ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση και Υλοποίηση των Μηχανισμών του Τηλέγραφου στο Λειτουργικό Σύστημα Mach

Κοσμάς Παπαχρήστος

Μεταπτυχιακή Εργασία

Ηράκλειο, Οκτώβριος 1995

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση και Υλοποίηση των Μηχανισμών του Τηλέγραφου στο Λειτουργικό Σύστημα Mach

Εργασία που υποβλήθηκε από τον
Κοσμά Παπαχρήστο
ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Κοσμάς Παπαχρήστος
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Χρήστος Νικολάου,Αναπληρωτής Καθηγητής, Επόπτης

Ευάγγελος Μαρκάτος,Επίκουρος Καθηγητής, Μέλος

Δημήτριος Σερπάνος,Επίκουρος Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος,Αναπληρωτής Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Οκτώβριος 1995

Σχεδίαση και Υλοποίηση των Μηχανισμών του Τηλέγραφου στο Λειτουργικό Σύστημα Mach

Κοσμάς Παπαχρήστος
Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Η διασύνδεση των υπαρχόντων σταθμών εργασίας με γρήγορα δίκτυα επικοινωνίας είναι μια από τις βασικές κατευθύνσεις της αρχιτεκτονικής παράλληλων υπολογιστών. Βασικός στόχος αυτών των συστημάτων είναι να πετύχουν επιδόσεις παραπλήσιες των μεγάλων πολυεπεξεργαστικών συστημάτων έχοντας όμως μειωμένο κόστος κατασκευής και διαθέτοντας μεγαλύτερη ευελιξία και επεκτασιμότητα. Ο Τηλέγραφος είναι μια αρχιτεκτονική που επιτυγχάνει τους παραπάνω στόχους.

Στην εργασία αυτή παρουσιάζεται η αρχιτεκτονική του πρωτοτύπου του Τηλέγραφου Ι και οι απαιτούμενες παρεμβάσεις στο λειτουργικό σύστημα για την έκδοση των ατομικών πράξεων που υποστηρίζονται. Παρουσιάζονται διαφορετικοί τρόποι έκδοσης των ατομικών πράξεων ανάλογα με τις παρεχόμενες από το λειτουργικό σύστημα δυνατότητες και την υποστήριξη που παρέχει το hardware. Βασικός στόχος του λογισμικού υποστήριξης είναι η αποφυγή προσθήκης επιπλέον κόστους στον χρόνο εκτέλεσης των λειτουργιών που υποστηρίζονται από τον Τηλέγραφο. Τέλος, παρουσιάζεται μια συγκριτική μελέτη του Τηλέγραφου με αρχιτεκτονικές διασύνδεσης παράλληλων και κατανεμημένων/πολυεπεξεργαστικών συστημάτων.

Επόπτης: Χρήστος Νικολάου
Αναπληρωτής Καθηγητής
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης.

Design and Implementation of the Telegraphos Operations for the Mach Operating System

Kosmas Papachristos
Master's Thesis

Computer Science Department
University of Crete

Abstract

Telegraphos is an architecture that aims at providing high speed interconnection among processors in order to achieve performance comparable to large−scale multiprocessor systems, without paying significant implementation cost or sacrificing flexibility and expandability.

In this thesis, we discuss the architecture of the Telegraphos I prototype and we describe the necessary modifications at the operating system level, in order to issue the supported atomic operations. Depending on the operating system and the hardware support provided, we can implement different ways of launching atomic operations. The basic goal of the software is to minimize its overhead on the total execution time of the supported operations. Finally, we compare Telegraphos I to other high−speed communication architectures.

Advisor: Christos Nikolaou
Associate Professor
Computer Science Department
University of Crete.

Ευχαριστίες

Ευχαριστώ θερμά τον επόπτη καθηγητή μου, κ. Χρήστο Νικολάου, ο οποίος μου έδωσε την δυνατότητα να ασχοληθώ με το θέμα της εργασίας αυτής. Ακόμη, ευχαριστώ πολύ την κ. Πηνελόπη Κωνσταντά–Φανουράκη και τον κ. Ευάγγελο Μαρκάτο που ήταν οι δύο άνθρωποι χωρίς την βοήθεια και την συμπαράσταση των οποίων, η εργασία αυτή δεν θα είχε ολοκληρωθεί. Ο κ. Σαράντος Καπιδάκης βοήθησε σημαντικά στα πρώτα στάδια της εργασίας.

Ενα μεγάλο ευχαριστώ αξίζει στον Γιώργο Δραμιτινό, του οποίου η εμπειρία στο λειτουργικό σύστημα Mach ήταν πολύ σημαντική για την ολοκλήρωση της εργασίας αυτής, αλλά και στα μέλη της ομάδας των Πλειάδων, του Ινστιτούτου Πληροφορικής, για την πολύ καλή συνεργασία που είχαμε καθ' όλη τη διάρκεια των σπουδών μου.

Επίσης, ευχαριστώ πολύ τους γονείς μου, των οποίων η αμέριστη ηθική στήριξη και συμπαράσταση σε όλη την διάρκεια των σπουδών μου, ήταν αυτή που μου έδινε δύναμη και κουράγιο για την συνέχεια.

Τέλος, θα ήθελα να ευχαριστήσω το Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης καθώς και το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Ερευνας για την υλικοτεχνική και οικονομική υποστήριξη που μου παρείχαν.

Περιεχόμενα

Περίληψη **i**

Abstract **iii**

Ευχαριστίες **v**

Περιεχόμενα **vii**

Κατάλογος Πινάκων **ix**

Κατάλογος Σχημάτων **xi**

1 Εισαγωγή **1**

2 Περιγραφή του Συστήματος **5**

2.1 Εισαγωγή 5

2.2 Η αρχιτεκτονική του Τηλέγραφου 5

2.2.1 Τηλέγραφος Ι 6

2.2.2 Περιγραφή των λειτουργιών του υποσυστήματος διασύνδεσης 8

2.3 Το λειτουργικό σύστημα Mach 11

2.3.1 Διασύνδεση του Τηλέγραφου με το λειτουργικό σύστημα 12

2.4 Κώδικας Pal 13

2.4.1 Χαρακτηριστικά 13

3 Προσομοιωτής **17**

3.1 Περιγραφή 17

3.2 Υλοποίηση 18

3.2.1 Επικοινωνία μεταξύ των πυρήνων 21

3.2.2 Η μνήμη του συστήματος 22

3.2.3 Ελεγχος της ορθότητας του προσομοιωτή 22

4 Βιβλιοθήκη υποστήριξης χρήστη **27**
4.1 Ατομικές πράξεις . . . 27
4.1.1 Υλοποίηση με κλήσεις συστήματος . . . 29
4.1.2 Υλοποίηση με κώδικα PAL . . . 30
4.1.3 Υποστήριξη συστημάτων χωρίς κώδικα Pal . . . 32
4.1.4 Σύγκριση της έκδοσης ατομικών πράξεων με τεχνικές άλλων συστημάτων 34
4.2 Συναρτήσεις υποστήριξης λίστας αντιγράφων σελίδων μνήμης . . . 35
4.2.1 Ελεγχος ύπαρξης αντιγράφων . . . 36
4.2.2 Λίστα αντιγράφων . . . 36
4.2.3 Δημιουργία αντιγράφων . . . 36
4.2.4 Ακύρωση αντιγράφων . . . 36
4.3 Απόδοση του πρωτοτύπου του Τηλέγραφου I . . . 37

5 Σύγκριση με άλλα συστήματα **39**
5.1 Flash . . . 39
5.2 Alewife . . . 41
5.3 KSR . . . 42
5.4 Meiko CS−2 . . . 44
5.5 PRAM . . . 45
5.6 SHRIMP . . . 46
5.7 Hamlyn . . . 48
5.8 Memory Channel . . . 49
5.9 Σύγκριση με τον Τηλέγραφο I . . . 49

6 Συμπεράσματα−Επεκτάσεις **53**
6.1 Μελλοντική εργασία . . . 54

A Κώδικας Pal **57**
A.1 Γενική περιγραφή . . . 57
A.2 Προγραμματισμός σε κώδικα Pal . . . 57

Βιβλιογραφία **61**

Κατάλογος Πινάκων

4.1 Διευθύνσεις των ειδικών καταχωρητών έκδοσης ατομικών πράξεων. 27
4.2 Κόστος εγγραφών/αναγνώσεων για το πρωτότυπο του Τηλέγραφου−I 37

5.1 Καθυστερήσεις της μνήμης της μηχανής KSR. 44
5.2 Χρόνος εκτέλεσης ατομικών πράξεων στις διάφορες αρχιτεκτονικές. 51

Κατάλογος Σχημάτων

2.1 Σχηματική αναπαράσταση τμήματος του δικτύου του Τηλέγραφου Ι 6
2.2 Διασύνδεση του HIB του Τηλέγραφου με τον υπολογιστή. 7
2.3 Αναπαράσταση του κώδικα Pal στην μνήμη του υπολογιστή 14

3.1 Προσομοίωση του δικτύου του Τηλέγραφου Ι 18
3.2 Τοποθέτηση των πινάκων και των μεταβλητών στην κοινή μνήμη του Τηλέγραφου για την εφαρμογή πολλαπλασιασμού πινάκων. 23
3.3 Χρήση της απομακρυσμένης μνήμης σαν αποθηκευτικός χώρος (swap space). . . 24

5.1 Το εσωτερικό ενός κόμβου του *FLASH* 40
5.2 Το εσωτερικό ενός κόμβου του *Alewife* 41
5.3 Η ιεραρχία των κόμβων της μηχανής KSR−1 43
5.4 Διασύνδεση του PRAM με τον δίαυλο του υπολογιστή και το δίκτυο. 45
5.5 Διασύνδεση του SHRIMP με τον δίαυλο του υπολογιστή και το δίκτυο. 47

Κεφάλαιο 1

Εισαγωγή

Οι ανάγκες για όλο και περισσότερη υπολογιστική ισχύ έχουν ωθήσει την έρευνα στην κατασκευή μεγάλων υπολογιστικών συστημάτων με πολλούς επεξεργαστές (multiprocessors). Αποτέλεσμα των ερευνητικών αυτών προσπαθειών είναι υπερ−υπολογιστές όπως το Alewife από το MIT [3] και το Flash [19] από το Πανεπιστήμιο Stanford αλλά και οι πολυεπεξεργαστικές μηχανές διαφόρων εταιριών (Cray, KSR).

Ταυτόχρονα υπάρχει η θέληση για όσο το δυνατό μεγαλύτερη αξιοποίηση των υπαρχόντων σταθμών εργασίας [4], οι οποίοι βασίζονται στις αρχιτεκτονικές υπολογιστών με έναν επε−ξεργαστή (uniprocessors), καθώς το κόστος ανάπτυξης και κατασκευής μεγάλων παράλληλων μηχανών τις καθιστά απαγορευτικές για την μεγαλύτερη κατηγορία χρηστών. Επίσης, η τεχνολογία των επεξεργαστών που χρησιμοποιούνται στις πολυεπεξεργαστικές μηχανές είναι πάντα ένα με δύο χρόνια πίσω σε σχέση με την τεχνολογία των επεξεργαστών των μονοεπε−ξεργαστικών συστημάτων, καθώς ο χρόνος ανάπτυξης και σχεδίασης των πολυεπεξεργαστών είναι μεγαλύτερος [15].

Κάτω από αυτό το πρίσμα της εξέλιξης της τεχνολογίας των υπολογιστών, το χάσμα ανάμεσα στα μεγάλα πολυεπεξεργαστικά συστήματα και τους κλασσικούς μονοεπεξεργα−στές έρχονται να καλύψουν δίκτυα με βάση τα υπάρχοντα μονοεπεξεργαστικά υπολογιστικά συστήματα. Τα δίκτυα αυτά, ξεκίνησαν από τις μικρές ταχύτητες των 1.2 MegaBytes/sec (Ethernet, token−ring) και επεκτάθηκαν φτάνοντας την ταχύτητα των δεκάδων MegaBytes/sec (FDDI, ATM). Τα δίκτυα όμως αυτά, δεν σχεδιάστηκαν για να παρέχουν δυνατότητες πα−ράλληλης επεξεργασίας (έχουν σχετικά μεγάλη καθυστέρηση λόγω πρωτοκόλλων) και κατά συνέπεια, δεν μπορούν να χρησιμοποιηθούν αποδοτικά για απαιτητικές κατανεμημένες ή παράλληλες εφαρμογές.

Ετσι υπάρχει ανάγκη για δίκτυα τα οποία θα υποστηρίζουν λειτουργίες που θα διευκολύνουν την παράλληλη επεξεργασία στους κόμβους του δικτύου και θα παρέχουν ακόμα μεγαλύτερες

ταχύτητες (της τάξης των Gigabits ανά δευτερόλεπτο) μειώνοντας όλο και περισσότερο το χρόνο ανταλλαγής μηνυμάτων μεταξύ απομακρυσμένων επεξεργαστών. Τα δίκτυα αυτά παρέχουν στους χρήστες την δυνατότητα να αξιοποιούν και να χρησιμοποιούν το σύνολο των μηχανών του δικτύου, δίνοντας στο χρήστη την αίσθηση μιας μεγάλης παράλληλης μηχανής. Ετσι ο χρήστης θα μπορεί να χρησιμοποιεί το σύνολο των πόρων του συστήματος για τις εφαρμογές του, χρησιμοποιώντας το σύνολο της μνήμης του δικτύου (μνήμη της ιδεατής παράλληλης μηχανής) αλλά και το σύνολο των διαθέσιμων επεξεργαστών του δικτύου που θα αποτελούν τους κόμβους της ιδεατής αυτής μηχανής.

Τα δίκτυα αυτά όμως, για να έχουν αποδοτική λειτουργία θα πρέπει να παρέχουν υψηλές και σταθερές ταχύτητες επικοινωνίας ανεξάρτητα από τις διακυμάνσεις του φορτίου, να επιτρέπουν απευθείας επικοινωνία των εφαρμογών χωρίς οι εφαρμογές να χρειάζεται να γνωρίζουν τις θέσεις των μεταβλητών στην κοινή μνήμη (transparency), να παρέχουν μηχανισμούς προστασίας, να ελαχιστοποιούν την επέμβαση του λειτουργικού συστήματος, και να έχουν μικρές καθυστερήσεις (low latency) ώστε να είναι δυνατή η εκτέλεση παράλληλων και κατανεμημένων εφαρμογών.

Τα δίκτυα αυτά συνδέονται με τους υπολογιστές είτε σαν επέκταση ενός γρήγορου διαύλου εισόδου – εξόδου (I/O bus), είτε σαν επέκταση του διαύλου επεξεργαστή – κύριας μνήμης. Ανάλογα με την σχεδίαση του δικτύου και τις υποστηριζόμενες λειτουργίες που αυτό παρέχει, ο προγραμματιστής μπορεί να χρησιμοποιεί κάποιο από τα δύο βασικά μοντέλα προγραμματισμού παράλληλων μηχανών (κοινόχρηστης μνήμης ή ανταλλαγής μηνυμάτων).

Υπάρχουν αρκετές προσπάθειες για την σχεδίαση και κατασκευή τέτοιων δικτύων. Το PRAM [29] παρουσιάζει ένα διαφορετικό τρόπο κατασκευής του μοντέλου κοινόχρηστης μνήμης χρησιμοποιώντας τεχνικές γρήγορης μεταφοράς πακέτων. Το Hamlyn [7] προτείνει γρήγορες τεχνικές για μαζικές μεταφορές δεδομένων (DMA). Το Memory Channel [12] διασυνδέει τους υπολογιστές παρέχοντας λειτουργίες που εμφανίζουν το χώρο διευθύνσεων του διαύλου εισόδου/εξόδου σαν επέκταση της κεντρικής μνήμης του συστήματος. Το SHRIMP [6] προτείνει τεχνικές κοινόχρηστης μνήμης όπως το PRAM και επιπλέον παρέχει μηχανισμούς για αποδοτικότερες μαζικές μεταφορές δεδομένων. Ο Τηλέγραφος [25, 22, 17] παρέχει ένα δίκτυο υψηλών ταχυτήτων για την διασύνδεση υπολογιστών και ταυτόχρονα προσφέρει ένα σύνολο από λειτουργίες για την αποδοτικότερη μετατροπή του δικτύου είτε σε μια ιδεατή μηχανή κοινόχρηστης μνήμης (shared memory) είτε σε μια μηχανή που σαν βασικό μέσο επικοινωνίας έχει την ανταλλαγή μηνυμάτων (message passing). Το υποσύστημα διασύνδεσης του δικτύου του Τηλέγραφου με την μηχανή (network interface) παρέχει ένα σύνολο από εντολές που περιλαμβάνει ‘‘μακρυνές’’ εγγραφές και αναγνώσεις, ατομικές πράξεις, ασύγχρονες αναγνώσεις και ασύγχρονη αποστολή μηνυμάτων.

Στην εργασία αυτή παρουσιάζουμε την σχεδίαση και την υλοποίηση του λογισμικού που απαιτείται για την βέλτιστη διασύνδεση του Τηλέγραφου με το περιβάλλον στο οποίο γίνεται η ανάπτυξη των εφαρμογών.

