ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Συγκριτική Μελέτη Δικτύων Διασύνδεσης ATM και Wormhole με Χρήση Ιχνών Πραγματικής Κίνησης

Μαρίνα Μωραΐτη

Μεταπτυχιακή Εργασία

Ηράκλειο, Οκτώβριος 1997

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Συγκριτική Μελέτη Δικτύων Διασύνδεσης ATM και Wormhole με Χρήση Ιχνών Πραγματικής Κίνησης

Εργασία που υποβλήθηκε από την
Μαρίνα Μωραΐτη
ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Μαρίνα Μωραΐτη
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Ευάγγελος Μαρκάτος, Επίκουρος Καθηγητής, Επόπτης

Δημήτριος Σερπάνος, Επίκουρος Καθηγητής, Μέλος

Γεώργιος Σταμούλης, Επίκουρος Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Οκτώβριος 1997

Συγκριτική Μελέτη Δικτύων Διασύνδεσης ATM και Wormhole με χρήση Ιχνών Πραγματικής Κίνησης

Μαρίνα Μωραΐτη

Μεταπτυχιακή εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

ΠΕΡΙΛΗΨΗ

Το δίκτυο διασύνδεσης αποτελεί βασικό δομικό στοιχείο των παράλληλων μηχανών, το οποίο συχνά επηρεάζει την απόδοση των παράλληλων εφαρμογών. Γι' αυτόν το λόγο, η εξέταση της συμπεριφοράς του δικτύου αποτέλεσε τα τελευταία χρόνια ένα από τα πιο δημοφιλή θέματα μελέτης της απόδοσης των παράλληλων συστημάτων. Κύριο χαρακτηριστικό των ερευνών που έχουν γίνει, είναι ότι οι περισσότερες παράγουν την κίνηση που κυκλοφορεί μέσα στο δίκτυο με τεχνητούς τρόπους, χρησιμοποιώντας μαθηματικά μοντέλα. Σε αντίθεση με την εξέταση των άλλων τμημάτων της παράλληλης μηχανής (caches, memories) σπάνια για την μελέτη του δικτύου χρησιμοποιείται η γνώση που πηγάζει από τα workloads παράλληλων εφαρμογών.

Η εργασία αυτή χωρίζεται σε δύο μέρη. Το πρώτο μέρος ασχολείται με την μελέτη της κίνησης που προέρχεται από 10 παράλληλες εφαρμογές, οι οποίες εκτελέστηκαν στην παράλληλη μηχανή IBM SP2 του Maui High Performance Computing Center. Η μελέτη επικεντρώνεται κυρίως στην συχνότητα παραγωγής των μηνυμάτων, στην κατανομή τους στους κόμβους του συστήματος και στο μέγεθος τους και δείχνει πως η πραγματική κίνηση διέπεται από ιδιότητες εντελώς διαφορετικές από αυτές που χαρακτηρίζουν την κίνηση που παράγεται με συνθετικά μέσα.

Στο δεύτερο μέρος η κίνηση που καταγράφηκε χρησιμοποιείται για τη σύγκριση των επιδόσεων δικτύων που αποτελούνται από μεταγωγείς που χρησιμοποιούν την μέθοδο ATM με έλεγχο ροής υλοποιημένο με εισιτήρια, με δίκτυα που αποτελούνται από μεταγωγείς που χρησιμοποιούν την μέθοδο wormhole. Γίνονται πειράματα για τρεις διαφορετικές τοπολογίες δικτύων (Butterfly, hypercube και mesh). Τα αποτελέσματα δείχνουν πως η χρήση των λωρίδων μειώνει σημαντικά την καθυστέρηση του δικτύου. Ομως, όσο αφορά στον συνολικό χρόνο εκτέλεσης των εφαρμογών οι βελτιώσεις που παρατηρούνται είναι πολύ μικρές. Αυτό οφείλεται στον πολύπλοκο τρόπο με τον οποίο

αλληλεπιδρούν οι παράλληλες εφαρμογές και οι παράλληλες αρχιτεκτονικές, ο οποίος εξαλείφει τα θετικά οφέλη από τη χρήση λωρίδων.

Επόπτης : Ευάγγελος Μαρκάτος
Επίκουρος Καθηγητής Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Trace-driven Simulation of ATM and Wormhole networks

Marina Moraiti

Master of Science Thesis

Department of Computer Science
University of Crete

ABSTRACT

Interconnection network is a vital component of a parallel machine and is often the limiting factor in the performance of several parallel applications. This is the reason why its performance evaluation has been a widely researched topic over the past years. However, unlike the other subsystems, interconnection network design and analysis has rarely used the knowledge of workloads generated by parallel applications. Instead, synthetic workloads have been used by most studies.

In our work, we first develop a framework for characterizing the communication properties of parallel applications. For our study, we use ten parallel applications that have been executed on IBM SP2 parallel machine of Maui High Performance Computing Center. Message generation frequency, spatial distribution of messages and message length are the three attributes that quantify any communication. Communication analysis shows that synthetic workloads do not capture the behavior of real applications.

Moreover, realistic workloads are used to measure and compare the performance of ATM networks with credit based flow control and wormhole. Measurements have been made for three network topologies, butterfly, mesh and hypercube respectively. Results show that by using lanes network latency can be reduced. However, with respect to overall execution time, the performance benefit is negligible, due to the complex interaction between a parallel architecture and an application.

Supervisor : Evangelos Markatos
Assistant Professor of Computer Science
University of Crete

Ευχαριστίες

Με το πέρας της εργασίας αυτής, θα ήθελα να ευχαριστήσω τον κ. Ε. Μαρκάτο για την καθοριστική συμβολή του καθ'όλη τη διάρκειά της. Με βοήθησε σημαντικά σε όλα τα στάδια, από την αρχή, με τη σύλληψη της αρχικής ιδέας, αλλά και στη συνέχεια με ουσιαστικές παρατηρήσεις και σχόλια. Επίσης θέλω να ευχαριστήσω τα μέλη της επιτροπής μου, τον κ. Δ. Σερπάνο και τον κ. Γ. Σταμούλη για τις υποδείξεις και τις παρατηρήσεις τους.

Ευχαριστώ το Ινστιτούτο Πληροφορικής του Ι.Τ.Ε. για την οικονομική και υλικοτεχνική υποστήριξη που μου παρείχε και το Maui High Performance Computing Center, για την παραχώρηση πόρων, οι οποίοι ήταν απολύτως απαραίτητοι για την ολοκλήρωση της εργασίας αυτής.

Από καρδιάς ευχαριστώ την Ανθούλα που με ενθάρρυνε και με βοήθησε σημαντικά στα τελευταία στάδια της εργασίας μου. Η Ανθούλα έκανε επίσης υποδείξεις που συνέβαλαν στην καλύτερη παρουσίαση της δουλειάς μου.

Τελειώνοντας, θα ήθελα να ευχαριστήσω τους γονείς μου Νίκο και Μαρία και τις αδελφές μου Κοραλία και Αμέλια για την αγάπη τους και τη συναισθηματική στήριξη που μου προσφέρουν στις δύσκολες στιγμές. Τέλος μιλώντας για την οικογένειά μου, δεν θα μπορούσα να μην αναφέρω τις Αναστασία Αναστασιάδη, Ανθή Γιώρτσου, Αντζυ Χλαπάνη, Φλώρα Χρυσού, Τζούλια Σεβασλίδου και τον Γιάννη Πάτρα που έγιναν για μένα αυτά τα 6 χρόνια στο Ηράκλειο μια δεύτερη οικογένεια.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Ο πολύπλοκος τρόπος με τον οποίο αλληλεπιδρούν οι παράλληλες αρχιτεκτονικές και οι εφαρμογές κάνει απαραίτητη την χρήση ρεαλιστικών workloads, για την αξιολόγηση των παράλληλων συστημάτων. Γι'αυτό το λόγο η ανάλυση της απόδοσης των επεξεργαστών, των κρυφών μνημών (caches), των κύριων μνημών και των I/O συσκευών έχει γίνει χρησιμοποιώντας benchmarks για παράλληλα συστήματα.

Το δίκτυο διασύνδεσης αποτελεί και αυτό ένα βασικό δομικό στοιχείο των παράλληλων μηχανών, το οποίο συχνά επηρεάζει αρνητικά την απόδοση των παράλληλων εφαρμογών. Γι' αυτόν το λόγο, η εξέταση της συμπεριφοράς και απόδοσής του δικτύου, αποτέλεσε τα τελευταία χρόνια ένα από τα πιο δημοφιλή θέματα μελέτης. Ομως σε αντίθεση με την εξέταση των άλλων τμημάτων της παράλληλης μηχανής, σπάνια για τη δική του μελέτη χρησιμοποιήθηκε η γνώση που πηγάζει από τα workloads των παράλληλων εφαρμογών.

Υπάρχουν δυο διαφορετικοί τρόποι αντιμετώπισης του δικτύου διασύνδεσης. Από την σκοπιά των σχεδιαστών λογισμικού και των προγραμματιστών συχνά γίνονται κάποιες υποθέσεις για το δίκτυο, όπως για παράδειγμα ότι ο χρόνος μεταφοράς ενός μηνύματος είναι σταθερός. Υιοθετείται δηλαδή ένα απλοποιημένο μοντέλο αναπαράστασης του δικτύου, το οποίο δεν εξετάζει τις λεπτομέρειες μετάδοσης του μηνύματος μέσα στο δίκτυο. Αυτές οι υποθέσεις επιτρέπουν στους σχεδιαστές να επικεντρωθούν στα αντικείμενα που τους ενδιαφέρουν, απλοποιώντας και επιταχύνοντας τις μελέτες τους.

Ομως για τους σχεδιαστές των δικτύων, το δίκτυο έρχεται σε πρώτη προτεραιότητα. Για αυτούς η βελτίωση της απόδοσης του δικτύου είναι πολύ σημαντική υπόθεση. Η τοπολογία του δικτύου, οι αρχιτεκτονικές των μεταγωγέων, οι τεχνικές δρομολόγησης (routing), ο έλεγχος ροής και ο φόρτος της επικοινωνίας αποτελούν τους κυριότερους παράγοντες που καθορίζουν την απόδοση του δικτύου. Μέχρι πρόσφατα, η έρευνα για τη βελτίωση της καθυστέρησης του δικτύου και της απόδοσης του είχε επικεντρωθεί στους τέσσερεις πρώτους παράγοντες.

Αποτελέσματα ερευνών [1] δείχνουν ότι δίκτυα όπως k-ary n-cubes και meshes, τα οποία χρησιμοποιούν wormhole switching και virtual channel flow control επιτυγχάνουν τη ζητούμενη απόδοση. Επιπλέον, adaptivity στη δρομολόγηση των μηνυμάτων επιφέρει επιπρόσθετες βελτιώσεις στην απόδοση του δικτύου [2, 3].

Πολλές από τις ιδέες αυτές υλοποιήθηκαν σε σύγχρονα εμπορικά συστήματα. Για παράδειγμα ο Cray T3E router, ο Intel Cavallino router και ο SGI router χρησιμοποιούν wormhole switching και έλεγχο ροής με virtual channels.

Κύριο χαρακτηριστικό των ερευνών που έχουν γίνει, είναι ότι οι περισσότερες παράγουν την κίνηση που κυκλοφορεί μέσα στο δίκτυο με τεχνητούς τρόπους χρησιμοποιώντας μαθηματικά μοντέλα. Στην πραγματικότητα, η κίνηση με την οποία τροφοδοτείται το δίκτυο κατά την εκτέλεση πραγματικών παράλληλων εφαρμογών έχει τα δικά της χαρακτηριστικά γνωρίσματα και διαφοροποιείται σε αρκετά σημεία από την κίνηση που παράγεται με τεχνητούς τρόπους. Μερικές από τις διαφορές των δύο τύπων κίνησης εντοπίζονται στη συχνότητα παραγωγής των μηνυμάτων, στο μέγεθός τους καθώς και στην κατανομή των παραγόμενων μηνυμάτων στους διάφορους προορισμούς.

Συνήθως στις μελέτες που χρησιμοποιούν συνθετική κίνηση υποθέτουμε ότι η συχνότητα παραγωγής των μηνυμάτων ακολουθεί την εκθετική κατανομή ή την κατανομή Poisson, ότι τα μηνύματα κατανέμονται ομοιόμορφα στους προορισμούς και ότι το μέγεθος τους είναι σταθερό και ίσο με ορισμένο αριθμό bytes. Αυτές όμως οι υποθέσεις δεν είναι απαραίτητο ότι ισχύουν για την κίνηση που δημιουργείται από την εκτέλεση παράλληλων εφαρμογών.

Ομως, η πιο σημαντική διαφορά ανάμεσα στα μοντέλα κίνησης που προσδιορίζονται από πραγματικές εφαρμογές και σε αυτά που καθορίζονται από μαθηματικές συναρτήσεις είναι πως τα πρώτα αντικατοπτρίζουν την πραγματικότητα, αφού περιγράφουν την κίνηση που παρατηρείται στο δίκτυο διασύνδεσης κατά τη διάρκεια εκτέλεσης μιας εφαρμογής. Οντας ρεαλιστικά, οδηγούν στην εξαγωγή

συμπερασμάτων που είναι ακριβή.

Η μέθοδος του Ασύγχρονου Τρόπου Μετάδοσης (Asynchronous Transfer Mode, ATM), που είναι ιδιαίτερα δημοφιλής στα δίκτυα, σε συνδυασμό με έλεγχο ροής εισιτηρίων (credit based flow control) χρησιμοποιείται σε δίκτυα επεξεργαστών. Παρουσιάζει αρκετές ομοιότητες με την μέθοδο wormhole, που είναι και αυτή μια μέθοδος αρκετά διαδεδομένη για τη διασύνδεση παράλληλων επεξεργαστών.

Σε παλαιότερη εργασία [4], είχε μελετηθεί η απόδοση ενός δικτύου ATM, το οποίο χρησιμοποιούσε έλεγχο ροής με εισιτήρια και παρείχε λωρίδες κυκλοφορίας των δεδομένων. Η απόδοση των δικτύων ATM είχε συγκριθεί με την απόδοση δικτύων των οποίων οι μεταγωγείς χρησιμοποιούν την μέθοδο wormhole. Συμπέρασμα της εργασίας αυτής ήταν ότι το δίκτυο ATM συμπεριφέρεται καλύτερα από το wormhole, παρουσιάζοντας μικρότερες καθυστερήσεις στην κίνηση του. Ειδικά για τα δίκτυα ATM παρατηρήθηκε πως η χρήση λωρίδων κυκλοφορίας των δεδομένων επιφέρει σημαντική βελτίωση στην καθυστέρηση στην οποία υποβάλλεται η κίνηση που διέρχεται μέσα από το δίκτυο.

Κύριο χαρακτηριστικό της εργασίας αυτής ήταν πως η κίνηση που διοχετεύεται στο δίκτυο παράγεται με τεχνητούς τρόπους. Ειδικότερα αντικείμενο μελέτης αποτέλεσαν τα ακόλουθα μοντέλα κίνησης, τα οποία όμως δεν παρατηρούνται αναγκαστικά στις πραγματικές εφαρμογές:

- κίνηση Poisson, όπου ο προορισμός ενός πακέτου επιλέγεται τυχαία και ομοιόμορφα
- εκρηκτική κίνηση, όπου μεταβάλλεται το μέγεθος των πακέτων αλλά ο προορισμός επιλέγεται και πάλι ομοιόμορφα
- κίνηση με θερμά σημεία, όπου κάποιοι προορισμοί επιλέγονται πιο συχνά από άλλους

Στο πρώτο μέρος αυτής της εργασίας μελετήθηκε η κίνηση που παράγεται από 10 δημοφιλείς παράλληλες εφαρμογές. Η ανάλυση της κίνησης αυτής οδηγεί στην εξαγωγή συμπερασμάτων σχετικά με τα χαρακτηριστικά της και τους τρόπους επικοινωνίας των κόμβων του παράλληλου συστήματος, κατά τη διάρκεια της εκτέλεσης ενός προγράμματος. Στη συνέχεια η κίνηση που καταγράφηκε κατά την εκτέλεση των εφαρμογών χρησιμοποιείται για την μελέτη των δύο τύπων δικτύων (ATM και Worm-hole), με σκοπό όχι μόνο την επαλήθευση της χρησιμότητας των λωρίδων κυκλοφορίας

αλλά και τον γενικότερο έλεγχο της αξιοπιστίας μελετών που γίνονται με τεχνητά workloads.

Οι μετρήσεις μας έδειξαν πως η χρήση των λωρίδων επιφέρει σημαντικές βελτιώσεις στην καθυστέρηση του δικτύου. Οσο αφορά στον χρόνο εκτέλεσης όμως οι βελτιώσεις είναι αμελητέες, γεγονός το οποίο οφείλεται στις ιδιαιτερότητες της πραγματικής κίνησης και στον πολύπλοκο τρόπο με τον οποίο αλληλεπιδρούν οι πραγματικές εφαρμογές και οι παράλληλες αρχιτεκτονικές.

Στις επόμενες δύο παραγράφους περιγράφονται συνοπτικά οι τεχνολογίες ATM και Wormhole και ο έλεγχος ροής που εφαρμόζεται με χρήση εισιτηρίων και λωρίδων για κάθε μια από τις δύο μεθόδους. Η υπόλοιπη εργασία έχει την ακόλουθη δομή. Στο κεφάλαιο 2 περιγράφονται οι παράλληλες εφαρμογές που χρησιμοποιήθηκαν και παρουσιάζονται συμπεράσματα σχετικά με τα χαρακτηριστικά της κίνησής τους. Τα συμπεράσματα αυτά προέκυψαν ύστερα από την μελέτη των εφαρμογών. Στο κεφάλαιο 3 παρουσιάζονται μια σειρά από παράλληλα εργαλεία που μελετήθηκαν στα πλαίσια της εργασίας αυτής. Στο κεφάλαιο 4 περιγράφονται τα μοντέλα που υλοποιήθηκαν στον προσομοιωτή. Τέλος στο κεφάλαιο 5 παρουσιάζονται και ερμηνεύονται τα αποτελέσματα από τα πειράματα που έγιναν.

1.1 Ασύγχρονος Τρόπος Μεταφοράς (ATM)

Η τεχνολογία ATM χρησιμοποιείται σε πολλά δίκτυα τοπικά και ευρείας περιοχής για την μεταφορά δεδομένων σε υψηλές ταχύτητες. Με την μέθοδο ATM, μία πηγή η οποία θέλει να στείλει πληροφορία σε κάποιον προορισμό, διαιρεί την πληροφορία αυτήν σε κομμάτια που ονομάζονται πακέτα. Τα πακέτα διαιρούνται με την σειρά τους σε μικρότερα κομμάτια, τα λεγόμενα κελλιά ή κύτταρα (cells). Τα cells είναι σταθερού μεγέθους (53 bytes) και μεταφέρουν εκτός από μέρος των δεδομένων και πληροφορίες απαραίτητες για τη δρομολόγησή τους μέσα στο δίκτυο.

Πριν ξεκινήσει η μετάδοση των δεδομένων, ο διαχειριστής του δικτύου συνδέει την πηγή με τον προορισμό με ένα μονοπάτι, το οποίο θα χρησιμοποιηθεί καθόλη τη διάρκεια μετάδοσης των δεδομένων (εικονική σύνδεση). Cells του ιδίου VC χρησιμοποιούν την ίδια εικονική και φυσική σύνδεση για την μετάδοσή τους και φτάνουν στον προορισμό τους με την ίδια σειρά με την οποία μεταδόθηκαν στο δίκτυο. Ο διαχειριστής του δικτύου

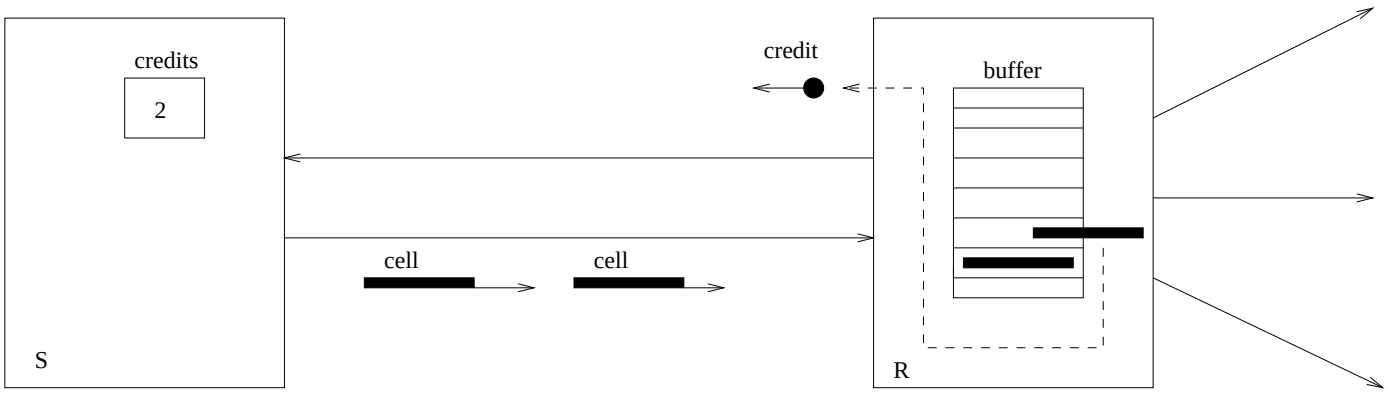
ενημερώνεται κατά το τέλος της μετάδοσης και ”απενεργοποιεί” την εικονική σύνδεση.

Η μέθοδος ATM κάνει δυνατή την πολύπλεξη στους συνδέσμους του δικτύου cells διαφορετικών πακέτων.

Προσθέτοντας έλεγχο ροής με εισιτήρια, τα δίκτυα ATM γίνονται κατάλληλα για την μεταφορά δεδομένων σε μικρές αποστάσεις και άρα μπορούν να χρησιμοποιηθούν σε δίκτυα επεξεργαστών.

1.1.1 Ελεγχος ροής με εισιτήρια

Οταν εφαρμόζεται έλεγχος ροής με εισιτήρια σε δίκτυα ATM, δεδομένα μεταδίδονται από ένα μεταγωγέα S σε ένα μεταγωγέα R μόνο εαν ο S ξέρει πως στον R υπάρχει χώρος διαθέσιμος για την παραλαβή τους. Αυτό υλοποιείται, όπως φαίνεται στο σχήμα 1.1:



Σχήμα 1.1: Ελεγχος ροής εισιτηρίων

Ο μεταγωγέας S έχει ένα μετρητή, ο οποίος δείχνει πόσος είναι ο ελεύθερος χώρος αποθήκευσης στον μεταγωγέα R. Κάθε φορά που δεδομένα μεταδίδονται από τον S στον R, ο μετρητής αυτός μειώνεται. Κάθε φορά που δεδομένα μεταδίδονται από τον R στον επόμενο στη σειρά μεταγωγέα, ελευθερώνεται χώρος αποθήκευσης στον R. Ο μεταγωγέας S ενημερώνεται για αυτήν την αποδέσμευση του χώρου αποθήκευσης του R μέσω ενός εισιτηρίου που αποστέλλει ο R σε αυτόν και αυξάνει τον αντίστοιχο μετρητή του.

Για να μην χάνεται εύρος ζώνης (bandwidth) θα πρέπει ο χώρος αποθήκευσης στον

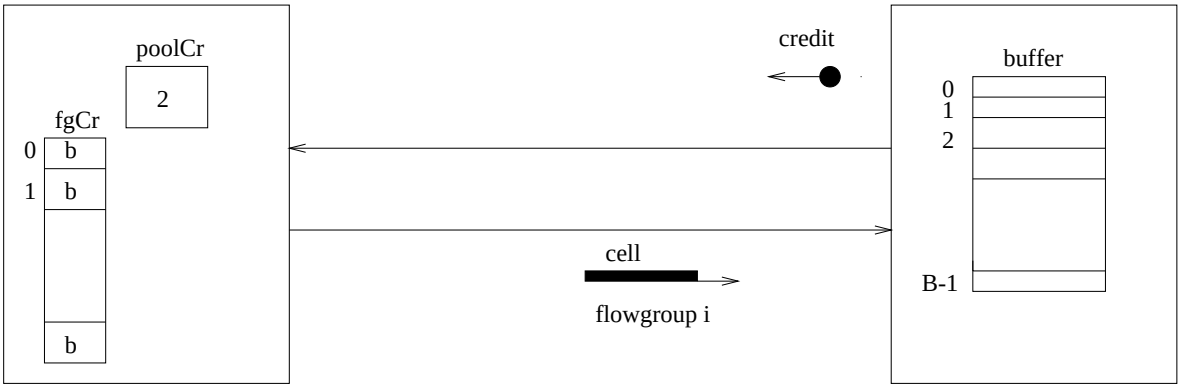
μεταγωγέα R να είναι τουλάχιστον ίσος με τη διέλευση (throughput) του δικτύου επί τον χρόνο επαναφοράς των δεδομένων και των εισιτηρίων. Ως χρόνος επαναφοράς (Round Trip Time, RTT) ορίζεται ο ελάχιστος χρόνος που μεσολαβεί μεταξύ της αναχώρησης ενός cell από τον μεταγωγέα S προς τον μεταγωγέα R, μέχρι τη στιγμή που η μετάδοση ενός νέου πακέτου θα ξεκινήσει από τον μεταγωγέα S χρησιμοποιώντας το εισιτήριο το οποίο δημιουργήθηκε από την αναχώρηση του προηγούμενου cell από τον μεταγωγέα R.

Ο έλεγχος ροής με εισιτήρια που απεικονίζεται στο σχήμα 1.1, διαμοιράζει τον χώρο αποθήκευσης ενός μεταγωγέα εξίσου σε όλα τα δεδομένα, ανεξάρτητα από τον προορισμό προς τον οποίο κατευθύνονται αυτά. Αυτό όμως μπορεί να επιφέρει αρνητικές συνέπειες όταν η υπάρχουσα κίνηση δεν είναι ομοιόμορφη. Σε αυτήν την περίπτωση, οι ενταμιευτές του μεταγωγέα γεμίζουν με δεδομένα, τα οποία στην πλειοψηφία τους ενδέχεται να κατευθύνονται προς τον ίδιο προορισμό. Εαν η κίνηση προς τον προορισμό αυτόν είναι αργή, εξαιτίας αυξημένης κυκλοφορίας, τα δεδομένα δεν μπορούν να μεταδοθούν και παραμένουν στον μεταγωγέα, εμποδίζοντας ταυτόχρονα τη διέλευση δεδομένων που κατευθύνονται προς διαφορετικές περιοχές, που χαρακτηρίζονται από μειωμένη κίνηση και τα οποία θα μπορούσαν να μεταδοθούν.

Αυτό το φαινόμενο είναι ανάλογο με το head of line blocking που παρατηρείται στο input queueing και έχει ως αποτέλεσμα την αύξηση της καθυστέρησης του δικτύου και την μείωση της χρήσης των συνδέσμων εξόδου του μεταγωγέα. Είναι ανάλογο με το φαινόμενο που παρατηρείται στην καθημερινή ζωή σε δρόμους με μία λωρίδα κυκλοφορίας. Ενα αμάξι, το οποίο έχει μπλοκαριστεί θέλοντας να στρίψει δεξιά καθυστερεί τα αμάξια που βρίσκονται πίσω του και τα οποία μπορεί να πηγαίνουν σε διαφορετικούς δρόμους.

