η μη αντιμετώπιση του θα εμπερικλείει σοβαρούς κινδύνους για την εύρυθμη λειτουργία της υπερεπιφάνειας.

Μια ακόμη ιδέα είναι το να μπορεί ο αλγόριθμος μέσω της αξιοποίησης της γνώσης που προέκυψε από την προηγούμενη του εκτέλεση, να προβαίνει σε μια γρηγορότερη αναγνώριση των σφαλμάτων. Εντούτοις η οποιαδήποτε μελέτη γίνει προς αυτή την κατεύθυνση θα πρέπει να λάβει υπόψη ότι ο πιθανός εντοπισμός παροδικών σφαλμάτων κατά τη διάρκεια μιας εκτέλεσης, καθιστά αδύνατη την απλή επέκταση του συνόλου των μη λειτουργικών ελεγκτών χωρίς τη διενέργεια επαναξιολόγησης.

Τέλος, για να μπορέσει να αξιοποιηθεί πλήρως ο αλγόριθμος αναγνώρισης σφαλμάτων που έχω σχεδιάσει, θα πρέπει σίγουρα να δημιουργηθεί και ένας αλγόριθμος δρομολόγησης στην παρουσία σφαλμάτων έτσι ώστε να είναι μπορεί να ολοκληρωθεί η διαδικασία αντιμετώπισης των σφαλμάτων που είναι δυνατό να παρουσιαστούν στο πλέγμα. Η δυνατότητα δημιουργίας δυναμικών μονοπατιών στο πλέγμα, κάτι που επιτυγχάνεται μέσω της εισαγωγής οδοφραγμάτων, ίσως να μπορεί να αποτελέσει το κλειδί για τη δημιουργία ενός τέτοιου αλγορίθμου.

Βιβλιογραφία

[1] K. Aisopos, A. DeOrio, L. Peh and V. Bertacco, "Ariadne: Agnostic reconfiguration in a disconnected network environment," in *Proc. International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2011, pp. 298-309.

[2] R. Bellman, "On a routing problem," *Q Appl Math*, vol. 16, no. 1, pp. 87-90, Dec. 1956.

[3] G. Chiu, "The odd-even turn model for adaptive routing," *IEEE Trans.Parallel Distrib.Syst.*, vol. 11, no. 7, pp. 729-738, Jul. 2000.

[4] E.W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269-271, Jun. 1959.

[5] L.R. Ford Jr, "Network flow theory," The *Rand Corporation*, Technical Report P-932, Aug. 1956.

[6] F.C. Gärtner, "Fundamentals of fault-tolerant distributed computing in asynchronous environments," *ACM Computing Surveys (CSUR)*, vol. 31, no. 1, pp. 1-26, Mar. 1999.

[7] M. Radetzki, C. Feng, X. Zhao and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 8, Jan. 2013.

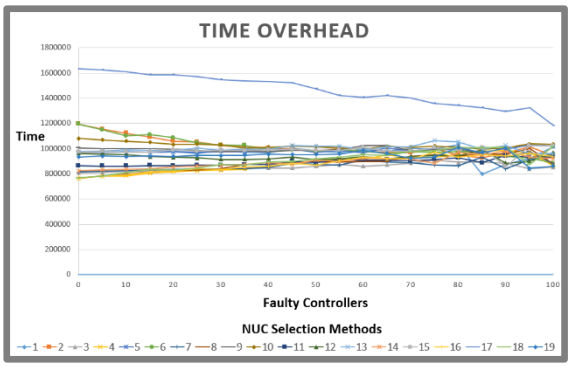
[8] X. Tran, J. Durupt, F. Bertrand, V. Beroulle and C. Robach, "A DFT architecture for asynchronous networks-on-chip," in *Proc. 11th European Test Symposium (ETS)*, 2006, pp. 219-224.

[9] A. Vitkovskiy, V. Soteriou and C. Nicopoulos, "Dynamic fault-tolerant routing algorithm for networks-on-chip based on localised detouring paths," *IET Computers & Digital Techniques*, vol. 7, no. 2, pp. 93-103, Mar. 2013.