Ενα από τα βασικότερα μειονεκτήματα των μεγάλων πολυεπεξεργαστικών συστημάτων είναι το μεγάλο κόστος ανταλλαγής μηνυμάτων που το κυριότερο μέρος του οφείλεται στις επιβαρύνσεις που προσθέτει το λειτουργικό σύστημα. Οι κύριες εντολές ανταλλαγής μηνυμάτων στα συστήματα αυτά, είναι υλοποιημένες σαν κλήσεις συστήματος (system calls) που έχει σαν συνέπεια, μαζί με τις επιπλέον καθυστερήσεις για την αντιγραφή των μεταφερόμενων δεδομένων, να προσθέτουν αρκετά microseconds στο συνολικό κόστος μετάδοσης ενός μηνύματος. Ετσι κύριος στόχος της σχεδίασης του λογισμικού για τον Τηλέγραφο είναι η αποφυγή της παρεμβολής του λειτουργικού συστήματος, κάτι στο οποίο βοηθά και η ίδια η αρχιτεκτονική του. Ταυτόχρονα όμως, θα πρέπει να εξασφαλίζεται η προστασία των προσπελάσεων διαφορετικών διεργασιών οι οποίες δεν θα πρέπει να θεωρούνται ‘‘ασφαλείς’’. Επιπλέον θα πρέπει να παρέχεται στον προγραμματιστή ένα περιβάλλον ανάπτυξης που θα επιτρέπει την ομαλή μετάβαση του, από το κλασσικό μοντέλο των μονοεπεξεργαστών σε αυτό των πολυεπεξεργαστών.

Στο 2^{o} κεφάλαιο της εργασίας περιγράφεται το σύστημα του Τηλέγραφου και το λειτουργικό σύστημα το οποίο χρησιμοποιήθηκε. Η εργασία αυτή αναπτύχθηκε παράλληλα με την κατασκευή του πρωτοτύπου του Τηλέγραφου Ι και έτσι δημιουργήθηκε ένας προσομοιωτής της λειτουργικότητας του Τηλέγραφου Ι για τον έλεγχο και την ανάπτυξη των υπολοίπων τμημάτων του λογισμικού. Ο προσομοιωτής περιγράφεται αναλυτικά στο 3^{o} κεφάλαιο. Στο 4^{o} κεφάλαιο περιγράφεται αναλυτικά η βιβλιοθήκη που υλοποιήθηκε για την υποστήριξη των λειτουργιών του Τηλέγραφου. Το 5^{o} κεφάλαιο περιέχει μια συγκριτική μελέτη μεταξύ του Τηλέγραφου και άλλων συστημάτων που παρέχουν παρόμοια λειτουργικότητα και συγκρίνουμε τις διαφορετικές αρχιτεκτονικές. Τέλος, στο 6^{o} κεφάλαιο, αναφέρουμε τα συμπεράσματα που προκύπτουν από την παρούσα εργασία και προτείνουμε μελλοντικές κατευθύνσεις.

Κεφάλαιο 2

Περιγραφή του Συστήματος

2.1 Εισαγωγή

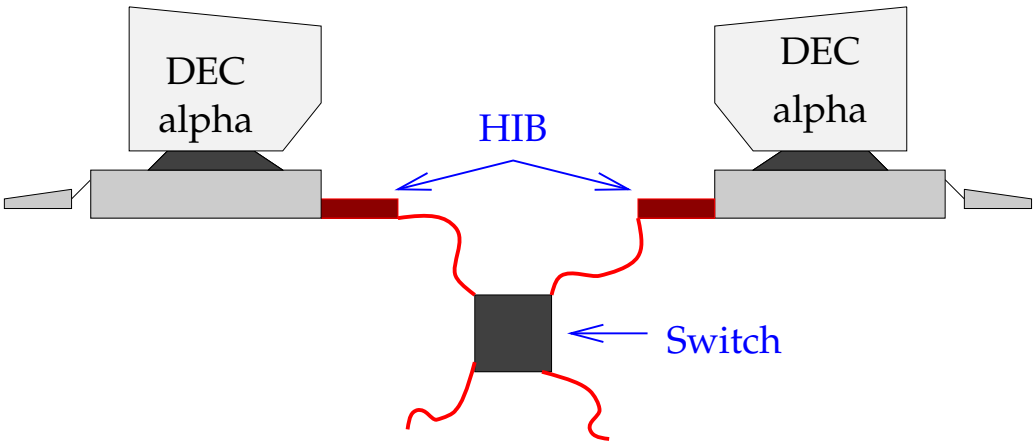
Η εργασία αυτή αναπτύχθηκε στο λειτουργικό σύστημα Mach [28] (έκδοση 3.0 από το Carnegie Mellon University [13][21]) χρησιμοποιώντας την έκδοσή του για τις μηχανές DEC 3000/300 [11] οι οποίες έχουν τον επεξεργαστή Alpha−AXP της DEC [9]. Ο Τηλέγραφος I, για τον οποίο αναπτύχθηκε η παρούσα εργασία, συνδέεται στον δίαυλο εισόδου/εξόδου του υπολογιστή που για την μηχανή που χρησιμοποιήθηκε είναι το TURBOchannel bus της DEC [10]. Στην συνέχεια του κεφαλαίου περιγράφεται η αρχιτεκτονική του Τηλέγραφου και συγκεκριμένα η σχεδίαση και οι υποστηριζόμενες λειτουργίες του Τηλέγραφου I (που είναι η πρώτη υλοποίηση της αρχιτεκτονικής αυτής) καθώς και εκείνα τα στοιχεία της μηχανής και του λειτουργικού συστήματος τα οποία χρησιμοποιήθηκαν κατά κύριο λόγο για την ανάπτυξη της εργασίας αυτής.

2.2 Η αρχιτεκτονική του Τηλέγραφου

Ο Τηλέγραφος έγινε στα πλαίσια ενός ερευνητικού προγράμματος[1] που αφορά στην σχεδίαση και κατασκευή πολυεπεξεργαστικών συστημάτων διασυνδέοντας τους επεξεργαστές με ένα δίκτυο υψηλών ταχυτήτων. Το δίκτυο αποτελείται από μεταγωγείς (switches) που παρέχουν έλεγχο ροής (flow control) και ανοχή της συμφόρησης (congestion tolerance) που μπορεί να προκύψει στο δίκτυο, χρησιμοποιώντας ενταμιευτές (buffers) αποκλειστικής χρήσης για κάθε ιδεατό μονοπάτι (virtual path) του δικτύου.

Ο Τηλέγραφος δεν ορίζει συγκεκριμένη τοπολογία διασύνδεσης και έτσι είναι δυνατή κάθε τοπολογία αρκεί να υπάρχει ιδεατό μονοπάτι για την επικοινωνία ανάμεσα σε κάθε κόμβο στο δίκτυο.

[1] "SHIPS" ESPRIT project 6253.



Σχήμα 2.1: Σχηματική αναπαράσταση τμήματος του δικτύου του Τηλέγραφου I

Βασικό δομικό στοιχείο της αρχιτεκτονικής του Τηλέγραφου είναι η πρωταρχική λειτουργία της τηλε−εγγραφής (remote write) από όπου προέρχεται και η ονομασία του Τηλέγραφου. Οι απομακρυσμένες εγγραφές επιτρέπουν την γρήγορη ανταλλαγή μηνυμάτων μεταξύ των υπολογιστών του δικτύου, ενώ η υλοποίησή της σαν απλή εντολή *store* για τον επεξεργαστή, εξαλείφει την ανάγκη για παρέμβαση του λειτουργικού συστήματος. Ο χώρος των διευθύνσεων της μνήμης του δικτύου είναι κοινός για όλους τους επεξεργαστές του δικτύου, ενώ η προστασία παρέχεται από τον υπάρχοντα μηχανισμό προστασίας που παρέχει η μετάφραση ιδεατών διευθύνσεων σε φυσικές. Αυτό έχει σαν αποτέλεσμα την ελαχιστοποίηση του χρόνου εκτέλεσης των απομακρυσμένων εγγραφών. Επίσης η αρχιτεκτονική του Τηλέγραφου προβλέπει επιπλέον λειτουργίες στην απομακρυσμένη μνήμη, όπως σύγχρονες/ασύγχρονες αναγνώσεις, μετρητές προσπελάσεων στις σελίδες της κοινής μνήμης του δικτύου και ατομικές πράξεις, παρέχοντας ένα βασικό σύνολο λειτουργιών για την αποδοτική υποστήριξη ιδεατής κοινόχρηστης μνήμης.

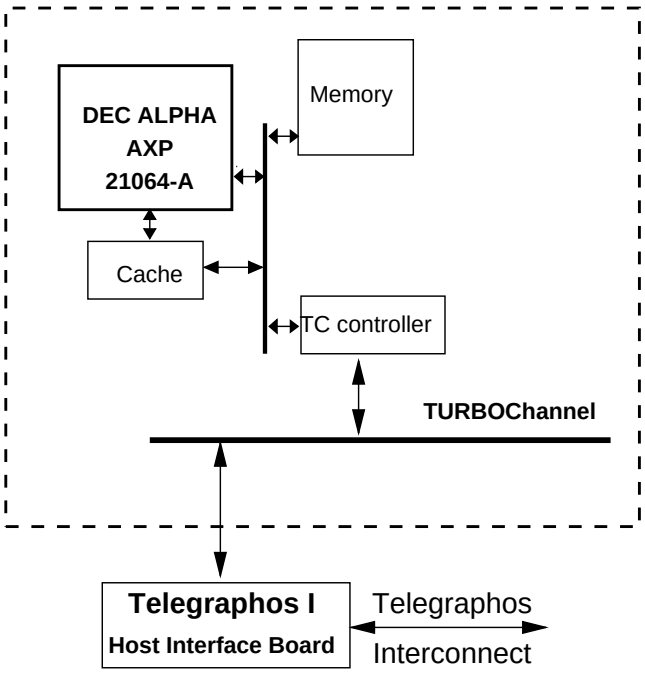
2.2.1 Τηλέγραφος I

Η παρούσα εργασία βασίζεται στο πρωτότυπο *Τηλέγραφος I* της αρχιτεκτονικής του Τηλέγραφου. Το πρωτότυπο αυτό είχε σαν βασικό στόχο να αναδείξει τα βασικά στοιχεία της αρχιτεκτονικής του Τηλέγραφου και ταυτόχρονα να αποτελέσει τη βάση για την περαιτέρω εξέλιξη της. Το δίκτυο, τμήμα του οποίου παρουσιάζεται στο σχήμα 2.1, αποτελούν σταθμοί εργασίας της εταιρίας Digital και συγκεκριμένα το μοντέλο DEC 3000/300 το οποίο χρησιμοποιεί τον επεξεργαστή Alpha AXP (DECchip 21064−AA), κάρτες διασύνδεσης των υπολογιστών με τον Τηλέγραφο (Host Interface Board ή HIB για συντομία) και μεταγωγείς

(switches) για την διασύνδεση του δικτύου.

Η διασύνδεση των υπολογιστών στο δίκτυο του Τηλέγραφου έγινε μέσω του διαύλου εισόδου/εξόδου που έχουν οι συγκεκριμένοι υπολογιστές, το TURBOchannel bus, με ρολόι χρονισμού 12.5MHz. Η κοινή μνήμη του δικτύου βρίσκεται στο χώρο διευθύνσεων του TURBOchannel και είναι τοποθετημένη στο HIB.

Οι προδιαγραφές του TURBOchannel καθορίζουν ένα χώρο 128 MBytes μνήμης (29 bits πλάτος διεύθυνσης) για κάθε θέση επέκτασης. Ωστόσο, για το συγκεκριμένο μοντέλο, το πλάτος διεύθυνσης είναι περιορισμένο στα 25 bits και έτσι ο μέγιστος κοινός χώρος μνήμης είναι 32 Mbytes. Το υποσύστημα διασύνδεσης υλοποιεί το πρωτόκολλο διασύνδεσης και έχει το τμήμα της κοινής μνήμης του. Η κύρια μνήμη του υπολογιστή δεν μπορεί να χρησιμοποιηθεί σαν τμήμα της κοινής μνήμης του δικτύου, διότι ο *Τηλέγραφος I* συνδέεται στο δίαυλο εισόδου/εξόδου και όχι στο δίαυλο επεξεργαστή−μνήμης[2]. Το σύστημα που έχει υλοποιηθεί και στο οποίο έχουν γίνει οι μετρήσεις, αποτελείται από δύο υπολογιστές και δύο κάρτες διασύνδεσης με 16 Mbytes μνήμης η κάθε μία.



Σχήμα 2.2: Διασύνδεση του HIB του Τηλέγραφου με τον υπολογιστή.

[2]Στον Τηλέγραφο II δεν θα υπάρχει αυτός ο περιορισμός

2.2.2 Περιγραφή των λειτουργιών του υποσυστήματος διασύνδεσης

Η κάρτα διασύνδεσης του Τηλέγραφου με τον υπολογιστή είναι υπεύθυνη για την εξυπηρέτηση αιτήσεων του επεξεργαστή για προσπελάσεις που αφορούν την κοινή μνήμη του δικτύου και την εξυπηρέτηση των αιτήσεων που φτάνουν από το δίκτυο. Ο τρόπος με τον οποίο συνδέεται η κάρτα με τον δίαυλο του υπολογιστή φαίνεται στο σχήμα 2.2

Συγκεκριμένα, οι εντολές που υποστηρίζονται από τον Τηλέγραφο Ι είναι:

Εγγραφές στην κοινή μνήμη του δικτύου

Οι εγγραφές στην κοινή μνήμη του δικτύου διακρίνονται σε τοπικές και απομακρυσμένες. Ο επεξεργαστής δεν χρειάζεται να γνωρίζει αυτόν τον διαχωρισμό. Μια εγγραφή είναι μια απλή *store* εντολή για τον επεξεργαστή. Για την κοινή μνήμη του δικτύου η διεύθυνση στην οποία γίνεται η εγγραφή είναι διεύθυνση του TURBOchannel και αναλαμβάνει ο ελεγκτής του TURBOchannel (TC Controller) να τη διοχετεύσει στον δίαυλο εισόδου/εξόδου. Η διάκριση των διευθύνσεων σε τοπικές και μη τοπικές γίνεται από το HIB. Τα 5 πιο σημαντικά ψηφία (bits) της διεύθυνσης καθορίζουν τον κόμβο του δικτύου, ενώ τα υπόλοιπα ορίζουν την σελίδα μνήμης και την συγκεκριμένη θέση μέσα σε αυτή. Αν η διεύθυνση αφορά μνήμη που βρίσκεται τοπικά στην κάρτα διασύνδεσης η εντολή εγγραφής ικανοποιείται άμεσα από το HIB. Αν η διεύθυνση αφορά απομακρυσμένη μνήμη, η κάρτα διασύνδεσης αναλαμβάνει να δημιουργήσει το κατάλληλο πακέτο και να το στείλει στο δίκτυο. Και στις δύο περιπτώσεις, ο επεξεργαστής δεν χρειάζεται να περιμένει για την ολοκλήρωση της εγγραφής και μπορεί να συνεχίσει την εκτέλεση των υπολοίπων εντολών του.

Αναγνώσεις από την κοινή μνήμη του δικτύου

Οπως και οι εγγραφές, οι αναγνώσεις διακρίνονται σε τοπικές και απομακρυσμένες. Οι αναγνώσεις για τον επεξεργαστή είναι απλές εντολές *load* και κατά συνέπεια πρέπει ο επεξεργαστής να περιμένει το τελικό αποτέλεσμα. Οι τοπικές προσπελάσεις ικανοποιούνται άμεσα από το HIB, ενώ για τις απομακρυσμένες το υποσύστημα διασύνδεσης στέλνει ένα πακέτο με την εντολή ανάγνωσης και περιμένει το πακέτο με την απάντηση.

Οι προδιαγραφές του TURBOchannel καθορίζουν ότι καμμία συσκευή δεν μπορεί να έχει τον έλεγχο του διαύλου για περισσότερους από 256 κύκλους (συχνότητας 25MHz). Ο Τηλέγραφος δεν εξασφαλίζει πάντα, την ικανοποίηση των αναγνώσεων σε λιγότερους από 256 κύκλους. Σε αυτή την περίπτωση, που εξαντλείται το όριο ελέγχου του διαύλου, ο επεξεργαστής διακόπτεται από τον ελεγκτή του διαύλου και το λογισμικό (διαχειριστής των διακοπών) είναι υπεύθυνο για να εξυπηρετήσει την διακοπή. Οταν τελικά το HIB πάρει

το αποτέλεσμα της ανάγνωσης, τοποθετεί το αποτέλεσμα σε έναν ειδικό καταχωρητή ενώ ταυτόχρονα θέτει την αντίστοιχη σημαία (flag) στον ειδικό καταχωρητή κατάστασης (status register). Τέτοιες ακραίες καταστάσεις δύναται να προκύψουν αν για παράδειγμα, πριν την ανάγνωση έχει προηγηθεί εγγραφή σε σελίδα μνήμης που έχει αντίγραφα σε πολλούς κόμβους.

Ενημέρωση των αντιγράφων των σελίδων μνήμης

Ο Τηλέγραφος υποστηρίζει την ύπαρξη πολλαπλών αντιγράφων για τις σελίδες της κοινής μνήμης του δικτύου. Οταν για παράδειγμα, κάποιος επεξεργαστής κάνει συχνές αναγνώσεις από σελίδες απομακρυσμένης μνήμης, το λειτουργικό σύστημα μπορεί να ζητήσει αντίγραφο της σελίδας τοπικά ώστε να αυξηθεί η απόδοση μειώνοντας τον χρόνο επικοινωνίας. Για την δημιουργία του αντιγράφου υπεύθυνος είναι ο διαχειριστής της μνήμης του συστήματος ο οποίος, για κάθε σελίδα με πολλαπλά αντίγραφα, διατηρεί μια λίστα με τους κόμβους που τα διαθέτουν. Η λίστα αυτή διατηρείται στην μνήμη που βρίσκεται στο HIB και είναι δεσμευμένη για τον σκοπό αυτό. Το υποσύστημα διασύνδεσης φροντίζει αυτόματα να ενημερώνει τα αντίγραφα των μεταβαλλόμενων σελίδων που αυτό έχει τοπικά. Για να μπορεί ο διαχειριστής μνήμης να διαχειρίζεται αποδοτικότερα τις λίστες αντιγράφων και να μπορεί να παίρνει σωστές αποφάσεις κάθε φορά που χρειάζεται να δημιουργηθεί κάποιο αντίγραφο σελίδας, ο Τηλέγραφος διατηρεί ειδικούς μετρητές προσπέλασης για κάθε σελίδα. Οι μετρητές αυτοί τίθενται αρχικά σε κάποια τιμή και μειώνονται κάθε φορά που γίνεται μια προσπέλαση στην συγκεκριμένη σελίδα. Οταν ένας μετρητής πάρει την τιμή μηδέν ο Τηλέγραφος προκαλεί διακοπή στον επεξεργαστή και το λειτουργικό σύστημα μπορεί να πάρει δυναμικά, αποφάσεις για την διαχείριση της μνήμης.