Θα θέλαμε δεδομένα τα οποία κινούνται προς περιοχές με μεγάλη κυκλοφορία, να μην καθυστερούν δεδομένα τα οποία κατευθύνονται προς προορισμούς με λιγότερη κίνηση και τα οποία θα μπορούσαν να μεταδοθούν. Κάτι τέτοιο μπορεί να γίνει εαν ο παραπάνω περιγραφόμενος έλεγχος ροής συνδυαστεί με την χρήση λωρίδων 1.2.

Η προσθήκη λωρίδων στον υπάρχοντα έλεγχο ροής με εισιτήρια, βασίζεται στην παρατήρηση ότι δεν έχει νόημα οι θέσεις μνήμης ενός μεταγωγέα να καταλαμβάνονται όλες από δεδομένα, τα οποία κατευθύνονται προς τον ίδιο προορισμό, δηλαδή από μία ροή κυκλοφορίας. Κάτι τέτοιο δεν πρόκειται να αυξήσει τον ρυθμό με τον οποίο η κίνηση φεύγει από το δίκτυο, αφού αυτό εξαρτάται από την ταχύτητα με την οποία οι προορισμοί μπορούν να βγάλουν κίνηση από το δίκτυο. Αρα η συσσώρευση μιας ροής



Σχήμα 1.2:

κυκλοφορίας σε ένα μεταγωγέα, δεν επιφέρει οφέλη και επιπλέον μπορεί να προκαλέσει δυσάρεστες συνέπειες, όπως αυτές που περιγράφονται παραπάνω.

Είναι λοιπόν πιο αποδοτικό ο χώρος αποθήκευσης ενός μεταγωγέα να διαμοιράζεται ανάμεσα σε διαφορετικές ροές κυκλοφορίας, κάτι το οποίο επιτυγχάνεται με την ύπαρξη των δύο ακολούθων εισιτηρίων:

1. Εισιτήριο αναχώρησης

   Πρόκειται για το εισιτήριο εκείνο που εξασφαλίζει την ύπαρξη χώρου στον επόμενο στην σειρά μεταγωγέα για την αποθήκευση του cell. Σε κάθε μεταγωγέα υπάρχει ένας μετρητής για κάθε σύνδεσμο εξόδου. Η αναχώρηση δεδομένων από τον μεταγωγέα συνοδεύεται πάντοτε από την κατανάλωση ενός εισιτηρίου αναχώρησης, η οποία υλοποιείται με μείωση του μετρητή, που αντιστοιχεί στον σύνδεσμο εξόδου που χρησιμοποιήθηκε .

   Αρα δεδομένα μπορούν να φύγουν εφόσον υπάρχει εισιτήριο αναχώρησης δηλαδή ο αντίστοιχος μετρητής είναι μεγαλύτερος από μηδέν, διαφορετικά τα δεδομένα παραμένουν στους χώρους αποθήκευσης του μεταγωγέα περιμένοντας την άφιξη ενός εισιτηρίου.

2. Εισιτήριο προορισμού

   Σε κάθε μεταγωγέα για κάθε σύνδεσμο εξόδου υπάρχει ένας πίνακας με θέσεις τόσες όσες και οι προορισμοί του δικτύου. Σε κάθε θέση του πίνακα υπάρχει και ένας μετρητής που εκφράζει τα ελεύθερα εισιτήρια για τον δεδομένο προορισμό.

Ενα cell δεν μπορεί να φύγει προς τον επόμενο μεταγωγέα, εφόσον δεν υπάρχει το αντίστοιχο εισιτήριο προορισμού. Κάθε φορά που δεδομένα μεταδίδονται από τον μεταγωγέα προς δεδομένο προορισμό, ο αντίστοιχος μετρητής μειώνεται.

Ετσι λοιπόν, ένα cell που κατευθύνεται προς τον προορισμό i, δηλαδή ανήκει στην ροή κυκλοφορίας i, μεταδίδεται εφόσον υπάρχουν εισιτήριο αναχώρησης ($poolCr > 0$) και εισιτήριο προορισμού ($fgCr[i] > 0$). Οταν το cell αυτό θα φύγει από τον επόμενο στη σειρά μεταγωγέα, ένα εισιτήριο θα φτάσει στον αρχικό μεταγωγέα, το οποίο θα περιλαμβάνει τον έλεγχο ροής i στον οποίο ανήκε το cell, αλλά και τον αριθμό του συνδέσμου εξόδου που χρησιμοποιήθηκε για την μετάδοσή του. Οι αντίστοιχοι μετρητές των εισιτηρίων αναχώρησης και προορισμού θα αυξηθούν.

Εστω b η τιμή στην οποία αρχικοποιούμε τους μετρητές των εισιτηρίων προορισμού και B το μέγεθος του ενταμιευτή που αφιερώνεται σε κάθε σύνδεσμο εισόδου. Καθώς σε κάθε μεταγωγέα μπορούν να βρίσκονται ταυτόχρονα το πολύ b cells, τα οποία κατευθύνονται προς τον ίδιο προορισμό, μπορούμε να έχουμε για γεμάτο buffer το λιγότερο $B/b = L$ διαφορετικές ροές κυκλοφορίας σε έναν μεταγωγέα ταυτόχρονα. Σε έναν μεταγωγέα μπορούμε να έχουμε ταυτόχρονα περισσότερες από L ροές κίνησης, εφόσον κάθε μία από αυτές καταλαμβάνει στους ενταμιευτές του μεταγωγέα λιγότερες από b θέσεις.

Εστω ένας μεταγωγέας ο οποίος έχει γεμίσει από ροές κίνησης που κατευθύνονται προς προορισμούς με αυξημένη κυκλοφορία. Οι ροές αυτές παραμένουν μπλοκαρισμένες μέσα στον μεταγωγέα. Η μετάδοση cells που ανήκουν σε διαφορετικές ροές κίνησης δεν εμποδίζεται, χάρη στην ύπαρξη των εισιτηρίων προορισμού που δεν επιτρέπουν στα cells που κατευθύνονται προς τους προορισμούς που χαρακτηρίζονται από συμφόρηση, να καταλάβουν όλες τις θέσεις μνήμης του μεταγωγέα.

Ομως είναι δυνατό η μετάδοση των cells που κατευθύνονται προς περιοχές χαμηλής κυκλοφορίας, να μην γίνεται με τη μέγιστη δυνατή ταχύτητα. Αυτό συμβαίνει όταν ο χώρος, τον οποίο καταλαμβάνουν στον μεταγωγέα τα cells που οδεύουν σε περιοχές με μειωμένη κυκλοφορία είναι μικρότερος από το γινόμενο του throughput τους επί τον χρόνο επαναφοράς. Σε αυτήν την περίπτωση παρατηρείται υποχρησιμοποίηση του συνδέσμου εξόδου (underutilization of outgoing link).

1.2 Wormhole routing

Η μέθοδος wormhole αποτελεί μια από τις πιο δημοφιλείς τεχνικές διασύνδεσης παράλληλων επεξεργαστών. Η μέθοδος wormhole παρουσιάζει αρκετές αναλογίες με την μέθοδο ATM. Οπως προαναφέρθηκε, σε ένα δίκτυο ATM τα πακέτα πριν μπουν στο δίκτυο διαιρούνται σε μικρότερα κομμάτια σταθερού μεγέθους, τα λεγόμενα κελιά (cells). Τα κελιά μεταδίδονται πάνω από virtual circuits (VC) που είναι σταθεροί δρόμοι μέσα στο δίκτυο. Cells του ιδίου πακέτου έχουν όλα το ίδιο VC και μεταδίδονται με σειρά μέσα στο δίκτυο.

Στο wormhole δίκτυο, κατά αναλογία με το ATM τα πακέτα διαιρούνται σε μικρότερα κομμάτια σταθερού μεγέθους, που ονομάζονται flits. Flits που ανήκουν στο ίδιο πακέτο δρομολογούνται σε σειρά μέσα στο δίκτυο, ακολουθώντας τον ίδιο δρόμο από τον παραγωγό τους προς τον προορισμό. Τα wormhole πακέτα αποτελούνται από flits τριών ειδών: από την επικεφαλίδα, στην οποία περιέχεται ο προορισμός του πακέτου και η οποία αναλαμβάνει να δημιουργήσει το δρόμο μέσα στο δίκτυο τον οποίο θα ακολουθήσουν τα υπόλοιπα flits του ιδίου πακέτου, από το σώμα, το οποίο εκτός από ορισμένες απαραίτητες για τη δρομολόγηση του flit πληροφορίες περιλαμβάνει τα δεδομένα του πακέτου και τέλος από την ουρά, της οποίας ρόλος είναι να αποσυνθέσει τη σύνδεση που δημιουργήθηκε για την μετάδοση του πακέτου.

Ενα μόνο πακέτο είναι αρκετό για να γεμίσει ο χώρος αποθήκευσης που είναι αφιερωμένος σε ένα σύνδεσμο ενός μεταγωγέα. Σε περίπτωση που το πακέτο αυτό μπλοκαριστεί, πακέτα τα οποία θέλουν να περάσουν από τον ίδιο σύνδεσμο και τα οποία κατευθύνονται προς διαφορετικούς προορισμούς μπλοκάρονται. Η κατάσταση αυτή επηρεάζει αρνητικά την απόδοση του δικτύου. Για να βετιωθούν οι επιδόσεις του δικτύου, η μνήμη του κάθε μεταγωγέα οργανώνεται με τρόπο τέτοιο ώστε σε κάθε πακέτο να αφιερώνεται μονάχα ένα μέρος της. Δηλαδή, η μνήμη του μεταγωγέα διαιρείται σε τμήματα, που ονομάζονται λωρίδες (lanes) επιτρέποντας με τον τρόπο αυτό την ταυτόχρονη ύπαρξη στον ίδιο μεταγωγέα περισσοτέρων του ενός πακέτων. Σε κάθε μεταγωγέα μπορούν να υπάρχουν ταυτόχρονα τόσα διαφορετικά πακέτα όσες είναι και οι λωρίδες του μεταγωγέα, αφού κάθε λωρίδα αναλογεί σε ακριβώς ένα πακέτο.

Τα δίκτυα wormhole χρησιμοποιούν τις λωρίδες με αρκετά διαφορετικό τρόπο από ότι τα δίκτυα ATM. Τα δίκτυα wormhole αφιερώνουν κάθε λωρίδα σε ένα πακέτο, το οποίο και την χρησιμοποιεί αποκλειστικά καθόλη τη διάρκεια της μετάδοσής του.

Αντίθετα στα δίκτυα ATM μια λωρίδα μπορεί να χρησιμοποιηθεί από cells που ανήκουν σε οποιοδήποτε πακέτο, εφόσον βέβαια η λωρίδα έχει προηγουμένως ελευθερωθεί από το cell που την χρησιμοποιούσε προηγουμένως.

Τα δίκτυα ATM κάνουν πιο αποδοτική χρήση των λωρίδων από ότι τα δίκτυα wormhole. Η δυνατότητα χρήσης της ίδιας λωρίδας από cells διαφορετικών πακέτων λειτουργεί πιο αποδοτικά για δίκτυα με λίγες λωρίδες. Αυτό οφείλεται στο γεγονός ότι στα δίκτυα wormhole είναι δυνατόν ένας ή περισσότεροι σύνδεσμοι εξόδου να μείνουν ανενεργοί επειδή ένα πακέτο που κατέχει τις λωρίδες σε μια σειρά μεταγωγέων έχει μπλοκαριστεί. Αυτή η εικόνα όμως δεν παρατηρείται ποτέ στα δίκτυα ATM.

Επίσης για δίκτυα με πιο πολλές λωρίδες, η μνήμη ενός μεταγωγέα wormhole είναι δυνατόν να γεμίσει από πακέτα τα οποία κατευθύνονται προς τον ίδιο προορισμό, γεγονός το οποίο δεν επιφέρει καμμία βελτίωση στην απόδοση του δικτύου, αφού εαν ένα από αυτά τα πακέτα μπλοκαριστεί θα μπλοκάρουν προφανώς και τα υπόλοιπα. Επιπλέον πακέτα τα οποία θα κατευθύνονται σε περιοχές με λιγότερη κίνηση δεν μπορούν να χρησιμοποιήσουν τις λωρίδες.

Κεφάλαιο 2

Παράλληλες εφαρμογές

Οπως τονίστηκε και στο προηγούμενο κεφάλαιο, η χρήση παράλληλων εφαρμογών συμβάλλει σημαντικά στην αξιολόγηση της απόδοσης των παράλληλων αρχιτεκτονικών αλλά και των δικτύων διασύνδεσης ειδικότερα. Χωρίς τη σωστή και σε βάθος κατανόηση των απαιτήσεων σε επικοινωνία της εφαρμογής δεν είναι εύκολη η πλήρης κατανόηση της παρατηρούμενης απόδοσης των παράλληλων μηχανών. Συνεπώς ο χαρακτηρισμός των απαιτήσεων σε επικοινωνία των εφαρμογών είναι απαραίτητος, προκειμένου να κατανοηθεί ο τρόπος αλληλεπίδρασης τους με τις παράλληλες αρχιτεκτονικές, να μεγιστοποιηθεί η απόδοση των υπαρχουσών αρχιτεκτονικών και να σχεδιαστούν καλύτερες αρχιτεκτονικές στο μέλλον.

Η ανάπτυξη τεχνικών και τρόπων για την καταγραφή των ιδιοτήτων σε επικοινωνία των παράλληλων εφαρμογών είναι μια εργασία που δεν έχει εξεταστεί αναλυτικά στο παρελθόν. Σε παλαιότερη εργασία [5] μια σειρά από εφαρμογές που εκτελέστηκαν στον Intel Touchstone Delta είχαν χρησιμοποιηθεί προκειμένου να αξιολογήσουν τις απαιτήσεις των εφαρμογών αυτών σε υπολογιστική ισχύ, μνήμη, επικοινωνία και I/O. Στην εργασία αυτή η περιγραφή των απαιτήσεων των εφαρμογών αυτών σε επικοινωνία περιορίζεται στην καταγραφή του πλήθους των μηνυμάτων που ανταλλάσσονται μεταξύ των επεξεργαστών και του μεγέθους των μηνυμάτων αυτών.

Μια πιο αναλυτική προσπάθεια περιγραφής και αξιολόγησης των απαιτήσεων εφαρμογών σε επικοινωνία έχει γίνει από τους Chodnekar, Srinivasan και Vaidya [6]. Στην εργασία αυτή αντικείμενο μελέτης είναι 5 εφαρμογές που εκτελούνται σε

μηχανές κοινόχρηστης μνήμης (shared memory) και 2 εφαρμογές που εκτελούνται σε αρχιτεκτονικές ανταλλαγής μηνυμάτων (message - passing architectures). Η μελέτη επικεντρώνεται στην περιγραφή με μαθηματικές συναρτήσεις των χαρακτηριστικών της κίνησης που παρατηρείται κατά την εκτέλεση των εφαρμογών αυτών.

Η ανάλυση των απαιτήσεων σε επικοινωνία μιας εφαρμογής βρίσκει χρησιμότητα σε πολλές διαφορετικές μελέτες: ένας σχεδιαστής αρχιτεκτονικών μπορεί να χρησιμοποιήσει την πληροφορία αυτή για τη σχεδίαση πιο αποδοτικών μηχανών, στον τομέα της ανάπτυξης αλγορίθμων μπορεί να συμβάλλει στη σχεδίαση πιο αποδοτικών αλγορίθμων αλλά και στην πιο ολοκληρωμένη και σε βάθος ανάλυση τους, ενώ τέλος μπορεί να βοηθήσει έναν αναλυτή συστημάτων να διαμορφώσει πιό ακριβή μοντέλα.

Τα **NAS Parallel Benchmarks** [8] είναι ένα σύνολο από οκτώ εφαρμογές, οι οποίες έχουν γραφεί με σκοπό να συμβάλλουν στην προσπάθεια μέτρησης, με αντικειμενικό τρόπο της απόδοσης, των παράλληλων υπολογιστών και σύγκρισής τους με τους παραδοσιακούς supercomputers. Τα προγράμματα σχετίζονται με σημαντικές κατηγορίες εφαρμογών αεροφυσικής και βασίζονται σε κώδικες για τον υπολογισμό της δυναμικής ρευστών. Η συμβολή τους στην μέτρηση της απόδοσης των παράλληλων υπολογιστών τα έχει κάνει πλέον ευρέως αποδεκτά και τα έχει καθιερώσει στην κατηγορία των πιο δημοφιλών εφαρμογών μέτρησης της συμπεριφοράς των παράλληλων συστημάτων.

Το NAS Benchmark [22] αποτελείται από πέντε καθαρά μαθηματικές εφαρμογές και τρείς ψευδο-εφαρμογές. Οι πέντε μαθηματικές εφαρμογές (οι EP (Embarrassingly parallel), IS (Integer Sort), CG (Conjugate gradient), MG (Multigrid solver) και FT (3D FFT)) αναπαράγουν τον υπολογισμό αριθμητικών μεθόδων που χρησιμοποιούνται σε εφαρμογές υπολογισμού της δυναμικής των ρευστών (Computational Fluid Dynamics). Οι άλλες τρεις (οι LU (SSOR solver), SP (scalar pentadiagonal system) και BT (block-tradiagonal)) περιλαμβάνουν εκτός από μαθηματικούς υπολογισμούς, την μετακίνηση των δεδομένων και το υπολογιστικό κομμάτι που βρίσκει κανείς σε πλήρεις CFD εφαρμογές. Τα NAS Benchmarks μαζί με άλλες δύο εφαρμογές (τις QR (QR decom-position) και PSTW (shallow water model)) οι οποίες περιλαμβάνονται στο PARKBENCH [9] αποτελούν το σύνολο των εφαρμογών που μελετήθηκαν.

Τα προγράμματα είναι γραμμένα σε **Fortran 77**, περιλαμβάνουν όμως μερικές επεκτάσεις, όπως ονόματα long μεταβλητών (long variable names) και δηλώσεις τύπου include. Για το κομμάτι της επικοινωνίας μεταξύ των κόμβων του συστήματος

χρησιμοποιείται το **MPI** (Message Passing Interface) [7], το οποίο έχει πλέον καθιερωθεί ως πρότυπο (standard).

Κατά τη διάρκεια της εκτέλεσης των εφαρμογών αυτών, η επικοινωνία η οποία δημιουργείται είναι πολυσύνθετη και συχνά αλλάζει μορφή κατά τη διάρκεια της εκτέλεσης, εμφανίζοντας διαφορετικά χαρακτηριστικά και ιδιότητες. Λόγω του γεγονότος ότι η κίνηση που εισέρχεται στο δίκτυο είναι πολυδιάστατη, πρέπει να αναλυθεί σε βάθος προκειμένου να γίνει κατανοητή.

Οι τρεις πιο χαρακτηριστικές συνιστώσες που περιγράφουν την κίνηση είναι η χρονική, η χωρική και η συνιστώσα του όγκου. Με την χρονική συνιστώσα εκφράζεται ο ρυθμός παραγωγής μηνυμάτων, η συχνότητα δηλαδή με την οποία οι επεξεργαστές τροφοδοτούν το δίκτυο με κίνηση κατά τη διάρκεια εκτέλεσης της εφαρμογής. Η χωρική συνιστώσα εκφράζει την κατανομή των μηνυμάτων στον χώρο δηλαδή στους κόμβους του συστήματος. Η παράμετρος αυτή περιγράφει τον τύπο της επικοινωνίας που παρατηρείται μεταξύ των κόμβων του συστήματος. Τέλος η συνιστώσα του όγκου αφορά τον αριθμό των μηνυμάτων που ανταλλάσσονται μεταξύ δύο συγκεκριμένων επεξεργαστών κατά τη διάρκεια εκτέλεσης της εφαρμογής αλλά και την κατανομή των μηνυμάτων αυτών με βάση το μέγεθος τους.

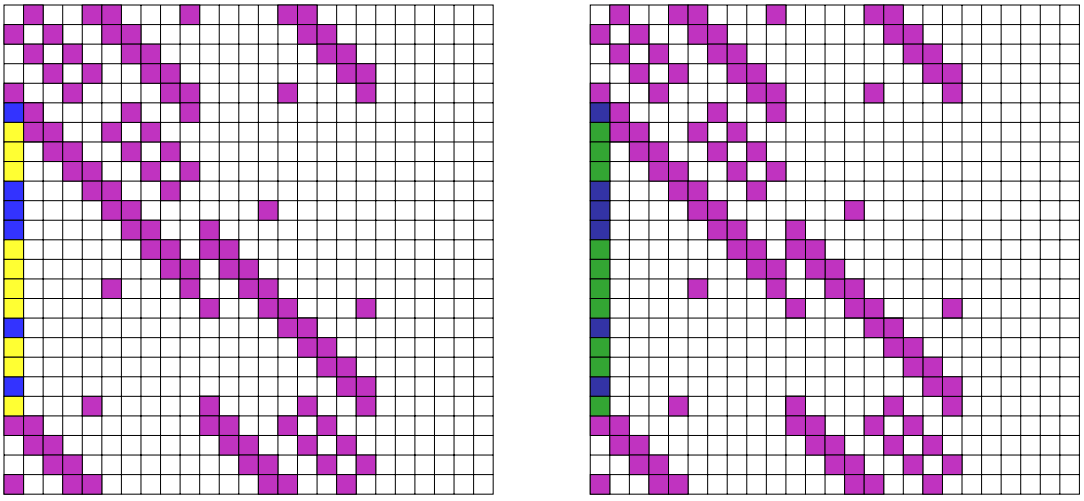
Κατά τη διάρκεια της εξέλιξης της εφαρμογής, συχνά παρατηρείται το φαινόμενο ένας προορισμός να δέχεται ταυτόχρονα μηνύματα από πολλούς διαφορετικούς αποστολείς (hot spot). Ο τρόπος με τον οποίο μεταβάλλεται στην εξέλιξη του χρόνου ο αριθμός των hot spots είναι μια ακόμη παράμετρος που παρουσιάζει ενδιαφέρον και αξίζει να μελετηθεί.

Τα αρχεία ιχνών (trace files), τα οποία δημιουργούνται με το τέλος της εκτέλεσης των εφαρμογών περιέχουν πληροφορία, η οποία μας βοηθάει να κατασκευάσουμε τις παραπάνω παραμέτρους. Ειδικότερα για κάθε αποστελλόμενο μήνυμα καταγράφεται ο κόμβος από τον οποίο προέρχεται το μήνυμα, ο κόμβος προς τον οποίο κατευθύνεται, η χρονική στιγμή της δημιουργίας του και τέλος το μέγεθός του σε bytes. Από τα στοιχεία αυτά μπορούμε να προσδιορίσουμε τις συνιστώσες της κίνησης, κατανοώντας τα χαρακτηριστικά της καθώς και τη συμπεριφορά της.

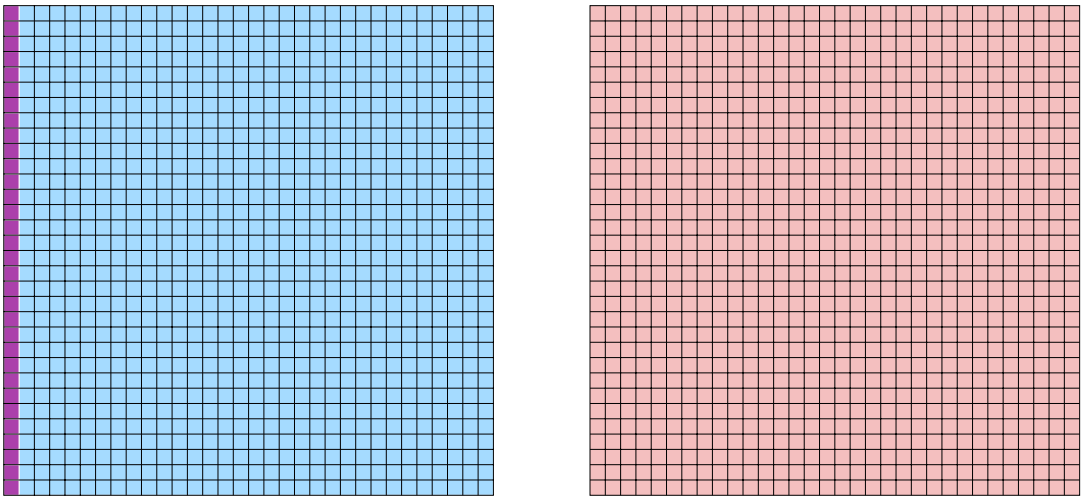
Οι συνιστώσες αυτές περιγράφουν αρκετά ικανοποιητικά την κίνηση που παρατηρείται στο δίκτυο και πρέπει πάντοτε να καθορίζονται όταν μελετάται η απόδοση των δικτύων διασύνδεσης - ανεξάρτητα από το αν η μελέτη αυτή γίνεται με προσομοίωση ή με αναλυτικό τρόπο. Για παράδειγμα, συχνά στις μελέτες που έχουν γίνει μέχρι τώρα

θεωρείται ότι η συχνότητα με την οποία οι παραγωγοί του δικτύου γεννούν μηνύματα ακολουθεί την εκθετική κατανομή ή την κατανομή Poisson. Επίσης συχνά θεωρείται ότι τα μηνύματα κατανέμονται ομοιόμορφα στους διάφορους προορισμούς του δικτύου και ότι έχουν σταθερό μέγεθος.

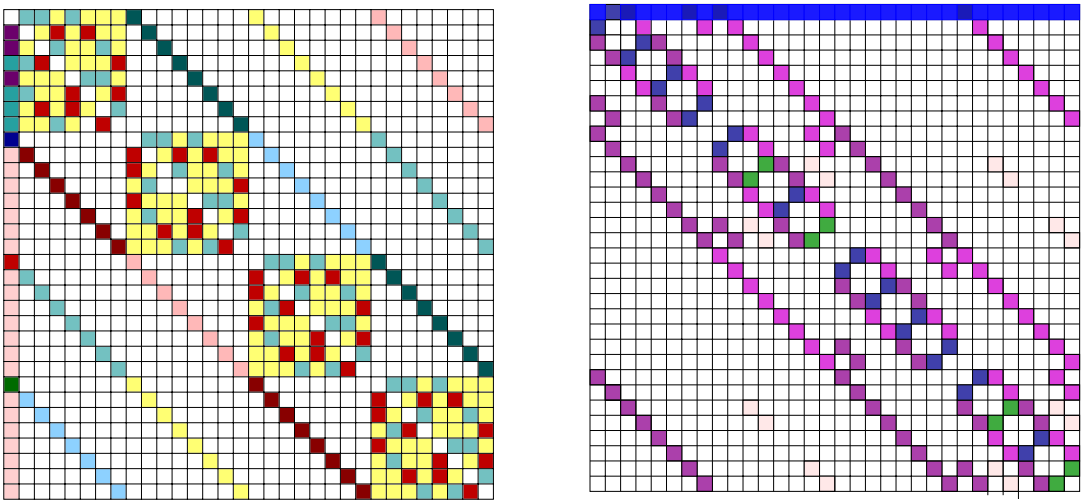
Στην ανάλυση της κίνησης που ακολουθεί στις επόμενες παραγράφους φαίνεται πως οι υποθέσεις που γίνονται για την τεχνητή κίνηση δεν ισχύουν για την πραγματική, αποδεικνύεται δηλαδή πως η πραγματική κίνηση διέπεται από χαρακτηριστικά εντελώς διαφορετικά από αυτά τα οποία προσδίδονται στην κίνηση που παράγεται με τεχνητούς τρόπους.