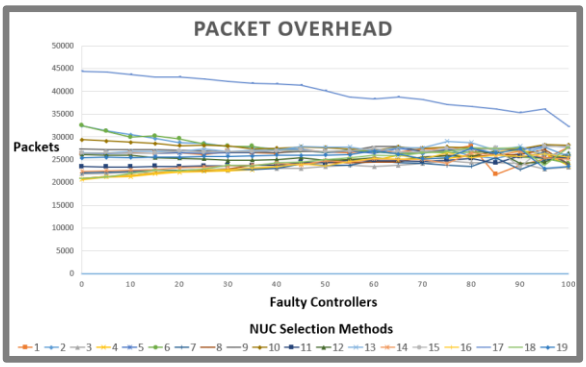
[10] W. Zhang, L. Hou, J. Wang, S. Geng and W. Wu, "Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip," in *Proc. WRI Global Congress on Intelligent Systems*, 2009, pp. 329-333.

[11] "Home," *VISORSURF*. [Online]. Available: http://www.visorsurf.eu/. [Accessed: 21-May-2019].

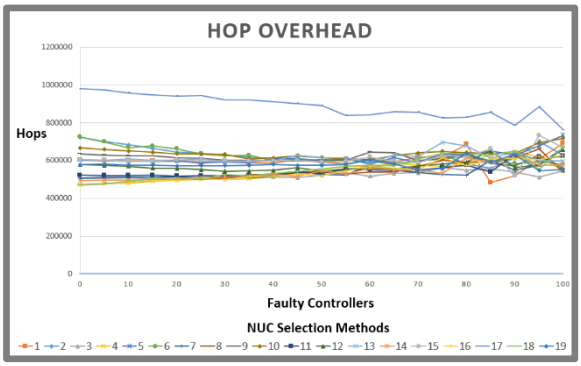
Παράρτημα Α



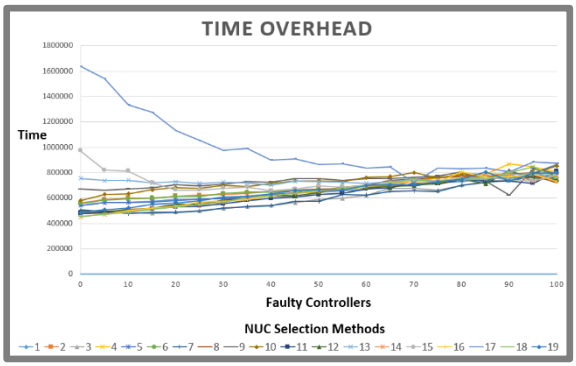
Σχήμα Α.1: Το χρονικό κόστος σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 1 του Κεφαλαίου 5.2.



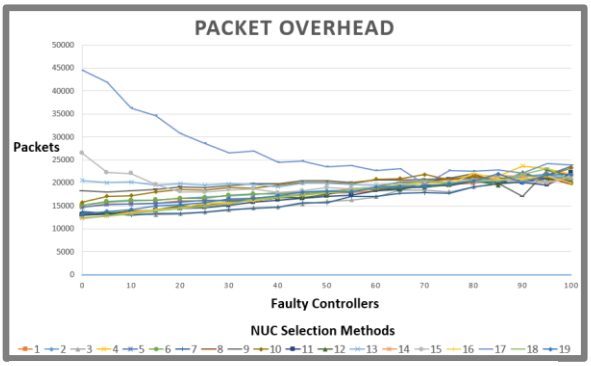
Σχήμα Α.2: Το κόστος σε πακέτα σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 1 του Κεφαλαίου 5.2.



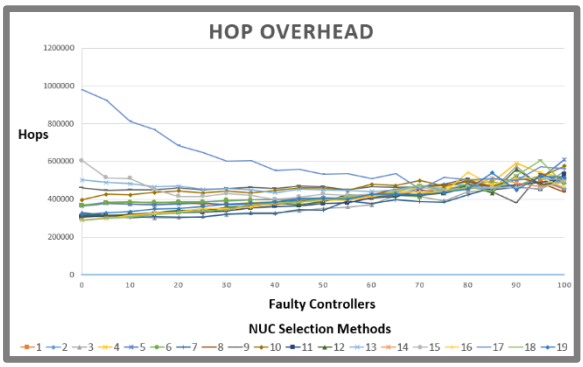
Σχήμα Α.3: Το κόστος σε μεταδόσεις σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 1 του Κεφαλαίου 5.2.