Ενημέρωση των μετρητών προσπέλασης

Ο Τηλέγραφος υποστηρίζει μετρητές προσπέλασης για κάθε σελίδα μνήμης. Συγκεκριμένα, σε κάθε HIB υπάρχουν μετρητές για κάθε σελίδα τοπικής μνήμης. Οι μετρητές αυτοί διακρίνονται σε μετρητές τοπικών προσπελάσεων και μετρητές απομακρυσμένων προσπελάσεων και είναι διαφορετικοί για τις αναγνώσεις και για τις εγγραφές. Σε κάθε εγγραφή ή ανάγνωση που εξυπηρετείται, ενημερώνεται ταυτόχρονα και ο αντίστοιχος μετρητής. Οι μετρητές αυτοί αρχικά τίθενται σε κάποια τιμή (την ορίζει το λειτουργικό σύστημα) και μειώνονται σε κάθε προσπέλαση.

Οταν κάποιος μετρητής που είχε τιμή 1 πάρει την τιμή 0, το HIB προκαλεί διακοπή την οποία διαχειρίζεται το λειτουργικό σύστημα και μέσω της οποίας μπορεί να πάρει δυναμικά αποφάσεις για την διάταξη των δεδομένων στην κοινή μνήμη του δικτύου. Αν το λειτουργικό σύστημα θέλει να ξαναπάρει διακοπή από τον ίδιο μετρητή πρέπει να του ξαναδώσει μη

μηδενική τιμή. Ολοι οι μετρητές αρχικά, έχουν τιμή 0, και έτσι το λειτουργικό σύστημα έχει την δυνατότητα να επιλέξει για ποιές σελίδες μνήμης θέλει κάθε φορά να προκαλείται διακοπή.

Ατομικές Πράξεις

Στις πολυεπεξεργαστικές μηχανές χρησιμοποιούνται ειδικές εντολές που επιτρέπουν τον συγχρονισμό των επεξεργαστών είτε για έλεγχο της ροής των προς εκτέλεση εντολών (instruction stream) είτε για τον έλεγχο της ροής των δεδομένων (data stream).

Ο Τηλέγραφος παρέχει ένα δομικό σύνολο ατομικών πράξεων που επιτρέπουν τέτοιου είδους συγχρονισμούς. Οι ατομικές αυτές πράξεις εκτελούνται σε σελίδες που δεν διαθέτουν αντίγραφα και έτσι εγγυώνται ότι όλοι οι επεξεργαστές θα δουν το ίδιο αποτέλεσμα μετά το τέλος της πράξης. Κάθε πράξη αποτελείται από ένα ζεύγος διεύθυνσης – τελεστέου. Ο τελεστέος καθορίζεται να έχει μέγεθος 24 bits. Για την εκτέλεση μιας ατομικής πράξης ο επεξεργαστής κάνει δύο εγγραφές που ορίζουν τον τελεστέο και την διεύθυνση στην οποία θα γίνει η ατομική πράξη. Το αποτέλεσμα προκύπτει κάνοντας μια ανάγνωση της διεύθυνσης στην οποία εκτελέστηκε η ατομική πράξη. Οι υποστηριζόμενες ατομικές πράξεις από το υποσύστημα διασύνδεσης είναι αναλυτικά οι εξής:

- **Fetch and Add:** επιστρέφει την λέξη που υπήρχε αποθηκευμένη στην διεύθυνση και της προσθέτει τον τελεστέο·
- **Fetch and Store:** επιστρέφει την λέξη που υπήρχε αποθηκευμένη στην διεύθυνση και αποθηκεύει στην θέση της τον τελεστέο·
- **Compare and Swap:** συγκρίνει την λέξη που υπάρχει ήδη, με μια λέξη μεγέθους 12 bits και αν είναι ίσες τότε αποθηκεύει μια λέξη μεγέθους 12 bits στην αρχική διεύθυνση. Είναι προφανές ότι και οι δύο λέξεις των 12 bits κωδικοποιούνται στον τελεστέο.

Ασύγχρονες αναγνώσεις

Ο *Τηλέγραφος I* παρέχει την δυνατότητα να εκτελούνται αναγνώσεις ασύγχρονα, χωρίς δηλαδή, να χρειάζεται ο επεξεργαστής να περιμένει για την ολοκλήρωση της πράξης (non−blocking execution). Ο μηχανισμός έκδοσης τέτοιων εντολών ανάγνωσης είναι ίδιος όπως και για την έκδοση ατομικών πράξεων αλλά στην θέση του τελεστέου υπάρχει η διεύθυνση προορισμού, η οποία θα πρέπει να είναι διεύθυνση της τοπικής μνήμης, διαφορετικα η πράξη δεν εκτελείται. Οι ασύγχρονες αναγνώσεις επιστρέφουν αμέσως, χωρίς να καθυστερούν την ροή εκτέλεσης των εντολών στον επεξεργαστή σε αντίθεση με τις ατομικές πράξεις, που ο επεξεργαστής περιμένει το αποτέλεσμα.

Προστασία διευθυνσιοδότησης

Στις υποστηριζόμενες λειτουργίες του Τηλέγραφου Ι αναφέρθηκε ότι οι προσπελάσεις στην μνήμη γίνονται μέσω απλών εντολών *store* και *load*[3]. Η δυνατότητα αυτή επιτυγχάνεται γιατί η μνήμη είναι κοινή για όλο το δίκτυο (με σταθερή διευθυνσιοδότηση) και κατά συνέπεια χρησιμοποιείται ο υπάρχων μηχανισμός προστασίας που παρέχει η ιδεατή μνήμη του λειτουργικού συστήματος για την προστασία των προσπελάσεων στην μνήμη του Τηλέγραφου. Οι διεργασίες που προσπελαύνουν διευθύνσεις στο χώρο του Τηλέγραφου έχουν αποκτήσει νωρίτερα μια αντιστοίχιση των φυσικών διευθύνσεων των σελίδων της κοινής μνήμης που τους ανήκουν, στις ιδεατές διευθύνσεις που τους έχει παραχωρήσει το λειτουργικό σύστημα. Ετσι κάθε προσπέλαση των διεργασιών στη κοινή μνήμη ακολουθεί το ίδιο μονοπάτι εκτέλεσης όπως και για τις υπόλοιπες διευθύνσεις στην τοπική μνήμη του υπολογιστή εξασφαλίζοντας ομοιόμορφη συμπεριφορά για όλες τις προσπελάσεις και κατά συνέπεια τον ίδιο βαθμό προστασίας.

2.3 Το λειτουργικό σύστημα Mach

Το λειτουργικό σύστημα το οποίο επιλέχθηκε για την παρούσα εργασία είναι το Mach το οποίο αναπτύχθηκε και υλοποιήθηκε από το πανεπιστήμιο Carnegie Mellon. Συγκεκριμένα, χρησιμοποιήθηκε η έκδοση 3.0, η οποία και είναι η έκδοση του λειτουργικού συστήματος Mach βασισμένη σε μικρο−πυρήνα (micro−kernel). Οι προσπάθειες που γίνονται για την καθιέρωση κοινών προτύπων για το μοντέλο προγραμματισμού που θα παρέχουν τα καινούρια συστήματα ώθησαν τους σχεδιαστές των λειτουργικών συστημάτων σε νέες τεχνικές που απομονώνουν από τον πυρήνα του λειτουργικού συστήματος όλα τα ανεξάρτητα από την μηχανή τμήματά του και αφήνουν σε αυτόν τις πρωταρχικές λειτουργίες διαχείρισης του επεξεργαστή, της μνήμης και των περιφερειακών συσκευών. Μια τέτοια σχεδίαση παρέχει μεγαλύτερη ευελιξία για το λειτουργικό σύστημα χωρίς το κόστος που προστίθεται να είναι αποτρεπτικό για την υλοποίηση της.

Η έκδοση 3.0 του λειτουργικού συστήματος Mach εκμεταλλεύεται αυτήν ακριβώς την παρεχόμενη ευελιξία και επιτρέπει την συνύπαρξη διαφορετικών εκδόσεων του Unix (π.χ BSD ή SysV) πάνω από τον ίδιο μικρο−πυρήνα. Ο μικρο−πυρήνας είναι υπεύθυνος μόνο για τον έλεγχο του επεξεργαστή, της φυσικής μνήμης του συστήματος και των περιφερειακών συσκευών και παρέχει τις βασικές μόνο δομές για την διαχείριση βασικών στοιχείων του συστήματος (ενδο−διεργασιακή επικοινωνία (interprocess communication), υποστήριξη για

[3]Για ετερογενή δίκτυα που θα περιέχουν και μηχανές με διαφορετική διάταξη μνήμης (byte ordering) το HIB θα πρέπει να εξασφαλίζει την σωστή διάταξη χρησιμοποιώντας ένα κοινό τρόπο μεταφοράς των δεδομένων για όλο το δίκτυο.

πολυεπεξεργαστικά συστήματα, διαχείριση κλήσεων συστήματος). Ολα τα υπόλοιπα στοιχεία, που αναγνωρίζονται σε ένα τυπικό σύστημα Unix, παρέχονται από μια εξωτερική διεργασία (Unix Server) που αναλαμβάνει να εξυπηρετεί όσα προγράμματα απαιτούν αυτές τις υπηρεσίες.

2.3.1 Διασύνδεση του Τηλέγραφου με το λειτουργικό σύστημα

Ο Τηλέγραφος συνδέεται με τον υπολογιστή μέσω του διαύλου εισόδου/εξόδου TURBOchannel. Είναι λοιπόν, μια επιπλέον συσκευή εισόδου/εξόδου και έτσι την χειρίζεται το λειτουργικό σύστημα. Οι συσκευές που τοποθετούνται στον δίαυλο TURBOchannel έχουν το ίδιο εύρος διευθύνσεων αλλά διαφορετική αρχική διεύθυνση. Για να μπορέσει το λειτουργικό σύστημα Mach να αναγνωρίσει το υποσύστημα διασύνδεσης του Τηλέγραφου θα πρέπει να υπάρχει ένας οδηγός συσκευής (device driver). Οι οδηγοί συσκευών στο λειτουργικό σύστημα Mach αποτελούνται από ένα σύνολο βασικών συναρτήσεων (open, close, reset, read, write, getstat, setstat, mmap, interrupt_handler), αλλά δεν είναι απαραίτητο να παρέχονται όλες οι συναρτήσεις. Η κάθε συσκευή, ανάλογα με την ιδαιτερότητά της παρέχει διαφορετικό υποσύνολο συναρτήσεων.

Ο έλεγχος για την ύπαρξη συσκευών στον δίαυλο εισόδου/εξόδου (TURBOchannel) γίνεται από τον διαχειριστή του διαύλου. Ο διαχειριστής ελέγχει αν υπάρχουν συσκευές και προσπαθεί να τις αναγνωρίσει. Η αναγνώριση γίνεται μέσω ενός πίνακα αντιστοίχισης που υπάρχει στον μικρο−πυρήνα του λειτουργικού συστήματος. Κάθε συσκευή πρέπει να έχει μια μνήμη ROM, όπου θα έχει τις απαραίτητες πληροφορίες για να γίνει η αναγνώρισή της από τον διαχειριστή του διαύλου.

Το υποσύστημα διασύνδεσης του Τηλέγραφου I δεν διαθέτει τέτοια μνήμη ROM και έτσι το λειτουργικό σύστημα δεν μπορεί να το αναγνωρίσει αυτόματα. Για το HIB του Τηλέγραφου−I ο πυρήνας του λειτουργικού συστήματος Mach αναγνωρίζει την ύπαρξη του ακολουθώντας διαφορετική μέθοδο αναγνώρισης. Το εύρος των διευθύνσεων στις οποίες τοποθετείται το HIB είναι ταυτόχρονα και η μνήμη του HIB. Ετσι, η μέθοδος ανίχνευσης εκτελεί μια εγγραφή και μια ανάγνωση από μια διεύθυνση που ανήκει στο εύρος διευθύνσεων του HIB και αν οι δύο πράξεις γίνουν σωστά (χωρίς την πρόκληση διακοπής), τότε το λειτουργικό σύστημα θεωρεί ότι υπάρχει το υποσύστημα διασύνδεσης του Τηλέγραφου.

Η μνήμη του Τηλέγραφου μπορεί να απεικονιστεί στις διεργασίες του χρήστη με δύο τρόπους. Ο ένας είναι μέσω του διαχειριστή μνήμης (memory manager) του συστήματος. Η τεχνική αυτή προϋποθέτει την δυνατότητα υποστήριξης από τον μεταφραστή (compiler) και τον συνδετήρα (linker) ειδικών συμβολικών λέξεων (keywords) για την ανάθεση μεταβλητών στον χώρο διευθύνσεων της κοινόχρηστης μνήμης του Τηλέγραφου. Η άλλη τεχνική είναι μέσω της συνάρτησης *mmap( )* του οδηγού της συσκευής του HIB. Με την χρήση του οδηγού

της συσκευής είναι δυνατόν οι διεργασίες του χρήστη να χρησιμοποιούν το σύνολο ή τμήματα της κοινόχρηστης μνήμης του δικτύου.

2.4 Κώδικας Pal

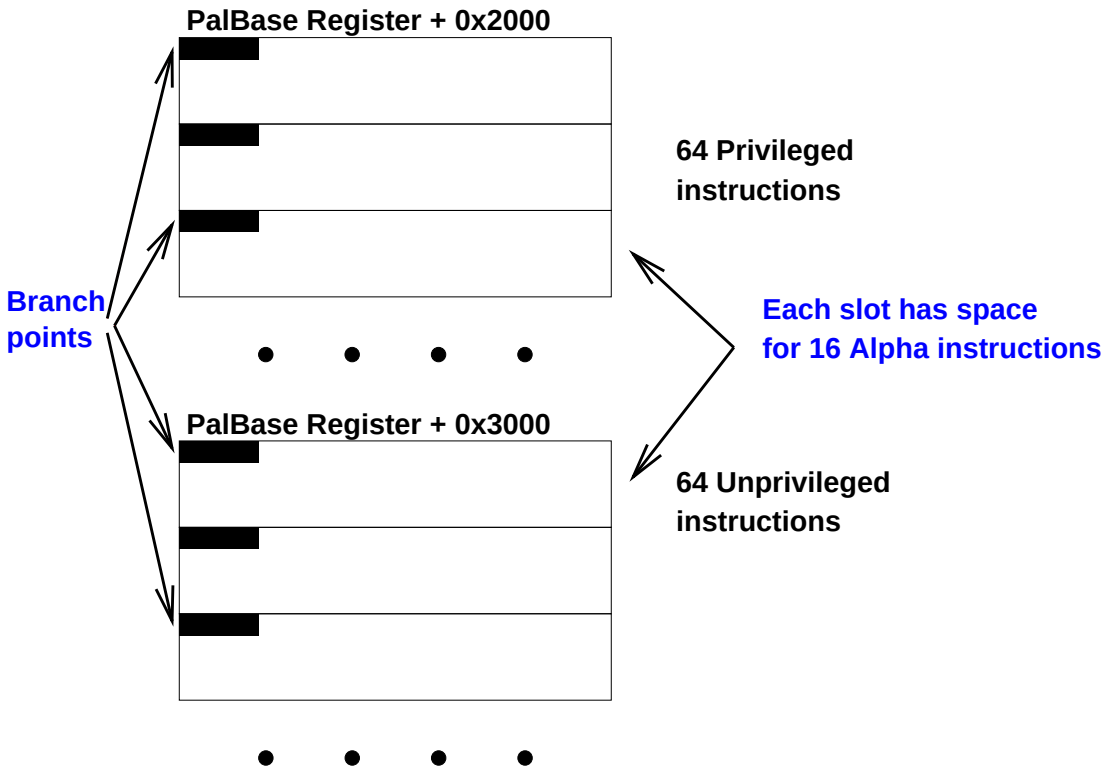
Η αρχιτεκτονική του επεξεργαστή Alpha AXP προσφέρει ένα επιπλέον περιβάλλον εκτέλεσης εντολών (pal mode) το οποίο παρέχει πλήρη έλεγχο του επεξεργαστή. Το όνομα PAL προκύπτει από τα αρχικά των λέξεων Privileged Architecture Library και είναι ένα σύνολο από συναρτήσεις που παρέχουν ζωτικής σημασίας λειτουργίες για τον υπολογιστή. Αυτό το περιβάλλον εκτέλεσης εντολών χρησιμοποιήθηκε ιδιαίτερα για την υλοποίηση εκτέλεσης ατομικών πράξεων.