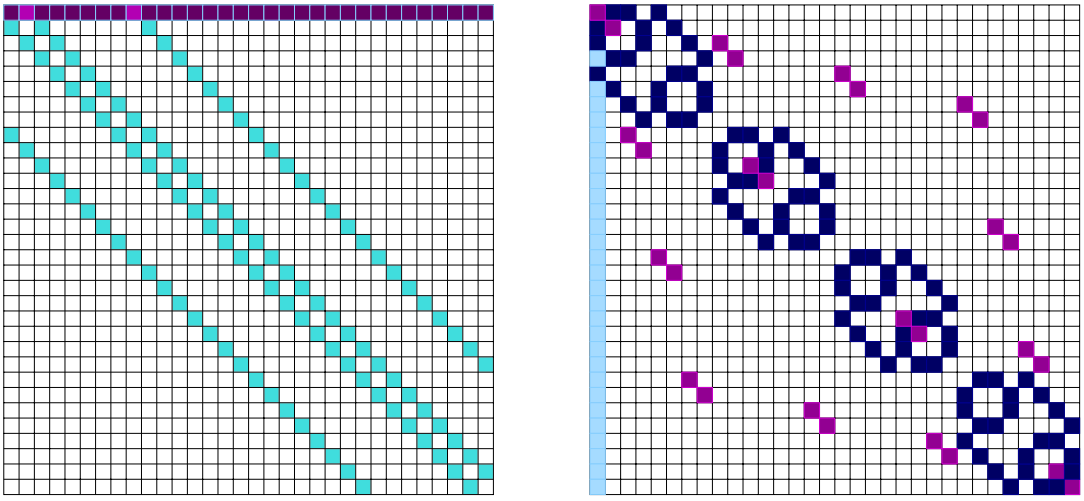
Σχήμα 2.1: Εφαρμογές BT και SP



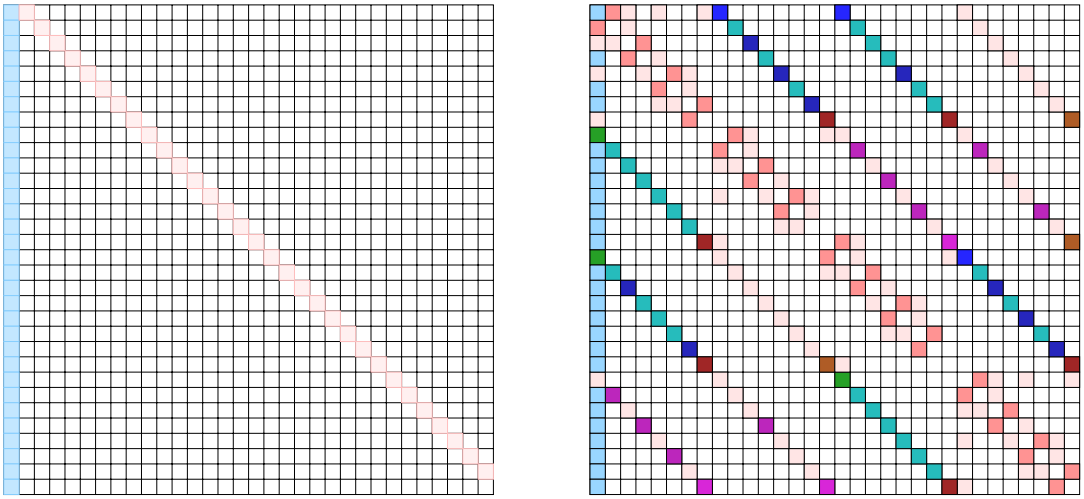
Σχήμα 2.2: Εφαρμογές FFT και EP



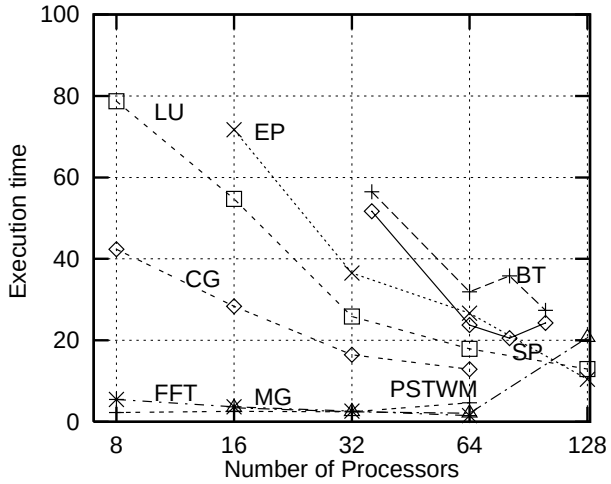
Σχήμα 2.3: Εφαρμογές PSTW και MG



Σχήμα 2.4: Εφαρμογές LU_solver και CG



Σχήμα 2.5: Εφαρμογές IS και QR



Σχήμα 2.6: Χρόνοι εκτέλεσης των εφαρμογών

Για κάθε εφαρμογή έγιναν διαδοχικές εκτελέσεις σε συνεχώς αυξανόμενο αριθμό επεξεργαστών. Για όλες τις εκτελέσεις μίας εφαρμογής χρησιμοποιήθηκε το ίδιο μέγεθος προβλήματος. Το πλήθος των κόμβων του συστήματος ξεκινούσε από οκτώ για να αυξηθεί διαδοχικά σε 16, 32, 64 μέχρι 128. Αύξηση στον αριθμό των επεξεργαστών που χρησιμοποιούνται για την εκτέλεση μιας εφαρμογής οδηγεί σε αύξηση του πλήθους των κόμβων του δικτύου και επομένως αναμένεται αύξηση της καθυστέρησης στην οποία υπόκεινται τα πακέτα μέσα στο δίκτυο. Επίσης είναι πιθανή η αύξηση του πλήθους των μηνυμάτων που αναλογούν σε κάθε επεξεργαστή, γεγονός το οποίο επιφέρει εντονότερη διαμάχη για τους πόρους του δικτύου. Παρόλα αυτά ο χρόνος που σπαταλάει κάθε επεξεργαστής για υπολογισμό (computation time) είναι πλέον μικρότερος.

Εκτελώντας την ίδια εφαρμογή σε διαδοχικά αυξανόμενο αριθμό επεξεργαστών, υπάρχει συνήθως ένα σημείο πέρα από το οποίο αύξηση του αριθμού των κόμβων του συστήματος επιφέρει αύξηση στον συνολικό χρόνο εκτέλεσης της εφαρμογής και όχι μείωση. Αυτό γίνεται φανερό από το σχήμα 2.6, όπου απεικονίζονται οι χρόνοι εκτέλεσης των εφαρμογών για τις εκτελέσεις που έγιναν για κάθε πλήθος επεξεργαστών. Πέρα από το σημείο αυτό η εκτέλεση της εφαρμογής μπορεί να χαρακτηριστεί μη κλιμακούμενη (unscalable) και δεν έχει ενδιαφέρον να μελετηθεί η συμπεριφορά ούτε της εφαρμογής ούτε του δικτύου.

Μελετώντας προσεκτικά τις εφαρμογές, παρατηρήθηκε πως οι κόμβοι του συστήματος επικοινωνούν μεταξύ τους με συγκεκριμένους τρόπους. Ξεχωρίζουν οι ακόλουθοι τύποι επικοινωνίας:

- Ενας προς όλους (Broadcast pattern)

  Ενας από τους επεξεργαστές του συστήματος στέλνει πληροφορία την οποία όλοι οι υπόλοιποι περιμένουν να λάβουν. Δηλαδή, ένας κόμβος του συστήματος επικοινωνεί με όλους τους υπόλοιπους. Αυτός ο τρόπος επικοινωνίας παρουσιάζεται συνήθως στην αρχή της εκτέλεσης της εφαρμογής. Συχνά όμως παρατηρείται σε κατοπινές φάσεις της εκτέλεσης του προγράμματος υποδηλώνοντας την έναρξη μιας καινούργιας φάσης υπολογισμού.

  Σε αυτόν τον τύπο επικοινωνίας ο αποστέλλων κόμβος στέλνει σε όλους τους υπόλοιπους ακριβώς τα ίδια σε μέγεθος αλλά και σε πλήθος μηνύματα. Δηλαδή επικοινωνεί εξίσου με όλους τους επεξεργαστές του συστήματος. Δεν είναι όμως απαραίτητο ότι ο επεξεργαστής που στέλνει τα μηνύματα στέλνει σε όλους ακριβώς το ίδιο σε περιεχόμενο μήνυμα. Για παράδειγμα, μπορεί να στέλνει σε κάθε κόμβο του συστήματος διαφορετικό σε περιεχόμενο μήνυμα καθορίζοντας τα όρια του πίνακα στα οποία ο ”παραλήπτης - επεξεργαστής” θα πρέπει να επικεντρώσει τους υπολογισμούς του.

- Ολοι προς έναν (Hot spot pattern)

  Πρόκειται για τον αντίθετο τρόπο επικοινωνίας από αυτόν που περιγράφεται από το μοντέλο επικοινωνίας ”ένας προς όλους”. Σε αυτόν τον τύπο επικοινωνίας όλοι οι επεξεργαστές του συστήματος στέλνουν ταυτόχρονα ένα μήνυμα προς τον ίδιο επεξεργαστή. Δηλαδή, μεταδίδεται ταυτόχρονα από όλους τους κόμβους του συστήματος πληροφορία, η οποία κατευθύνεται προς τον ίδιο προορισμό. Αυτός ο τύπος κυκλοφορίας σημειώνεται συνήθως στο τέλος της εκτέλεσης της παράλληλης εφαρμογής. Παρατηρείται όμως και σε ενδιάμεσα στάδια σηματοδοτώντας το τέλος κάποιων φάσεων επεξεργασίας του παράλληλου προγράμματος.

  Το μέγεθος των μηνυμάτων τα οποία σχετίζονται με αυτόν τον τύπο κίνησης είναι συνήθως μικρό, της τάξεως μερικών bytes, γεγονός το οποίο οφείλεται στο ότι αυτός ο τρόπος επικοινωνίας δεν σχετίζεται με την μεταφορά δεδομένων αλλά εκφράζει την ανάγκη των μελών του συστήματος να ενημερώσουν τον ”αρχηγό” είτε για την κατάστασή τους (τελείωσαν κάποια φάση υπολογισμού) είτε για το αποτέλεσμα κάποιου υπολογισμού που εκτέλεσαν.

Ενδιαφέρον παρουσιάζει το γεγονός ότι αυτή η φάση επικοινωνίας δεν χαρακτηρίζεται από μεγάλη διάρκεια. Ο κάθε κόμβος του συστήματος στέλνει προς τον κοινό προορισμό ακριβώς ένα μήνυμα και όχι περισσότερα. Ποτέ στο δείγμα των εφαρμογών που χρησιμοποιήθηκαν δεν παρατηρήθηκε το φαινόμενο κατά το στάδιο της "Όλοι προς έναν" επικοινωνίας οι επεξεργαστές να στέλνουν προς τον "αρχηγικό κόμβο" δύο ή περισσότερα μηνύματα. Σε όλες τις εφαρμογές που μελετήθηκαν τον ρόλο αυτό, του "αρχηγικού κόμβου" δηλαδή, κατείχε πάντοτε ο κόμβος 0. Τέλος μια ακόμη ενδιαφέρουσα παρατήρηση που έγινε κατά την μελέτη των εφαρμογών είναι ότι σε όλο το δείγμα των εφαρμογών μόνο ένας επεξεργαστής εμφανίζεται, ο οποίος να συγκεντρώνει σε τέτοιο μεγάλο βαθμό την προτίμηση των αποστολέων.

- Ολοι σε όλους (All to all pattern)

  Ολοι οι κόμβοι του συστήματος επικοινωνούν μεταξύ τους είτε για να ενημερώσουν ο ένας τον άλλον για κάποιο αποτέλεσμα είτε για να δώσουν ο ένας στον άλλο δεδομένα, που είναι απαραίτητα για την συνέχιση της εκτέλεσης της εφαρμογής. Το μέγεθος των μηνυμάτων που παρατηρούνται σε αυτήν την φάση επικοινωνίας είναι άλλες φορές μικρό, της τάξεως μερικών bytes (όταν η πληροφορία που διαδίδεται σε αυτήν την φάση είναι το αποτέλεσμα ενός υπολογισμού) και άλλοτε μεγάλο (αρκετά Kbytes, όταν το δίκτυο υπερφορτώνεται από δεδομένα).

  Ενδιαφέρον παρουσιάζει το γεγονός ότι αυτός ο τύπος της επικοινωνίας είναι που δίνει τη δυνατότητα σε κάθε επεξεργαστή να επικοινωνήσει με ορισμένους κόμβους του συστήματος. Εκτός από αυτά τα μηνύματα, δεν παρατηρείται άλλη ανταλλαγή μηνυμάτων μεταξύ αυτών των κόμβων καθόλη τη διάρκεια εκτέλεσης της εφαρμογής.

- Γείτονες (Neighbours)

  Πρόκειται για τον τύπο επικοινωνίας που εμφανίζεται πιο συχνά στις παράλληλες εφαρμογές. Μελετώντας το δείγμα των εφαρμογών έγινε φανερό πως οι κόμβοι του συστήματος χωρίζονται κατά τη διάρκεια εκτέλεσης του προγράμματος σε ομάδες. Ο κάθε επεξεργαστής επικοινωνεί αποκλειστικά με μερικούς ή με όλους τους κόμβους που ανήκουν στην ίδια με αυτόν ομάδα. Επικοινωνεί με τους υπόλοιπους μόνο σε φάσεις επικοινωνίας της κατηγορίας "όλοι προς όλους".

  Ο τρόπος με τον οποίο οι κόμβοι του συστήματος χωρίζονται σε ομάδες διαφοροποιείται ανάλογα με την εφαρμογή και εξαρτάται από το πλήθος των

κόμβων του συστήματος και από την φάση του υπολογισμού στην οποία βρισκόμαστε. Στη διάρκεια μιας εφαρμογής οι κόμβοι του συστήματος οργανώνονται σε διαφορετικές ομάδες, ανάλογα με το τρέχον στάδιο του υπολογισμού.

Μερικοί από τούς πιο χαρακτηριστικούς τύπους ομάδων που παρατηρήθηκαν είναι οι ακόλουθοι:

-- Ομάδα των τεσσάρων

Οι κόμβοι του συστήματος χωρίζονται σε ομάδες των τεσσάρων. Κάθε ομάδα υποδιαιρείται με τη σειρά της σε δύο υποομάδες των 2 μελών. Κάθε μέλος μιας υποομάδας επικοινωνεί αποκλειστικά με τα μέλη της άλλης υποομάδας, της ίδιας βέβαια ομάδας. Τέτοιος τύπος επικοινωνίας παρατηρείται στις εφαρμογές PSTWM, MG, CG.

-- Ομάδες με έναν προς έναν επικοινωνία

Οι επεξεργαστές χωρίζονται σε ομάδες των 8 μελών ή ακόμη μεγαλύτερες "γειτονιές" των 16 ή 32 επεξεργαστών. Και πάλι κάθε ομάδα χωρίζεται σε δύο υποομάδες. Οι δύο υποομάδες δεν είναι απαραίτητο να έχουν το ίδιο πλήθος μελών. Κάθε μέλος της μίας υποομάδας επικοινωνεί ακριβώς με έναν κόμβο της αντίστοιχης υποομάδας. Μπορεί δηλαδή να ορισθεί μια ενα προς ένα και επί συνάρτηση, η οποία να αντιστοιχεί σε κάθε αποστολέα τον παραλήπτη των μηνυμάτων του.

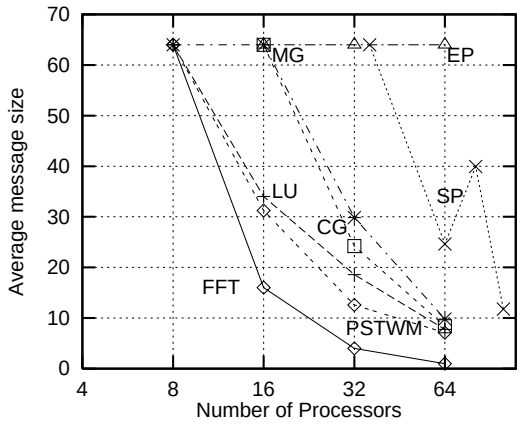
-- Ομάδες ευρείας επικοινωνίας

Στις εφαρμογές συχνά οι κόμβοι του συστήματος συνενώνονται σε μεγαλύτερες ομάδες, στις οποίες ο κάθε επεξεργαστής επικοινωνεί εξίσου με όλα τα υπόλοιπα μέλη της ομάδας. Δηλαδή στέλνει μηνύματα με την ίδια συχνότητα σε όλα τα υπόλοιπα μέλη της ομάδας αλλά και λαβαίνει μηνύματα εξίσου συχνά από όλους τους επεξεργαστές της "γειτονιάς".

-- Επικοινωνία μεταξύ εκπροσώπων ομάδων

Σε ακόλουθες φάσεις υπολογισμού οι κόμβοι μιας ομάδας επικοινωνούν με τους κόμβους μιας άλλης ομάδας μέσω των εκπροσώπων τους. Δηλαδή η κάθε ομάδα ορίζει έναν ή περισσότερους εκπροσώπους οι οποίοι αναλαμβάνουν να την "αντιπροσωπεύσουν" κατά τη διάρκεια της επικοινωνίας με κάποια άλλη ομάδα.

Στα σχήματα 2.1, 2.2, 2.3, 2.4 και 2.5 φαίνεται για κάθε εφαρμογή η κατανομή



Σχήμα 2.7: Μεταβολή του μέσου μεγέθους του μηνύματος

των μηνυμάτων στους διάφορους κόμβους του συστήματος. Στα σχήματα αυτά γίνονται εμφανείς οι τρόποι επικοινωνίας που περιγράφηκαν προηγουμένως. Παρατηρεί κανείς ότι η εφαρμογή LU_solver επικοινωνεί χωρίζοντας τους επεξεργαστές του συστήματος σε ομάδες, αλλά με κανέναν από τους τρόπους που αναφέρονται πιο πάνω. Αυτό οφείλεται στο γεγονός ότι ο τρόπος με τον οποίο χωρίζονται οι επεξεργαστές σε ομάδες δεν είναι ιδιαίτερος. Ο τύπος επικοινωνίας ”Γείτονες” δεν προσδιορίζεται τόσο εύκολα όπως οι υπόλοιποι τρόποι επικοινωνίας.

Εξετάζοντας το μέγεθος των μηνυμάτων τα οποία ανταλλάσονται μεταξύ των επεξεργαστών του συστήματος κατά τη διάρκεια της εκτέλεσης της εφαρμογής, μπορεί κανείς να τα ταξινομήσει στα μηνύματα τα οποία σχετίζονται με την μεταφορά δεδομένων μεταξύ των κόμβων του συστήματος, σε αυτά τα οποία μεταφέρουν το αποτέλεσμα ενός υπολογισμού και σε αυτά τα οποία μεταφέρουν πληροφορία για την κατάσταση του κόμβου (π.χ. ο κόμβος έφτασε στο τέλος κάποιας φάσης υπολογισμού). Τα μηνύματα τα οποία σχετίζονται με την μεταφορά δεδομένων είναι πάντοτε πολύ μεγάλα, πολύ μεγαλύτερα από τα μηνύματα που ανήκουν στις άλλες δύο κατηγορίες. Το μέγεθος τους όμως μειώνεται καθώς αυξάνεται το πλήθος των επεξεργαστών του συστήματος, αφού το ίδιο μέγεθος δεδομένων κατανέμεται πλέον σε περισσότερους επεξεργαστές. Τα μηνύματα τα οποία ανήκουν στις δύο άλλες κατηγορίες έχουν μικρότερο μέγεθος το οποίο είναι ανεξάρτητο του αριθμού των κόμβων, από τους οποίους αποτελείται το παράλληλο σύστημα.

2.2 Γενικά συμπεράσματα

Συνοψίζοντας την μελέτη του τύπου της επικοινωνίας που παρατηρείται στο δείγμα των εφαρμογών καταλήγει κανείς στα ακόλουθα συμπεράσματα:

- Υπάρχει μόνο ένα θερμό σημείο (hot spot)
  Ορίζουμε ως θερμό σημείο έναν κόμβο του συστήματος εαν αποτελεί τον προορισμό των μηνυμάτων που αποστέλλονται ταυτόχρονα από ένα μεταβλητό αλλά αναλογικά μεγάλο αριθμό κόμβων του συστήματος, ο οποίος καθορίζεται από τον χρήστη.

  Η μελέτη των εφαρμογών οδήγησε στο συμπέρασμα πως για πλήθος αποστολέων μεγαλύτερο ή ίσο από το ένα τέταρτο των κόμβων του συστήματος, δεν εμφανίζεται κατά τη διάρκεια της εκτέλεσης της εφαρμογής ποτέ περισσότερα από ένα θερμά σημεία.
  Η διαμάχη για τους πόρους του δικτύου που οδηγούν προς το θερμό σημείο γίνεται ολοένα και εντονότερη καθώς αυξάνεται το πλήθος των κόμβων του συστήματος.

- Δεν υπάρχει ομοιόμορφος τύπος επικοινωνίας
  Από την μελέτη των εφαρμογών γίνεται φανερό πως οι επεξεργαστές του συστήματος δεν επικοινωνούν με την ίδια συχνότητα κατά τη διάρκεια της εφαρμογής. Κάθε κόμβος του συστήματος επικοινωνεί κυρίως με ένα μικρό υποσύνολο των κόμβων του συστήματος, ενώ η επικοινωνία του με τους υπόλοιπους παρατηρείται μόνο σε φάση ”Όλοι προς όλους” επικοινωνίας.

- Δεν υπάρχει ενιαίο μέσο μέγεθος μηνύματος
  Τα μηνύματα τα οποία αποστέλλονται κατά τη διάρκεια της εφαρμογής δεν έχουν κάποια σταθερή τιμή ούτε κυμαίνονται μέσα σε κάποιο όριο. Για κάθε εφαρμογή υπάρχουν ορισμένες κατηγορίες μεγεθών μηνυμάτων στις οποίες ανήκουν τα μηνύματα του προγράμματος.

Υστερα από μελέτη των εφαρμογών γίνεται φανερό πως η κίνηση που παράγεται με τεχνητά μέσα (συνθετικά traces) διαφοροποιείται σε πάρα πολλά σημεία από την κίνηση που παράγεται από τις πραγματικές εφαρμογές. Αυτό οφείλεται στο ότι η τεχνητή κίνηση δημιουργείται έχοντας ως βάση κάποιες παραδοχές, οι οποίες δεν παρατηρούνται ποτέ στην πραγματικότητα. Αυτές οι λανθασμένες υποθέσεις στις

οποίες στηρίζεται η παραγωγή της τεχνητής κίνησης αποτελούν την αιτία για τις παρατηρούμενες διαφορές στις μετρήσεις των δικτύων ATM και Wormhole που έγιναν στο δεύτερο μέρος της εργασίας αυτής.

Ομως η κίνηση που δημιουργείται κατά την εκτέλεση των πραγματικών εφαρμογών υπακούει σε μαθηματικούς κανόνες. Χωρίζοντας την κάθε εφαρμογή σε στάδια και μελετώντας ξεχωριστά για την κάθε φάση, τον τρόπο που επικοινωνούν οι επεξεργαστές, τη συχνότητα επικοινωνίας και το μέγεθος των μηνυμάτων που ανταλλάσσονται είναι δυνατόν να ορισθούν μαθηματικές συναρτήσεις οι οποίες να περιγράφουν τη φάση του παράλληλου προγράμματος που μελετάται. Στο σύνολο της, η εφαρμογή περιγράφεται από τη συνάρτηση που προκύπτειαπό το άθροισμα των συναρτήσεων, που προσδιορίζουν τα επιμέρους στάδια υπολογισμού.

Εχοντας προσδιορίσει τις μαθηματικές συναρτήσεις που περιγράφουν την παράλληλη εφαρμογή είναι δυνατή η παραγωγή τεχνητής κίνησης, η οποία υπόκειται στους ίδιους νόμους με την πραγματική. Υπό αυτήν την προυπόθεση η ανάλυση βάσει τεχνητής κίνησης δεν είναι παραπλανητική αλλά μπορεί να θεωρηθεί εξίσου χρήσιμη για την μελέτη της συμπεριφοράς του δικτύου με την πραγματική.

Κεφάλαιο 3

Εργαλεία Ιχνηλασίας παράλληλων εφαρμογών

Οι παράλληλες μηχανές ανάλογα με τον τρόπο με τον οποίο επικοινωνούν μεταξύ τους χωρίζονται σε δύο βασικές κατηγορίες. Σε αυτές που επικοινωνούν με χρήση διαμοιραζόμενης μνήμης (shared memory) και σε αυτές στις οποίες η επικοινωνία γίνεται με ανταλλαγή μηνυμάτων (message passing).

Στις αρχιτεκτονικές ανταλλαγής μηνυμάτων κάθε κόμβος της μηχανής έχει τη δική του τοπική μνήμη, την οποία μπορεί εύκολα να προσπελάσει η κεντρική μονάδα επεξεργασίας του κόμβου και μόνο αυτή (CPU) . Η ανταλλαγή των δεδομένων μεταξύ των επεξεργαστών του συστήματος γίνεται με μηνύματα μέσω του δικτύου διασύνδεσης. Τα δεδομένα μορφοποιούνται σε πακέτα και αποστέλλονται από τον ένα επεξεργαστή στον άλλο. Ενα μήνυμα μπορεί να αποτελείται από ένα ή περισσότερα πακέτα, στα οποία εκτός από την αποστελλόμενη πληροφορία περιλαμβάνονται στοιχεία δρομολόγησης ή άλλα δεδομένα ελέγχου.

Από την άλλη μεριά, στις αρχιτεκτονικές κοινόχρηστης μνήμης είναι δυνατή η προσπέλαση του ίδιου τμήματος μνήμης από πολλούς επεξεργαστές. Σε αυτήν την κατηγορία των μηχανών η κυκλοφορία που παρατηρείται στο δίκτυο κατά την εκτέλεση μιας εφαρμογής οφείλεται όχι μόνο στην ίδια την εφαρμογή αλλά και στα πρωτόκολλα που χρησιμοποιούνται για την προσπέλαση και ενημέρωση της μνήμης (coherence traffic).

Γι'αυτό και η κίνηση που γεννάται κατά την εκτέλεση μιας εφαρμογής διαφοροποιείται, ανάλογα με την μηχανή που χρησιμοποιείται για την εκτέλεσή της. Επίσης, στις αρχιτεκτονικές κοινόχρηστης μνήμης το μέγεθος των μηνυμάτων που ανταλλάσσονται εξαρτάται από το μέγεθος του μπλοκ της κρυφής μνήμης, γεγονός το οποίο δεν ισχύει για τα συστήματα ανταλλαγής μηνυμάτων.

Οι παράλληλες αρχιτεκτονικές αποτελούν το ιδανικό περιβάλλον για την εκτέλεση πολλών επιστημονικών εφαρμογών, οι οποίες χαρακτηρίζονται από τις μεγάλες τους απαιτήσεις σε υπολογιστική ισχύ. Προσφέρουν μια οικονομική λύση εκμεταλλευόμενες το κόστος των μικροεπεξεργαστών, το οποίο παραμένει χαμηλό σε αντίθεση με το κόστος των supercomputers το οποίο συνεχώς αυξάνει. Για αυτό η μελέτη τους αποτέλεσε και συνεχίζει ακόμη να αποτελεί ένα από τα πιο δημοφιλή θέματα έρευνας.