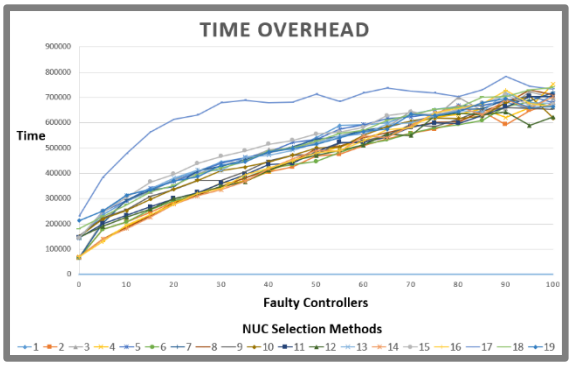
Σχήμα A.4: Το χρονικό κόστος σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 2 του Κεφαλαίου 5.2.



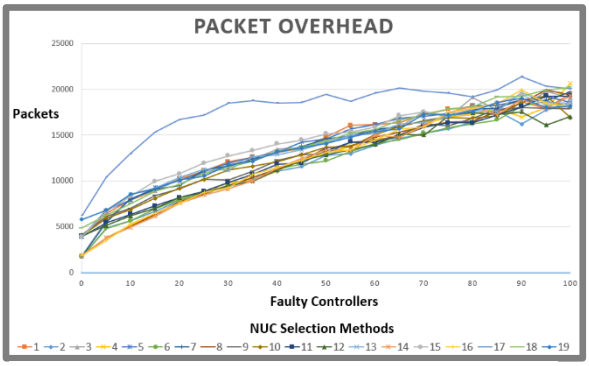
Σχήμα A.5: Το κόστος σε πακέτα σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 2 του Κεφαλαίου 5.2.



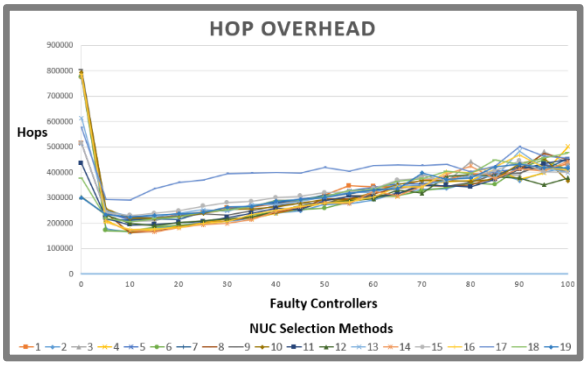
Σχήμα A.6: Το κόστος σε μεταδόσεις σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 2 του Κεφαλαίου 5.2.



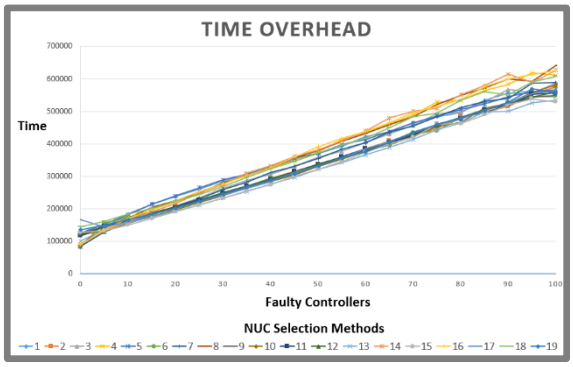
Σχήμα Α.7: Το χρονικό κόστος σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 3 του Κεφαλαίου 5.2.



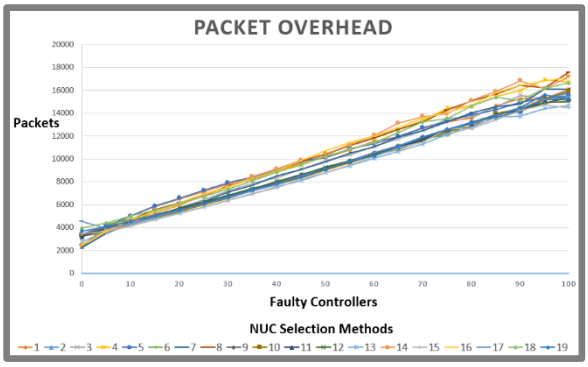
Σχήμα Α.8: Το κόστος σε πακέτα σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 3 του Κεφαλαίου 5.2.