Ο κώδικας Pal [30] αποτελείται από τις κανονικές εντολές του επεξεργαστή και επιπλέον διαθέτει κάποιες ειδικές εντολές (ανάλογα με την υλοποίηση) που επιτρέπουν προσπέλαση στους εσωτερικούς καταχωρητές του επεξεργαστή. Η κλήση του κώδικα Pal γίνεται είτε μέσω διακοπής (interrupt) από κάποια συσκευή του υπολογιστή είτε μέσω λογισμικού με χρήση της εντολής CALL_PAL [8]. Η βιβλιοθήκη του κώδικα Pal βρίσκεται σε συγκεκριμένες διευθύνσεις στην μνήμη του επεξεργαστή, ενώ οι βασικές λειτουργίες για τις οποίες χρησιμοποιείται είναι η αρχική εκκίνηση του υπολογιστή (boot), διαχείριση των διακοπών, διαδικασίες για την διαχείριση του πίνακα μετάφρασης σελίδων (Translation Buffer) αλλά και ακολουθίες εντολών που πρέπει να εκτελεστούν ατομικά χωρίς διακοπή (long instruction sequences) ή δεν υποστηρίζονται από την συγκεκριμένη υλοποίηση του επεξεργαστή. Η αρχιτεκτονική δεν καθορίζει τρόπους προσθήκης επιπλέον συναρτήσεων Pal αλλά επιτρέπει την αλλαγή ολόκληρου το κώδικα εκτός από ορισμένες λειτουργίες που απαιτεί να υπάρχουν σε κάθε διαφορετική υλοποίηση. Κάθε Alpha workstation περιέχει σε μνήμη ROM έκδοση του κώδικα Pal που έχει την δυνατότητα να υποστηρίξει τα λειτουργικά συστήματα Digital Unix (πρώην DEC OSF/1) και Open VMS. Ωστόσο, παρέχεται η δυνατότητα στο λειτουργικό σύστημα, κατά την εκκίνηση του, να αλλάξει τον κώδικα Pal. Ετσι για παράδειγμα μπορεί κάθε λειτουργικό σύστημα να χρησιμοποιεί διαφορετικό κώδικα Pal. Η έκδοση του λειτουργικού συστήματος Mach για τα Alpha workstations, χρησιμοποιεί την έκδοση του κώδικα Pal που χρησιμοποιεί και το λειτουργικό σύστημα Ditigal Unix.

2.4.1 Χαρακτηριστικά

Το περιβάλλον εκτέλεσης του κώδικα Pal έχει ένα σύνολο από ιδιότητες που καθορίζουν και την ιδιαιτερότητα του. Συγκεκριμένα:

- προσφέρει πλήρη έλεγχο του επεξεργαστή·



Σχήμα 2.3: Αναπαράσταση του κώδικα Pal στην μνήμη του υπολογιστή

- δεν επιτρέπονται διακοπές (interrupts)·
- ενεργοποιούνται οι επιπλέον εντολές του επεξεργαστή που επιτρέπουν την προσπέλαση στους εσωτερικούς καταχωρητές του·
- δεν εξυπηρετούνται διακοπές που οφείλονται σε λάθη προσπέλασης σε σελίδες μνήμης που περιέχουν κώδικα (I−stream memory management traps are prevented).

Για τον προγραμματιστή, ο κώδικας Pal φαίνεται να είναι σαν μια ακολουθία εντολών του επεξεργαστή τις οποίες μπορεί να εκτελέσει κάνοντας διακλάδωση μόνο σε συγκεκριμένα σημεία. Σχηματικά αυτό φαίνεται στο Σχήμα 2.3 Η διακλάδωση στον κώδικα Pal γίνεται με την ειδική εντολή *call_pal*.

Κάθε διάστημα (slot) έχει χώρο μόνο για 16 εντολές όπου η τελευταία πρέπει να είναι η εντολή HW_REI που επιστρέφει από το περιβάλλον του κώδικα Pal, στο κανονικό περιβάλλον εκτέλεσης εντολών. Υπάρχουν 64 διαστήματα για μακροεντολές που επιτρέπεται να εκτε−λούνται μόνο στο προστατευμένο περιβάλλον του πυρήνα (kernel mode) και 64 διαστήματα για μακροεντολές που επιτρέπεται να εκτελούνται στο κανονικό περιβάλλον εκτέλεσης εντολών για τον χρήστη (user mode).

Ενα βασικό χαρακτηριστικό του περιβάλλοντος εκτέλεσης είναι ότι δεν πρέπει ο εκτελούμενος κώδικας να περιέχει σφάλματα ή ανεξέλεγκτες προσπελάσεις στη μνήμη καθώς η αρχιτεκτονική δεν καθορίζει την κατάσταση στην οποία θα βρεθεί το σύστημα αν προκύψει διακοπή που θα οφείλεται στον εκτελούμενο κώδικα Pal. Ωστόσο, στο λειτουργικό σύστημα Digital Unix σφάλματα κατά την εκτέλεση του κώδικα Pal εξυπηρετούνται μέσω του μηχανισμού εξυπηρέτησης διακοπών και μεταφέρονται στον χρήστη σαν διακοπή λόγω εκτέλεσης ‘‘παράνομης’’ εντολής (illegal instruction trap). Αυτή η δυνατότητα μας επιτρέπει να μπορούμε να ελέγχουμε την προστασία των διευθύνσεων και από το περιβάλλον εκτέλεσης του κώδικα Pal απλουστεύοντας έτσι, τους απαιτούμενους ελέγχους πριν την μετάβαση από το ένα περιβάλλον εκτέλεσης στο άλλο. Το λειτουργικό σύστημα Mach δεν παρέχει παρόμοιο μηχανισμό και τα σφάλματα κατά την εκτέλεση του κώδικα Pal οδηγούν σε ανεξέλεγκτες καταστάσεις.

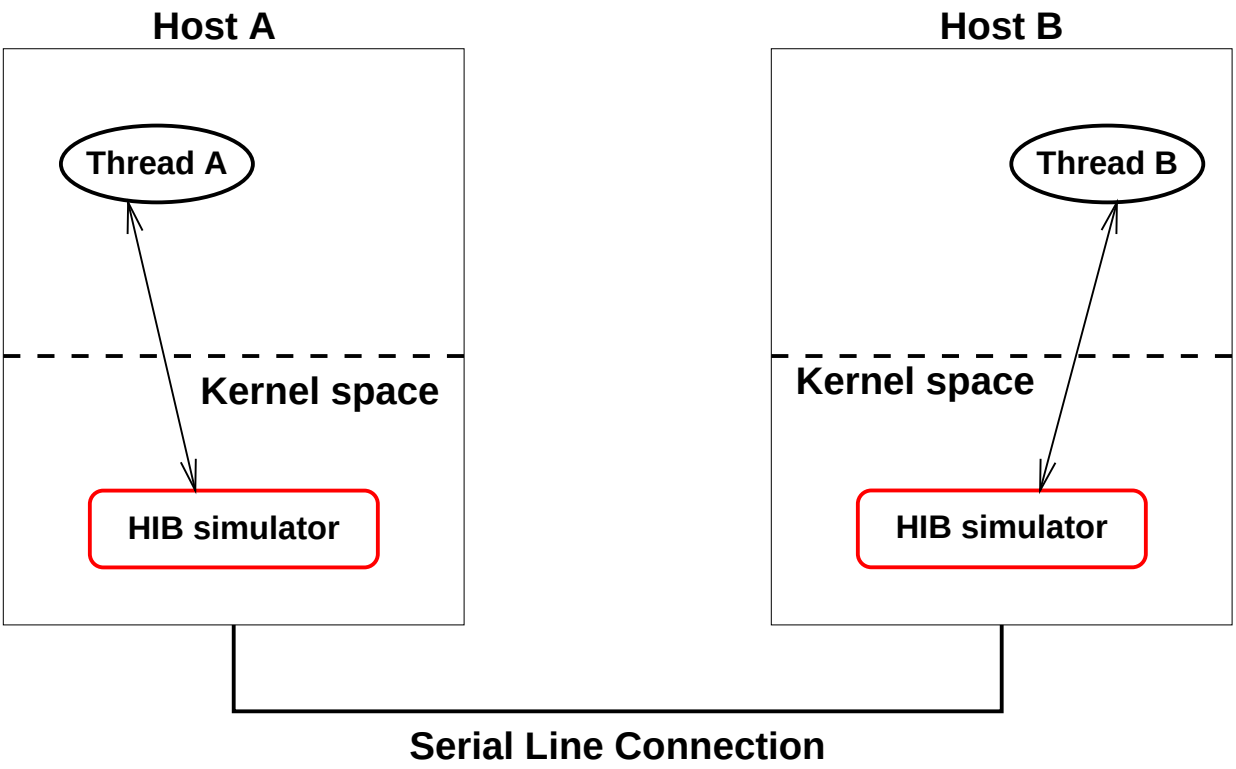
Κεφάλαιο 3

Προσομοιωτής

Για την σχεδίαση και μελέτη όλων των τμημάτων του λογισμικού υποστήριξης κατασκευάστηκε, παράλληλα με το πραγματικό πρωτότυπο, ένας προσομοιωτής λειτουργίας του υποσυστήματος διασύνδεσης του υπολογιστή με το δίκτυο του Τηλέγραφου (Host Interface Board). Ο προσομοιωτής έπρεπε να καλύπτει όλες τις λειτουργίες του Τηλέγραφου Ι και ταυτόχρονα να παρέχει μια όσο τον δυνατόν ρεαλιστικότερη εικόνα προσομοίωσης. Στη συνέχεια του κεφαλαίου περιγράφεται ο προσομοιωτής, αναλύονται βασικά σημεία της υλοποίησής του και των μηχανισμών που χρησιμοποιήθηκαν και τέλος περιγράφεται ο τρόπος διασύνδεσης του προσομοιωτή με το υπόλοιπο τμήμα του λογισμικού.

3.1 Περιγραφή

Η προσομοίωση αφορά το υποσύστημα διασύνδεσης (HIB), το οποίο περιγράφεται στην υποενότητα 2.2.2 και το οποίο, είναι υπεύθυνο για το σύνολο των λειτουργιών που υποστηρίζονται στο δίκτυο του Τηλέγραφου. Παρέχεται μόνο λειτουργική προσομοίωση του υποσυστήματος διασύνδεσης και όχι προσομοίωση σε επίπεδο γλώσσας μηχανής. Αυτό αρκεί για την υλοποίηση των υπολοίπων τμημάτων του λογισμικού. Ο προσομοιωτής είναι ενσωματωμένος στον πυρήνα του λειτουργικού συστήματος και οι λειτουργίες του παρέχονται στον χρήστη είτε σαν κλήσεις συστήματος είτε σαν κλήσεις της βιβλιοθήκης διασύνδεσης. Σαν μνήμη του υποσυστήματος διασύνδεσης χρησιμοποιείται η κύρια μνήμη του υπολογιστή. Σαν ‘‘πόρτα’’ διασύνδεσης με το υπόλοιπο δίκτυο χρησιμοποιήθηκε η σειριακή έξοδος του υπολογιστή. Το περιβάλλον προσομοίωσης αποτελούν 2 υπολογιστές DEC 3000/300, οι οποίοι είναι συνδεδεμένοι μέσω της σειριακής εισόδου/εξόδου που διαθέτουν. Σε κάθε έναν από τους υπολογιστές, ο πυρήνας του λειτουργικού τους συστήματος περιλαμβάνει τον προσομοιωτή του υποσυστήματος διασύνδεσης. Στο σχήμα 3.1 παρουσίαζεται σχηματικά η σχεδίαση του προσομοιωτή και το περιβάλλον ανάπτυξης της εργασίας αυτής.



Σχήμα 3.1: Προσομοίωση του δικτύου του Τηλέγραφου I

3.2 Υλοποίηση

Η έκδοση του λειτουργικού συστήματος Mach 3.0 για τους υπολογιστές που είχαμε στην διάθεση μας δεν ήταν πλήρης. Για τους υπολογιστές που έχουν τον επεξεργαστή Alpha–AXP έχει μεταφερθεί (ported) μόνο ο μικρο–πυρήνας και όχι και ο Unix Server ο οποίος παρέχει ένα πλήρες περιβάλλον προγραμματισμού Unix. Ετσι, για τις ανάγκες της εργασίας αυτής έπρεπε να περιοριστούμε στον μικρο–πυρήνα. Η εκτέλεση προγραμμάτων εκτός του πυρήνα (user mode) είναι περιορισμένη σε μία μόνο διεργασία (init process) από την οποία έχουμε την δυνατότητα εκτέλεσης κώδικα που χρησιμοποιεί μόνο κλήσεις συστήματος του πυρήνα και όχι κλήσεις συστήματος του Unix Server. Το κύριο τμήμα του προσομοιωτή αποτελεί η υποδιεργασία (thread) του πυρήνα που προσομοιώνει το υποσύστημα διασύνδεσης του Τηλέγραφου με τον υπολογιστή. Στην υποδιεργασία αυτή είναι υλοποιημένες όλες οι λειτουργίες του HIB.

Οι υποδιεργασίες του πυρήνα δεν μπορούν να διακοπούν κατά την εκτέλεσή τους για να εκτελεστεί κάποια άλλη υποδιεργασία. Η υποδιεργασία του προσομοιωτή ενεργοποιείται μόνο όταν υπάρχει είσοδος από την σειριακή είσοδο του υπολογιστή. Κατά συνέπεια,

η υποδιεργασία αυτή εξυπηρετεί μόνο τις αιτήσεις για πράξεις που προέρχονται από το προσομοιούμενο ‘‘δίκτυο’’ του Τηλέγραφου. Οι πράξεις που προέρχονται από τοπικές, στον υπολογιστή, διεργασίες, υλοποιούνται από την βιβλιοθήκη υποστήριξης ή από κλήσεις συστήματος που προστέθηκαν στον πυρήνα του λειτουργικού συστήματος για τον σκοπό αυτό.

Οταν το λειτουργικό σύστημα ξεκινά την εκτέλεσή του, αρχικοποιεί τις εσωτερικές δομές του, ελέγχει τις περιφερειακές συσκευές του συστήματος και ξεκινά τις υποδιεργασίες του που είναι υπεύθυνες για την διαχείριση της μνήμης, του επεξεργαστή και των συσκευών εισόδου/εξόδου. Στην φάση της αρχικοποίησης των υποδιεργασιών αρχικοποιείται και ο προσομοιωτής, αποκτά τον έλεγχο της σειριακής εισόδου/εξόδου και καθορίζεται η ταχύτητα και το είδος της επικοινωνίας που θα υπάρχει. Στην ίδια φάση αρχικοποιούνται οι εσωτερικές δομές του προσομοιωτή (μνήμη του HIB, λίστα αντιγράφων σελίδων (multicast list)) και ξεκινά η συνάρτηση κορμός του προσομοιωτή η οποία περιμένει συνεχώς για είσοδο από την σειριακή γραμμή εισόδου/εξόδου.

Η είσοδος από την σειριακή γραμμή ελέγχεται από άλλη υποδιεργασία του πυρήνα που είναι υπεύθυνη για την είσοδο και έξοδο από όλες τις συσκευές. Ο μικρο–πυρήνας του λειτουργικού συστήματος τροποποιήθηκε ώστε οι δύο υποδιεργασίες να συνεργάζονται και η είσοδος από την σειριακή γραμμή να ανακατευθύνεται στην υποδιεργασία του προσομοιωτή. Ετσι στην εσωτερική δομή που αναπαριστά τις αιτήσεις εισόδου/εξόδου από τις συσκευές του συστήματος προστέθηκε ένα επιπλέον πεδίο που καθορίζει αν η αίτηση προέρχεται από εξωτερική διεργασία ή αν πρόκειται για αίτηση εσωτερικής υποδιεργασίας (προσομοιωτής).

Ο διαχωρισμός αυτός είναι απαραίτητος γιατί είναι διαφορετική η αντιμετώπιση των αιτήσεων ανάλογα με την προέλευση τους. Οι εσωτερικές αιτήσεις προσπέλασης των συσκευών έχουν ήδη δεσμευμένους ενταμιευτές στον χώρο διευθύνσεων του πυρήνα ενώ για τις αιτήσεις των εξωτερικών διεργασιών (διεργασίες του χρήστη) πρέπει να γίνει αντιγραφή των ενταμιευτών από τον χώρο διευθύνσεων του χρήστη στον χώρο των διευθύνσεων του πυρήνα. Επίσης, όταν η αίτηση εξυπηρετηθεί, ο τρόπος με τον οποίο στέλνεται η απάντηση διαφέρει ανάλογα με τον χώρο προέλευσης. Οι εξωτερικές αιτήσεις απαντώνται στέλνοντας μήνυμα μέσω του μηχανισμού αποστολής μηνυμάτων (IPC mechanism) που διαθέτει ο μικρο–πυρήνας του Mach. Για τις εσωτερικές αιτήσεις η απάντηση δεν χρειάζεται να γίνει μέσω μηνύματος αλλά αρκεί να σταλεί ένα σήμα στην υποδιεργασία του πυρήνα που έκανε την αίτηση για να την ενημερώσει ότι έχει εξυπηρετηθεί η αίτηση. Το αποτέλεσμα το παίρνει μέσω κοινών μεταβλητών (global variables) του πυρήνα.

Οι αιτήσεις εξόδου από την σειριακή γραμμή εξυπηρετούνται και αυτές από την υπο–διεργασία που διαχειρίζεται τις συσκευές. Και σε αυτή την περίπτωση ο διαχωρισμός των αιτήσεων σε εξωτερικές και εσωτερικές είναι απαραίτητος, ώστε να μην γίνονται αντιγραφές

των ενταμιευτών και να αποστέλονται στον προσομοιωτή εσωτερικά σήματα και όχι μηνύματα, κάθε φορά που μια αίτηση εξόδου ολοκληρώνεται.

Οι παραπάνω μεταβολές έγιναν στο επίπεδο διαχείρισης των συσκευών εισόδου χαρακτήρων (character devices) και όχι στο επίπεδο διαχείρισης της σειριακής γραμμής εισόδου/εξόδου. Αυτό σημαίνει ότι είναι ανεξάρτητες από την σειριακή γραμμή εισόδου/εξόδου και κατά συνέπεια, ο προσομοιωτής μπορεί να χρησιμοποιήσει οποιαδήποτε άλλη διαθέσιμη συσκευή επιτρέπει την μεταφορά χαρακτήρων, με την προϋπόθεση ότι διαθέτει έλεγχο ροής.