Ομως η ανάπτυξη εφαρμογών που να εκμεταλλεύονται πλήρως τις δυνατότητες των παράλληλων μηχανών δεν είναι τόσο εύκολη όσο η ανάπτυξη σειριακών προγραμμάτων. Ο προγραμματιστής της παράλληλης εφαρμογής θα πρέπει να κατανοεί το μοντέλο επικοινωνίας που αναπτύσσεται κατά τη διάρκεια της εκτέλεσης της εφαρμογής μεταξύ των κόμβων του συστήματος, την κατανομή των εργασιών της εφαρμογής στους πόρους της παράλληλης μηχανής, τις ενέργειες που σχετίζονται με το I/O και ένα πλήθος άλλων παραμέτρων προκειμένου να σχεδιάσει σωστά και αποδοτικά προγράμματα.

Οι δυσκολίες κατά τη σχεδίαση και ανάπτυξη παράλληλων προγραμμάτων οδήγησαν τα τελευταία χρόνια στην ανάπτυξη και κατασκευή πολλών εργαλείων τα οποία σχετίζονται με θέματα όπως το λογισμικό του συστήματος, γλώσσες προγραμματισμού σε παράλληλα συστήματα, μεταγλωτιστές και αλγόριθμοι ανάπτυξης προγραμμάτων. Επίσης αναπτύχθηκαν μια σειρά από εργαλεία τα οποία μελετούν την απόδοση του συστήματος και των εφαρμογών. Στόχος όλων αυτών των εργαλείων είναι η καλύτερη κατανόηση του τρόπου με τον οποίο αλληλεπιδρούν οι παράλληλες μηχανές και τα παράλληλα προγράμματα γεγονός το οποίο θα οδηγήσει όχι μόνο στη σχεδίαση πιο αποδοτικών μηχανών αλλά καί στην ανάπτυξη εφαρμογών που θα εκμεταλλεύονται όσο δυνατόν περισσότερο τους υπάρχοντες πόρους.

Ανάμεσα στους τρόπους εξέτασης της απόδοσης των αρχιτεκτονικών διακρίνονται ως πιο βασικοί οι ακόλουθοι:

1. **Μελέτη με χρήση αναλυτικών μοντέλων**

   Οι ιδιαιτερότητες των παράλληλων αρχιτεκτονικών δυσχεραίνουν την εφαρμογή

αυτής της μεθόδου αφού την κάνουν αρκετά πολύπλοκη. Γι'αυτόν το λόγο η μέθοδος αυτή δεν χρησιμοποιείται.

2. **Μελέτη με εκτέλεση πειραμάτων πάνω σε πραγματικές μηχανές**
   Στην περίπτωση αυτή είναι απαραίτητο να έχει κατασκευαστεί η μηχανή η οποία θα αποτελέσει το αντικείμενο μελέτης. Αφού τελειώσει η κατασκευή της, η μηχανή χρησιμοποιείται για τη διεξαγωγή πειραμάτων. Τα αποτελέσματα της μεθόδου αυτής είναι ακριβή, αφού δείχνουν ακριβώς αυτό που θα συνέβαινε στην πραγματικότητα. Ομως το κόστος για την κατασκευή της μηχανής αυτής κάνει την παραπάνω μέθοδο ασύμφορη. Επιπλέον, όπως είναι φανερό, με την μέθοδο αυτή δεν υπάρχει η δυνατότητα ευελιξίας κατά τη διεξαγωγή των πειραμάτων, αφού η αλλαγή των χαρακτηριστικών της μηχανής είναι δαπανηρή και όχι πάντοτε εφικτή.

3. **Μελέτη με προσομοίωση**
   Η προσομοίωση αποτελεί τον πιο δημοφιλή τρόπο μελέτης των παράλληλων αρχιτεκτονικών. Το κόστος της σε σχέση με την μέθοδο της εκτέλεσης των πειραμάτων σε πραγματικές μηχανές είναι πολύ μικρό, αφού δεν προυποθέτει την κατασκευή της μηχανής. Ως μέθοδος χαρακτηρίζεται από ευελιξία, αφού εύκολα μπορεί κανείς να αλλάξει κάποιες παραμέτρους της αρχιτεκτονικής που μελετάται. Επίσης είναι αποτελεσματική γιατί ανάλογα με τις παρατηρήσεις επιτρέπει την επικέντρωση της μελέτης στα σημεία που παρουσιάζουν μεγαλύτερο ενδιαφέρον. Ομως συχνά οι προσομοιωτές ανάλογα με τον βαθμό στον οποίο προσομοιώνουν τα διάφορα κομμάτια της παράλληλης μηχανής είναι αργοί και κάνουν δύσκολη την εκτέλεση μεγάλων πειραμάτων.

Από τα παραπάνω γίνεται φανερό πως ο μόνος αποτελεσματικός τρόπος μελέτης των παραλλήλων αρχιτεκτονικών είναι η μέθοδος της προσομοίωσης. Ανάλογα με τον τρόπο που αυτή διεξάγεται, μπορούμε να διακρίνουμε τις ακόλουθες μεθόδους προσομοίωσης:

1. **Προσομοίωση με χρήση ιχνών (trace - driven)**
   Κατά τη διάρκεια εκτέλεσης της εφαρμογής συλλέγεται σε αρχεία πληροφορία, η οποία περιγράφει με ακρίβεια την κυκλοφορία η οποία δημιουργήθηκε από την εφαρμογή. Η πληροφορία αυτή χρησιμοποιείται ως είσοδος για την προσομοίωση των λειτουργιών, στις οποίες επικεντρώνεται το ενδιαφέρον μελέτης.

2. **Προσομοίωση που "οδηγείται" από τα γεγονότα (Execution-driven)**

Η προσομοίωση επικεντρώνεται και πάλι στη συμπεριφορά του δικτύου, αφού αυτό αποτελεί και το βασικό κομμάτι μελέτης. Ομως η εκτέλεση της παράλληλης εφαρμογής γίνεται παράλληλα με την προσομοίωση της παράλληλης αρχιτεκτονικής και όχι ακολουθιακά όπως προηγουμένως. Εαν κατά τη διάρκεια εκτέλεσης της εφαρμογής δημιουργηθεί κάποιο γεγονός που έχει ενδιαφέρον για την προσομοίωση τότε ενημερώνεται ο προσομοιωτής, που εκτελείται παράλληλα, με την εφαρμογή και το προσομοιώνει.

Εχουν αναπτυχθεί μια σειρά απο προσομοιωτές για την μελέτη των παράλληλων αρχιτεκτονικών και των εφαρμογών που τρέχουν πάνω σε αυτές. Ο Proteus [10], ο Mint [11], ο Tango-Lite [14], ο Talisman [16] και το Augmint [15] είναι μερικά από τους πιο γνωστά εργαλεία που έχουν δημιουργηθεί για την μελέτη με προσομοίωση μηχανών κοινόχρηστης μνήμης. Καθώς όμως το κόστος της προσομοίωσης είναι μεγάλο όταν αυτή γίνεται σε μεγάλο βάθος μοντελοποίησης και με αρκετή λεπτομέρεια, οι παραπάνω προσομοιωτές έχουν επικεντρωθεί κυρίως στην μελέτη της αρχιτεκτονικής των μηχανών αφήνοντας σε δεύτερη μοίρα την μελέτη του δικτύου διασύνδεσης.

Για συστήματα ανταλλαγής μηνυμάτων έχουν αναπτυχθεί πολλά χρήσιμα πακέτα και βιβλιοθήκες, όπως είναι το PICL [12], το MPI [7] και το PVM [17], τα οποία και μελετήθηκαν. Υστερα από την επιτυχία του PICL/Paragraph, του Oak Ridge National Laboratory, κάθε εταιρεία υπολογιστών η οποία κατασκεύαζε ένα καινούργιο πολυεπεξεργαστή κατασκεύαζε και μια σειρά εργαλείων που είχαν ως σκοπό την μελέτη των παράλληλων εφαρμογών και την οπτικοποίηση των αποτελεσμάτων των εφαρμογών αυτών. Μερικά από τα πιο γνωστά από αυτά τα εργαλεία είναι το ParAide της Intel Paragon, το MPP Apprentice της CRI T-2D, το PRISM της TMC CM-5, το PV της IBM SP2 καθώς και το CXTrace της Convex SPP1. Αλλα γνωστά εργαλεία τα οποία κατασκευάστηκαν από πανεπιστήμια είναι το Pablo του UIUC, το XPVM [20] και το AIMS της NASA.

Κατά τη διάρκεια της εργασίας αυτής μελετήθηκαν πολλά από αυτά τα εργαλεία. Κύριο ζητούμενο κατά την μελέτη τους ήταν η πιθανή χρησιμοποίηση τους για τη συλλογή ιχνών (traces) τα οποία να περιγράφουν το τι συμβαίνει κατά την εκτέλεση μιας εφαρμογής κυρίως όσο αφορά το δίκτυο διασύνδεσης. Στις επόμενες δύο παραγράφους περιγράφουμε συνοπτικά τα εργαλεία αυτά καθώς και μερικές χρήσιμες παρατηρήσεις που συλλέχθησαν κατά την μελέτη τους.

3.1 Εργαλεία προσομοίωσης μηχανών κοινόχρηστης μνήμης

- **PROTEUS**

  Ο Proteus [10] είναι ένας προσομοιωτής για μηχανές κοινόχρηστης μνήμης, του οποίου η εκτέλεση κατευθύνεται από τα γεγονότα που δημιουργούνται κατά τη διάρκεια της προσομοίωσης (execution driven). Στον Proteus η λειτουργία κάθε επεξεργαστή της αρχιτεκτονικής που μελετάται προσομοιώνεται με ξεχωριστά νήματα (threads). Οι λειτουργίες κάθε επεξεργαστή χωρίζονται σε τοπικές και μη τοπικές (local και non-local operations αντίστοιχα). Ως τοπικές θεωρούνται οι λειτουργίες ενός επεξεργαστή που μπορούν να εξυπηρετηθούν από τον ίδιο ενώ ως μη τοπικές αυτές που εμπλέκουν και άλλους επεξεργαστές, όπως είναι η προσπέλαση της μνήμης ενός άλλου επεξεργαστή. Οι τοπικές εντολές μεταφράζονται από τον Proteus σε κώδικα που εκτελείται από την μηχανή πάνω στην οποία τρέχει ο προσομοιωτής (native execution). Οι μη τοπικές προσομοιώνονται από τον Proteus.

  Εκτός από τα νήματα που υπάρχουν για την προσομοίωση των λειτουργιών των επεξεργαστών της αρχιτεκτονικής που μελετάται, ο Proteus χρησιμοποιεί ακόμη ένα νήμα, το οποίο είναι υπεύθυνο για την ομαλή εξέλιξη της προσομοίωσης. Το νήμα αυτό κρατάει μια λίστα στην οποία αποθηκεύονται όλες οι αιτήσεις που γίνονται προς τον προσομοιωτή. Οι αιτήσεις αυτές αναφέρονται σε μη τοπικές εντολές οι οποίες όπως αναφέρεται παραπάνω προσομοιώνονται από τον Proteus. Το νήμα αυτό είναι υπεύθυνο για την εξυπηρέτηση των αιτήσεων αυτών και για την μετάβαση του ελέγχου ροής από το ένα νήμα στο άλλο.

  Η προσομοίωση της παράλληλης εφαρμογής δεν γίνεται κύκλο προς κύκλο. Κάθε thread εκτελείται μέχρι ο επεξεργαστής τον οποίο προσομοιώνει να εκδόσει μια μη τοπική εντολή. Δηλαδή κάθε επεξεργαστής μπορεί να προχωρά στην εκτέλεσή του για αρκετούς κύκλους μηχανής. Η έκδοση μιας μη τοπικής εντολής δημιουργεί μια αίτηση για προσομοίωση, η οποία εισάγεται στη λίστα για αιτήσεις που κρατάει ο προσομοιωτής και ο έλεγχος μεταφέρεται στο κεντρικό νήμα. Αυτό εξετάζει τον επόμενο κόμβο της λίστας και ανάλογα μεταφέρει τον έλεγχο είτε σε κάποιο άλλο νήμα (συνέχεια εκτέλεσης της παράλληλης εφαρμογής σε κάποιον άλλο κόμβο) είτε σε κάποιον request handler, ο οποίος και προσομοίωνει μια μη τοπική εντολή.

  Ο προσομοιωτής μπορεί με κατάλληλες προσθήκες να τροποποιηθεί έτσι ώστε να παράγει traces, που να περιγράφουν τα μηνύματα που ανταλλάσονται μεταξύ των επεξεργαστών κατά τη διάρκεια της εκτέλεσης της εφαρμογής. Καθώς η αποστολή

και η λήψη ενός μηνύματος είναι μη τοπικές εντολές, ισοδυναμούν με τη δημιουργία στο κεντρικό νήμα του προσομοιωτή μιας αίτησης, η οποία και εξυπηρετείται από κάποιον request handler. Με κατάλληλες μετατροπές ο handler αυτός, ο οποίος έχει στη διάθεσή του όλες τις πληροφορίες που θέλουμε να καταγραφούν, μπορεί να διοχετεύσει την πληροφορία αυτή στο trace file.

Το κύριο μειονέκτημα του Proteus είναι ότι δέχεται μόνο εφαρμογές που είναι γραμμένες σε μια ειδική γλώσσα, υπερσύνολο της C και επομένως ήδη υπάρχουσες εφαρμογές θα πρέπει να τροποποιηθούν προκειμένου να χρησιμοποιηθούν ως είσοδος στον προσομοιωτή. Η τροποποίηση αυτή δεν γίνεται αυτόματα, γεγονός που αποτελεί ένα από τα μεγαλύτερα μειονεκτήματα του εργαλείου. Επίσης ο Proteus εξαιτίας του τρόπου κατασκευής του κάνει δύσκολη την προσομοίωση εφαρμογών που χρησιμοποιούν εντολές όπως η fork().

- **MINT**

Ο MINT [11] είναι ένα εργαλείο, το οποίο βοηθά στην κατασκευή event-driven προσομοιωτών ιεραρχίας μνήμης για multiprocessors. Περιλαμβάνει έναν memory reference generator καθώς και μια βιβλιοθήκη και δίνει στον χρήστη τη δυνατότητα να κατασκευάσει τον προσομοιωτή του συστήματος που θέλει να μελετήσει, επικεντρώνοντας το ενδιαφέρον του στα σημεία που αυτός θεωρεί πιο σημαντικά.

Ο MINT δέχεται ως είσοδο statically linked Irix executables, που έχουν μεταγλωτιστεί για τον MIPS R3000 επεξεργαστή. Σε αντίθεση λοιπόν με τον Proteus, δεν μεταβάλλει καθόλου τις εφαρμογές που εκτελεί.

Ο MINT δίνει τη δυνατότητα για την κατασκευή προσομοιωτών, που επικεντρώνουν το ενδιαφέρον τους σε ορισμένα τμήματα. Με προσεχτική σχεδίαση μπορεί να κατασκευαστεί ένας ολοκληρωμένος program driven προσομοιωτής, ο οποίος μελετά τη συμπεριφορά του δικτύου διασύνδεσης.

- **Αλλα εργαλεία**

Αλλοι προσομοιωτές οι οποίο μελετήθηκαν στα πλαίσια αυτής της εργασίας είναι ο Tango-Lite [14], απόγονος του προσομοιωτή Tango, ο Talisman [16] και ο Augmint [15].

Ολα αυτά τα εργαλεία δεν προσομοιώνουν με λεπτομέρεια την εκτέλεση των εντολών σε κάθε επεξεργαστή. Αυτό διότι θεωρούν πως η προσομοίωση

της εκτέλεσης κάθε εντολής δεν επηρεάζει ιδιαίτερα την συμπεριφορά του συστήματος. Κατά συνέπεια το μεγαλύτερο μέρος των εφαρμογών εκτελείται από τον επεξεργαστή, που χρησιμοποιείται για την εκτέλεση του προσομοιωτή. Προσομοίωση χρησιμοποιείται μόνο τις εντολές που παρουσιάζουν ενδιαφέρον για την εξέλιξη του συστήματος. Τα εργαλεία αυτά δεν προσομοιώνουν τις λειτουργίες του δικτύου.

3.2 Συστήματα ανταλλαγής μηνυμάτων

- **PICL**

  Το PICL (Portable Instrumentated Communication Library) [12] είναι μια βιβλιοθήκη, η οποία υλοποιεί ένα γενικό μοντέλο ανταλλαγής μηνυμάτων σε πολυεπεξεργαστές. Προγράμματα τα οποία έχουν γραφεί χρησιμοποιώντας PICL ρουτίνες και όχι τις εντολές για επικοινωνία μεταξύ των επεξεργαστών, που υποστηρίζονται άμεσα από την μηχανή (native commands), είναι συμβατό (portable), με την έννοια ότι μπορούν να εκτελεστούν σε οποιαδήποτε μηχανή, η οποία υποστηρίζεται από την βιβλιοθήκη PICL.

  Το PICL παρέχει μια σειρά από ρουτίνες, οι οποίες επιτελούν τη συλλογή των ιχνών (traces). Με απλή προσθήκη των ρουτινών αυτών στον πηγαίο κώδικα της εφαρμογής, παράγεται ένα αρχείο, στο οποίο καταγράφονται πληροφορίες για τα γεγονότα τα οποία συνέβησαν κατά τη διάρκεια της εκτέλεσης της εφαρμογής.

  Η εντολή αυτή δίνει επίσης στον προγραμματιστή τη δυνατότητα να διαλέξει την μορφή, την οποία θα έχει το παραγόμενο αρχείο. Το αρχείο αυτό μπορεί να είναι είτε αναλυτική μορφή, είτε σε μορφή πιο συμπαγή . Η πρώτη κάνει την ανάγνωση και κατανόηση του παραγόμενου trace αρχείου δυνατή από το χρήστη. Μειονέκτημα της το μέγεθος του παραγόμενου αρχείου. Η δεύτερη μορφή μειώνει κατά πολύ το μέγεθος του αρχείου αλλά για να διαβαστεί αυτό από τον χρήστη απαιτείται η μετατροπή του.

  Για κάθε γεγονός που συμβαίνει καταγράφονται στο trace αρχείο 2 γραμμές με πληροφορίες. Η πρώτη αναφέρεται στην στιγμή που αρχίζει η εκτέλεση του γεγονότος αυτού και η δεύτερη καταγράφει την στιγμή που αυτό τερματίζει. Τα αρχεία που παράγονται από το PICL μπορεί κανείς να τα δεί χρησιμοποιώντας

εργαλεία οπτικοποίησης όπως το Paragraph και το Upshot.

Σε κατανεμημένα συστήματα, η χρονική πληροφορία που σχετίζεται με τα γεγονότα που καταγράφονται στα ίχνη δεν είναι πάντοτε σωστή και ακριβής. Αυτό οφείλεται στις διαφοροποιήσεις που υπάρχουν μεταξύ των τοπικών ρολογιών των επεξεργαστών. Για να εξασφαλιστεί ότι τα γεγονότα που καταγράφονται είναι στην σωστή χρονική σειρά απαιτείται επιπλέον εργασία η οποία αυξάνει το κόστος της συλλογής των ιχνών. Το PICL επιτυγχάνει τον συγχρονισμό των τοπικών ρολογιών των επεξεργαστών με χρήση της εντολής tracenode.

Παρά το γεγονός ότι το PICL είναι ένα πολύ καλό εργαλείο για παραγωγή traces, έχει ένα μειονέκτημα, το οποίο εμποδίζει την ευρεία χρήση του. Το PICL παράγει traces μόνο για προγράμματα, τα οποία επιτελούν την ανταλλαγή των μηνυμάτων τους με χρήση PICL ρουτινών. Δεν συλλέγει πληροφορίες για μηνύματα, τα οποία στέλνονται με χρήση άλλου τύπου ρουτινών. Αυτό σημαίνει ότι όλα τα προγράμματα θα πρέπει να ξαναγραφούν σε PICL μορφή για να είναι δυνατή η συλλογη ιχνών.

Ομως τα πιο πολλά benchmarks, τα οποία είναι διαθέσιμα είναι γραμμένα είτε σε PVM [17] είτε σε MPI, ενώ ελάχιστες είναι οι διαθέσιμες εφαρμογές σε PICL. Το κόστος για την μετατροπή των υπαρχουσών εφαρμογών σε μορφή PICL είναι ασύμφορο και αποτελεί την κυριότερη αιτία για την οποία το συγκεκριμένο εργαλείο δεν χρησιμοποιήθηκε στην παρούσα εργασία για την συλλογή των ιχνών.

Τέλος η μορφή των αρχείων τα οποία παράγονται από το PICL δεν είναι ικανοποιητική. Παρότι το PICL καταγράφει πληροφορία για όλα τα γεγονότα που σχετίζονται με την ανταλλαγή μηνυμάτων μεταξύ των επεξεργαστών κατά τη διάρκεια εκτέλεσης της εφαρμογής, δεν συλλέγει στοιχεία για άλλου τύπου γεγονότα, όπως για παράδειγμα συγχρονισμός με barriers μεταξύ των επεξεργαστών. Η καταγραφή των γεγονότων αυτών είναι σημαντική προκειμένου να μελετηθεί προσεχτικά η εφαρμογή.

- **PVM**

Το PVM (Parallel Virtual Machine) [17] είναι ένα πακέτο λογισμικού, το οποίο κάνει δυνατή τη συνένωση πολλών ετερογενών υπολογιστών Unix σε μια παράλληλη μηχανή. Το σύστημα που δημιουργείται κάνει πιο εύκολη την επίλυση μεγάλων υπολογιστικών προβλημάτων, αφού ο χρήστης έχει πλέον στη διάθεσή του την υπολογιστική ισχύ και την μνήμη όχι μόνο ενός αλλά πολλών υπολογιστών.

Αυτός είναι και ένας από τους κυριότερους στόχους του PVM, δηλαδή το να επιτρέψει στον χρήστη τη δημιουργία μιας ”ιδεατής παράλληλης μηχανής”, η οποία θα αποτελείται από ετερογενείς κόμβους.

Το PVM παρέχει πληθώρα ρουτινών, όχι μόνο για την ανταλλαγή μηνυμάτων μεταξύ των κόμβων του συστήματος, αλλά και για τον έλεγχο των εργασιών που εκτελούνται στο σύστημα, τον έλεγχο της κατάστασης των κόμβων του συστήματος καθώς και δυνατότητες για τη συλλογή ιχνών. Πέρα όμως από τις ρουτίνες του PVM για συλλογή ιχνών έχουν αναπτυχθεί πολλά πακέτα, τα οποία συλλέγουν πληροφορίες για την εκτέλεση των PVM προγραμμάτων. Τα πακέτα έχουν κατασκευαστεί με σκοπό η συλλογή των traces να επηρεάζει όσο το δυνατόν λιγότερο την εκτέλεση της εφαρμογής. Τα πιο γνωστά από αυτά ειναι το PG_PVM [18], το PGPVM [19] και το XPVM [20]. Από αυτά τα δύο πρώτα παράγουν αρχεία που έχουν την μορφή που καθορίζεται από το πακέτο PICL. Η μορφή αυτή επιτρέπει την χρησιμοποίηση των αρχείων αυτών από το Paragraph, που είναι ένα εργαλείο οπτικοποίησης παράλληλων εφαρμογών. Το XPVM παράγει τις πληροφορίες σε διαφορετική μορφή (self defining data format, SDDF).

Αρχεία που παράγονται από το XPVM μπορεί κανείς να τα δει χρησιμοποιώντας το Pablo, ένα εργαλείο οπτικοποίησης σχεδιασμένο για να αποδίδει με γραφικό τρόπο τη συμπεριφορά των παράλληλων εφαρμογών.

- **MPI**

Το MPI (Message Passing Interface)[7] είναι ένα interface, το οποίο αποτελεί ένα κοινώς αποδεκτό και ευρέως χρησιμοποιούμενο σημείο αναφοράς για τη δημιουργία προγραμμάτων ανταλλαγής μηνυμάτων. Είναι όχι μόνο γρήγορο (συγκρινόμενο με βιβλιοθήκες που έχουν σχεδιαστεί για συκεκριμένες μηχανές) αλλά και portable (μπορεί να υλοποιηθεί στις πιο πολλές αρχιτεκτονικές κατανεμημένης μνήμης). Το MPI υποστηρίζει πολλούς τρόπους επικοινωνίας.

Υπάρχουν πολλές υλοποιήσεις του, όπως το mpich που είναι μια portable υλοποίηση του MPI γραμμένη από το Argonne National Lab και το Mississippi State University, η οποία βασίζεται στη βιβλιοθήκη P4, υλοποιήσεις που στηρίζονται στο PVM καθώς και η υλοποίηση της IBM, η οποία επιτυγχάνει την καλύτερη απόδοση σε αρχιτεκτονικές SP2. Οι τρόποι συλλογής των ιχνών από εφαρμογές MPI διαφέρει ανάλογα με την υλοποίηση του MPI.

3.3 Εργαλεία ιχνηλασίας σε υπολογιστές SP2

- **Δυνατότητες παραγωγής ιχνών από το περιβάλλον AIX**

  Το λειτουργικό σύστημα της IBM AIX/6000, το οποίο είναι κατασκευασμένο με βάση το σύστημα Unix V, παρέχει τη δυνατότητα της συλλογής πληροφορίας που περιγράφει τις ενέργειες του χρήστη αλλά και του συστήματος. Μερικά από τα γεγονότα για τα οποία συγκεντρώνονται στοιχεία είναι κλήσεις συστήματος, λάθη σελίδας, η κατάσταση των διεργασιών του συστήματος και τέλος I/O γεγονότα όπως reads και writes. Επιπλέον το AIX κάνει δυνατή τη συλλογή ιχνών από προγράμματα γραμμένα είτε σε MPI είτε σε MPL (Message Passing Library).

  Οι ρουτίνες του AIX αποτελούν τη βάση για την κατασκευή του UTE [13], ενός περιβάλλοντος ειδικά κατασκευασμένου για την παραγωγή, ανάλυση και οπτικοποίηση ιχνών εφαρμογών που εκτελούνται σε IBM SP2 υπολογιστές.

- **Unified Trace Environment (UTE)**

  Το UTE [13] αποτελεί ένα περιβάλλον για την ανάλυση παράλληλων εφαρμογών και τη συλλογή ιχνών. Η κατασκευή του στηρίζεται στις δυνατότητες για συλλογή ιχνών που παρέχονται από το AIX γεγονός το οποίο συντελεί στη δημιουργία ενός ενοποιημένου και εύκολα επεκτάσιμου περιβάλλοντος, το οποίο μπορεί να συγκεντρώνει στοιχεία από διαφορετικού τύπου εφαρμογές όπως για παράδειγμα προγράμματα γραμμένα σε MPI, MPL, PIOFS (parallel file system) ή HPF (High Performance Fortran).

  Τα πιο πολλά συστήματα που συλλέγουν ίχνη από εφαρμογές απαιτούν την μετατροπή του πηγαίου κώδικα των εφαρμογών αυτών. Ενα από τα μεγαλύτερα πλεονεκτήματα του UTE είναι ότι δεν απαιτεί την μετατροπή του πηγαίου κώδικα της εφαρμογής, γεγονός που κάνει τη χρήση του πολύ απλή. Σε περίπτωση όμως που αυτός είναι διαθέσιμος ο χρήστης μπορεί με απλή εισαγωγή κώδικα να επικεντρώσει την προσοχή του σε κάποια σημεία του προγράμματος π.χ. loops ή ρουτίνες της εφαρμογής και να τα μελετήσει εκτενέστερα και προσεκτικότερα.