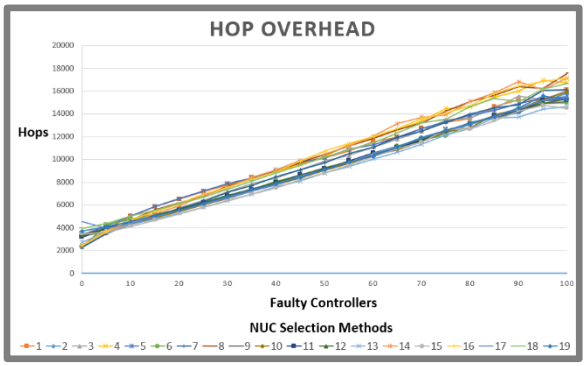
Σχήμα Α.9: Το κόστος σε μεταδόσεις σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 3 του Κεφαλαίου 5.2.



Σχήμα Α.10: Το χρονικό κόστος σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 5 του Κεφαλαίου 5.2.



Σχήμα Α.11: Το κόστος σε πακέτα σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 5 του Κεφαλαίου 5.2.



Σχήμα Α.12: Το κόστος σε μεταδόσεις σε σχέση με τον αριθμό των σφαλμάτων που υπάρχουν στο πλέγμα, για κάθε μέθοδο επιλογής NUC, όπως προέκυψε κατά την εκτέλεση του πειράματος 5 του Κεφαλαίου 5.2.

Παράρτημα Β

Μέθοδος επιλογής NUC που προκαλεί το μικρότερο κόστος ανά αριθμό σφαλμάτων																					
	Ποσότητα Σφαλμάτων																				
Είδος Κόστους	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**
Χρονικό Κόστος	4	4	4	4	4	4	4	4	3	3	3	7	3	3	3	7	7	1	7	3	3
Κόστος σε πακέτα	4	4	4	4	4	4	4	4	3	3	3	7	3	3	3	7	7	1	7	3	3
Κόστος σε μεταδόσεις	18	18	4	4	4	8	16	4	7	3	16	7	3	3	3	7	7	1	15	3	8

Πίνακας Β.1: Παρουσίαση των μεθόδων επιλογής NUC που επιφέρουν το μικρότερο κόστος ανά αριθμό σφαλμάτων, όταν χρησιμοποιείται ο αλγόριθμο δημιουργίας μονοπατιού 1 και ο αλγόριθμος δρομολόγησης Linear Motion. Τα αποτελέσματα εξάχθηκαν από τις γραφικές παραστάσεις Α.1, Α.2 και Α.3 του Παραρτήματος Α.

Μέθοδος επιλογής NUC που προκαλεί το μικρότερο κόστος ανά αριθμό σφαλμάτων																					
	Ποσότητα Σφαλμάτων																				
Είδος Κόστους	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**
Χρονικό Κόστος	14	16	7	3	3	7	3	7	7	3	7	3	3	7	7	7	7	12	9	11	8
Κόστος σε πακέτα	14	16	7	3	3	7	3	7	7	3	7	3	3	7	7	7	7	12	9	11	8
Κόστος σε μεταδόσεις	16	16	7	3	3	7	3	7	7	3	7	3	3	7	7	7	7	12	9	11	8

Πίνακας Β.2: Παρουσίαση των μεθόδων επιλογής NUC που επιφέρουν το μικρότερο κόστος ανά αριθμό σφαλμάτων, όταν χρησιμοποιείται ο αλγόριθμο δημιουργίας μονοπατιού 2 και ο αλγόριθμος δρομολόγησης Linear Motion. Τα αποτελέσματα εξάχθηκαν από τις γραφικές παραστάσεις Α.4, Α.5 και Α.6 του Παραρτήματος Α.

Μέθοδος επιλογής NUC που προκαλεί το μικρότερο κόστος ανά αριθμό σφαλμάτων																					
	Ποσότητα Σφαλμάτων																				
Είδος Κόστους	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**
Χρονικό Κόστος	6	16	14	14	14	14	14	12	2	2	6	2	2	6	12	2	6	6	2	12	10
Κόστος σε πακέτα	6	16	14	14	14	14	14	12	2	2	6	2	2	6	12	2	6	6	2	12	10
Κόστος σε μεταδόσεις	19	6	8	14	14	14	14	14	4	2	6	14	2	4	12	2	11	6	2	12	10

Πίνακας Β.3: Παρουσίαση των μεθόδων επιλογής NUC που επιφέρουν το μικρότερο κόστος ανά αριθμό σφαλμάτων, όταν χρησιμοποιείται ο αλγόριθμο δημιουργίας μονοπατιού 3 και ο αλγόριθμος δρομολόγησης Linear Motion. Τα αποτελέσματα εξάχθηκαν από τις γραφικές παραστάσεις Α.7, Α.8 και Α.9 του Παραρτήματος Α.