Ολες οι λειτουργίες του Τηλέγραφου είναι συναρτήσεις της βιβλιοθήκης υποστήριξης. Οταν η διεργασία θέλει να εκτελέσει κάποια πράξη σε διευθύνσεις μνήμης που είναι τοπικές στον υπολογιστή αυτές εκτελούνται με κλήση συστήματος. Οι κλήσεις συστήματος εκτελούνται στο περιβάλλον του πυρήνα και με αυτό τον τρόπο εξασφαλίζουμε την ατομική εκτέλεση των ειδικών λειτουργιών. Για τις απλές αναγνώσεις και εγγραφές δεν υπάρχει η ανάγκη να γίνονται με κλήση συστήματος. Οι πράξεις αυτές υλοποιούνται μέσω της βιβλιοθήκης υποστήριξης χωρίς να χρειάζεται η μετάβαση στο περιβάλλον του πυρήνα. Αυτό γιατί στην αρχή της διεργασίας η μνήμη του δικτύου του Τηλέγραφου (μνήμη στον πυρήνα) αντιστοιχίζεται στις ιδεατές διευθύνσεις της διεργασίας του χρήστη και έτσι έχει την δυνατότητα άμεσης προσπέλασης.

Για τις πράξεις σε "μακρυνές" διευθύνσεις, η βιβλιοθήκη υποστήριξης στέλνει την αίτηση για εκτέλεση πράξης στο μακρυνό υπολογιστή μέσω του πυρήνα του λειτουργικού συστήματος, όπως περιγράφεται στην επόμενη υποενότητα. Οι λειτουργίες των ατομικών πράξεων και των απομακρυσμένων αναγνώσεων είναι σύγχρονες και πρέπει οι διεργασίες που τις εκτελούν να περιμένουν χωρίς όμως να εμποδίζουν την εκτέλεση των άλλων διεργασιών. Αυτό επιτυγχάνεται γιατί όλες οι απομακρυσμένες πράξεις εκτελούνται σαν κλήσεις συστήματος. Οταν μια διεργασία περιμένει για το αποτέλεσμα μιας απομακρυσμένης πράξης η υποδιεργασία του πυρήνα στην οποία εκτελείται, μπαίνει σε μια ουρά αναμονής για το αποτέλεσμα. Η υποδιεργασία του προσομοιωτή είναι υπεύθυνη να την βγάλει από την ουρά αναμονής αμέσως μόλις έρθει το αποτέλεσμα της πράξης της. Υπάρχει περίπτωση, πολλές διεργασίες να περιμένουν για το αποτέλεσμα απομακρυσμένων λειτουργιών. Κάθε έκδοση απομακρυσμένης πράξης έχει έναν μοναδικό αριθμό που την διακρίνει και με βάση αυτόν τον αριθμό γίνεται η αφύπνιση της διεργασίας που περιμένει το αποτέλεσμα.

Η πράξη της ασύγχρονης ανάγνωσης εκτελείται από την υποδιεργασία του προσομοιωτή χωρίς να εμποδίζεται η εκτέλεση της διεργασίας που την κάλεσε. Οταν η διεργασία ζητήσει μια ασύγχρονη ανάγνωση, η κλήση συστήματος επιστρέφει αμέσως. Το αποτέλεσμα, όταν έρθει, αποθηκεύεται στην δοσμένη διεύθυνση.

Ο προσομοιωτής είναι υπεύθυνος ακόμη, για την ενημέρωση των τοπικών σελίδων που

έχουν αντιγράφα σε άλλους υπολογιστές. Κάθε φορά που γίνεται κάποια πράξη εγγραφής σε τοπικές διευθύνσεις ο προσομοιωτής ελέγχει αν η σελίδα διαθέτει αντίγραφα σε άλλους υπολογιστές και στέλνει αυτόματα σε αυτούς πράξεις εγγραφής στις αντίστοιχες διευθύνσεις[1].

3.2.1 Επικοινωνία μεταξύ των πυρήνων

Ενα από τα αρχικά βασικά προβλήματα της υλοποίησης του προσομοιωτή ήταν η επικοινωνία των υποδιεργασιών που εκτελούνται στους δύο διαφορετικούς υπολογιστές. Ο πυρήνας του Mach λόγω της micro−kernel αρχιτεκτονικής του, παρέχει μόνο βασική υποστήριξη των συσκεύων που διαθέτει ο υπολογιστής. Παρέχονται οι βασικές λειτουργίες για είσοδο και έξοδο δεδομένων και για έλεγχο της κατάστασης των συσκευών αλλά δεν υλοποιεί κανένα πρωτόκολλο επικοινωνίας για τις υποστηριζόμενες συσκευές. Την ευθύνη για τα πρωτόκολλα αυτά την έχει ο Unix Server.

Η υποδιεργασία του προσομοιωτή ωστόσο, είχε ανάγκη από ένα σταθερό πρωτόκολλο επικοινωνίας που θα πρόσφερε ασφαλή μεταφορά δεδομένων. Ετσι, για την επικοινωνία των δύο πυρήνων χρησιμοποιήθηκε η σειριακή γραμμή εισόδου/εξόδου του υπολογιστή[2]. Ο ελεγκτής της σειριακής γραμμής [2][1] υποστηρίζει έλεγχο ροής (hardware flow control). Στην έκδοση του λειτουργικού συστήματος Mach για τους υπολογιστές που χρησιμοποιήσαμε, ο κώδικας διαχείρισης του ελεγκτή της σειριακής γραμμής δεν υποστήριζε το συγκεκριμένο μοντέλο και χρειάστηκε να υποστηριχθεί για να μπορέσουμε να την χρησιμοποιήσουμε. Τελικά, υλοποιήθηκε σαν μέρος του μικρο−πυρήνα, ένα πρωτόκολλο για hardware flow control και έτσι είχαμε την δυνατότητα να χρησιμοποιούμε την σειριακή γραμμή εισόδου/εξόδου με τη μέγιστη δυνατή ταχύτητα των $38.4 Kbps$. Το πρωτόκολλο χρησιμοποιεί δύο επιπλέον σήματα της σειριακής γραμμής. Το σήμα CTS (clear to send) του ενός υπολογιστή συνδέεται με το σήμα RTS (ready to send) του άλλου. Ο κάθε υπολογιστής μπορεί να στείλει δεδομένα μόνο αν το σήμα RTS είναι ενεργό. Το σήμα RTS απενεργοποιείται κάθε φορά που η ουρά εισόδου του παραλήπτη ξεπερνά το πάνω όριο (high watermark) και επανενεργοποιείται όταν το μέγεθος της ουράς εισόδου γίνει μικρότερο από το κάτω όριο (low watermark).

Η είσοδος χαρακτήρων από την σειριακή γραμμή είναι ασύγχρονη και γίνεται μέσω διακοπής. Οταν συγκεντρωθούν στην ουρά εισόδου, χαρακτήρες πάνω από ένα συγκεκριμένο άνω όριο που έχει τεθεί, η υποδιεργασία του πυρήνα που ελέγχει την είσοδο χαρακτήρων από την σειριακή γραμμή, στέλνει σήμα στην υποδιεργασία του προσομοιωτή και την βγάζει

[1]Στην υπάρχουσα υλοποίηση χρησιμοποιήθηκε σταθερό σύνολο σελίδων μνήμης που αντιγράφονται στην μνήμη του άλλου Host Interface Board καθώς δεν έχει υλοποιηθεί σύστημα διαχείρισης μνήμης (memory manager) για τις σελίδες μνήμης του δικτύου του Τηλέγραφου

[2]Στην αρχική φάση της υλοποίησης προτιμήθηκε να χρησιμοποιηθεί η συσκευή διασύνδεσης με το δίκτυο Ethernet, ώστε να υπάρχει η δυνατότητα περισσοτέρων υπολογιστών στο δίκτυο. Δυστυχώς θα έπρεπε να υλοποιηθεί κάποιο πρωτόκκολο αντίστοιχο του TCP, κάτι που ξεφεύγει από τα πλαίσια της εργασίας αυτής.

από την κατάσταση αναμονής για είσοδο δεδομένων. Για την έξοδο δεδομένων υπάρχει μια ουρά αναμονής στην οποία εισέρχονται οι αιτήσεις για έξοδο δεδομένων και ικανοποιούνται ασύγχρονα αλλά πάντα με την ίδια σειρά με την οποία εισέρχονται.

Για να αποφευχθούν τα προβλήματα της ασύγχρονης μετάδοσης ο προσομοιωτής χρησιμοποιεί μηνύματα σταθερού μεγέθους και χρησιμοποιεί επιπλέον ενταμιευτές που κρατούν την είσοδο των χαρακτήρων για να εξασφαλίζεται έτσι, η ασφαλής εκτέλεση όλων των αιτήσεων που έρχονται από το προσομοιούμενο δίκτυο.

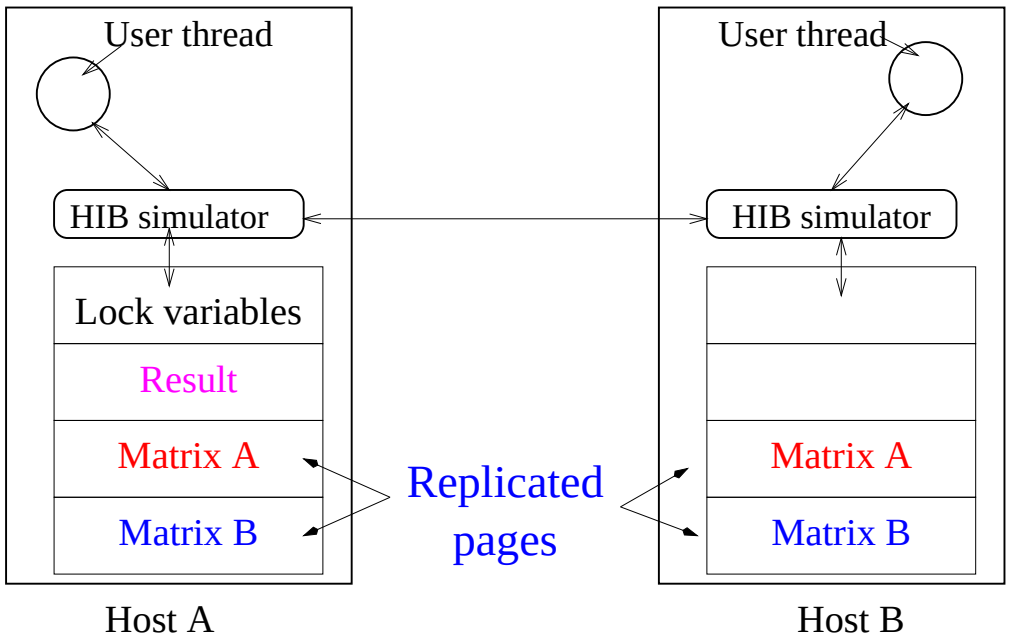
3.2.2 Η μνήμη του συστήματος

Ο προσομοιωτής χρησιμοποιεί την μνήμη του υπολογιστή στον οποίο εκτελείται για να προσομοιώσει την μνήμη του υποσυστήματος διασύνδεσης του Τηλέγραφου. Για την μνήμη αυτή δεν υπάρχει σύστημα διαχείρισης και έτσι η διευθυνσιοδότηση είναι απόλυτη (χωρίς μετάφραση). Οι δοκιμαστικές εφαρμογές που αναπτύχθηκαν χρησιμοποιούν τις απόλυτες διευθύνσεις. Ωστόσο, επειδή η μνήμη αυτή είναι ακέραιος αριθμός σελίδων του πυρήνα, η τοπική μνήμη του δικτύου αντιστοιχίζεται στην διεργασία και έτσι υπάρχει η δυνατότητα οι πράξεις εγγραφής και ανάγνωσης από αυτές τις διευθύνσεις να γίνονται χωρίς παρέμβαση του λειτουργικού. Η κλήση συστήματος που υλοποιήθηκε για αυτό το σκοπό υλοποιεί και την αντιστοίχιση της φυσικής μνήμης του Τηλέγραφου Ι στον χώρο των διευθύνσεων του χρήστη.

3.2.3 Ελεγχος της ορθότητας του προσομοιωτή

Ο προγραμματιστής μπορεί να χρησιμοποιήσει τον προσομοιωτή για να αναπτύξει εφαρμογές που χρησιμοποιούν την λειτουργικότητα της αρχιτεκτονικής του Τηλέγραφου. Στον προγραμματιστή παρέχεται μια βιβλιοθήκη υποστήριξης (περιγράφεται αναλυτικά στο επόμενο κεφάλαιο) η οποία περιλαμβάνει συναρτήσεις για όλες τις λειτουργίες του Τηλέγραφου. Για τον έλεγχο της ορθότητας του προσομοιωτή δοκιμάστηκαν απλές εφαρμογές που κάνουν χρήση των λειτουργιών του Τηλέγραφου και δείχνουν τρόπους χρήσης του δικτύου του Τηλέγραφου. Ετσι, μια εφαρμογή είναι η υλοποίηση ενός απλού προγράμματος παραγωγού–καταναλωτή (producer–consumer) όπου ο παραγωγός είναι μια διεργασία στον έναν υπολογιστή και ο καταναλωτής είναι μια διεργασία στον άλλο και κάνουν χρήση των ατομικών πράξεων για να υλοποιούν αμοιβαίο αποκλεισμό (mutual exclusion) στην χρήση των δεδομένων. Επίσης, με τις παρεχόμενες ατομικές πράξεις, γίνεται και μια απλή υλοποίηση σημαφόρων (semaphores). Η εφαρμογή δείχνει την χρήση του Τηλέγραφου για γρήγορη επικοινωνία και ανταλλαγή μηνυμάτων μεταξύ διεργασιών διαφορετικών υπολογιστών.

Μια άλλη εφαρμογή που δοκιμάστηκε, είναι ο παράλληλος πολλαπλασιασμός πινάκων. Η εφαρμογή πολλαπλασιάζει δύο πίνακες οι οποίοι είναι αποθηκευμένοι στην μνήμη του ενός

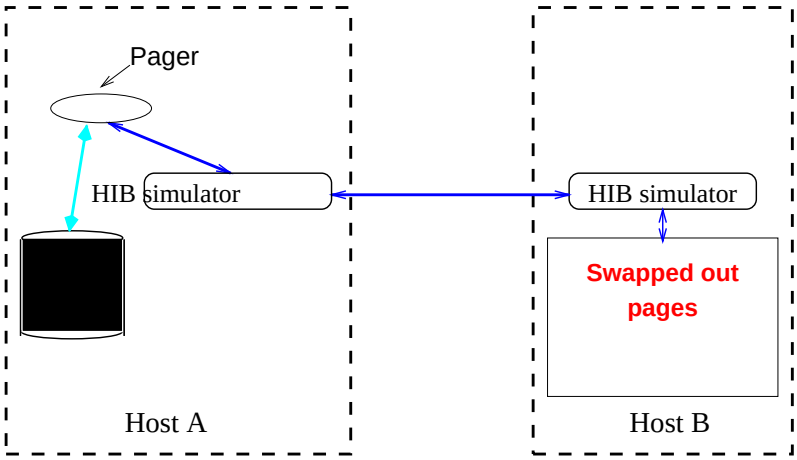


Σχήμα 3.2: Τοποθέτηση των πινάκων και των μεταβλητών στην κοινή μνήμη του Τηλέγραφου για την εφαρμογή πολλαπλασιασμού πινάκων.

υπολογιστή αλλά αντίγραφα των σελίδων μνήμης υπάρχουν και στον άλλον υπολογιστή. Οι μεταβλητές στις οποίες γίνονται ατομικές πράξεις δεν μπορούν να είναι σε σελίδες μνήμης οι οποίες διαθέτουν αντίγραφα. Σχηματικά, η τοποθέτηση των μεταβλητών φαίνεται στο σχήμα 3.2. Η εφαρμογή αυτή δείχνει ότι ο Τηλέγραφος παρέχει ένα ικανοποιητικό περιβάλλον εκτέλεσης για εφαρμογές κοινόχρηστης μνήμης.

Remote paging

Χρησιμοποιώντας τον προσομοιωτή υλοποιήθηκε μια εφαρμογή που επιτρέπει την χρήση της απομακρυσμένης μνήμης σαν αποθηκευτικό χώρο του συστήματος διαχείρισης ιδεατής μνήμης (paging device). Ο κλασσικός χώρος αποθήκευσης των σελίδων μνήμης που δεν χρησιμοποιούνται συχνά είναι ο σκληρός δίσκος (hard disk) του υπολογιστή. Η τεχνική της απομακρυσμένης αποθήκευσης σελίδων μνήμης, η οποία περιγράφεται με περισσότερες λεπτομέρειες στα [24, 23], μας επιτρέπει να χρησιμοποιούμε την μνήμη του υπόλοιπου δικτύου σαν ένα επιπλέον επίπεδο στην ιεραρχία της ιδεατής μνήμης. Η ταχύτητα προσπέλασης στην απομακρυσμένη μνήμη κάνει πολύ ελκυστική μια τέτοια εφαρμογή για τα πραγματικά συστήματα και ιδιαίτερα για την εκτέλεση σειριακών εφαρμογών που απαιτούν μεγάλα ποσά μνήμης.



Σχήμα 3.3: Χρήση της απομακρυσμένης μνήμης σαν αποθηκευτικός χώρος (swap space).