  Ενα ιδανικό περιβάλλον συλλογής ιχνών θα πρέπει να συλλέγει τις απαιτούμενες πληροφορίες επιβαρύνοντας όσο το δυνατόν λιγότερο την εκτέλεση της εφαρμογής. Εαν το κόστος για την παραγωγή των ιχνών είναι μεγάλο η χρονική πληροφορία που σχετίζεται με τα καταγραφόμενα γεγονότα μπορεί να υποστεί αρκετές αλλαγές με αποτέλεσμα τα στατιστικά στοιχεία και η πληροφορία η οποία συγκεντρώνεται

να μην έχουν νόημα. Οι ρουτίνες του AIX κάνουν δυνατή τη συλλογή πληροφορίας χωρίς να επηρεάζουν σε μεγάλο βαθμό την εκτέλεση της εφαρμογής. Τα δεδομένα αποθηκεύονται στην κύρια μνήμη των κόμβων της παράλληλης μηχανής γεγονός το οποίο μειώνει το κόστος για τη συλλογή τους, αφού περιορίζει συμβάντα όπως λάθη σελίδας.

Πριν ξεκινήσει η εκτέλεση της εφαρμογής, ο χρήστης πρέπει να καθορίσει την τιμή της μεταβλητής περιβάλλοντος TRACEOPT, από την οποία εξαρτώνται μια σειρά από γεγονότα όπως οι τύποι των γεγονότων για τους οποίους θα συλλεγεί πληροφορία, το μέγεθος του ενταμιευτή που βρίσκεται στην μνήμη και χρησιμοποιείται για την αποθήκευση των γεγονότων, η κατάληξη του αρχείου στο οποίο αποθηκεύονται τα ίχνη. Εαν ο χρήστης δεν θέσει κάποια τιμή στην μεταβλητή TRACEOPT τότε η εκτέλεση του προγράμματος δεν θα συνοδευτεί από την παραγωγή ιχνών.

Το UTE μπορεί να παράγει ίχνη για τις ακόλουθες κατηγορίες γεγονότων:

-- ανταλλαγές μηνυμάτων μεταξύ των επεξεργαστών

-- κλήσεις συστήματος, page faults, interrupts και process dispatch

-- ενέργειες που σχετίζονται με εγγραφές και αναγνώσεις (I/O)

-- ενέργειες που αφορούν το TCP/IP

Το UTE αποτελείται από:

-- τη βιβλιοθήκη libute.a

-- το Nupshot, ένα εργαλείο για την οπτικοποίηση της εφαρμογής, απόγονο του Upshot [21]

-- διάφορα προγράμματα για την μετατροπή, και συνένωση των παραγόμενων αρχείων καθώς επίσης και προγράμματα για τη συλλογή στατιστικών στοιχείων που αφορούν την εφαρμογή

Η πληροφορία η οποία συλλέγεται από το UTE είναι σε δυαδική μορφή (binary) και πρέπει να μετατραπεί σε άλλη μορφή προκειμένου να οπτικοποιηθεί. Το UTE περιλαμβάνει μια σειρά από προγράμματα μετατροπής των παραγόμενων αρχείων προκειμένου αυτά να μπορούν να χρησιμοποιηθούν από εργαλεία οπτικοποίησης όπως τα:

-- Nupshot , που είναι ένα εργαλείο οπτικοποίσης παράλληλων εφαρμογών που αναπτύχθηκε στο Argonne National Laboratory. Απεικονίζει trace files που βρίσκονται είτε σε ALOG είτε σε PICL format.

-- Pablo Self Defining Data Format (SDDF)

-- Visualization Tool (VT)

Καθένα από τα παραπάνω εργαλεία οπτικοποίησης απαιτεί το αρχείο εισόδου να είναι σε μια συγκεκριμένη μορφή. Το VT απαιτει .trc, το Nupshot .ups και το Pablo .sddf αρχεία. Οι απαραίτητες μετατροπές από την μία μορφή στην άλλη γίνονται με τα ακόλουθα εργαλεία:

-- alog2ups μετατρέπει αρχεία από alog μορφή (περίπου ίδια με την PICL μορφή) σε ups μορφή. Τα αρχεία σε alog μορφή χρησιμοποιούνται από το εργαλείο upshot ενώ τα ups αρχεία από το Nupshot, που είναι η μεταγενέστερη μορφή του Upshot.

-- mp2sddf μετατρέπει αρχεία από την μορφή VT (.trc) σε SSDF μορφή

-- ute2sddf μετατρέπει αρχεία από την μορφή UTE (.ups) σε μορφή SDDF

- **Visualization Tool**

Το Visualization Tool (VT) του παράλληλου περιβάλλοντος AIX της IBM είναι ένα εργαλείο οπτικοποίησης, το οποίο αναπαριστά με γραφικό τρόπο εφαρμογές που είναι γραμμένες χρησιμοποιώντας το Message Passing Interface (MPI). Το VT παρέχει πληθώρα από εικόνες βοηθώντας με τον τρόπο αυτό τον χρήστη να αποκτήσει σαφή εικόνα του τι συμβαίνει κατά τη διάρκεια εκτέλεσης της εφαρμογής.

Πριν ξεκινήσει η εκτέλεση της εφαρμογής, ο χρήστης καθορίζει τους τύπους των γεγονότων που έχουν ενδιαφέρον για αυτόν. Για τις κατηγορίες αυτές των γεγονότων θα συλλεγεί πληροφορία κατά την εκτέλεση της εφαρμογής. Τα γεγονότα αυτά ανήκουν σε μία από τις ακόλουθες κατηγορίες:

-- γεγονότα που σχετίζονται με τα μηνύματα που ανταλλάσονται μεταξύ των επεξεργαστών κατά τη διάρκεια της εφαρμογής. Τέτοια γεγονότα είναι για παράδειγμα ένα send ή ένα receive.

-- γεγονότα που αφορούν την χρήση του συστήματος. Τα γεγονότα αυτά δίνουν στον χρήστη πληροφορία σχετικά με το ποσοστό χρήσης της CPU, την κυκλοφορία που σχετίζεται με το δίσκο ή στατιστικά στοιχεία που αφορούν

την μνήμη. Η συχνότητα με την οποία γίνεται η συλλογή των στατιστικών αυτών στοιχείων επιλέγεται από τον χρήστη, πριν την εκτέλεση της εφαρμογής

-- γεγονότα, τα οποία αφορούν κάποια σημεία της εφαρμογής που έχουν ενδιαφέρον για τον χρήστη

Στην συνέχεια εκτελείται η εφαρμογή και κάθε φορά που κάποιο ενδιαφέρον γεγονός λαβαίνει χώρα, πληροφορία σχετική με αυτό καταγράφεται σε ένα αρχείο. Δημιουργούνται τόσα αρχεία όσοι και οι κόμβοι που χρησιμοποιούνται για την εκτέλεση της εφαρμογής. Τα αρχεία αυτά συνενώνονται στο τέλος σε ένα μοναδικό αρχείο. Το εργαλείο οπτικοποίησης (VT) χρησιμοποιεί το τελικό αυτό αρχείο για να "παίξει ξανά" τα γεγονότα που δημιουργήθηκαν προηγουμένως δίνοντας στον χρήστη μια παραστατική και σαφή εικόνα της εφαρμογής και του συστήματος.

Το VT παρέχει στο χρήστη την πληροφορία αυτή μέσα από μια πληθώρα εικόνων, τις οποίες μπορεί κανείς να κατατάξει στις ακόλουθες κατηγορίες:

-- Υπολογισμός, το πόσο δηλαδή χρησιμοποιήθηκε ο κάθε επεξεργαστής κατά την εκτέλεση του προγράμματος

-- Επικοινωνία, πληροφορία η οποία σχετίζεται με τα μηνύματα που ανταλλάσουν οι επεξεργαστές κατά την εκτέλεση της εφαρμογής

-- Σύστημα, πληροφορία σχετικά με τη δραστηριότητα του συστήματος, όπως το ποσοστό των λαθών σελίδας (page faults) ή των context switches

-- Δίκτυο, αριθμός των TCP/IP πακέτων που στάλθησαν και παραλείφθησαν

-- Δίσκος, δραστηριότητες του δίσκου, όπως διάβασμα ή εγγραφή σε δίσκο

Οι εικόνες με τις οποίες το VT απεικονίζει την εφαρμογή είναι περίπου ίδιες με αυτές που παρέχονται από το Paragraph. Ομως το VT δίνει στον χρήστη περισσότερη ευελιξία όσο αφορά στον τρόπο με τον οποίο αυτός μπορεί να δεί τα δεδομένα. Επίσης η πληροφορία την οποία παρέχει είναι περισσότερη, σε σύγκριση με αυτήν την οποία παριστά το Paragraph.

Το Visualization Tool χρησιμοποιεί δικές του ρουτίνες για τη δημιουργία των ιχνών και όχι τις ρουτίνες που παρέχει το λειτουργικό σύστημα AIX. Για τη συλλογή των ιχνών απαιτούνται μερικές απλές μετατροπές του πηγαίου κώδικα της εφαρμογής. Ειδικότερα, χρειάζεται η προσθήκη μερικών απλών ρουτινών στην αρχή και στο τέλος της εφαρμογής. Για τα προγράμματα που είναι γραμμένα σε Fortran οι ρουτίνες αυτές είναι οι:

-- INTEGER FUNCTION VT_TRC_START (INTEGER FLAG)

-- INTEGER FUNCTION VT_TRC_STOP

ενώ για τα C προγράμματα είναι οι:

-- int VT_trc_start_c(int)

-- int VT_trc_stop_c()

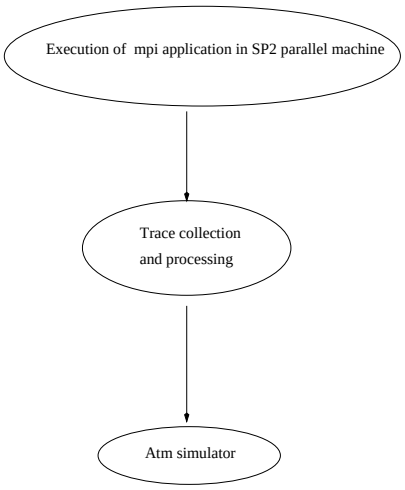
Το όρισμα στην ρουτίνα VT_TRC_START / VT_trc_start_c ελέγχει το ποιοί τύποι δεδομένων θα καταγραφούν. Εκτός από την προσθήκη των ρουτινών αυτών ο χρήστης πρέπει να θέσει και την μεταβλητή περιβάλλοντος MP_TRACELEVEL σε κάποια τιμή (ίδια με αυτήν του ορίσματος των ρουτινών VT_TRC_START / VT_trc_start_c).

Το αρχείο που παράγεται έχει το ίδιο όνομα με την εφαρμογή από την οποία προέρχεται και την κατάληξη .trc . Βρίσκεται σε δυαδική μορφή (binary).

Κεφάλαιο 4

Προσομοίωση

Η στρατηγική, η οποία ακολουθείται για την μελέτη της συμπεριφοράς του δικτύου και της επίδρασης του στην εκτέλεση των παράλληλων εφαρμογών απεικονίζεται στο σχήμα 4.1:



Σχήμα 4.1: Μοντέλο προσομοίωσης

Αρχικά εκτελούνται οι εφαρμογές οι οποίες θα αποτελέσουν την πηγή για την μελέτη της απόδοσης του δικτύου. Πρόκειται για πολύ δημοφιλή προγράμματα σε MPI, όπως τα NASA benchmarks [8], τα οποία έχουν γραφεί για να συμβάλλουν στην εξέταση

της απόδοσης των παράλληλων αρχιτεκτονικών. Για την εκτέλεση των εφαρμογών χρησιμοποιήθηκαν οι πόροι του Maui High Performance Computing Center (MHPCC).

Το MHPCC διαθέτει ένα IBM Scalable POWERparallel SP2 σύστημα, το οποίο αποτελείται από 400 κόμβους. Ο SP2 είναι ο πιο πρόσφατος scalable παράλληλος υπολογιστής κατανεμημένης μνήμης, που έχει κατασκευάσει η IBM. Κάθε κόμβος της παράλληλης μηχανής είναι ένας RS/6000 επεξεργαστής αρχιτεκτονικής POWER2. Πρόκειται για ένα τσιπάκι superscalar, με δυνατότητα εκτέλεσης έξι εντολών σε ένα κύκλο ρολογιού. Οι επεξεργαστές "τρέχουν" στα 66.7 MHz, παρουσιάζοντας μια απόδοση γύρω στα 266 MFLOPS ανά επεξεργαστή. Η συνολική απόδοση για το όλο σύστημα των 400 κόμβων είναι πάνω από 100 Gigaflops.

Οι κόμβοι του συστήματος επικοινωνούν μεταξύ τους αποκλειστικά με ανταλλαγή μηνυμάτων. Στο MHPCC οι κόμβοι του συστήματος επικοινωνούν με διάφορους τύπους δικτύου, όπως το ethernet και ο SP2 cross-bar μεταγωγέας υψηλής απόδοσης, για τον οποίο επιτυγχάνεται και το μεγαλύτερο bandwidth (40 MB/sec). Η καθυστέρηση εξαιτίας του υλικού (hardware latency) είναι περίπου 500 nanoseconds ανά switch "hop".

Η IBM έχει κατασκευάσει μια σειρά από προγράμματα για την καταγραφή των γεγονότων που συμβαίνουν κατά την εκτέλεση των παράλληλων εφαρμογών καθώς και για την οπτικοποίησή τους όπως το UTE (Unified Trace Environment) και το VT (Visualization Tool). Το τελευταίο χρησιμοποιήθηκε για την καταγραφή των μηνυμάτων που ανταλλάσσονται κατά την εκτέλεση των εφαρμογών, καθώς και άλλων γεγονότων που έχουν ενδιαφέρον (π.χ. πράξεις συγχρονισμού). Η εκτέλεση λοιπόν των εφαρμογών τερματίζει με την καταγραφή σε αρχεία χρήσιμης για την κατοπινή προσομοίωση πληροφορίας.

Τα αρχεία τα οποία δημιουργούνται με το τέλος της εκτέλεσης της εφαρμογής είναι σε δυαδική μορφή (binaries) και χρειάζονται μετατροπή προκειμένου να αποτελέσουν είσοδο για τον προσομοιωτή του δικτύου. Προκειμένου να πάρουν την τελική τους μορφή τα παραγόμενα αρχεία επεξεργάζονται από το πρόγραμμα rtrc. Το rtrc είναι μια εφαρμογή η οποία δέχεται ως είσοδο δυαδικά αρχεία ιχνών (trace files) που παράγονται από το εργαλείο VT και τα μετατρέπει σε ascii μορφή, κρατώντας μονάχα όση πληροφορία είναι απαραίτητη για την αναπαράσταση της κυκλοφορίας που δημιουργήθηκε στο δίκτυο κατά τη διάρκεια της εκτέλεσης της εφαρμογής.

Για κάθε μήνυμα το οποίο ανταλλάσεται μεταξύ των κόμβων του παράλληλου συστήματος καταγράφεται ο χρόνος που ξεκίνησαν και ολοκληρώθηκαν τόσο η

αποστολή όσο και η λήψη του, ο αποστολέας, ο παραλήπτης καθώς και το μέγεθος του μηνύματος σε bytes. Επίσης καταγράφεται ο τύπος της εντολής στην οποία αποδίδεται η δημιουργία του συγκεκριμένου μηνύματος. Η καταγραφή του τύπου της εντολής είναι απαραίτητη για την σωστή προσομοίωση της εφαρμογής.

Οι εντολές ανταλλαγής μηνυμάτων μπορούν να ταξινομηθούν σε δύο κατηγορίες:

- σε αυτές για τις οποίες η ολοκλήρωση της ρουτίνας ανταλλαγής του μηνύματος εξαρτάται από ορισμένα γεγονότα (blocking). Για παράδειγμα, για την αποστολή ενός μηνύματος, η ρουτίνα αποστολής της πληροφορίας τερματίζει είτε όταν τα δεδομένα έχουν σταλεί ή όταν έχουν αντιγραφεί στον ενταμιευτή του συστήματος, έτσι ώστε ο ενταμιευτής της εφαρμογής να μπορεί να ξαναχρησιμοποιηθεί.
- σε αυτές που δεν χρειάζονται την ολοκλήρωση ορισμένων γεγονότων για τον τερματισμό τους (non - blocking). Για παράδειγμα, για να τερματίσει μια non blocking ρουτίνα αποστολής ενός μηνύματος, δεν απαιτείται η αντιγραφή των αποστελλόμενων δεδομένων από τον ενταμιευτή της εφαρμογής στον ενταμιευτή του συστήματος. Σε αυτήν την περίπτωση ο προγραμματιστής θα πρέπει από μόνος του να διαπιστώσει πότε είναι δυνατόν να ξαναχρησιμοποιήσει τον ενταμιευτή της εφαρμογής.

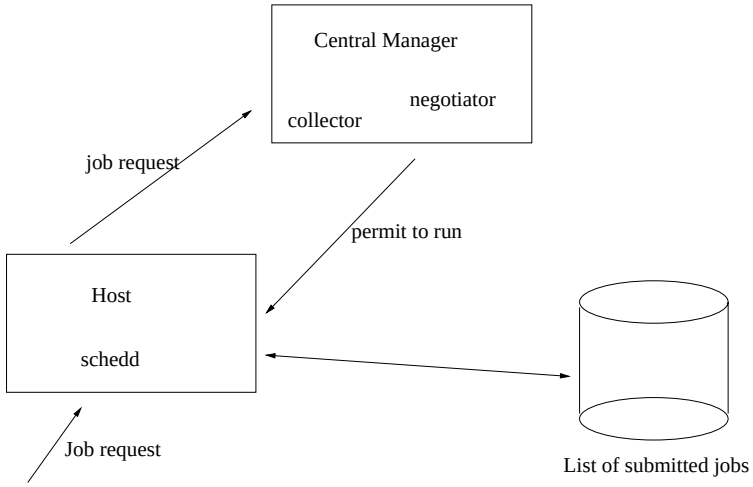
Ο τύπος της εντολής που αντιστοιχεί στην αποστολή ή τη λήψη ενός μηνύματος καταγράφεται λοιπόν προκειμένου να διαπιστωθεί εαν πρόκειται για blocking ή non-blocking εντολή και κατόπιν να ακολουθηθεί ο ανάλογος τρόπος προσομοίωσής της.

Αντί για την προσέγγιση που ακολουθείται (χρήση ιχνών για την μελέτη της συμπεριφοράς του δικτύου (trace driven approach)), θα μπορούσε να ακολουθηθεί ένας διαφορετικός τρόπος προσέγγισης του προβλήματος. Θα μπορούσε κανείς όχι να εκτελεί τις εφαρμογές αλλά να προσομοιώσει την εκτέλεση τους και κάθε φορά που αποστέλλεται κάποιο μήνυμα να ενημερώνεται αμέσως ο προσομοιωτής. Με αυτόν τον τρόπο προσομοίωσης (execution driven), η εκτέλεση της παράλληλης εφαρμογής και του προσομοιωτή του δικτύου γίνονται ταυτόχρονα αποφεύγοντας την αποθήκευση της πληροφορίας σε μεγάλα αρχεία. Ομως το κόστος αυτής της προσέγγισης είναι πολύ μεγάλο αφού η εκτέλεση των εφαρμογών είναι πολύ πιο γρήγορη και λιγότερο δαπανηρή από την προσομοίωσή τους. Για αυτό τον λόγο ακολουθείται trace driven τρόπος προσέγγισης και όχι execution driven.

Οι εφαρμογές εκτελέστηκαν δύο φορές, μια σε 32 και μια σε 64 κόμβους του

συστήματος. Για την εκτέλεση τους χρησιμοποιήθηκε το πρόγραμμα LoadLeveler της IBM. Ο LoadLeveler είναι ένα πρόγραμμα υπεύθυνο για τη δημιουργία, επεξεργασία και εκτέλεση εργασιών στο παράλληλο σύστημα. Φροντίζει να διαμοιράζει τις διάφορες εργασίες με τον καλύτερο δυνατό τρόπο, ώστε να εκμεταλλεύεται όσο το δυνατόν πιο σωστά και αποδοτικά τους πόρους του συστήματος. Σε συστήματα με πολλούς χρήστες ένα τέτοιο πρόγραμμα είναι απαραίτητο προκειμένου να επιτύχουμε την καλύτερη απόδοση και την καλύτερη δυνατή εξυπηρέτηση των χρηστών.

Απλοποιημένα, ένας κύκλος δουλειάς του LoadLeveler είναι ως εξής:



Σχήμα 4.2: Μοντέλο LoadLeveler

Ο χρήστης κατασκευάζει την εργασία, την οποία θέλει να εκτελέσει μέσω του προγράμματος LoadLeveler και την στέλνει από την τοπική μηχανή στον LoadLeveler. Υπεύθυνοι για αυτήν την αποστολή είναι ο δαίμονας του LoadLeveler που τρέχει στην τοπική μηχανή (schedd) και ο δαίμονας του Central Manager (negotiator). Ο Central Manager είναι η μηχανή του συστήματος που αποφασίζει για το που και πότε θα εκτελεστεί μια εργασία στο σύστημα. Καθώς έχει στη διάθεση του στοιχεία για την κατάσταση κάθε μηχανής του συστήματος μπορεί να κάνει την ορθότερη και αποδοτικότερη ανάθεση των εργασιών στους πόρους της παράλληλης μηχανής.

Ο Central Manager διαλέγει την επόμενη προς εκτέλεση εργασία από μια ουρά στην οποία αποθηκεύονται οι εργασίες μέχρι τη στιγμή της εκτέλεσής τους, την οποία αποφασίζει ο Central Manager έχοντας εκτιμήσει μια σειρά από παραμέτρους όπως τον

χρόνο παραμονής των εργασιών στην ουρά, τον χρήστη στον οποίο ανήκει η κάθε εργασία, τους διαθέσιμους πόρους, τις υπολογιστικές απαιτήσεις της εργασίας.

Επιπλέον ο LoadLeveler μπορεί να μας εξασφαλίσει ότι οι κόμβοι της μηχανής θα χρησιμοποιηθούν αποκλειστικά και μόνο για την εκτέλεση της εφαρμογής και ότι δεν θα τρέχουν ταυτόχρονα κανένα άλλο πρόγραμμα. Αυτό είναι απαραίτητο για να μπορέσουμε να καταγράψουμε τους ακριβείς χρόνους υπολογισμού της εφαρμογής.

Χάρη λοιπόν στην παράλληλη μηχανή SP2 και στο πρόγραμμα LoadLeveler μπορούμε να καταγράψουμε με ακρίβεια τη σειρά με την οποία έγιναν οι ανταλλαγές των μηνυμάτων καθώς και τους χρόνους υπολογισμού που σπαταλάει κάθε επεξεργαστής. Η εκτέλεσή τους δεν ήταν δυνατόν να γίνει χρησιμοποιώντας μονάχα τις υπολογιστικές δυνατότητες του Ινστιτούτου Πληροφορικής. Αυτό οφείλεται στο ότι η πλατφόρμα που προσφέρεται για την εκτέλεση των εφαρμογών δεν είναι η κατάλληλη.

Οι υπολογιστές του Ινστιτούτου μπορούν να συνενωθούν σχηματίζοντας μια ιδεατή παράλληλη μηχανή. Ομως η σχηματιζόμενη μηχανή ήταν δομημένη από λίγους κόμβους, με αποτέλεσμα να αντιστοιχεί περισσότερες από μια διεργασίες σε έναν υπολογιστή. Ετσι traces τα οποία συλλέγονταν χρησιμοποιώντας τους πόρους του Ινστιτούτου δεν ήταν ακριβή, όσον αφορά στην καταγραφή των απαιτήσεων κάθε διεργασίας σε υπολογισμό (computation time). Γι' αυτό και ήταν απαραίτητη η χρησιμοποίηση μιας παράλληλης μηχανής με κόμβους τόσους όσες και οι διεργασίες της εφαρμογής.

Στις επόμενες παραγράφους περιγράφεται αναλυτικά ο προσομοιωτής, ο οποίος κατασκευάστηκε για την μελέτη της συμπεριφοράς του δικτύου διασύνδεσης και της επίδρασης του στην εκτέλεση των παράλληλων εφαρμογών. Κατασκευάστηκαν προσομοιωτές για δύο διαφορετικούς τύπους δικτύου, για δίκτυο ATM και για δίκτυο Wormhole. Οι δύο προσομοιωτές είναι ίδιοι όσον αφορά στην κεντρική τους δομή, διαφοροποιούνται όμως στην αρχιτεκτονική του μοντέλου του μεταγωγέα του δικτύου.

4.0.1 Περιγραφή του προσομοιωτή

Ο προσομοιωτής μελετά τη συμπεριφορά δικτύων ATM και Wormhole, με εισιτήρια για τρεις διαφορετικές τοπολογίες, butterfly, mesh και hypercube. Πρόκειται για πρόγραμμα γραμμένο σε κώδικα C. Αποτελείται από δύο βασικά μέρη:

1. Κεντρική ουρά γεγονότων

2. Δευτερεύουσες ουρές, μία για κάθε διεργασία

Στις δευτερεύουσες ουρές αποθηκεύονται πληροφορίες σχετικές με την αποστολή και την λήψη των μηνυμάτων, που αντιστοιχούν σε κάθε επεξεργαστή. Η πληροφορίες αυτές αντλούνται από το trace file και καθορίζουν την ακριβή χρονική σειρά με την οποία κάθε επεξεργαστής θα παραλάβει και θα στείλει μηνύματα, κατά τη διάρκεια της προσομοίωσης. Οι δευτερεύουσες ουρές τροφοδοτούν με γεγονότα την κεντρική ουρά, η οποία και καθορίζει την εξέλιξη της προσομοίωσης.

Σε ένα κύκλο προσομοίωσης,προσομοιώνονται τα γεγονότα τα οποία βρίσκονται στην κεφαλή της ουράς. Τα γεγονότα αυτά σχετίζονται με την αποστολή και τη λήψη cells και εισιτηρίων ελέγχου. Η εκτέλεση τους συνοδεύεται από την εισαγωγή νέων γεγονότων στην κεντρική ουρά αλλά και την πιθανή διαγραφή κάποιων γεγονότων από τις επιμέρους ουρές. Αφού γίνουν όλες οι ενημερώσεις, η προσομοίωση μεταβαίνει στον επόμενο κύκλο, οπότε και προσομοιώνονται τα γεγονότα εκείνη τη χρονική στιγμή έχουν έρθει πλέον στην κορυφή της ουράς.

Η προσομοίωση τελειώνει όταν η κεντρική ουρά αδειάσει. Εαν έχουν αδειάσει και οι επιμέρους ουρές η προσομοίωση ήταν επιτυχής, αφού όλα τα μηνύματα έχουν αποσταλλεί και έχουν φτάσει στον προορισμό τους. Σε διαφορετική περίπτωση, η προσομοίωση τελειώνει με ένδειξη λάθους.

Περιγραφή μοντέλου επεξεργαστή

Με βάση τα communication events που συμβαίνουν σε κάθε επεξεργαστή, αυτός κατά τη διάρκεια της προσομοίωσης μεταβαίνει ανάμεσα στις ακόλουθες καταστάσεις:

- Κατάσταση Αποστολής (Send state)
  Ο επεξεργαστής στέλνει κάποιο μήνυμα στο δίκτυο. Κατά τη διάρκεια της αποστολής , είναι δυνατή η αποδοχή, από τον επεξεργαστή, μηνυμάτων που φτάνουν από το δίκτυο, τα οποία όμως παραμένουν στους ενταμιευτές (buffers) του επεξεργαστή μέχρι να ολοκληρωθεί η αποστολή του συγκεκριμένου μηνύματος.