Στην υλοποίηση που έγινε, στο λειτουργικό σύστημα Mach, η υποδιεργασία του λειτουργικού συστήματος, που αναλαμβάνει να μεταφέρει τις σελίδες της ιδεατής μνήμης στον δίσκο τροποποιήθηκε ώστε να χρησιμοποιεί πρώτα, την διαθέσιμη απομακρυσμένη μνήμη. Η υποδιεργασία του σελιδοδιαχειριστή (pager) εκτελείται κάθε φορά που η διαθέσιμη ελεύθερη μνήμη του συστήματος ξεπεράσει το κάτω όριο που έχει θέσει ο διαχειριστής της μνήμης. Οι σελίδες που μεταφέρονται στον αποθηκευτικό χώρο είναι αυτές που έχουν παραμείνει για μεγάλο χρονικό διάστημα ανενεργές. Ο τροποποιημένος σελιδοδιαχειριστής γνωρίζει την διαθέσιμη απομακρυσμένη μνήμη στο σύστημα του Τηλέγραφου και στέλνει τις σελίδες που πρέπει να μεταφερθούν στον αποθηκευτικό χώρο, στην απομακρυσμένη μνήμη. Για κάθε σελίδα μνήμης που αποστέλλεται, ενημερώνεται μια δομή που κρατά τον κόμβο και την απομακρυσμένη σελίδα στην οποία έχει αποθηκευτεί. Η αποστολή των σελίδων γίνεται εκτελώντας απομακρυσμένες εγγραφές. Αντίστοιχα η ανάκτηση των σελίδων γίνεται με απομακρυσμένες αναγνώσεις. Οταν η διαθέσιμη τηλε–μνήμη εξαντληθεί, χρησιμοποιείται ο σκληρός δίσκος του συστήματος.

Κάθε σελίδα μνήμης έχει μέγεθος 8192 bytes. Αυτό σημαίνει ότι για την αποστολή της χρειάζονται 2048 απομακρυσμένες εγγραφές. Κάθε απομακρυσμένη εγγραφή αποστέλλεται σαν ένα πακέτο που περιέχει τα δεδομένα και την διεύθυνση στην οποία εγγράφονται. Για την αποστολή όμως, δεδομένων που γράφονται σε συνεχόμενες θέσεις μνήμης χρειάζεται να ξέρουμε μόνο την αρχική διεύθυνση και το μέγεθος των δεδομένων που θα αποσταλούν. Για να εξαλείψουμε το (μη αναγκαίο) κόστος των ξεχωριστών απομακρυσμένων εγγραφών ο προσομοιωτής τροποποιήθηκε ώστε να υποστηρίζει εντολές αποστολής μηνυμάτων μεγέθους

σελίδας μνήμης. Μια τέτοια τεχνική μειώνει τον αριθμό των μεταφερόμενων δεδομένων και μοιάζει με τις τεχνικές DMA που παρέχουν άλλα συστήματα διασύνδεσης υπολογιστών[3].

Η τεχνική της χρήσης της απομακρυσμένης μνήμης για αποθήκευση των ανενεργών σελίδων του συστήματος σε συνδυασμό με την υποστήριξη των απομακρυσμένων αναγνώσεων από τον Τηλέγραφο παρέχει την δυνατότητα βελτίωσης της απόκρισης του συστήματος διαχείρισης της ιδεατής μνήμης.

[3]Ο Τηλέγραφος ΙΙ παρέχει υποστήριξη για DMA.

Κεφάλαιο 4

Βιβλιοθήκη υποστήριξης χρήστη

Η βιβλιοθήκη υποστήριξης περιλαμβάνει όλες εκείνες τις ρουτίνες που χρειάζονται ώστε να παρέχεται στον προγραμματιστή ένα πλήρες περιβάλλον διασύνδεσης με το δίκτυο του Τηλέγραφου αξιοποιώντας ταυτόχρονα τα χαρακτηριστικά του. Το πρώτο μέρος του κεφαλαίου αφορά την υποστήριξη των ατομικών πράξεων. Περιγράφονται τρεις διαφορετικοί τρόποι υλοποίησης, ανάλογα με την υποστήριξη που μας παρέχεται από το λειτουργικό σύστημα και τον επεξεργαστή. Στην συνέχεια περιγράφεται ένα σύνολο από επικουρικές διαδικασίες που είναι απαραίτητες για την υποστήριξη άλλων υποσυστημάτων του λειτουργικού (π.χ διαχείρισης μνήμης).

4.1 Ατομικές πράξεις

Ο Τηλέγραφος Ι παρέχει ένα σύνολο από ατομικές πράξεις (υποενότητα 2.2.2) οι οποίες εκδίδονται χρησιμοποιώντας τους ειδικούς καταχωρητές του πίνακα 4.1.

Operation	Address
Fetch and Store	7FE0008
Fetch and Add	7FE0009
Compare and Swap	7FE000A
Remote Fetch	7FE000B

Πίνακας 4.1: Διευθύνσεις των ειδικών καταχωρητών έκδοσης ατομικών πράξεων.

Για κάθε ατομική πράξη υπάρχει ένας διαφορετικός καταχωρητής κοινός για όλες τις διεργασίες που εκτελούνται στον υπολογιστή. Ετσι, η έκδοση των ατομικών πράξεων πρέπει να εξασφαλίζει ότι γίνονται όλες οι προσπελάσεις στους ειδικούς καταχωρητές από την ίδια

διεργασία ατομικά, χωρίς να υπάρχει περίπτωση διακοπής από κάποια άλλη διεργασία. Οι ατομικές πράξεις μπορούν να γίνουν σε οποιαδήποτε διεύθυνση της κοινής μνήμης αρκεί η σελίδα στην οποία ανήκει να μην έχει κάποιο αντίγραφο.

Η βιβλιοθήκη ορίζει τον τύπο της ατομικής μεταβλητής (atomic variable) και επιτρέπει μόνο σε τέτοιου τύπου μεταβλητές να εκτελούνται ατομικές πράξεις. Οι ατομικές μεταβλητές ανατίθενται δυναμικά (dynamically allocated) από το σύστημα διαχείρισης μνήμης, το οποίο επιτρέπει ανάθεση μνήμης σε σελίδες για τις οποίες εξασφαλίζει ότι δεν αποκτούν αντίγραφα. Ετσι, η βιβλιοθήκη υποστήριξης, πριν ξεκινήσει την ατομική πράξη, πρέπει να εξασφαλίσει ότι η ατομική πράξη γίνεται σε σελίδα μνήμης που δεν έχει αντίγραφα. Η λύση που ακολουθείται είναι ότι πρώτα ακυρώνονται τα αντίγραφα και ο κόμβος που θέλει να κάνει την ατομική πράξη κρατά την αποκλειστική κυριότητα της σελίδας. Καθώς λοιπόν, δεν είναι δυνατόν να γίνονται ατομικές πράξεις μόνο με προσπελάσεις σε τοπική μνήμη και αναγκαστικά κάποιες διεργασίες θα πρέπει να κάνουν απομακρυσμένες, η πολιτική που επιλέχτηκε δεν περιορίζει την απόδοση του συστήματος. Στη βιβλιογραφία υπάρχει αλγόριθμος ο οποίος εξασφαλίζει σταθερό αριθμό απομακρυσμένων προσπελάσεων ($O(1)$), όταν πολλές διεργασίες συναγωνίζονται να προσπελάσουν αποκλειστικά μια ατομική μεταβλητή [26]. Ο αλγόριθμος αυτός (MCS lock) χρησιμοποιεί μια λίστα με όλους τους κόμβους που θέλουν να κάνουν αποκλειστική χρήση της ίδιας μεταβλητής. Η λίστα είναι προσπελάσιμη από όλους τους κόμβους και η απομακρυσμένη προσπέλαση που γίνεται αφορά την εισαγωγή στην λίστα. Κάθε κόμβος που εισάγει τον εαυτό του στην λίστα περιμένει για να πάρει την αποκλειστική πρόσβαση εκτελώντας επαναληπτικές προσπελάσεις σε τοπική διεύθυνση μνήμης (busy waiting). Ο κάτοχος του δικαιώματος αποκλειστικής πρόσβασης είναι υπεύθυνος για να μεταβιβάσει το δικαίωμα αποκλειστικής χρήσης στον επόμενο κόμβο της λίστας. Από όσο μας είναι γνωστό, δεν υπάρχει σύστημα που να μπορεί να εκτελεί ατομικές πράξεις σε σελίδα μνήμης που διαθέτει ταυτόχρονα, πολλά αντίγραφα.

Τα βήματα εκτέλεσης μιας ατομικής πράξης είναι τα ακόλουθα:

- γίνεται εγγραφή στον ειδικό καταχωρητή της ατομικής πράξης του τελεστέου της ατομικής πράξης·
- γίνεται μια ‘‘δοκιμαστική’’ εγγραφή στην διεύθυνση στην οποία θα γίνει η ατομική πράξη. Η εγγραφή αυτή δεν γίνεται, αλλά ο σκοπός της είναι να γίνει η μεταφορά της φυσικής διεύθυνσης στο υποσύστημα διασύνδεσης του υπολογιστή με τον Τηλέγραφο και ταυτόχρονα να γίνει έλεγχος της προστασίας που παρέχει το σύστημα·
- τελικά, γίνεται μια ανάγνωση από την διεύθυνση της ατομικής πράξης. Οταν η πράξη της ανάγνωσης τελειώσει, έχουμε εξασφαλίσει ότι έχει γίνει η ατομική πράξη.

Οπως αναφέρεται και στην υποενότητα 2.2.2, δεν εξασφαλίζεται ότι οι αναγνώσεις στον Τηλέγραφο Ι μπορούν να ολοκληρωθούν πριν το υποσύστημα διασύνδεσης (HIB) χάσει τον έλεγχο του διαύλου. Ετσι η τελευταία εντολή κάθε ατομικής πράξης, που είναι η ανάγνωση πρέπει να αντιμετωπιστεί με ασφαλή τρόπο ώστε να εξασφαλίζεται η ατομικότητα.

4.1.1 Υλοποίηση με κλήσεις συστήματος

Οι προϋποθέσεις εκτέλεσης των ατομικών πράξεων εξασφαλίζονται πλήρως, αν αυτές εκτελούνται στο περιβάλλον του πυρήνα του λειτουργικού συστήματος.

Ετσι μια υλοποίηση των ατομικών πράξεων έγινε με κλήσεις συστήματος. Για το σκοπό αυτό επεκτάθηκε το σύνολο των κλήσεων συστήματος του λειτουργικού συστήματος Mach. Κάθε κλήση συστήματος παίρνει σαν ορίσματα έναν δείκτη σε μια ατομική μεταβλητή και τον τελεστέο της ατομικής πράξης. Μέσα στο περιβάλλον του πυρήνα μπορούμε να ελέγχουμε τις διακοπές εξασφαλίζοντας την εκτέλεση του συνόλου των προσπελάσεων στην ατομική μεταβλητή. Ο έλεγχος της προστασίας της διεύθυνσης της ατομικής μεταβλητής γίνεται αυτόματα από το λειτουργικό σύστημα, με την δοκιμαστική εγγραφή που γίνεται στο δεύτερο βήμα έκδοσης των ατομικών πράξεων. Σε περίπτωση λάθους, το σύστημα προκαλεί διακοπή της διεργασίας (segmentation fault). Το βασικό μειονέκτημα της υλοποίησης των ατομικών πράξεων με κλήσεις συστήματος είναι το μεγάλο κόστος εκτέλεσής τους. Με μετρήσεις που έγιναν στο λειτουργικό σύστημα Mach υπολογίστηκε ότι το κόστος εκτέλεσης μιας κλήσης συστήματος χωρίς κώδικα (null system call) είναι 6.7 μsecs. Ωστόσο για να μπορέσουμε να εξασφαλίσουμε την ασφαλή εκτέλεση των ατομικών πράξεων και των αναγνώσεων είμαστε αναγκασμένοι να τις υλοποιήσουμε σαν κλήσεις συστήματος.

Αυτός ο περιορισμός προκύπτει γιατί η σχεδίαση του Τηλέγραφου Ι δεν εξασφαλίζει ότι θα εκτελούνται οι αναγνώσεις εντός των χρονικών ορίων που καθορίζει η υλοποίηση του TURBOchannel . Ετσι αν κάποια ανάγνωση διακοπεί είμαστε αναγκασμένοι να διαχειριστούμε την διακοπή που θα προκαλέσει ο ελεγκτής του TURBOchannel και ο μόνος ασφαλής τρόπος για να το κάνουμε αυτό στο λειτουργικό σύστημα Mach, είναι να βρισκόμαστε στο περιβάλλον εκτέλεσης του πυρήνα. Στον πυρήνα μόνο, μπορούμε να διαχειριστούμε τις διακοπές που προκύπτουν από τον ελεγκτή του TURBOchannel και να επιστρέψουμε το αποτέλεσμα μέσω του ειδικού καταχωρητή ανάγνωσης. Αυτός ο περιορισμός επιβαρύνει κατά πολύ το κόστος όλων των ατομικών πράξεων και όλων των αναγνώσεων του συστήματος, ενώ ταυτόχρονα δεν μπορούμε πια να εκτελούμε αναγνώσεις από την κοινή μνήμη του συστήματος με απλές εντολές *load* αλλά είμαστε αναγκασμένοι να εκτελέσουμε κλήση συστήματος.

4.1.2 Υλοποίηση με κώδικα PAL

Για την αρχιτεκτονική του επεξεργαστή Alpha AXP, υπάρχει ένας επιπλέον τρόπος για να μπορούμε να εξασφαλίσουμε την εκτέλεσης ατομικών πράξεων. Αυτός είναι με την χρήση κώδικα Pal του οποίου τα χαρακτηριστικά περιγράφονται στην υποενότητα 2.4. Το περιβάλλον εκτέλεσης του κώδικα Pal μας εξασφαλίζει από μόνο του την ατομικότητα της ακολουθίας των εντολών που απαιτούνται για την έκδοση μιας ατομικής πράξης. Ωστόσο στο περιβάλλον του κώδικα Pal δεν έχουμε την δυνατότητα να χειριστούμε λάθη σε περίπτωση που η ''δοκιμαστική'' εγγραφή στην διεύθυνση της ατομικής μεταβλητής αποτύχει. Ετσι θα πρέπει ο έλεγχος της διεύθυνσης να γίνει έξω από το περιβάλλον εκτέλεσης του κώδικα Pal. Για το σκοπό αυτό η ατομική μεταβλητή έχει την παρακάτω δομή:

```
union atomic {
    double_data_t atomic_variable;
    struct  {
        data_t address_checker;
        data_t real_address;
    } item ;
};
```

Ο Τηλέγραφος κάνει προσπελάσεις σε μεταβλητές μεγέθους τεσσάρων bytes μόνο, και έτσι ο τύπος *data_t* είναι μεγέθους τεσσάρων bytes και αντιστοιχεί στον τύπο *integer* της γλώσσας προγραμματισμού C. Ο τύπος της μεταβλητής *atomic_variable* της δομής της ατομικής μεταβλητής είναι μεγέθους οκτώ bytes (τύπος *long* για τους υπολογιστές μας) και ο σκοπός του είναι να μας εξασφαλίζει ότι κάθε ατομική μεταβλητή θα ανατίθεται στην ίδια σελίδα μνήμης.

Για να γίνει ο έλεγχος της προστασίας της διεύθυνσης της ατομικής μεταβλητής γίνεται μια εγγραφή στην διεύθυνση της μεταβλητής *address_checker*, και μέσω αυτής της δοκιμαστικής εγγραφής ελέγχεται η προστασία της σελίδας και η δυνατότητα προσπέλασης της. Αν δεν υπάρχει σφάλμα κατά την δοκιμαστική αυτή προσπέλαση, γίνεται η μετάβαση στο περιβάλλον εκτέλεσης του κώδικα Pal.

Ο κώδικας Pal δεν έχει μεγάλο κόστος εκτέλεσης. Από μετρήσεις που έγιναν στους υπολογιστές που χρησιμοποιήσαμε, υπολογίστηκε ότι το κόστος μετάβασης και επιστροφής

από το περιβάλλον εκτέλεσης κώδικα Pal είναι 15 με 20 κύκλοι του επεξεργαστή[1]. Αυτό μεταφράζεται σε περίπου 100 με 130 ns για κύκλο ρολογιού 150 MHz. Επίσης, ο κώδικας Pal που αφορά την υποστήριξη των ατομικών πράξεων είναι ανεξάρτητος από το λειτουργικό σύστημα. Στην υλοποίησή μας δοκιμάσαμε τον ίδιο κώδικα και για το λειτουργικό σύστημα Mach αλλά και για το λειτουργικό σύστημα Digital Unix. Ωστόσο, το μειονέκτημα του κώδικα Pal είναι ότι μπορεί να χρησιμοποιηθεί μόνο στους επεξεργαστές Alpha AXP. Ακόμη, δεν μπορούμε να χρησιμοποιήσουμε κώδικα Pal, αν δεν μπορούμε να εξασφαλίσουμε την εκτέλεση των αναγνώσεων χωρίς να υπάρχει περίπτωση διακοπής από τον ελεγκτή του διαύλου. Αν προκύψει διακοπή κατά την διάρκεια μιας ανάγνωσης που εκτελείται σε περιβάλλον εκτέλεσης κώδικα Pal, ο επεξεργαστής δεν θα μπορέσει ποτέ να εξυπηρετήσει την διακοπή και δεν θα μπορέσει να συνεχίσει.

Στο πρωτότυπο του Τηλέγραφου I, οι τοπικές αναγνώσεις εξασφαλίζεται ότι μπορούν να εκτελεστούν μέσα στα χρονικά περιθώρια που καθορίζει ο ελεγκτής του διαύλου του TURBOchannel. Αυτό γιατί δεν έχει υλοποιηθεί η δυνατότητα ύπαρξης αντιγράφων των τοπικών σελίδων μνήμης. Το παραπάνω χαρακτηριστικό, σε συνδυασμό με την κατάλληλη υλοποίηση των αναγνώσεων ώστε να επιστρέφουν αμέσως τον έλεγχο στον επεξεργαστή, μας επιτρέπει να υλοποιήσουμε τις ατομικές πράξεις αλλά και τις αναγνώσεις, σε κώδικα Pal.

Εξασφαλίζοντας ότι οι τοπικές αναγνώσεις θα είναι πάντα επιτυχείς μπορούμε να τις εκτελέσουμε με ασφάλεια στον κώδικα Pal. Και τα τρία στάδια έκδοσης μια ατομικής πράξης μπορούν να εκτελεστούν σε περιβάλλον Pal. Τώρα όμως, η τελευταία ανάγνωση (αν δεν γίνεται στην τοπική μνήμη) δεν επιστρέφει το σωστό αποτέλεσμα. Το σωστό αποτέλεσμα θα το πάρουμε κάνοντας επαναληπτικές προσπελάσεις (polling) στον ειδικό καταχωρητή κατάστασης. Οταν το πεδίο που καθορίζει ότι έχει έρθει το αποτέλεσμα τεθεί, μπορούμε να πάρουμε το αποτέλεσμα κάνοντας μια ανάγνωση από τον καταχωρητή που κρατά το αποτέλεσμα. Ολες οι παραπάνω αναγνώσεις είναι τοπικές και μπορούν να εκτελεστούν με ασφάλεια στο περιβάλλον του κώδικα Pal. Οπως προαναφέρθηκε, το κόστος εκτέλεσης κώδικα Pal είναι πολύ μικρό και δεν επιβαρύνει το κόστος των απομακρυσμένων αναγνώσεων, καθώς σε κάθε περίπτωση ο επεξεργαστής πρέπει να περιμένει το αποτέλεσμα της πράξης.

Η υλοποίηση όμως, των ρουτινών έκδοσης ατομικών πράξεων δεν μπορεί να γίνει σε 16 μόνο εντολές κώδικα Pal[2]. Για το σκοπό αυτό, έπρεπε να βρεθεί τρόπος ώστε να εξασφαλίσουμε την δυνατότητα εκτέλεσης περισσότερων εντολών στο περιβάλλον εκτέλεσης κώδικα Pal. Η ίδια η σχεδίαση του περιβάλλοντος αυτού έχει προβλέψει και υπάρχει δεσμευμένο ένα ενιαίο

[1]Η μέτρηση δεν είναι ακριβής γιατί εξαρτάται από τον χρόνο που θα κάνει ο επεξεργαστής να "αδειάσει" την pipeline.

[2]Στην πραγματικότητα, οι χρήσιμες εντολές που μπορούν να εκτελεστούν είναι πολύ λιγότερες επειδή ο επεξεργαστής εκτελεί 2 εντολές ταυτόχρονα (dual issue) αλλά και λόγω περιορισμών που θέτει το περιβάλλον εκτέλεσης κώδικα Pal.

τμήμα μνήμης, που χρησιμοποιείται για αυτό το σκοπό. Για την περιοχή της μνήμης αυτής είχαμε γνωστά μόνο την διεύθυνση από όπου ξεκινούσε και το μέγεθος που καταλάμβανε, ενώ για να μπορούμε να την χρησιμοποιήσουμε και εμείς θα έπρεπε να ξέρουμε και ποιά τμήματά της χρησιμοποιούνται από τις άλλες ρουτίνες του κώδικα Pal. Κάτι τέτοιο βέβαια, προϋποθέτει την ύπαρξη του πηγαίου κώδικα.

Ετσι έπρεπε να μπορέσουμε να χρησιμοποιήσουμε άλλες σελίδες μνήμης για τον ίδιο σκοπό. Ομως οι εντολές διακλάδωσης που υποστηρίζει ο επεξεργαστής Alpha δεν μας επέτρεπαν να χρησιμοποιήσουμε οποιαδήποτε σελίδα μνήμης και βέβαια θα έπρεπε να ξέρουμε εκ των προτέρων την φυσική διεύθυνση των σελίδων αυτών. Το πρόβλημα όμως λύθηκε αξιοποιώντας ένα χαρακτηριστικό που υπάρχει στο περιβάλλον εκτέλεσης κώδικα Pal. Στο περιβάλλον του κώδικα Pal υποστηρίζεται η ειδική εντολή *hw_rei*, η οποία επιτρέπει την μετάβαση από περιβάλλον εκτέλεσης Pal στο κανονικό περιβάλλον εκτέλεσης αλλά ακόμη, μπορεί να συμπεριφερθεί και σαν απλή εντολή διακλάδωσης, εσωτερικά του περιβάλλοντος Pal. Ετσι, αν βάλουμε στον ειδικό καταχωρητή που καθορίζει την ιδεατή διεύθυνση της επόμενης εντολής που θα εκτελεστεί (VPC internal register), την διεύθυνση στην οποία έχουμε τον δικό μας κώδικα Pal, μπορούμε να εκτελέσουμε περισσότερες εντολές στο περιβάλλον Pal. Με το τέχνασμα αυτό μπορούμε τελικά να εκτελέσουμε τον δικό μας Pal κώδικα δεσμεύοντας μερικές σελίδες του πυρήνα, ενώ μας χρειάζεται μόνο μια ρουτίνα Pal που θα εκτελεί τη διακλάδωση στο διεύθυνση που θα παίρνει σαν όρισμα. Η ρουτίνα αυτή θα πρέπει να ανήκει στο σύνολο των ρουτινών που έχουν δικαίωμα να εκτελούνται μόνο από το περιβάλλον του πυρήνα, ώστε να εξασφαλίσουμε την ασφαλή χρήση της. Λεπτομέρειες για την συγκεκριμένη υλοποίηση υπάρχουν στο παράρτημα Α.

4.1.3 Υποστήριξη συστημάτων χωρίς κώδικα Pal

Συστήματα που υποστηρίζουν κώδικα Pal (ή έχουν παρόμοια λειτουργικότητα) έχουν την δυνατότητα υλοποίησης ατομικών πράξεων από το περιβάλλον του χρήστη (user space) χωρίς την ανάγκη παρεμβολής του λειτουργικού συστήματος, όπως περιγράφηκε στην προηγούμενη υποενότητα. Επίσης, σε πολλές περιπτώσεις είναι σημαντικό να αποφύγουμε το επιπλέον κόστος των κλήσεων συστήματος, όταν το κόστος που προσθέτουν είναι πολύ μεγαλύτερο από το κόστος εκτέλεσης των ατομικών πράξεων από το hardware.

Σε συστήματα που δεν έχουμε περιβάλλον προγραμματισμού αντίστοιχο με το περιβάλλον εκτέλεσης κώδικα Pal μπορούμε εναλλακτικά να υλοποιήσουμε ένα πρωτόκολλο ανάμεσα στον πυρήνα του λειτουργικού συστήματος και στις διεργασίες του χρήστη, ώστε να είναι δυνατή η εκτέλεση τμημάτων διεργασιών χωρίς διακοπές από το λειτουργικό σύστημα ενώ ταυτόχρονα δεν θα επιβαρύνει τις διεργασίες αυτές με μεγάλο κόστος. Ενα τέτοιο πρωτόκολλο

παρουσίασαν για πρώτη φορά οι Jan Elder, Jim Lipkis και Edith Schonberg [16]. Το πρωτόκολλο αυτό, προτάθηκε αρχικά για να επιτρέπει την ασφαλή εκτέλεση κρίσιμων τμημάτων (critical sections) του κώδικα των διεργασιών (π.χ ασφαλή προσπέλαση σε κοινές δομές μεταξύ 2 διεργασιών), αλλά εξασφαλίζει και την δυνατότητα υποστήριξης ατομικών πράξεων από κλήσεις διεργασιών του χρήστη. Η λύση που προτείνεται είναι η ακόλουθη:

Ο πυρήνας του λειτουργικού συστήματος θα πρέπει να διαμοιράζεται με κάθε διεργασία, που θα θέλει να εκτελέσει τμήμα του κώδικα της ατομικά, σελίδα(ες) μνήμης ώστε να μπορούν να ανταλλάσουν μηνύματα. Η διεργασία που θα πρέπει να εκτελέσει κάποια ατομική πράξη θα πρέπει να ενημερώνει τον πυρήνα (μέσω μιας σημαίας (flag) της κοινής μνήμης) ότι θέλει να εκτελεστεί χωρίς διακοπές. Το σύστημα τότε, θα επιτρέπει στην διεργασία να εκτελείται για όσο χρονικό διάστημα θέλει, χωρίς να χάνει τον έλεγχο της CPU. Το χρονικό αυτό διάστημα δεν μπορεί να είναι απεριόριστο και συνήθως είναι μέχρι την επόμενη διακοπή του ρολογιού (clock interrupt).

Οταν το λειτουργικό σύστημα δει ότι η διεργασία έχει εξαντλήσει το χρονικό όριο για συνεχόμενη εκτέλεση χωρίς διακοπή, θα θέτει μια δεύτερη σημαία που θα ενημερώνει την διεργασία ότι δεν μπορεί να κρατά άλλο την CPU. Οι διεργασίες που θέλουν να ακολουθούν αυτό το πρωτόκολλο θα πρέπει να ελέγχουν, στα κρίσιμα σημεία του κώδικα τους, αν το λειτουργικό σύστημα τους έχει διακόψει (λόγω εξάντλησης του χρονικού ορίου) και θα πρέπει να φροντίζουν να αποδεσμεύουν το λειτουργικό σύστημα. Για το σκοπό αυτό θα υπάρχει μια ειδική κλήση συστήματος (*yield*) που θα αλλάζει την τρέχουσα διεργασία του συστήματος. Αν η ακολουθία των εντολών μπορεί να επαναληφθεί χωρίς να δημιουργείται πρόβλημα, η διεργασία μπορεί να επαναλάβει τα βήματα αργότερα.

Αν η διεργασία δεν ‘‘συμμορφωθεί’’, το λειτουργικό σύστημα θα διακόπτει την εκτέλεση της διεργασίας. Η διακοπή θα μπορεί να γίνεται αμέσως ή μετά από την πάροδο χρονικού διαστήματος ίσο με την περισσότερο αργή υποστηριζόμενη ατομική πράξη. Αν η διεργασία που είχε τον έλεγχο της CPU δεν έχει τελειώσει ακόμα, το λειτουργικό σύστημα δεν εξασφαλίζει την ορθότητα των αποτελεσμάτων της.

Η λύση αυτή μας επιτρέπει να εκτελούμε ατομικά τμήματα του προγράμματός μας χωρίς να χρειάζεται να γίνει κάποια κλήση συστήματος. Η ιδέα στηρίζεται στο γεγονός ότι το λειτουργικό σύστημα ‘‘εμπιστεύεται’’ τις διεργασίες του χρήστη και τους επιτρέπει (με προϋποθέσεις) να κρατήσουν τον έλεγχο του επεξεργαστή για όσο χρονικό διάστημα είναι απαραίτητο. Πρόβλημα παρουσιάζεται, αν κάποια ‘‘κακή’’ διεργασία θελήσει να δεσμεύσει τον επεξεργαστή για μεγάλο χρονικό διάστημα (π.χ., λόγω προγραμματιστικού λάθους). Αν όμως μπορούμε να εξασφαλίσουμε άνω φράγμα για τον χρόνο εκτέλεσης των ατομικών πράξεων, μπορούμε να ορίσουμε κατάλληλα τον μέγιστο επιτρεπόμενο χρόνο εκτέλεσης

μιας διεργασίας χωρίς διακοπές. Το λειτουργικό σύστημα μπορεί να επιβάλλει πολιτική δικαιοσύνης για την χρήση του επεξεργαστή μειώνοντας την προτεραιότητα όσων διεργασιών κάνουν χρήση του ‘‘δικαιώματος’’ εκτέλεσης χωρίς διακοπή.

Η λύση αυτή έχει το πλεονέκτημα ότι δεν χρειάζεται κάποιο ειδικό περιβάλλον εκτέλεσης εντολών (π.χ κώδικα Pal) για να υλοποιηθεί, και κατά συνέπεια είναι ανεξάρτητη από τον επεξεργαστή. Ταυτόχρονα το κόστος που προσθέτει είναι ελάχιστο. Απαιτούνται μόνο λίγες σελίδες μνήμης για να διαμοιράζονται με τις διεργασίες του χρήστη και μια παρέμβαση στον κώδικα του scheduler του λειτουργικού συστήματος, ώστε να ελέγχει αν μια διεργασία έχει ζητήσει να εκτελείται χωρίς διακοπές (έλεγχος μιας θέσης μνήμης) και να ελέγχει αν κάποια διεργασία έχει ξεπεράσει το μέγιστο επιτρεπόμενο χρονικό όριο αδιάλλειπτης εκτέλεσης. Ωστόσο, απαιτεί αλλαγή του πυρήνα του λειτουργικού συστήματος και συγκεκριμένα του χρονοδρομολογητή των διεργασιών (scheduler) και άρα πρόσβαση στον πηγαίο κώδικα του.

Για την αρχιτεκτονική του Τηλέγραφου, μπορούμε να ορίσουμε μέγιστο επιτρεπόμενο χρόνο εκτέλεσης για κάθε ατομική πράξη, ο οποίος είναι συνάρτηση του αριθμού των κόμβων του δικτύου. Ο αριθμός των κόμβων του δικτύου δεν μεταβάλλεται δυναμικά.

4.1.4 Σύγκριση της έκδοσης ατομικών πράξεων με τεχνικές άλλων συστημάτων

Ο *Τηλέγραφος Ι* διαθέτει ένα σύνολο καταχωρητών για την έκδοση των ατομικών πράξεων. Αυτό συνεπάγεται την ανάγκη για ατομική εκτέλεση των προσπελάσεων στους ειδικούς καταχωρητές έκδοσης ατομικών πράξεων. Η αρχιτεκτονική του Flash [19] χρησιμοποιεί για τον σκοπό αυτό διαφορετικό σύνολο καταχωρητών για κάθε διεργασία, οι οποίοι βρίσκονται στον ειδικό επεξεργαστή ελέγχου του δικτύου. Κάθε φορά που το λειτουργικό σύστημα επιλέγει μια άλλη διεργασία προς εκτέλεση, ενημερώνει και τον ειδικό επεξεργαστή ώστε να γνωρίζει κάθε φορά, ποιό είναι το ενεργό σύνολο καταχωρητών από όπου εκτελούνται ειδικές εντολές. Η αρχιτεκτονική του Hamlyn [7] προσφέρει και αυτή διαφορετικό σύνολο καταχωρητών σε κάθε διεργασία (terminus).

Η χρήση ενός συνόλου καταχωρητών έχει το μειονέκτημα ότι πρέπει το σύστημα να υποστηρίζει εκτέλεση πολλών εντολών ατομικά (π.χ κώδικας Pal). Η τεχνική της χρήσης διαφορετικών συνόλων καταχωρητών δεν έχει ανάγκη τέτοιων τεχνικών καθώς δεν επηρεάζεται η έκδοση των ατομικών πράξεων από τις αλλαγές των εκτελούμενων διεργασιών του συστήματος. Ωστόσο, η αρχιτεκτονική του Flash απαιτεί πρόσβαση στον διαχειριστή διακοπών του λειτουργικού συστήματος για να μπορεί να ενημερώνει τον ειδικό επεξεργαστή με την τρέχουσα προς εκτέλεση διεργασία. Αντίθετα, ο *Τηλέγραφος Ι* μπορεί να εκτελέσει ατομικές πράξεις χωρίς να χρειάζεται πρόσβαση στον χειριστή διακοπών ή σε άλλο τμήμα του πυρήνα, αρκεί να υπάρχει ειδικό περιβάλλον εκτέλεσης εντολών ατομικά (περιβάλλον

εκτέλεσης κώδικα Pal).

Η αρχιτεκτονική του Hamlyn δεν χρειάζεται αλλαγή στον διαχειριστή διακοπών του λειτουργικού συστήματος αλλά η ανάθεση του συνόλου των καταχωρητών γίνεται αρχικά, κατά την δημιουργία κάθε νέας διεργασίας. Η προστασία της διεύθυνσης στην οποία θα εκτελεστεί η ειδική εντολή εξασφαλίζεται με την χρήση ειδικών κλειδιών που έχει κάθε διεργασία που μπορεί να εκτελεί ειδικές εντολές (π.χ. DMA). Η αρχιτεκτονική του *Τηλέγραφου II* [22] προτείνει και αυτή την ύπαρξη ξεχωριστών συνόλων καταχωρητών για κάθε διεργασία και την προστασία των προσπελάσεων την εξασφαλίζει με την χρήση κρυφής διευθυνσιοδότησης (shadow addressing). Με την τεχνική της κρυφής διευθυνσιοδότησης είναι δυνατή η μετατροπή ιδεατών διευθύνσεων σε φυσικές και ο ελέγχος της προστασίας της διεύθυνσης κατευθείαν από το περιβάλλον εκτέλεσης εντολών του χρήστη.

4.2 Συναρτήσεις υποστήριξης λίστας αντιγράφων σελίδων μνήμης

Η βιβλιοθήκη υποστήριξης περιλαμβάνει ακόμη, βοηθητικές συναρτήσεις για την διαχείριση του υποσυστήματος διασύνδεσης και συγκεκριμένα για την υποστήριξη του συστήματος διαχείρισης μνήμης. Οι συναρτήσεις αυτές χειρίζονται το τμήμα της μνήμης του υποσυστήματος διασύνδεσης που διατηρεί τις λίστες με τους κόμβους που διατηρούν αντίγραφα των τοπικών σελίδων μνήμης.

Η δομή που ορίζεται από την αρχιτεκτονική του Τηλέγραφου είναι απλή συνδεδεμένη λίστα. Για κάθε τοπική σελίδα μνήμης υπάρχει ένας δείκτης για την αρχή της λίστας. Ο Τηλέγραφος αναλαμβάνει να εκτελεί "τηλε−εγγραφές" σε κάθε κόμβο που περιέχεται στην λίστα της αντίστοιχης σελίδας. Η διαθέσιμη μνήμη για τις λίστες αυτές δεν είναι αρκετή για να διαθέτουν όλες οι σελίδες μνήμης αντίγραφα σε όλους τους κόμβους, αλλά αυτό δεν αποτελεί περιορισμό καθώς μια τέτοια κατανομή των αντιγράφων είναι πολύ κακή για την συνολική απόδοση του συστήματος.

Ο διαχειριστής της μνήμης του Τηλέγραφου ελέγχει και διατηρεί τις λίστες αυτές για την αποδοτικότερη λειτουργία του συστήματος. Η ενημέρωση τους βασίζεται στους μετρητές προσπέλασης που διατηρεί ο Τηλέγραφος.