- Κατάσταση Λήψης (Receive state)
  Ο επεξεργαστής ολοκληρώνει τη λήψη κάποιου μήνυματος. Μετά την ολοκλήρωση της λήψης του μηνύματος, αρχίζει η προσομοίωση του επομένου γεγονότος που αντιστοιχεί σε αυτό τον επεξεργαστή, όπως αυτό καθορίζεται από το trace file.

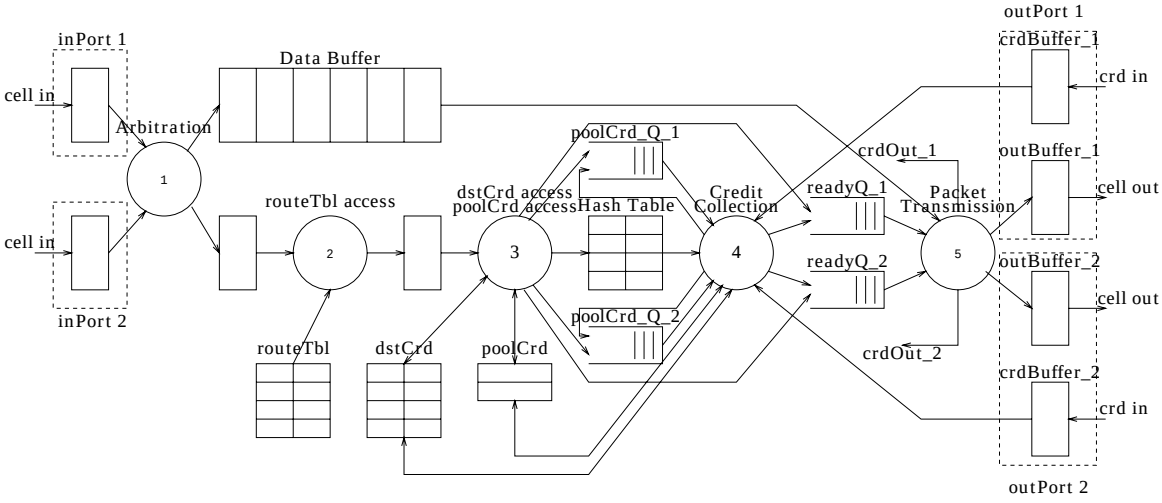
- Κατάσταση Αδράνειας (Idle state)
  Ο επεξεργαστής είναι ανενεργός, περιμένοντας για τη λήψη κάποιου συγκεκριμένου μηνύματος το οποίο καθυστερεί. Κατά τη διάρκεια της αναμονής, cells άλλων μηνυμάτων, τα οποία φτάνουν στον επεξεργαστή αποθηκεύονται στους ενταμιευτές του.

- Κατάσταση Υπολογισμού (Computing state)
  Ο επεξεργαστής εκτελεί εντολές του παράλληλου προγράμματος. Στην κατάσταση αυτή δεν αποστέλλονται μηνύματα στο δίκτυο ενώ μηνύματα ποθ φτάνουν στο συγκεκριμένο κόμβο αποθηκεύονται στους ενταμιευτές του μέχρι την μετάβαση του επεξεργαστή στην κατάσταση λήψης.

- Κατάσταση Ολοκλήρωσης (Done state)
  Ο επεξεργαστής έχει ολοκληρώσει την αποστολή και την λήψη των μηνυμάτων που του αντιστοιχούσαν.

Περιγραφή μοντέλου μεταγωγέα ATM

Ως μοντέλο μεταγωγέα χρησιμοποιήθηκε αυτό το οποίο αναπτύχθηκε κατά την μεταπτυχιακή εργασία του Ε. Σπυριδάκη [4]. Το μοντέλο αποτελείται από πέντε εξυπηρετητές, καθένας από τους οποίους εκτελεί λειτουργίες σε διάρκεια ενός κύκλου ρολογιού του μεταγωγέα.



Σχήμα 4.3: Μοντέλο μεταγωγέα ATM 2×2

Ο εξυπηρετητής 1 είναι υπεύθυνος για την παραλαβή των cells που φτάνουν

στον μεταγωγέα. Η άφιξη ενός cell συνοδεύεται και από τη δέσμευση χώρου στον ενταμιευτή του μεταγωγέα, για την αποθήκευση της πληροφορίας που αυτό μεταφέρει. Η επικεφαλίδα του cell διέρχεται από τους υπόλοιπους εξυπηρετητές με σκοπό τη δρομολόγηση της αποστολής του cell στο δίκτυο.

Αρχικά, η επικεφαλίδα του cell μεταβαίνει στον εξυπηρετητή 2, όπου και εξετάζεται για να διαπιστωθεί ποιός από τους συνδέσμους εξόδου του μεταγωγέα θα χρησιμοποιηθεί κατά την έξοδο του cell από αυτόν στο δίκτυο.

Στην συνέχεια, η επικεφαλίδα του cell μεταβαίνει στον εξυπηρετητή 3, όπου και αποφασίζεται σε ποιά από τις ουρές του μεταγωγέα θα αποθηκευτεί, μέχρι να λάβει χώρα η αποστολή του στον επόμενο στη σειρά μεταγωγέα. Για τον σκοπό αυτό, ο εξυπηρετητής 3 ελέγχει στους πίνακες Destination Credit και Pool Credit για την ύπαρξη εισιτηρίων αναχώρησης και προορισμού. Ανάλογα με την ύπαρξη ή όχι των εισιτηρίων αυτών, η επικεφαλίδα του cell αποθηκεύεται:

- στην ουρά readyQ_x, όπου x ο σύνδεσμος εξόδου, από τον οποίο θα φύγει το cell. Αυτό συμβαίνει μόνο εφόσον υπάρχουν και οι δύο τύποι εισιτηρίων, οπότε επιτρέπεται η άμεση αποστολή του cell. Στην περίπτωση αυτή καταναλώνεται ένα εισιτήριο από κάθε είδος.

- στον πίνακα Hash Table, εφόσον δεν υπάρχει εισιτήριο προορισμού. Το cell θα παραμείνει εκεί μέχρι να έρθει ένα εισιτήριο προορισμού.

- στην ουρά poolCrd_Q_x, όπου x ο σύνδεσμος εξόδου, εφόσον υπάρχει εισιτήριο προορισμού αλλά όχι και το αντίστοιχο εισιτήριο αναχώρησης. Σε αυτήν την περίπτωση το εισιτήριο προορισμού καταναλώνεται και το cell αποθηκεύεται στην ουρά poolCrd_Q_x περιμένοντας για εισιτήριο αναχώρησης.

Ο εξυπηρετητής 4 είναι υπεύθυνος για τις λειτουργίες που πρέπει να γίνουν όταν ένα εισιτήριο φτάνει στον μεταγωγέα. Το εισιτήριο αυτό περιλαμβάνει ένα εισιτήριο αναχώρησης για τον επόμενο στην σειρά μεταγωγέα και ένα εισιτήριο προορισμού, για τον προορισμό προς τον οποίο κατευθύνεται το cell, του οποίου η αναχώρηση από τον επόμενο μεταγωγέα προκάλεσε την αποστολή αυτού του εισιτηρίου.

Αρχικά ελέγχεται εαν υπάρχει cell, το οποίο φτάνοντας στον μεταγωγέα αποθηκεύτηκε στον Hash πίνακα εξαιτίας της έλλειψης εισιτηρίου προορισμού και το οποίο ανήκει στην ίδια ροή κυκλοφορίας με αυτήν που εκφράζεται από το εισιτήριο

που μόλις έφτασε. Τότε το εισερχόμενο εισιτήριο προορισμού καταναλώνεται και το cell προωθείται στην poolCr_x ουρά, όπου x ο σύνδεσμος εξόδου που θα χρησιμοποιηθεί για την έξοδό του στο δίκτυο. Κατόπιν ελέγχεται εαν υπάρχει cell στην poolCr ουρά του μεταγωγέα το οποίο περιμένει για εισιτήριο αναχώρησης. Εαν βρεθεί τέτοιο cell αυτό χρησιμοποιεί το εισιτήριο αναχώρησης για την μετάβαση του στην readyQ_x ουρά. Σε διαφορετική περίπτωση οι αντίστοιχοι μετρητές εισιτηρίων ενημερώνονται ανάλογα.

Τέλος, ο εξυπηρετητής 5 είναι υπεύθυνος για την αποστολή των cells στο δίκτυο. Μόλις τελειώσει η αποστολή ενός cell, ο εξυπηρετητής ελέγχει εαν υπάρχει κάποια επικεφαλίδα από cell, αναμένουσα στην ουρά εξόδου readyQ, που αντιστοιχεί στο σύνδεσμο που χρησιμοποιήθηκε. Εαν βρεθεί μια τέτοια επικεφαλίδα, δημιουργείται ξανά το cell, συνενώνοντας την επικεφαλίδα αυτή με την πληροφορία που βρίσκεται αποθηκευμένη στους ενταμιευτές του μεταγωγέα. Κατόπιν ξεκινά η αποστολή του cell στο δίκτυο. Επίσης, ο εξυπηρετητής 5 είναι υπεύθυνος για την αποστολή στον πιο πίσω στην σειρά μεταγωγέα των credits, που συνοδεύουν απαραίτητα την αποστολή κάθε cell στο δίκτυο.

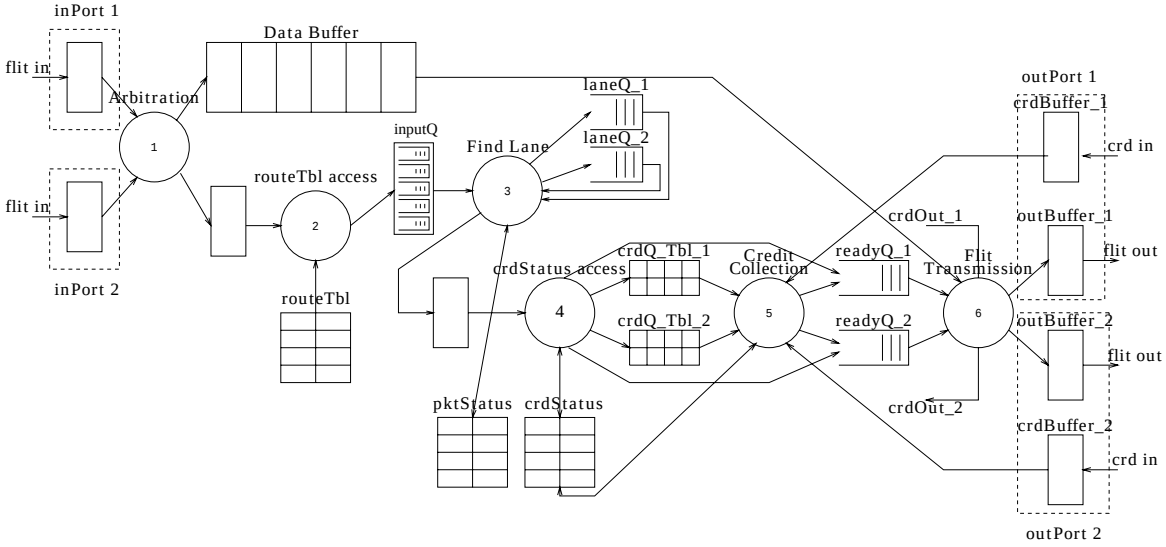
Περιγραφή μοντέλου μεταγωγέα Wormhole

Ως μοντέλο μεταγωγέα για την προσομοίωση του δικτύου Wormhole χρησιμοποιήθηκε και πάλι αυτό το οποίο αναπτύχθηκε κατά την μεταπτυχιακή εργασία του Ε. Σπυριδάκη [4]. Η σχεδίαση του μεταγωγέα έγινε με βάση το μοντέλο του μεταγωγέα ATM, το οποίο περιγράφεται στην προηγούμενη παράγραφο, έτσι ώστε ανάμεσα στους δύο τύπους μεταγωγέων να υπάρχουν όσο το δυνατόν λιγότερες διαφορές. Η ομοιότητα αυτή ανάμεσα στους δύο τύπους μεταγωγέων διευκολύνει τη σύγκριση των δύο τύπων δικτύων.

Το μοντέλο αποτελείται από έξι εξυπηρετητές, καθένας από τους οποίους εκτελεί λειτουργίες σε διάρκεια ενός κύκλου ρολογιού του μεταγωγέα.

Ο εξυπηρετητής 1 είναι υπεύθυνος για την παραλαβή των flits που φτάνουν στον μεταγωγέα. Η άφιξη ενός flit σε έναν μεταγωγέα συνοδεύεται από την εύρεση μιας άδειας θέσης στους ενταμιευτές του μεταγωγέα η οποία και χρησιμοποιείται για την αποθήκευση του flit. Παράλληλα μια σειρά από ενέργειες ελέγχου επιτελούνται από τους υπόλοιπους εξυπηρετητές.

Ο εξυπηρετητής 2 εξετάζει την επικεφαλίδα του flit και καθορίζει ποιος είναι ο



Σχήμα 4.4: Μοντέλο μεταγωγέα Wormhole 2×2

σύνδεσμος εξόδου που θα χρησιμοποιηθεί από αυτό κατά την έξοδο του στο δίκτυο.

Ο εξυπηρετητής 3 εξετάζει εαν υπάρχει ελεύθερη λωρίδα στον σύνδεσμο εξόδου από τον οποίο θέλει να βγεί το flit. Ο έλεγχος αυτός γίνεται μονάχα για το πρώτο flit ενός πακέτου, αφού μονάχα αυτό είναι υπεύθυνο για την εύρεση λωρίδας στον σύνδεσμο εξόδου. Εαν βρεθεί μια τέτοια λωρίδα αυτή χρησιμοποιείται κατόπιν από όλα τα flits που ανήκουν στο δεδομένο πακέτο. Διαφορετικά το flit μπλοκάρει και αποθηκεύεται στην ουρά lane_Q_x, όπου x ο σύνδεσμος εξόδου που έχει καθοριστεί από τον εξυπηρετητή 2. Τα υπόλοιπα flits του ιδίου πακέτου μπλοκάρουν και αυτά στον πίνακα inputQ, όπου και περιμένουν μέχρι την εύρεση μιας λωρίδας στον σύνδεσμο εξόδου.

Ο εξυπηρετητής 4 αποφασίζει για το τι θα συμβεί σε ένα flit, στο οποίο έχει αποδοθεί λωρίδα εξόδου. Στο δίκτυο wormhole η απόδοση λωρίδας εξόδου σε ένα πακέτο ισοδυναμεί με την κατανάλωση ενός εισιτηρίου αναχώρησης για το δίκτυο ATM. Συνεπώς για το δίκτυο Wormhole υπάρχει μονάχα ένας τύπος εισιτηρίων, τα εισιτήρια προορισμού. Ανάλογα λοιπόν από την υπαρξη ή μη ενός τέτοιου εισιτηρίου, ο εξυπηρετητής 4 τοποθετεί το flit,

- στην ουρά readyQ_x, όπου x ο σύνδεσμος εξόδου, εφόσον υπάρχει εισιτήριο αναχώρησης. Στην περίπτωση αυτή το εισιτήριο καταναλώνεται και το flit ετοιμάζεται για την αποστολή του στον επόμενο στη σειρά μεταγωγέα.
- στην ουρά crdQ_Tbl_x, όπου x ο σύνδεσμος εξόδου, εφόσον δεν υπάρχει εισιτήριο

αναχώρησης. Το flit θα παραμείνει εκεί μέχρι να φτάσει στον μεταγωγέα το απαραίτητο εισιτήριο. Η ουρά στην οποία αποθηκεύεται το flit σχετίζεται με τον σύνδεσμο εξόδου αλλά και την λωρίδα εξόδου που χρησιμοποιείται από το flit.

Ο εξυπηρετητής 5 είναι υπεύθυνος για τι ενέργειες που πρέπει να γίνουν στον μεταγωγέα όταν φτάνει σε αυτόν ένα νέο εισιτήριο. Εαν υπάρχει flit το οποίο περιμένει για αυτό το εισιτήριο, τότε αυτό καταναλώνεται και το flit μεταφέρεται από την ουρά crdQ_Tbl_x στην οποία ήταν αποθηκευμένο ενόσω περίμενε για την άφιξη του εισιτηρίου, στην ουρά readyQ_x. Διαφορετικά ο αντίστοιχος μετρητής εισιτηρίων που αντιστοιχεί στην λωρίδα εξόδου για την οποία ήρθε το εισιτήριο αυξάνει.

Τέλος ο εξυπηρετητής 6 είναι υπεύθυνος για την αποστολή των flits στο δίκτυο. Η αποστολή κάθε flit συνοδεύεται από την δημιουργία και αποστολή ενός εισιτηρίου στον προηγούμενο στη σειρά μεταγωγέα.

Κεφάλαιο 5

Πειραματικά αποτελέσματα

Εγιναν πειράματα για τρείς διαφορετικούς τύπους δικτύων. Προσομοιώθηκε και μελετή0ηκε η συμπεριφορά ενός δικτύου Banyan Butterfly, ενός εξαδιάστατου υπερκύβου (hypercube) και ενός διδιάστατου mesh. Τα πειράματα έγιναν για τοπολογίες 64 εισόδων εξόδων, όσοι δηλαδή και οι επεξεργαστές που χρησιμοποιήθηκαν κατά την εκτέλεση των παράλληλων εφαρμογών.

Το μέσο μέγεθος του πακέτου δεν είναι σταθερό, αλλά κυμαίνεται ανάλογα με την εφαρμογή από μερικές δεκάδες μέχρι μερικές εκατοντάδες cells/flits. Σε όλες τις ομάδες πειραμάτων, κάθε μεταγωγέας του δικτύου έχει 2 εισόδους και 2 εξόδους.

Ενα μέγεθος, το οποίο μελετήθηκε ήταν ο συνολικός χρόνος εκτέλεσης της εφαρμογής, δηλαδή η συνολική διάρκεια της προσομοίωσης. Αλλα μεγέθη, τα οποία μετρήθηκαν είναι:

- Καθυστέρηση δικτύου (In network time)
  Πρόκειται για το συνολικό χρόνο, που χρειάζεται ένα cell/flit από τη στιγμή που εισέρχεται στο δίκτυο μέχρι τη στιγμή που φτάνει στον προορισμό του. Είναι δηλαδή ο χρόνος που απαιτείται για την μετάδοση του cell/flit μέσω των μεταγωγέων και των συνδέσμων που συνδέουν αυτούς.

- Χρόνος παραμονής ενός πακέτου στους ενταμιευτές του παραγωγού (Last cell/flit injection time)
  Πρόκειται για τον χρόνο από τη στιγμή της δημιουργίας ενός πακέτου μέχρι τη

στιγμή που το τελευταίο cell/flit του εισέρχεται στο δίκτυο.

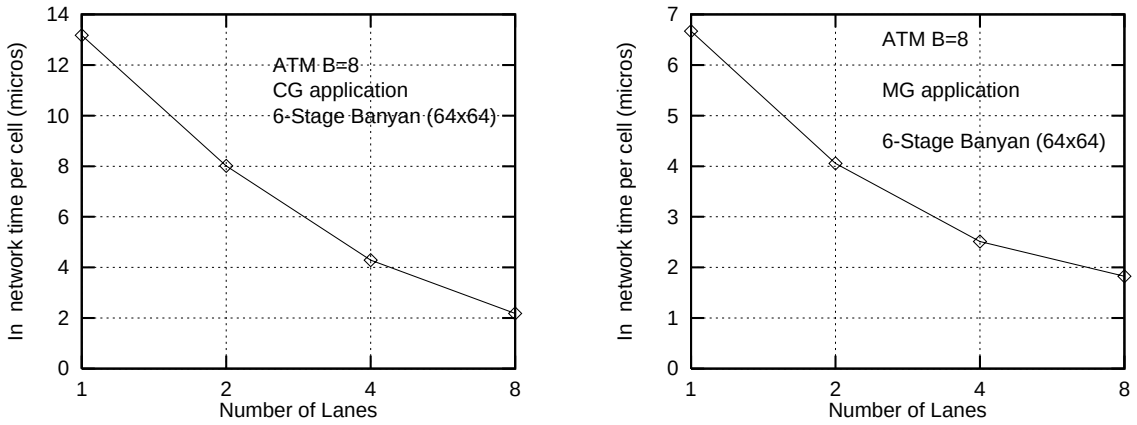
- Χρόνος άφιξης ενός πακέτου στον προορισμό του.
  Πρόκειται για τον χρόνο από τη στιγμή της δημιουργίας του πακέτου μέχρι τη στιγμή που το τελευταίο του cell/flit και επομένως ολόκληρο το πακέτο, φτάνει στον καταναλωτή. Η άφιξη ενός πακέτου στον προορισμό του δεν συνοδεύεται απαραίτητα από την άμεση παραλαβή του πακέτου αυτού από τον καταναλωτή. Η σειρά με την οποία γίνεται η αποστολή και η λήψη των μηνυμάτων κατά τη διάρκεια της προσομοίωσης καθορίζεται αυστηρά από την πληροφορία η οποία είναι καταγεγραμμένη στο trace file. Ετσι, παραδείγματος χάριν εαν το trace file προσδιορίζει πως ο καταναλωτής θα πρέπει να παραλάβει κάποιο άλλο πακέτο πρώτα, τότε το πακέτο που μόλις έφτασε στον καταναλωτή θα πρέπει να περιμένει στους ενταμιευτές του μέχρι να έρθει η στιγμή της παραλαβής του, όπως αυτή καθορίζεται από το trace file.

- Χρόνος παραλαβής πακέτου
  Πρόκειται για το χρόνο από τη στιγμή της δημιουργίας του πακέτου, (η οποία προσδιορίζεται από την ύπαρξη στο trace file μιας MPI εντολής αποστολής πληροφορίας στο δίκτυο) μέχρι τη στιγμή που αυτό παραλαμβάνεται από τον καταναλωτή (η οποία καθορίζεται από μια MPI εντολή παραλαβής). Στην παράμετρο αυτή περιλαμβάνονται ο χρόνος που καταναλώνει το πακέτο στους ενταμιευτές του παραγωγού περιμένοντας να ολοκληρωθεί η μετάδοση του μέσα στο δίκτυο, ο χρόνος που περνάει μέσα στο δίκτυο κατά την πορεία του προς τον καταναλωτή καθώς τέλος και ο χρόνος που παραμένει στους ενταμιευτές του καταναλωτή.

5.1 Πειράματα σε δίκτυο Butterfly 64 εισόδων εξόδων

5.1.1 Σε δίκτυο ATM

Σε αυτή την σειρά πειραμάτων το δίκτυο του οποίου η συμπεριφορά μελετάται είναι ένα δίκτυο Banyan Butterfly με 64 εισόδους και ισάριθμες εξόδους, το οποίο αποτελείται από μεταγωγείς 2×2. Για όλα τα πειράματα, το μέγεθος του χώρου αποθήκευσης, που είναι αφιερωμένο ανά σύνδεσμο εξόδου είναι σταθερό και ίσο με 8 cells. Το πλήθος των



Σχήμα 5.1: Καθυστέρηση δικτύου, ανά cell για δύο εφαρμογές

λωρίδων είναι 1,2,4 και 8.

Στα γραφήματα του σχήματος 5.1, απεικονίζεται η μέση καθυστέρηση στο δίκτυο, ανά cell, για τις εφαρμογές CG και MG. Η καθυστέρηση αυτή μετριέται σε μs. Από τις γραφικές παραστάσεις φαίνεται πως ο χρόνος που απαιτείται για την μετάδοση ενός cell στους συνδέσμους του δικτύου και μέσα στους μεταγωγείς μειώνεται καθώς αυξάνει ο αριθμός των λωρίδων που αντιστοιχούν σε κάθε σύνδεσμο. Αυτό το φαινόμενο παρατηρείται σε όλες τις εφαρμογές που προσομοιώθηκαν.

Ενα cell, εισερχόμενο στο δίκτυο, μέχρι να φτάσει στον προορισμό του καταναλώνει χρόνο πάνω στους συνδέσμους και μέσα στους μεταγωγείς του δικτύου. Εξαιτίας της τοπολογίας του δικτύου, ο χρόνος πάνω στους συνδέσμους του δικτύου είναι σταθερός και ίδιος για όλα τα cells. Συνεπώς, αύξηση του αριθμού των λωρίδων οδηγεί σε μείωση του μέσου χρόνου παραμονής των cells μέσα στους μεταγωγείς.

Η χρήση μεγάλου αριθμού λωρίδων (L) συνεπάγεται μείωση του αριθμού των cells που κατευθύνονται προς τον ίδιο προορισμό και βρίσκονται μέσα στους ενταμιευτές ενός μεταγωγέα περιμένοντας να βγουν στο δίκτυο (αφού από τη σχέση $b = B/L$ φαίνεται πως το b είναι αντιστρόφως ανάλογο με το L). Αρα όταν έχουμε μεγάλο αριθμό λωρίδων στους μεταγωγείς συσσωρεύονται λιγότερα cells τα οποία και μεταδίδονται πιο γρήγορα προς τον επόμενο μεταγωγέα της σύνδεσης.

Με βάση τις γραφικές παραστάσεις που απεικονίζουν την μεταβολή της καθυστέρησης δικτύου σε συνάρτηση με τον αριθμό των λωρίδων, θα περίμενε κανείς να δει ανάλογα αποτελέσματα και στις γραφικές παραστάσεις της απεικόνισης του συνολικού χρόνου εκτέλεσης των εφαρμογών. Ομως κάτι τέτοιο δεν επιβεβαιώνεται από τα αποτελέσματα της προσομοίωσης των 10 εφαρμογών. Αρκετά συχνά για 8 lanes έχουμε

	Application	1 lane	2 lanes	4 lanes	8 lanes
1	**ft**	1.402	1.326	1.404	1.490
2	**mg**	2.089	2.245	2.288	2.432
3	**lu**	17.5	16.8	17.1	16.95
4	**ep**	25.5	24.9	24.8	25.1
5	**cg**	12.47	11.32	11.9	12.3
6	**is**	8.5	8.3	8.6	8.01
7	**pstw**	4.27	4.55	4.46	4.63
8	**qr**	1.10	1.086	1.079	1.07
9	**sp**	20.60	19.156	19.67	19.07
10	**bt**	8.0168	7.96	7.864	8.4

Σχήμα 5.2: Μεταβολή του συνολικού χρόνου εκτέλεσης των εφαρμογών

τον χειρότερο χρόνο εκτέλεσης (εφαρμογές FFT και MG), ενώ για τις περιπτώσεις που ο καλύτερος χρόνος εκτέλεσης επιτυγχάνεται για 8 λωρίδες (εφαρμογή IS) η διαφοροποίηση που παρατηρείται ανάμεσα στον καλύτερο και τον χειρότερο χρόνο είναι πολύ μικρή, σε σχέση με συτήν που παρατηρήθηκε στην καθυστέρηση του δικτύου. Γενικά για όλες τις εφαρμογές παρατηρείται μικρή διαφοροποίηση στους χρόνους εκτέλεσης.

Ο συνολικός χρόνος εκτέλεσης της εφαρμογής εξαρτάται από την ταχύτητα με την οποία λαβαίνουν χώρα τα γεγονότα στους επεξεργαστές. Οσο πιο γρήγορα εξελίσσονται οι αποστολές και οι λήψεις των μηνυμάτων που αναλογούν σε κάθε επεξεργαστή, τόσο πιο γρήγορα τερματίζει η προσομοίωση της εφαρμογής και ανάλογα πιο μικρός είναι ο χρόνος εκτέλεσης που παρατηρείται. Αρκεί λοιπόν να αναλυθεί ο μέσος χρόνος άφιξης του πακέτου, για να εξηγηθούν τα αποτελέσματα που προκύπτουν.

Στο δίκτυο, ο δρόμος τον οποίο ακολουθούν τα cells που πηγάζουν από συγκεκριμένο επεξεργαστή και κατευθύνονται σε συγκεκριμένο προορισμό είναι μοναδικός. Επιπλέον, εξαιτίας της κατασκευής του μεταγωγέα, ένα cell, το οποίο φτάνει σε ένα μεταγωγέα και κατευθύνεται προς ορισμένο προορισμό θα βγεί στο δίκτυο πιο γρήγορα από ένα cell που κατευθύνεται προς τον ίδιο προορισμό αλλά έφτασε αργότερα στον μεταγωγέα.

Με βάση τα παραπάνω ισχύει η σχέση:

$$Average \quad packet \quad arrival \quad time = Last \quad Cell \quad Injection \quad time + In \quad network \quad time$$

Στη σχέση αυτή, ο δεύτερος προσθετέος είναι πάντοτε φθίνουσα συνάρτηση του αριθμού των λωρίδων που αντιστοιχούν σε ένα σύνδεσμο. Ο πρώτος παράγοντας όμως δεν

	Application	1 lane	2 lanes	4 lanes	8 lanes
1	**ft**	29.475	28.678	31.879	34.567
2	**mg**	37.089	38.245	40.88	40.97
3	**lu**	31.026	31.96	31.3	32.95
4	**ep**	151.5	149.9	152.8	153.1
5	**cg**	454.47	451.32	464.9	469.3
6	**is**	12.786	12.334	12.765	12.365
7	**pstw**	11.978	10.456	11.54	12.63
8	**qr**	56.97	56.92	56.643	56.97
9	**sp**	16.673	15.156	15.67	16.07
10	**bt**	24.78	22.627	23.864	23.9

Σχήμα 5.3: Μεταβολή του μέσου χρόνου παραμονής των πακέτων στους ενταμιευτές του παραγωγού

έχει τόσο προβλέψιμη συμπεριφορά. Ο πίνακας του σχήματος 5.3 δείχνει την μεταβολή του χρόνου που παραμένει ένα πακέτο στους ενταμιευτές του παραγωγού, σε συνάρτηση με τον αριθμό των λωρίδων για όλες τις εφαρμογές.

Ο χρόνος παραμονής ενός πακέτου στους ενταμιευτές του παραγωγού επηρεάζεται από τους ακόλουθους παράγοντες:

- **Το μέγεθος του πακέτου σε cells**
  Ενα πακέτο με μικρό μέγεθος εισέρχεται πιο γρήγορα στο δίκτυο από ότι ένα μεγάλο πακέτο.

- **Το throughput του δικτύου**
  Οσο πιο μεγάλο το throughput του δικτύου, τόσο πιο γρήγορα εισέρχεται ολόκληρο το πακέτο στο δίκτυο.

- **Ο αριθμός των λωρίδων που αντιστοιχούν στον σύνδεσμο εισόδου**
  Μικρός αριθμός λωρίδων επιτρέπει πιο πολλά cells του ιδίου πακέτου μέσα στον μεταγωγέα και άρα επιταχύνει την είσοδο του πακέτου στο δίκτυο.

- **Χρόνος παραμονής των cells στον πρώτο μεταγωγέα του δικτύου**
  Οσο πιο γρήγορα αναχωρούν τα cells του πακέτου από τον πρώτο μεταγωγέα, τόσο πιο γρήγορα εισέρχεται το πακέτο στο δίκτυο.

Οι πρώτοι δύο παράγοντες είναι ανεξάρτητοι του αριθμού των λωρίδων, που χρησιμοποιούνται. Το throughput του δικτύου είναι 622 Mbits/sec για όλα τα πειράματα

που έγιναν, ενώ το μέγεθος του πακέτου εξαρτάται από την εφαρμογή.

Κατά την πορεία του μέσα από το δίκτυο, ένα cell διέρχεται πάντοτε από 6 στάδια, ένα για κάθε μεταγωγέα. Μετρήθηκε ο μέσος χρόνος παραμονής του cell, στα διάφορα στάδια της διαδρομής. Στο σχήμα 5.4 απεικονίζονται οι χρόνοι παραμονής ανά cell στα 6 στάδια του δικτύου για τις διάφορες τιμές των λωρίδων για την εφαρμογή CG. Οι χρόνοι είναι μετρημένοι σε μs.

Lanes	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 4**	**Stage 5**	**Stage 6**
1	4.591	2.514	2.074	1.654	1.391	0.803
2	2.261	1.742	1.423	1.101	0.887	0.455
4	1.155	0.911	0.767	0.586	0.470	0.256
8	0.576	0.468	0.397	0.310	0.251	0.154

Σχήμα 5.4: Χρόνοι παραμονής ανά cell στα διάφορα στάδια του δικτύου για την εφαρμογή CG

Με ανάλογο τρόπο μεταβάλλονται οι χρόνοι παραμονής των cells στους μεταγωγείς του δικτύου και για τις υπόλοιπες εφαρμογές. Οπως φαίνεται δεδομένου ενός αριθμού λωρίδων, ο χρόνος που περνάει ένα cell στο πρώτο στάδιο είναι πάντοτε μεγαλύτερος από τους χρόνους που περνάει το cell στα επόμενα στάδια της διαδρομής. Επίσης παρατηρήθηκε ότι καθώς το cell πλησιάζει προς τον καταναλωτή ο χρόνος αυτός μειώνεται.

Το φαινόμενο αυτό οφείλεται στις ιδιαιτερότητες της τοπολογίας του δικτύου. Οσο πιο κοντά προς τον παραγωγό βρισκόμαστε - αρχικά στάδια - στους μεταγωγείς συσσωρεύονται cells από πακέτα, τα οποία δεν κατευθύνονται απαραίτητα προς τον ίδιο προορισμό αλλά θέλουν να βγούν στο δίκτυο από τον ίδιο σύνδεσμο. Εχουμε λοιπόν διαμάχη για τον ίδιο σύνδεσμο εξόδου, η οποία οδηγεί σε αύξηση του χρόνου παραμονής μέσα στον μεταγωγέα. Το φαινόμενο αυτό επαναλαμβάνεται και στα επόμενα στάδια της διαδρομής, αλλά σε μικρότερη κλίμακα αφού i)εξαιτίας της τοπολογίας του δικτύου, η πιθανότητα 2 cells, τα οποία κατευθύνονται προς διαφορετικό προορισμό και εισέρχονται από διαφορετική είσοδο να θέλουν να χρησιμοποιήσουν τον ίδιο σύνδεσμο εξόδου μειώνεται και ii)cells, τα οποία στο προηγούμενο στάδιο συνυπήρχαν στον ίδιο μεταγωγέα και θέλουν και σε αυτό το στάδιο να χρησιμοποιήσουν τον ίδιο σύνδεσμο εξόδου, εισέρχονται στον μεταγωγέα σειριακά, το ένα μετά το άλλο.

Για το δίκτυο Butterfly 64 εισόδων, έχουμε 6 στάδια, δηλαδή ένα cell θα πρέπει να

περάσει από 6 μεταγωγείς πριν να φτάσει στον προορισμό του. Οσο αφορά στους χρόνους παραμονής στους μεταγωγείς των 4 πρώτων σταδίων, οι χρόνοι πάντοτε μειώνονται, καθώς απομακρυνόμαστε από τον παραγωγό και προχωράμε σε μεγαλύτερο στάδιο. Η φθίνουσα αυτή πορεία δεν παρατηρείται πάντοτε στα στάδια 5 και 6. Είναι δυνατόν, για ορισμένες εφαρμογές να έχουμε μεγαλύτερο χρόνου παραμονής των cells στο πέμπτο στάδιο, από το τέταρτο. Αυτό συμβαίνει, όταν έχουμε κίνηση προς γειτονικούς καταναλωτές, η οποία προέρχεται από μακρινούς παραγωγούς. Τότε τα πακέτα φτάνουν στον προορισμό τους διασχίζοντας μονοπάτια, τα οποία ταυτίζονται προς το τέλος της διαδρομής. Δηλαδή η κίνηση συσσωρεύεται στα τελευταία στάδια του δικτύου, για αυτό το λόγο και η παρατηρούμενη αύξηση στους χρόνους παραμονής από το στάδιο 4 στο στάδιο 5.

Αρα, για δεδομένο αριθμό λωρίδων ο χρόνος παραμονής στο πρώτο στάδιο είναι πάντοτε μεγαλύτερος από τους χρόνους παραμονής στα επόμενα στάδια, γεγονός το οποίο επηρεάζει αρνητικά τον μέσο χρόνο παραμονής στους ενταμιευτές των παραγωγών.

Επιστρέφοντας στη σχέση

$$Average\ \ packet\ \ arrival\ \ time = Last\ \ Cell\ \ Injection\ \ time + In\ \ network\ \ time$$

παρατηρούμε ότι ο μέσος χρόνος άφιξης ενός πακέτου στον προορισμό του, αναλύεται σε 2 συνιστώσες εκ των οποίων η μια, η καθυστέρηση στο δίκτυο, μειώνεται πάντοτε εφόσον αυξάνει ο αριθμός των λωρίδων ενώ η δεύτερη, ο μέσος χρόνος παραμονής του πακέτου στους ενταμιευτές των παραγωγών, συμπεριφέρεται με απροσδιόριστο τρόπο. Οπως φαίνεται από τις μετρήσεις που έγιναν πάνω στο δείγμα των εφαρμογών, η συνιστώσα αυτή αποτελεί τον κύριο παράγοντα καθυστέρησης για την άφιξη του πακέτου στον παραγωγό.

Είναι λοιπόν φανερό ότι ο μέσος χρόνος καθυστέρησης των πακέτων στους ενταμιευτές του παραγωγού επηρεάζει αρνητικά το συνολικό χρόνο αποστολής του πακέτου και εκμηδενίζει τα όποια θετικά οφέλη υπάρχουν από το γρήγορο πέρασμα των cells μέσα από το δίκτυο.

Με βάση τα παραπάνω, θα περίμενε κανείς να υπάρχει πάντοτε για όλες τις εφαρμογές αναλογία ανάμεσα στους χρόνους άφιξης των πακέτων και στον συνολικό χρόνο εκτέλεσης της εφαρμογής. Δηλαδή, για δεδομένη εφαρμογή, εαν για κάποιο αριθμό λωρίδων σημειώνεται μικρότερος χρόνος άφιξης των πακέτων στον καταναλωτή από ότι για κάποιον άλλο αριθμό λωρίδων, θα ήταν αναμενόμενο η βελτίωση αυτή να σημειωθεί

και στους χρόνους εκτέλεσης της εφαρμογής. Αυτό όμως δεν παρατηρείται πάντοτε.

Ενα μήνυμα, το οποίο φτάνει στον προορισμό του δεν είναι απαραίτητο ότι θα ”παραληφθεί” αμέσως από τον καταναλωτή. Η σειρά με την οποία γίνεται η αποστολή και λήψη των μηνυμάτων προσδιορίζεται αυστηρά από την πληροφορία, η οποία έχει καταγραφεί κατά τη διάρκεια της εκτέλεσης της εφαρμογής. Επιπλέον η αποστολή και η λήψη ενός μηνύματος (send και receive) είναι γεγονότα αμοιβαία αποκλειστικά, δεν συμβαίνουν ποτέ ταυτόχρονα σε έναν επεξεργαστή. Δηλαδή, ενόσω κάποιος παραγωγός στέλνει μηνύματα στο δίκτυο, δεν μπορεί να λαμβάνει μηνύματα, μπορεί μόνο να αποθηκεύει τα μηνύματα που φτάνουν στους ενταμιευτές του. Αντίστοιχα, ενώ κάποιος επεξεργαστής περιμένει να λάβει κάποιο μήνυμα, δεν μπορεί να σημειωθεί αποστολή κάποιου πακέτου στο δίκτυο. Επίσης, επειδή όλα τα γεγονότα τα οποία συμβαίνουν σε έναν επεξεργαστή γίνονται σειριακά, ένα γεγονός δεν μπορεί να εκτελεστεί, εφόσον δεν έχουν τελειώσει όλα τα γεγονότα τα οποία ήταν καταγεγραμμένα πρίν από αυτό. Για παράδειγμα, η σειρά με την οποία γίνεται η λήψη των μηνυμάτων καθορίζεται από το trace file της εφαρμογής, δηλαδή ένα μήνυμα Α θα ληφθεί πριν από ένα μήνυμα Β, εφόσον κάτι τέτοιο καθορίζεται από το trace file ακόμη και αν το Β φτάσει στον καταναλωτή πιο πριν από το Α.

Ετσι, ένα πακέτο φτάνοντας στον καταναλωτή, μπορεί να περιμένει για αρκετή ακόμη ώρα στους ενταμιευτές του, γιατί:

- ο επεξεργαστής στέλνει πληροφορία στο δίκτυο. Το πακέτο, που μόλις έφτασε θα πρέπει να περιμένει να τελειώσει η αποστολή της πληροφορίας από τον επεξεργαστή. Σε αυτήν την περίπτωση, δεν υπάρχουν οφέλη από την γρήγορη αφιξή του πακέτου, όσο αφορά στον χρόνο εκτέλεσης της εφαρμογής.
- ο επεξεργαστής είναι ανενεργός, περιμένοντας τη λήψη ενός συγκεκριμένου πακέτου, η οποία καθυστερεί. Αυτό σημαίνει καθυστέρηση, όχι μονάχα στην εξέλιξη των γεγονότων για το συγκεκριμένο επεξεργαστή, αλλά και για άλλους επεξεργαστές, εφόσον στην ακολουθία των γεγονότων υπάρχουν αποστολές πακέτων, οι οποίες και καθυστερούν.

Το φαινόμενο αυτό, να παραμένει δηλαδή ένα πακέτο στους ενταμιευτές του προορισμού του, παρατηρείται συχνά και φανερώνει ότι κατά τη διάρκεια της εκτέλεσης της εφαρμογής οι κόμβοι του συστήματος δεν είναι όλοι εξίσου γρήγοροι στην εκτέλεση των υπολογισμών αλλά και στην αποστολή και λήψη των διαφόρων μηνυμάτων. Κάποιοι

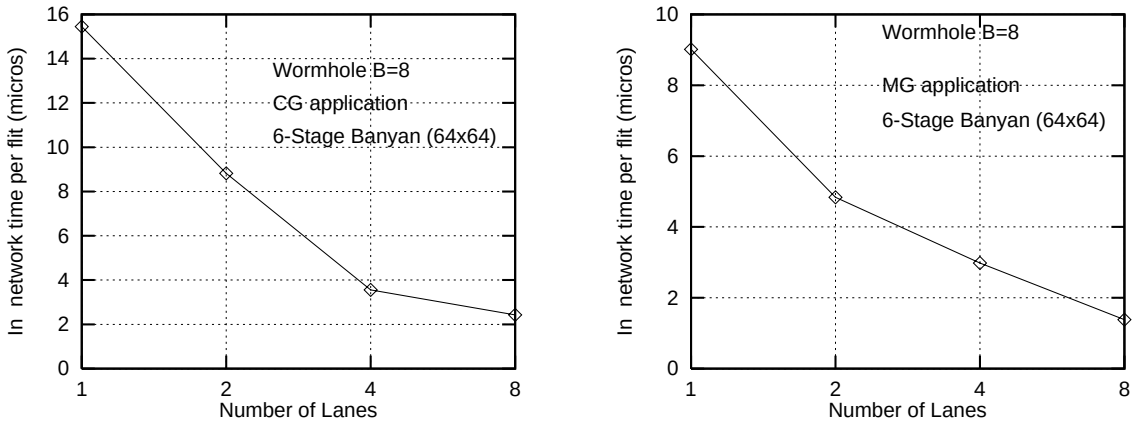
από τους επεξεργαστές οφελούνται σε μεγαλύτερο βαθμό από τις συνθήκες του δικτύου και εκτελούν τα γεγονότα που τους αναλογούν με μεγαλύτερη ταχύτητα. Αυτοί όμως αναγκάζονται τελικά να περιμένουν τους πιο αργούς, με αποτέλεσμα όσα οφέλη σημειώθηκαν κατά τη διάρκεια της εκτέλεσης να εξαλείφονται τελικά. Οπως φαίνεται στο παράλληλο πρόγραμμα συχνά κάποια γεγονότα δρουν ως ”φράχτες” περιορίζοντας την ταχύτητα της εκτέλεσης του προγράμματος.

5.1.2 Πειράματα σε δίκτυο Wormhole

Η ίδια σειρά πειραμάτων επαναλήφθηκε και πάλι για δίκτυο Butterfly 64 εισόδων εξόδων, η αρχιτεκτονική των μεταγωγέων του οποίου ήταν αυτήν την φορά wormhole. Το μέγεθος του χώρου αποθήκευσης είναι και πάλι σταθερό και ίσο με 8 flits ενώ το πλήθος των λωρίδων κυμαίνεται από 1 εως 8 για όλες τις δυνάμεις του 2 μέσα σε αυτά τα όρια.

Μετρήθηκε η καθυστέρηση του δικτύου ανά flit και μελετήθηκε η μεταβολή της. Για όλες τις εφαρμογές που προσομοιώθηκαν, παρατηρήθηκε πως η καθυστέρηση του δικτύου ανά flit μειώνεται καθώς αυξάνει ο αριθμός των λωρίδων που αντιστοιχεί σε κάθε σύνδεσμο. Οπως φαίνεται Στο σχήμα 5.5, όπου απεικονίζεται η καθυστέρηση του δικτύου για δύο εφαρμογές (CG και MG), η μείωση είναι πάντοτε εντονότερη καθώς μεταβαίνουμε από μία λωρίδα σε δύο. Αυτό οφείλεται στις αρνητικές συνέπειες που παρατηρούνται όταν μία μόνο λωρίδα είναι αφιερωμένη σε κάθε σύνδεσμο. Στην περίπτωση αυτή είναι αρκετά συχνό το φαινόμενο ένας σύνδεσμος να παραμένει ανενεργός επειδή το πακέτο το οποίο έχει καταλάβει την μνήμη του μεταγωγέα στον οποίο αντιστοιχεί ο σύνδεσμος αυτός έχει μπλοκαριστεί. Ο σύνδεσμος αυτός θα μπορούσε να χρησιμοποιηθεί από flits άλλων πακέτων, τα οποία βρίσκονται ήδη στο δίκτυο και τα οποία κατευθύνονται προς διαφορετικούς προορισμούς. Ομως ο τρόπος με τον οποίο λειτουργούν οι λωρίδες στα δίκτυα wormhole, δεν επιτρέπει στα flits των άλλων πακέτων τα οποία βρίσκονται στο δίκτυο να προχωρήσουν στο δεδομένο μεταγωγέα και να αξιοποιήσουν τον ανενεργό σύνδεσμο εξόδου.

Εξετάζοντας το συνολικό χρόνο εκτέλεσης των εφαρμογών (σχήμα 5.6) και πάλι παρατηρούμε ότι δεν υπάρχει αντιστοιχία ανάμεσα στον τρόπο που αυτός μεταβάλεται και στον τρόπο με τον οποίο μεταβάλλεται η καθυστέρηση στο δίκτυο. Οπως φαίνεται και στο σχήμα 5.6 για καμμία εφαρμογή δεν παρατηρείται ο καλύτερος



Σχήμα 5.5: Καθυστέρηση δικτύου για τις εφαρμογές CG και MG

	Application	1 lane	2 lanes	4 lanes	8 lanes
1	**ft**	1.304	1.093	1.131	1.277
2	**mg**	2.108	1.99	2.01	2.113
3	**lu**	16.34	16.45	16.9	16.8
4	**ep**	25.3	24.4	24.7	24.9
5	**cg**	12.56	11.56	11.89	11.67
6	**is**	8.556	8.0	7.95	7.64
7	**pstw**	4.9	3.90	3.85	3.56
8	**qr**	1.016	0.998	1.0129	1.0074
9	**sp**	21.8	18.56	18.54	18.6
10	**bt**	8.656	7.138	7.354	7.4

Σχήμα 5.6: Μεταβολή του χρόνου εκτέλεσης των εφαρμογών για δίκτυο Wormhole

χρόνος εκτέλεσης για 8 λωρίδες. Ομως εδώ τα αποτελέσματα παρουσιάζουν για όλες τις εφαρμογές τον καλύτερο χρόνο εκτέλεσης για 2 λωρίδες ενώ ο χειρότερος χρόνος εκτέλεσης παρατηρείται για 8 λωρίδες.

Ο συνολικός χρόνος εκτέλεσης μιας εφαρμογής που εκτελείται πάνω από δίκτυο wormhole, εξαρτάται από την ταχύτητα με την οποία λαβαίνουν χώρα τα γεγονότα στους επεξεργαστές. Ο σημειωθείς χρόνος εκτέλεσης είναι μικρότερος εφόσον οι αποστολές και οι λήψεις των μηνυμάτων που αναλογούν σε κάθε επεξεργαστή είναι συντομότερες. Για τον χρόνο παραλαβής ενός πακέτου ισχυεί ως γνωστόν η σχέση:

$$Average \quad packet \quad arrival \quad time = Last \quad Flit \quad Injection \quad time + In \quad network \quad time$$

όπου με τον όρο Last flit Injection Time εκφράζεται ο χρόνος παραμονής ενός πακέτου στους ενταμιευτές του παραγωγού.

	Application	1 lane	2 lanes	4 lanes	8 lanes
1	**ft**	35.422	24.773	27.795	28.46
2	**mg**	38.10	40.67	41.9	44.10
3	**lu**	29.34	28.45	29.6	29.8
4	**ep**	154.3	148.4	150.7	152.9
5	**cg**	500.67	440.6	455.9	470.9
6	**is**	13.6	12.98	13.05	13.12
7	**pstw**	12.67	7.90	8.15	8.56
8	**qr**	51.189	49.98	51.984	52.1534
9	**sp**	17.8	14.56	15.54	15.24
10	**bt**	24.76	21.324	21.987	21.884

Σχήμα 5.7: Μεταβολή του χρόνου παραμονής των πακέτων στους ενταμιευτές για όλες εφαρμογές

Από τα πειράματα που έγιναν έγινε φανερό πως ο πρώτος προσθετέος του παραπάνω αθροίσματος δεν έχει εξίσου προβλέψιμη και σταθερή συμπεριφορά με το δεύτερο προσθετέο. Στα γραφήματα του σχήματος 5.7 απεικονίζεται για δύο εφαρμογές η μεταβολή του χρόνου παραμονής ενός πακέτου στους ενταμιευτές του παραγωγού και ο τρόπος με τον οποίο ο χρόνος αυτός μεταβάλλεται συναρτήσει του αριθμού των λωρίδων.

Ο χρόνος παραμονής ενός πακέτου στους ενταμιευτές του παραγωγού επηρεάζεται από τους ακόλουθους παράγοντες:

- **Το μέγεθος του πακέτου σε flits**
  Οσο πιο μικρό το μέγεθος του πακέτου τόσο πιο μικρή και η παράμετρος Last flit injection time.

- **Το throughput του δικτύου**
  Μεγαλύτερο throughput προκαλεί πιο γρήγορη είσοδο των πακέτων στο δίκτυο

- **Ο αριθμός των λωρίδων που αντιστοιχούν στους σύνδεσμους των μεταγωγέων**
  Η παράμετρος αυτή μπορεί ανάλογα με τις συνθήκες να επηρεάσει άλλοτε θετικά και άλλοτε αρνητικά την μεταβολή του χρόνου παραμονής των πακέτων στους ενταμιευτές των παραγωγών.

Οπως φαίνεται και από τις γραφικές παραστάσεις του σχήματος 5.7 για μία λωρίδα μπορεί να επιτευχθεί άλλοτε ο καλύτερος και άλλοτε ο χειρότερος Last flit injection time. Οταν έχουμε μόνο μια λωρίδα τότε ολόκληρος ο χώρος μνήμης του μεταγωγέα

χρησιμοποιείται από flits του πακέτου, στο οποίο έχει ανατεθεί η λωρίδα. Αυτό μπορεί να οδηγήσει σε γρήγορη διάδοση του πακέτου μέσα από το δίκτυο.

Εαν όμως κατά την προώθηση του πακέτου μέσα από τους μεταγωγείς του δικτύου κάποιο άλλο πακέτο το οποίο θέλει για δεδομένο μεταγωγέα να χρησιμοποιήσει τον ίδιο σύνδεσμο εξόδου (ανεξάρτητα από το εαν το πακέτο αυτό κατευθύνεται προς τον ίδιο προορισμό με το πρώτο ή όχι) του διακόψει το δρόμο, τότε το πρώτο πακέτο εγκλωβίζεται μέσα στο δίκτυο αλλά και μέσα στους ενταμιευτές του παραγωγού του, περιμένοντας μέχρι να ολοκληρωθεί η μετάδοση του πακέτου που το έχει αποκλείσει. Σε αυτήν την περίπτωση ο χρόνος παραμονής στους ενταμιευτές του παραγωγού που σημειώνεται είναι και ο χειρότερος.

Αυξάνοντας τον αριθμό των λωρίδων σε 4 ή 8, όπως φαίνεται και από τις γραφικές παραστάσεις παρατηρούμε ότι αυξάνεται η τιμή της παραμέτρου Last flit injection time αφού ο χώρος αποθήκευσης που αφιερώνεται σε κάθε πακέτο στους μεταγωγείς είναι πλέον μικρότερος.

Παρά το γεγονός ότι ο χρόνος παραμονής των πακέτων στους ενταμιευτές των παραγωγών μεταβάλλεται με απροσδιόριστο τρόπο, η μεταβολή του συνολικού χρόνου εκτέλεσης των εφαρμογών φαίνεται να έχει πιο προβλέψιμη συμπεριφορά. Για όλες τις εφαρμογές που προσομοιώθηκαν ο χειρότερος χρόνος εκτέλεσης σημειώνεται για τα πειράματα που έγιναν για μία λωρίδα, ενώ για δύο και περισσότερες λωρίδες οι παρατηρούμενες διακυμάνσεις του χρόνου εκτέλεσης των εφαρμογών ήταν ελάχιστες. Στην πλειοψηφία των εφαρμογών ο καλύτερος χρόνος εκτέλεσης σημειώνεται για τα πειράματα που έγιναν με δύο λωρίδες.

Το φαινόμενο αυτό μπορεί να εξηγηθεί ανατρέχοντας στους τύπους επικοινωνίας που παρατηρούνται στο δείγμα των εφαρμογών και οι οποίο παρουσιάστηκαν σε προηγούμενο κεφάλαιο. Κατά τη διάρκεια εκτέλεσης μιας εφαρμογής οι επεξεργαστές του συστήματος χωρίζονται σε ομάδες και ο κάθε επεξεργαστής επικοινωνεί αποκλειστικά με μέλη της ομάδας του. Αυτό σημαίνει πως ένας επεξεργαστής άλλοτε στέλνει κάποιο μήνυμα σε κάποιο κόμβο της ομάδας του και άλλοτε περιμένει να λάβει μήνυμα από κάποιον επεξεργαστή της ίδιας πάντοτε ομάδας. Αυτό πρακτικά σημαίνει πως στους μεταγωγείς του δικτύου σπάνια βρίσκονται ταυτόχρονα περισσότερα από δύο διαφορετικά πακέτα, τα οποία κατευθύνονται προς τον ίδιο ή γειτονικούς προορισμούς και επομένως η ύπαρξη 4 ή περισσοτέρων λωρίδων δεν επιφέρει καμμία απολύτως βελτίωση, αφού αυτές οι λωρίδες παραμένουν στην πληοψηφεία του χρόνου εκτέλεσης

αχρησιμοποίητες.

Ομοίως σε περιπτώσεις επικοινωνίας "ένας προς όλους" (Broadcast) το δίκτυο γεμίζει πρώτα με flits που πηγαίνουν προς τον κόμβο 0, κατόπιν με flits που κατευθύνονται προς τον κόμβο 1 κ.ο.κ, γεγονός το οποίο και πάλι σημαίνει πως η ύπαρξη σε έναν μεταγωγέα περισσότερων από 2 λωρίδων είναι άχρηστη.

Η ύπαρξη πολλών λωρίδων είναι χρήσιμη για τους τύπους επικοινωνίας "Όλοι προς Ενα" και "Όλοι σε Ολους", ο οποίος και ισοδυναμεί με μια ακολουθία από "Όλοι προς Εναν" τρόπους επικοινωνίας. Ομως σε αυτήν την περίπτωση τα κέρδη τα οποία επιτυγχάνονται από τη γρήγορη άφιξη ενός πακέτου στον προορισμό του χάνονται αφού για να συνεχιστεί η εκτέλεση του προγράμματος θα πρέπει να φτάσουν στον προορισμό τους όλα τα πακέτα. Ο χρόνος παραλαβής όλων των πακέτων είναι όμως περίπου ίδιος είτε οι μεταγωγείς του δικτύου χρησιμοποιούν λίγες είτε πολλές λωρίδες.

Για να επιβεβαιωθούν αυτές οι διαισθητικές παρατηρήσεις, μετρήθηκε κατά τη διάρκεια των προσομοιώσεων το πλήθος των διαφορετικών πακέτων, τα οποία βρίσκονται ταυτόχρονα σε ένα μεταγωγέα. Η μέτρηση αυτή έγινε για τα πειράματα που σημειώθηκαν με 4 και 8 λωρίδες. Για όλες τις εφαρμογές, για όλους τους μεταγωγείς παρατηρήθηκε ότι ο μέσος όρος των πακέτων κυμαίνεται ανάμεσα στο 1 και το 2, χωρίς να ξεπερνάει ποτέ το φράγμα του 2, γεγονός το οποίο φανερώνει πως ακόμη και όταν σε έναν μεταγωγέα ανατίθενται περισσότερες των 2 λωρίδες, αυτές είναι πρακτικά άχρηστες αφού δεν φτάνουν ποτέ πακέτα να τις χρησιμοποιήσουν.

5.2 Σύγκριση αρχιτεκτονικών Wormhole και ATM

Τα πειράματα που έγιναν σε δίκτυο Butterfly 64 εισόδων εξόδων έδειξαν πως η χρήση λωρίδων στους μεταγωγείς του δικτύου βελτιώνει σημαντικά την καθυστέρηση των cells/ flits μέσα στο δίκτυο, τόσο για τα δίκτυα ATM όσο και για τα δίκτυα Wormhole. Ομως οι παρατηρούμενες βελτιώσεις στον συνολικό χρόνο εκτέλεσης των εφαρμογών είναι πολύ μικρές, γεγονός το οποίο θέτει υπό αμφισβήτηση την χρησιμότητα των λωρίδων για τις δύο κατηγορίες δικτύων.

Οι λόγοι για τους οποίους η χρήση λωρίδων δεν επιφέρει τα προσδοκώμενα αποτελέσματα στο συνολικό χρόνο εκτέλεσης των εφαρμογών είναι οι ακόλουθοι:

- Ο τρόπος με τον οποίο η χρήση λωρίδων μεταβάλλει τον χρόνο παραμονής των πακέτων στους παραγωγούς.

  Οπως περιγράφηκε στις προηγούμενες παραγράφους, ο αριθμός των λωρίδων άλλοτε αυξάνει και άλλοτε μειώνει τον χρόνο παραμονής των πακέτων στους ενταμιευτές των παραγωγών με αποτέλεσμα άλλοτε να επιφέρει θετικές και άλλοτε αρνητικές επιπτώσεις στον χρόνο παραλαβής των πακέτων από τους καταναλωτές, εξαλείφοντας όλα τα οφέλη από την συνεχώς ελλατούμενη, χάρη στην ύπαρξη των λωρίδων, καθυστέρηση των cells/flits μέσα στο δίκτυο.

- Η μη άμεση παραλαβή των πακέτων από τους κατανωτές εξαιτίας της αναγκαστικής παραμονής τους στους ενταμιευτές των καταναλωτών.

  Οφέλη που παρατηρούνται στον χρόνο άφιξης των πακέτων στους προορισμούς τους, δεν συνοδεύονται από αντίστοιχα οφέλη στο συνολικό χρόνο εκτέλεσης των εφαρμογών καθώς τα πακέτα είναι υποχρεωμένα εξαιτίας της πληροφορίας που έχει καταγραφεί στα αρχεία ιχνών (trace files) να παραμείνουν στους ενταμιευτές των καταναλωτών μέχρι να έρθει η στιγμή της παραλαβής τους.

- Η κάλυψη του χρόνου αποστολής ενός πακέτου με υπολογισμό

  Οι εφαρμογές συχνά χρησιμοποιούν εντολές αποστολής πακέτων, τα οποία επιτρέπουν στους επεξεργαστές να συνεχίσουν με την εκτέλεση κάποιων υπολογισμών, χωρίς να χρειάζεται να περιμένουν να ολοκληρωθεί η αποστολή στο δίκτυο ενός μηνύματος. Αυτό εξαλείφει ακόμη περισσότερο τις διαφοροποιήσεις στο συνολικό χρόνο εκτέλεσης των εφαρμογών για τις διάφορες τιμές των λωρίδων.

- Οι τρόποι επικοινωνίας μεταξύ των κόμβων του συστήματος

  Οπως περιγράφηκε οι κόμβοι του συστήματος επικοινωνούν μεταξύ τους με τρόπους τέτοιους ώστε σπάνια να βρίσκονται ταυτόχρονα σε κάποιον μεταγωγέα cells, τα οποία να κατευθύνονται σε περισσότερους από 2 διαφορετικούς προορισμούς ή flits τα οποία να ανήκουν σε περισσότερα από δύο διαφορετικά πακέτα. Κατά συνέπεια η χρήση περισσότερων από 2 λωρίδων είναι άχρηστη για εφαρμογές που παρουσιάζουν τους δεδομένους τρόπους επικοινωνίας.

Από τα πειράματα που έγιναν για το δίκτυο Butterfly φάνηκε πως η χρήση των λωρίδων δεν επιφέρει σημαντικές βελτιώσεις ούτε στα δίκτυα ATM, ούτε στα Wormhole. Ενδιαφέρον όμως έχει η σύγκριση των δύο αρχιτεκτονικών. Στο σχήμα 5.8 καταγράφονται για κάθε εφαρμογή ο καλύτερος χρόνος εκτέλεσης σε δίκτυο ATM και

	Εφαρμογή	ATM	Wormhole
1	**ft**	1.326 (2)	1.093 (2)
2	**mg**	2.089 (1)	1.99 (2)
3	**lu**	16.8 (2)	16.45 (2)
4	**ep**	24.8 (4)	24.4 (2)
5	**cg**	11.9 (4)	11.56 (2)
6	**is**	8.01 (8)	7.64 (8)
7	**pstw**	4.27 (1)	3.56 (8)
8	**qr**	1.07 (8)	0.998 (2)
9	**sp**	19.156 (2)	18.54 (4)
10	**bt**	7.864 (4)	7.138 (2)

Σχήμα 5.8: Καλύτεροι χρόνοι εκτέλεσης για δίκτυα ATM και Wormhole

ο καλύτερος χρόνος εκτέλεσης σε δίκτυο Wormhole. Δίπλα σε κάθε χρόνο εκτέλεσης καταγράφεται μέσα σε παρένθεση το πλήθος των λωρίδων για το οποίο σημειώθηκε η καλύτερη απόδοση.

Συγκρίνοντας τους χρόνους εκτέλεσης των εφαρμογών τόσο για το δίκτυο ATM όσο και για το δίκτυο Wormhole, παρατηρείται πως τα δίκτυα Wormhole παρουσιάζουν καλύτερη απόδοση από τα δίκτυα ATM. Ενδιαφέρον επίσης παρουσιάζει το γεγονός ότι στην πλειοψηφία των πειραμάτων σε δίκτυο Wormhole ο καλύτερος χρόνος εκτέλεσης παρατηρείται για 2 λωρίδες.

Τα δίκτυα Wormhole εμφανίζουν καλύτερη απόδοση από τα ATM παρότι η καθυστέρηση του δικτύου είναι περίπου η ίδια και για τις δύο αρχιτεκτονικές δικτύων, όταν το πλήθος των λωρίδων που χρησιμοποιείται είναι μεγαλύτερο από ένα. Για πειράματα σε μία λωρίδα η καθυστέρηση δικτύου είναι μεγαλύτερη σε δίκτυα Wormhole από ότι σε δίκτυα ATM. Αυτό οφείλεται στο γεγονός ότι στα δίκτυα Wormhole αν ένα πακέτο Α μπλοκάρει σε κάποιο μεταγωγέα επειδή κάποιο άλλο πακέτο Β έχει προλάβει να καταλάβει τη λωρίδα του switch, το πακέτο Α θα χρειαστεί να περιμένει να τελειώσει η μετάδοση ολόκληρου του πακέτου Β μέχρι να μπορέσει να χρησιμοποιήσει τη λωρίδα. Αντίθετα στα ATM, cells του πακέτου Β μπορούν να χρησιμοποιήσουν το σύνδεσμο εξόσου αμέσως μόλις cells του πακέτου Α που κατείχαν τη λωρίδα βγουν στο δίκτυο.

Ομως, όσο αφορά στον χρόνο παραμονής των πακέτων στους ενταμιευτές των παραγωγών, τα δίκτυα Wormhole με δύο λωρίδες φαίνονται να συμπεριφέρονται καλύτερα από τα ATM. Δηλαδή ένα πακέτο ολοκληρώνει πιο γρήγορα την μετάδοση του

	Application	1 lane	2 lanes	4 lanes	8 lanes
1	**ft**	1.1917	1.025	1.24	1.31
2	**mg**	1.92	1.88	1.97	1.99
3	**lu**	15.29	15.43	15.4	15.55
4	**ep**	24.2	24.1	24.15	24.17
5	**cg**	11.35	11.31	11.63	11.75
6	**is**	7.89	7.76	7.85	7.81
7	**pstw**	1.99	2.02	2.001	1.987
8	**qr**	0.9725	0.9703	0.9738	0.9772
9	**sp**	19.75	19.67	19.69	19.70
10	**bt**	7.93	7.85	7.88	7.92

Σχήμα 5.9: Χρόνοι εκτέλεσης σε υπερκύβο για δίκτυο ATM

στο δίκτυο όταν αυτό είναι Wormhole με δύο λωρίδες από ότι όταν αυτό είναι ATM. Στα δίκτυα wormhole η μνήμη των μεταγωγέων ανατίθενται στα flits με βάση το πακέτο στο οποίο αυτά ανήκουν και όχι με βάση τον προορισμό προς τον οποίο κατευθύνονται. Αυτό σε συνδυασμό με το γεγονός ότι λόγω των ιδιαιτεροτήτων της κίνησης που παρατηρείται στο δίκτυο δεν συσσωρεύονται ταυτόχρονα πολλά διαφορετικά πακέτα στους μεταγωγείς του δικτύου κάνει τα δίκτυα Wormhole πιο αποδοτικά .

5.3 Πειράματα σε Υπερκύβο και Mesh

Εγιναν πειράματα για δύο ακόμη τοπολογίες δικτύου και συγκεκριμένα για:

- **Εξαδιάστατο Υπερκύβο**
  Το δίκτυο είναι ένας εξαδιάστατος υπερκύβος, ο οποίος αποτελείται από 64 κόμβους. Κάθε κόμβος του δικτύου αναλύεται σε μια συνδεσμολογία εννέα μεταγωγέων 2 x 2, η οποία συνδέει τον κόμβο αυτόν με τους υπόλοιπους κόμβους του δικτύου και με έναν από τους επεξεργαστές του. Το δίκτυο αποτελείται πλέον από 576 μεταγωγείς σε αντίθεση με το δίκτυο Butterfly, το οποίο σχηματίζεται από 192 μεταγωγείς.

- **Διδιάστατο mesh**
  Και αυτό το δίκτυο αποτελείται από 64 κόμβους καθένας από τους οποίους αναλύεται σε 9 μεταγωγείς 2 x 2.

	Application	1 lane	2 lanes	4 lanes	8 lanes
1	**ft**	1.203	1.097	1.21	1.205
2	**mg**	2.02	2.01	2.07	2.09
3	**lu**	17.01	16.98	16.4	16.3
4	**ep**	24.5	24.2	24.8	24.45
5	**cg**	11.22	11.12	11.46	11.32
6	**is**	7.74	7.65	7.89	7.85
7	**pstw**	2.015	2.003	2.014	2.18
8	**qr**	0.854	0.855	0.8554	0.8609
9	**sp**	19.93	19.84	19.92	19.91
10	**bt**	7.85	7.82	7.89	7.915

Σχήμα 5.10: Χρόνοι εκτέλεσης σε mesh για δίκτυο ATM

	Application	1 lane	2 lanes	4 lanes	8 lanes
1	**ft**	1.084	1.082	1.14	1.18
2	**mg**	1.75	1.69	1.73	1.79
3	**lu**	15.93	15.89	15.91	15.97
4	**ep**	24.1	23.9	23.92	23.97
5	**cg**	11.07	11.09	11.27	11.32
6	**is**	7.75	7.70	7.89	7.92
7	**pstw**	1.997	1.902	1.957	1.932
8	**qr**	0.9245	0.913	0.928	0.912
9	**sp**	19.52	19.34	19.42	19.56
10	**bt**	7.89	7.815	7.81	7.82

Σχήμα 5.11: Χρόνοι εκτέλεσης σε υπερκύβο για δίκτυο Wormhole

Στους πίνακες των σχημάτων 5.9, 5.10, 5.11 και 5.12, παρατίθενται τα αποτελέσματα από την προσομοίωση των δέκα εφαρμογών για τις δύο διαφορετικές τοπολογίες (υπερκύβος και mesh) για δίκτυα ATM και Wormhole. Οι χρόνοι μετριούνται σε δευτερόλεπτα.

Παρατηρούμε πως για συγκεκριμένη τοπολογία και τύπο δικτύου, οι διαφορές στον χρόνο εκτέλεσης εξαιτίας της χρήσης λωρίδων είναι πολύ μικρές. Κάτι τέτοιο είχε παρατηρηθεί και στα αποτελέσματα των προσομοιώσεων που έγιναν σε δίκτυο Butterfly, όμως τώρα οι διαφορές ανάμεσα στον καλύτερο και τον χειρότερο χρόνο εκτέλεσης μιας εφαρμογής έχουν γίνει ακόμη μικρότερες. Ο αριθμός των μεταγωγέων που αποτελούν τη συνδεσμολογία του δικτύου καθώς και η τοπολογία του δίνουν στις ροές της κίνησης, που διέρχονται μέσα από το δίκτυο περισσότερο χώρο για την μετάδοση τους. Η πιθανότητα

	Application	1 lane	2 lanes	4 lanes	8 lanes
1	**ft**	1.063	1.056	1.059	1.061
2	**mg**	1.71	1.70	1.71	1.715
3	**lu**	15.89	15.82	15.80	15.82
4	**ep**	23.98	23.96	24.01	24.015
5	**cg**	11.015	11.003	11.107	11.131
6	**is**	7.68	7.65	7.69	7.81
7	**pstw**	1.89	1.82	1.853	1.869
8	**qr**	0.9012	0.917	0.919	0.921
9	**sp**	20.01	19.52	19.67	19.61
10	**bt**	7.79	7.78	7.73	7.79

Σχήμα 5.12: Χρόνοι εκτέλεσης σε mesh για δίκτυο Wormhole

μια ροή κίνησης να μπλοκαριστεί από κάποια άλλη είναι τώρα πιο μικρή, γι'αυτό και η χρήση των λωρίδων δεν φαίνεται να επιφέρει καμμιά ουσιαστική διαφορά στο συνολικό χρόνο εκτέλεσης των εφαρμογών.

Για δίκτυα της ίδιας κατηγορίας (π.χ. υπερκύβος ATM και mesh ATM) παρατηρούμε ότι άλλοτε ο καλύτερος χρόνος εκτέλεσης επιτυγχάνεται για την μία τοπολογία και άλλοτε για την άλλη. Το ποια τοπολογία θα σημειώσει για δεδομένη εφαρμογή τα καλύτερα αποτελέσματα εξαρτάται από την κατανομή των επεξεργαστών πάνω στους κόμβους του δικτύου. Συμφόρηση του δικτύου η οποία παρατηρείται για δεδομένη τοπολογία δεν παρατηρείται για την άλλη, αφού τα πακέτα δρομολογούνται με τέτοιο τρόπο ώστε να μην μπλοκάρει μια ροή κίνησης κάποια άλλη ροή.

Τα δίκτυα hypercube και mesh αποτελούνται από 576 μεταγωγούς ενώ το δίκτυο Butterfly από 192, γεγονός που σημαίνει πως τα δύο πρώτα δίκτυα είναι πιο πλούσια σε πόρους από το τελευταίο. Η προσθήκη πολλών μεταγωγέων προσθέτει "δρόμους" μέσα στο δίκτυο και μειώνει την πιθανότητα μια ροή κίνησης η οποία κατευθύνεται προς δεδομένο προορισμό να μπλοκαριστεί από κάποια άλλη ροή κίνησης. Γι' αυτό και είναι λογικό να παρατηρούμε μικρότερους χρόνους εκτέλεσης στα δίκτυα mesh και hypercube από ότι στο Butterfly.

Κεφάλαιο 6

Συμπεράσματα

Μελετήσαμε τις επιδόσεις δικτύων που αποτελούνται από μεταγωγείς ATM με έλεγχο ροής με εισιτήρια σε σύγκριση με τις επιδόσεις δικτύων wormhole με ανάλογες διαμορφώσεις. Η μελέτη έγινε με προσομοίωση. Κύριο χαρακτηριστικό της μελέτης που έγινε είναι ότι η κίνηση με την οποία τροφοδοτείται το δίκτυο είναι πραγματική, αφού πρόκειται για κίνηση που καταγράφηκε κατά την εκτέλεση 10 πραγματικών message - passing εφαρμογών.

Η μελέτη που έγινε μας οδηγεί στο συμπέρασμα πως τόσο για δίκτυα ATM όσο και για δίκτυα Wormhole η χρήση λωρίδων επιφέρει βελτιώσεις μόνο όσο αφορά στη καθυστέρηση δικτύου, στην οποία υποβάλλονται τα cells/flits. Δεν παρατηρούνται όμως βελτιώσεις στο συνολικό χρόνο εκτέλεσης των εφαρμογών. Αυτό οφείλεται κυρίως στις ιδιαιτερότητες της κίνησης που προέρχεται από αυτές τις πραγματικές εφαρμογές, οι οποίες και εξαλείφουν όλα τα θετικά οφέλη που σημειώνονται από τη χρήση των λωρίδων.

Ενδιαφέρον παρουσιάζει το γεγονός πως η καθυστέρηση δικτύου που παρατηρείται είναι περίπου ίδια και για τις δύο αρχιτεκτονικές δικτύου, όταν το πλήθος λωρίδων είναι μεγαλύτερο από ένα. Για πειράματα με μία λωρίδα η καθυστέρηση δικτύου είναι μεγαλύτερη για δίκτυα Wormhole από ότι για δίκτυα ATM, γεγονός το οποίο οφείλεται στο διαφορετικό τρόπο με τον οποίο γίνεται η ανάθεση των λωρίδων για τις δύο αρχιτεκτονικές.

Συγκρίνοντας τις αρχιτεκτονικές Wormhole και ATM καταλήγουμε στο

συμπέρασμα πως για την κατηγορία των εφαρμογών που χρησιμοποιήθηκαν για την μελέτη, τα δίκτυα Wormhole με 2 λωρίδες σημειώνουν σχετικά καλύτερη επίδοση από τα δίκτυα ATM. Αυτό συμβαίνει γιατί όσο αφορά στον χρόνο παραμονής των πακέτων στους ενταμιευτές των παραγωγών τα δίκτυα Wormhole με δύο λωρίδες επιτυγχάνουν καλύτερες επιδόσεις από τα ATM. Δηλαδή η μετάδοση ενός πακέτου στο δίκτυο ολοκληρώνεται γρηγορότερα σε δίκτυο Wormhole με δύο λωρίδες από ότι σε δίκτυο ATM. Στα wormhole η μνήμη των μεταγωγέων ανατίθεται στα flits με βάση το πακέτο στο οποίο ανήκουν και όχι με βάση τον προορισμό προς τον οποίο κατευθύνονται, γεγονός το οποίο επιτυγχάνει καλύτερη χρήση της μνήμης των μεταγωγέων και επιταχύνει την περάτωση της αποστολής των πακέτων από τους παραγωγούς βελτιώνοντας το συνολικό χρόνο εκτέλεσης.

Ενδιαφέρον θα είχε η μελέτη δικτύων που έχουν σχεδιαστεί με βάση τον τρόπο με τον οποίο επικοινωνούν μεταξύ τους οι κόμβοι του συστήματος για το συγκεκριμένο δείγμα των εφαρμογών. Οπως φάνηκε από την μελέτη που έγινε οι κόμβοι του συστήματος επικοινωνούν κυρίως σε ομάδες. Επομένως σημαντική είναι η βελτίωση της επικοινωνίας μεταξύ ομάδων κόμβων και όχι μεταξύ όλων των κόμβων του συστήματος. Η μελέτη ενός δικτύου στο οποίο στα πακέτα αποδίδονται προτεραιότητες ανάλογα με το πόσο γρήγορα ή αργά αυτά πρέπει να φτάσουν στον προορισμό τους και η χρήση λωρίδων διαφορετικής ταχύτητας πιθανά να οδηγήσουν σε καλύτερα αποτελέσματα όσο αφορά στον συνολικό χρόνο εκτέλεσης των εφαρμογών.

Ενδιαφέρον επίσης θα είχε η προσεχτικότερη μελέτη των τεσσάρων τρόπων επικοινωνίας που αναπτύσσονται κατά τη διάρκεια της εκτέλεσης των εφαρμογών που μελετήθηκαν και ο καθορισμός του βαθμού στον οποίο ο κάθε ένας από αυτούς τους τρόπους επηρεάζει τον χρόνο εκτέλεσης της εφαρμογής. Μια τέτοια μελέτη θα μπορούσε να οδηγήσει στον σχεδιασμό τεχνικών που επιταχύνουν τους συγκεκριμένους τρόπους επικοινωνίας και οδηγούν σε γρηγορότερη εκτέλεση των εφαρμογών.

Συνολικά μπορούμε να πούμε πως για την κατηγορία των εφαρμογών που μελετήθηκαν η χρήση των λωρίδων δεν συμβάλει στη βελτίωση του χρόνου εκτέλεσής τους, εξαιτίας των χαρακτηριστικών και των ιδιαιτεροτήτων της κίνησης που εισέρχεται στο δίκτυο.

Βιβλιογραφία

[1] W.J.Dally. Virtual-Channel Flow Control. IEEE Trans. on Parallel and Distributed Systems, 3(2):194-205, May 1992

[2] J.Duato. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. IEEE Trans. on Parallel and Distributed Systems, 4(12):1320-1331, December 1993

[3] W.J.Dally and C.L.Ceitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. IEEE Trans. on Computers, C-36(5):547-553, May 1987

[4] M.Katevenis, D.Serpanos and E.Spyridakis. Credit-Flow-Controlled ATM versus Wormhole Routing. Technical Report 171, ICS-FORTH, Heraklio, Crete, Greece, July 1996.

[5] R.Cypher, A.Ho, S.Konstantinidou, and P.Messina, "Architectural requirements of parallel scientific applications with explicit communication," in Proceedings of 20th Annual International Symposium on Computer Architecture, pp 2-13, May 1993

[6] S.S.Chodnekar, V. Srinivasan, A.S. Vaidya, A.Sivasubramanian, and C.R.Das. Towards a Communication Characterization Methodology of Parallel Applications. In Proc. Third Intl. Symp. High Performance Computer Architecture (HPCA-3), pages 310-319, February 1997

[7] Message Passing Interface Forum: MPI: A Message-Passing Interface Standard, Version 1.1, July, 1995, http://www.mcs.anl.gov/mpi/mpi-report1.1/mpi-report.html

[8] B. Bailey et al. "The NAS Parallel Benchmarks", International Journal of Supercomputer Applications, vol. 5, no. 3, pp. 63-73, 1991

[9] PARKBENCH Comittee, Public International Benchmarks for Parallel Computers, February 1994. Report-1, assembled by R.Hockney and M.Berry.

[10] E.A.Brewer, N. Dellarocas, A. Colbrook, and W.E. Weihl. PROTEUS: A high-performance parallel-architecture simulator. Technical report MIT-LCS-TR-516, Massachusetts Institute of Technology, Cambridge, MA 02139, September 1991

[11] J.E. Veenstra and R.J. Fowler, "MINT: A Front End for Efficient Simulation and Shared Meemory Multiprocessors", in Proceedings of MASCOTS '94, pp. 201-207, February 1994

[12] G.A.Geist, M.T.Heath, B.W.Peyton, P.Worley, "A Users' guide to PICL: A Portable Instrumented Communication Library", Oak Ridge National Laboratory, Mathematical Sciences Section

[13] C.Eric Wu, Hubertus Franke, Yew-Huey Liu, "UTE: A Unified Trace Environment for IBM SP Systems", IBM T.J. Watson Research Center

[14] Stephen Alan Herrod, "Tango Lite: A Multiprocessor Simulation Environment: Introduction and User's Guide", Computer Systems Laboratory, Stanford University, Stanford CA 94305

[15] Arun Sharma, Antony Trung Nguyen and Joseph Torellas "Augmint: A multiprocessor Simulation Environment for Intel x86 architectures", Center for Supercomputing Research and Development, university of Illinois at Urbana-Champaign

[16] Robert C. Bedichek "Talisman: Fast and Accurate Multicomputer Simulation", Laboratory for Computer Sciemce, NE43-629, Massachissetts Institute for Technology, Cambridge, MA 02139

[17] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, "PVM 3: User's Guide and reference Manual", Oak Ridge National Laboratory, September 1994

[18] Edward Walker, Roger Thien, Chea Chee Weng, "PG_PVM User Manual", National Supercomputing REsearch Center, Singapore

[19] Brad Topol, V. Sunderam, and Anders Alund, "PGPVM Performance visualization support for PVM", Technical Report CSTR-940801, Emory University, August 1991

[20] James Arthur Kohl, G.A.Geist, "The PVM 3.4 Tracing Facility and XPVM 1.1", Computer Science and Mathematics Division, Oak Ridge National Laboratory

[21] Virginia Herrarte, Ewing Lusk, "Studying Parallel Program Behavior with Upshot"

[22] D. Bailey, E.Barszcz, J.Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T.Lanski, R. Schreiber, H. Simon, V. Venkatakrishnan and S. Weeretunga, RNR Technical Report RNR-94-007, March 1994