Η λίστα διατηρείται σε έναν πίνακα συνεχόμενων διευθύνσεων μνήμης. Οι ελεύθερες θέσεις οι οποίες μπορούν να χρησιμοποιηθούν για να δημιουργηθεί ένα επιπλέον αντίγραφο είναι όλες τοποθετημένες σε μια απλά συνδεδεμένη λίστα. Αρχικά όλες οι θέσεις του πίνακα ανήκουν σε αυτή τη λίστα. Η συνάρτηση *get_free_slot()* επιστρέφει έναν δείκτη στην διαθέσιμη ελεύθερη θέση που είναι πάντα η αρχή της λίστας. Ετσι ο χρόνος εύρεσης μιας ελεύθερης θέσης είναι πάντα σταθερός (O(1)).

4.2.1 Ελεγχος ύπαρξης αντιγράφων

Η διαδικασία *has_replica(page)* ελέγχει αν η συγκεκριμένη τοπική σελίδα διαθέτει αντίγραφο σε άλλους κόμβους. Η συνάρτηση επιστρέφει μια λογική τιμή (true/false). Ο έλεγχος αυτός είναι απαραίτητος στην περίπτωση που στην σελίδα αυτή θέλουμε να έχουμε μεταβλητές που θα προσπελαύνουμε ατομικά. Οπως αναφέρθηκε και στην ενότητα 4.1, πρέπει να εξασφαλίζεται ότι οι ατομικές μεταβλητές βρίσκονται σε σελίδες μνήμης που δεν διαθέτουν αντίγραφα. Το σύστημα διαχείρισης μνήμης μπορεί να χρησιμοποιεί αυτή τη συνάρτηση προτού αναθέσει μνήμη στις ατομικές μεταβλητές. Ο έλεγχος αυτός μπορεί να γίνει με μια προσπέλαση στην αρχή της λίστας για την δοσμένη σελίδα και άρα ο χρόνος εκτέλεσης είναι σταθερός (O(1)). Ο χρόνος αυτός είναι ο ίδιος με τον χρόνο μιας τοπικής ανάγνωσης από την μνήμη του Τηλέγραφου.

4.2.2 Λίστα αντιγράφων

Η συνάρτηση *get_multicast_list(page)* επιστρέφει μια λίστα με όλους τους κόμβους που διαθέτουν αντίγραφο της τοπικής σελίδας *page*. Μια τέτοια λίστα είναι χρήσιμη στην περίπτωση που θα πρέπει να ακυρωθούν τα αντίγραφα μερικών ή όλων των κόμβων. Το κόστος της συνάρτησης αυτής εξαρτάται από τον αριθμό των αντιγράφων που υπάρχουν (O($number_of_replicas$)). Ωστόσο, οι εφαρμογές που έχουν πολλαπλά αντίγραφα πρέπει να φροντίζουν να μην διατηρούν μεγάλες λίστες αντιγράφων, για να αξιοποιούν καλύτερα την απόδοση του συστήματος.

4.2.3 Δημιουργία αντιγράφων

Η συνάρτηση *create_replica(page,node)* δημιουργεί αντίγραφο της τοπικής σελίδας *page* στον κόμβο *node*. Η δομή του κόμβου περιλαμβάνει και την αντίστοιχη σελίδα στην οποία θα δημιουργηθεί το αντίγραφο. Ο χρόνος εισαγωγής στην λίστα είναι σταθερός (O(1)). Ενα ελεύθερο στοιχείο το βρίσκουμε σε σταθερό χρόνο, όπως επίσης σε σταθερό χρόνο γίνεται η εισαγωγή στην λίστα αφού το νέο στοιχείο τοποθετείται στην αρχή της.

Ο διαχειριστής μνήμης θα πρέπει να φροντίζει να ορίζει ελεύθερες σελίδες μνήμης για την δημιουργία των αντιγράφων, καθώς η συνάρτηση δεν έχει γνώση της κατάστασης της σελίδας στην οποία θα δημιουργηθεί το αντίγραφο.

4.2.4 Ακύρωση αντιγράφων

Η συνάρτηση *drop_replica(page,node)* αφαιρεί τον κόμβο $node$ από την λίστα των κόμβων που διαθέτουν αντίγραφο της τοπικής σελίδας $page$. Η ακύρωση του αντίγραφου απαιτεί γραμμικό χρόνο εκτέλεσης ($O(n)$) καθώς θα πρέπει να γίνει σειριακή διάσχιση της λίστας για να βρεθεί

το στοιχείο της που αντιστοιχεί στο αντίγραφο που θέλουμε να διαγράψουμε. Η θέση που ελευθερώνεται εισάγεται στην λίστα που διατηρεί τις κενές θέσεις του πίνακα. Η νέα θέση εισάγεται στην αρχή της λίστας κενών θέσεων και το κόστος εισαγωγής είναι σταθερό (O(1)).

Ο διαχειριστής της μνήμης θα πρέπει να φροντίσει να ενημερώσει τον κόμβο από τον οποίο αφαιρείται το αντίγραφο, και η διεργασία που διατηρούσε το αντίγραφο θα πρέπει να εκτελεί πλέον απομακρυσμένες αναγνώσεις των δεδομένων της συγκεκριμένης σελίδας.

4.3 Απόδοση του πρωτοτύπου του Τηλέγραφου I

Στο πρωτότυπο του Τηλέγραφου I πήραμε μετρήσεις για τις βασικές λειτουργίες εγγραφών και αναγνώσεων οι οποίες αρχικά υλοποιήθηκαν. Αν και το πρωτότυπο δεν είναι πλήρες και οι μετρήσεις είναι ενδεικτικές της τελικής απόδοσης, ωστόσο, μπορούμε να εξάγουμε συμπεράσματα για την απόδοση και των υπολοίπων λειτουργιών. Στον πίνακα 4.2 παρουσιάζονται τα αποτελέσματα που προκύπτουν εκτελώντας συνεχόμενες εγγραφές/αναγνώσεις από την μνήμη του Τηλέγραφου.

Εντολή	Επαναλήψεις	Κόστος μιας εντολής
Τοπικές εγγραφές	100	326 ns
	1000	441 ns
	5000	459 ns
	10000	461 ns
Τοπικές αναγνώσεις	100	890 ns
	1000	897 ns
	5000	879 ns
	10000	872 ns
Απομακρυσμένες εγγραφές	100	494 ns
	500	636 ns
	1000	701 ns
Απομακρυσμένες αναγνώσεις	100	7128 ns
	500	7177 ns
	1000	7205 ns

Πίνακας 4.2: Κόστος εγγραφών/αναγνώσεων για το πρωτότυπο του Τηλέγραφου−I

Οι μετρήσεις αυτές έγιναν στο λειτουργικό σύστημα Mach, στο οποίο υπήρχε μόνο μια διεργασία χρήστη η οποία διενεργούσε τις επαναληπτικές προσπελάσεις, ενώ ο χρόνος μετρήθηκε με βάση το ειδικό καταχωρητή του επεξεργαστή Alpha ο οποίος μετρά τους κύκλους που έχει εκτελέσει ο επεξεργαστής. Οι χρόνοι των μετρήσεων προκύπτουν αφού αφαιρέσουμε τους κύκλους των εντολών που εκτελεί ο επαναληπτικός βρόχος. Η μέτρηση

αυτή μπορεί να θεωρηθεί πολύ ακριβής για πολλές επαναλήψεις. Δυστυχώς, ο αριθμός των επαναλήψεων του πίνακα 4.2 για τις απομακρυσμένες πράξεις δεν είναι πάρα πολύ μεγάλος αλλά το σύστημα δεν διατηρούσε σταθερή κατάσταση αν ο αριθμός των απομακρυσμένων πράξεων ξεπερνούσε τις 1000. Ο επαναληπτικός βρόχος εκτελούσε εντολές εγγραφών ή αναγνώσεων[3] από τον χώρο διευθύνσεων του Τηλέγραφου και αφορούσαν λέξεις μεγέθους τεσσάρων bytes. Για τις εντολές εγγραφής και ανάγνωσης από τον Τηλέγραφο δεν απαιτείται κλήση κάποιας ρουτίνας από την βιβλιοθήκη υποστήριξης.

Από τα αποτελέσματα αυτά βλέπουμε ότι οι τοπικές εγγραφές μπορούν να εκτελεστούν σε περίπου 400−480 ns (5−6 κύκλοι του TURBOchannel). Αντίθετα οι τοπικές αναγνώσεις απαιτούν περίπου τον διπλάσιο χρόνο καθώς ο επεξεργαστής πρέπει να περιμένει την απάντηση.

Για τις απομακρυσμένες εγγραφές βλέπουμε ότι απαιτούνται περίπου 720 ns για την εκτέλεση τους (9 κύκλοι TURBOchannel), το οποίο είναι και θεωρητικά αναμενόμενο καθώς το σύστημα διασύνδεσης του Τηλέγραφου στέλνει στό δίκτυο 1 byte σε κάθε κύκλο TURBOchannel και το μέγεθος του πακέτου για την αποστολή μιας πράξης εγγραφής είναι 9 bytes.

Για τις απομακρυσμένες αναγνώσεις το κόστος μεγαλώνει αρκετά και χρειάζονται περίπου 7200 ns για την ολοκλήρωση μιας πράξης ανάγνωσης. Η πράξη της απομακρυσμένης ανάγνωσης καθορίζει και την απόδοση των ατομικών πράξεων του Τηλέγραφου Ι, αφού κάθε ατομική πράξη εκτελεί δύο τοπικές εγγραφές και καταλήγει σε μια ανάγνωση, που μπορεί να είναι είτε τοπική είτε απομακρυσμένη. Αν η υλοποίηση των ατομικών πράξεων γίνει με κώδικα Pal, το συνολικό τους κόστος αναμένεται να είναι περίπου 1930 ns (αν η διεύθυνση είναι τοπική) και να ξεπερνά τα 8250 ns αν η διεύθυνση δεν είναι τοπική. Για τους προαναφερθέντες χρόνους θεωρήθηκε ότι η τελευταία πράξη της ανάγνωσης επιστρέφει πάντα με επιτυχία. Σε περίπτωση που αυτό δεν ισχύει, ο χρόνος εκτέλεσης της ανάγνωσης είναι μεταβλητός και ανάλογα μεταβάλλεται και ο συνολικός χρόνος εκτέλεσης της ατομικής πράξης και φυσικά θα πρέπει να προστεθεί το κόστος εκτέλεσης μιας κλήσης συστήματος (6.7μs) στον συνολικό χρόνο εκτέλεσης.

[3]Οι αναγνώσεις εκτελούνται χωρίς να χρειάζεται να γίνονται επαναληπτικές προσπελάσεις για την επιστροφή του αποτελέσματος.

Κεφάλαιο 5

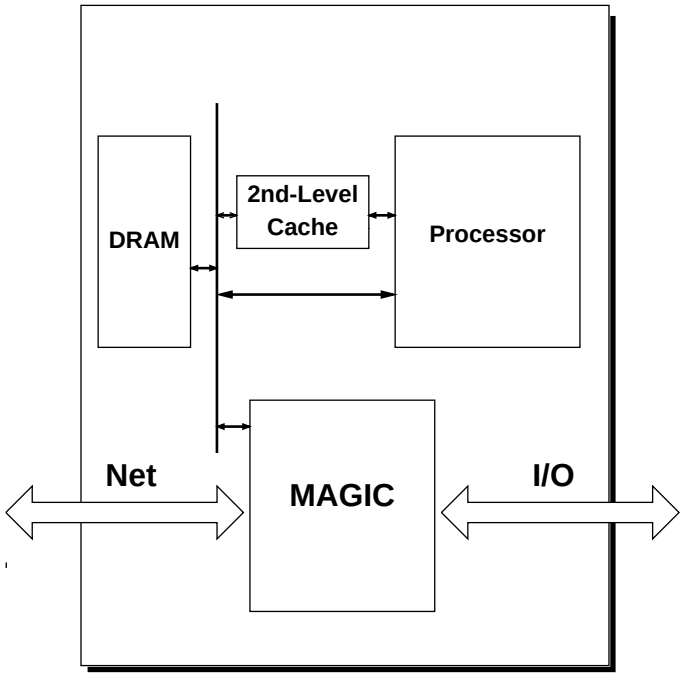
Σύγκριση με άλλα συστήματα

Στο κεφάλαιο αυτό παρουσιάζεται μία συγκριτική μελέτη του Τηλέγραφου Ι και της αρχιτεκτονικής του, με άλλες αρχιτεκτονικές που εμφανίστηκαν είτε σε παράλληλες πολυεπεξεργαστικές μηχανές όπως είναι το FLASH, το Alewife και το KSR−I είτε σαν αυτόνομες αρχιτεκτονικές για διασύνδεση υπολογιστών όπως είναι το PRAM, το SHRIMP και το Hamlyn. Για κάθε μία από τις συγκρινόμενες αρχιτεκτονικές, γίνεται μια συνοπτική παρουσίασή της ενώ στο τέλος γίνεται μια αξιολόγηση των διαφόρων αρχιτεκτονικών και εντοπίζουμε τις διαφορές ή τις ομοιότητες που παρουσιάζουν σε σύγκριση με τον Τηλέγραφο.

5.1 Flash

Το *FLASH* [19] είναι μια πολυεπεξεργαστική μηχανή, κατανεμημένης μνήμης, με ένα πεδίο διευθύνσεων για όλους τους κόμβους της. Οι κόμβοι του *FLASH* αποτελούνται από έναν υπάρχοντα εμπορικό μονοεπεξεργαστή και έναν ελεγκτή διασύνδεσης του κόμβου με το δίκτυο διασύνδεσης, που ονομάζεται MAGIC [14] .

Το MAGIC, όπως φαίνεται και στο σχήμα 5.1, ελέγχει όλες τις βασικές λειτουργίες του κόμβου (μνήμη, I/O, δίκτυο) και έχει την δυνατότητα να προγραμματίζεται για την εκτέλεση διαφορετικών πρωτοκόλλων. Στην πραγματικότητα, το MAGIC είναι ένας ειδικός επεξεργαστής με δική του κύρια και κρυφή μνήμη όπου έχει αποθηκεύμενο τον κώδικα που υλοποιεί τα διαφορετικά πρωτόκολλα που υποστηρίζει (handlers). Ανάλογα με τον τύπο του μηνύματος που δέχεται, εκτελείται ο αντίστοιχος κώδικας. Ετσι παρέχει την δυνατότητα για την υποστήριξη και πρωτοκόλλων ανταλλαγής μηνυμάτων αλλά και την δυνατότητα υποστήριξης cache coherency. Επίσης το MAGIC έχει την δυνατότητα για υλοποίηση εντολών συγχρονισμού. Υλοποιώντας εντολές συγχρονισμού στο MAGIC κερδίζουμε το κόστος διακοπής του επεξεργαστή, ενώ εξασφαλίζεται η ατομικότητα των εντολών αυτών γιατί ο κώδικας που εκτελείται από το MAGIC δεν μπορεί να διακοπεί.



Σχήμα 5.1: Το εσωτερικό ενός κόμβου του *FLASH*

Οι διεργασίες του χρήστη επικοινωνούν με το MAGIC χωρίς την παρεμβολή του λειτουργικού συστήματος. Αυτό επιτυγχάνεται αντιστοιχίζοντας τους καταχωρητές του MAGIC με διευθύνσεις στον χώρο της διεργασίας. Το MAGIC διατηρεί, για κάθε διεργασία, μια δομή λειτουργίας του πρωτοκόλλου (protocol operation record). Για το σκοπό αυτό, το λειτουργικό σύστημα ενημερώνει το MAGIC με τον κωδικό (PID) της διεργασίας που πρόκειται να αρχίσει να εκτελείται μετά από κάθε αλλαγή εκτελούμενης διεργασίας (context switch).

Η επικοινωνία των διεργασιών του χρήστη με το MAGIC γίνεται μέσω ενός απλού πρωτοκόλλου που περιλαμβάνει μερικές εγγραφές για να περάσουν οι διευθύνσεις και τα δεδομένα και τερματίζει με μια ανάγνωση που καθορίζει αν η εντολή είναι αποδεκτή ή ακυρώθηκε. Αν η ακολουθία των εντολών γίνει αποδεκτή δεν σημαίνει ότι έχει τερματίσει η εκτέλεσή της. Ο έλεγχος του τερματισμού γίνεται, στα περισσότερα πρωτόκολλα, με επαναληπτικές προσπελάσεις (polling) σε προκαθορισμένες διευθύνσεις. Σε περίπτωση που η διεργασία διακοπεί κατά την διάρκεια μιας "συνομιλίας" με το MAGIC, εξασφαλίζεται ότι θα μπορέσει να συνεχιστεί γιατί κάθε διεργασία έχει τη δική της δομή μέσω της οποίας επικοινωνεί με το MAGIC.

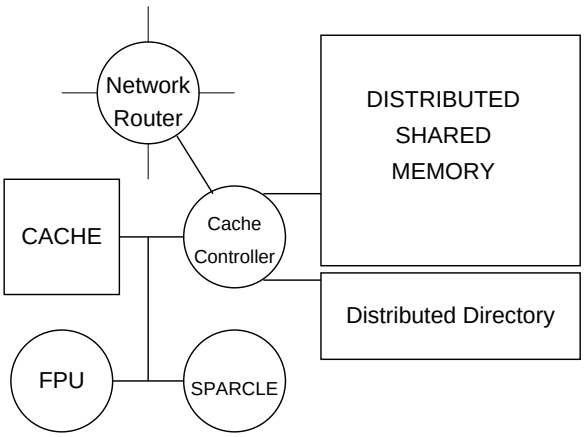
Στο MAGIC υλοποιήθηκε ένα είδος απλών ενεργών μηνυμάτων (active messages [31]) και ένα παράδειγμα τους είναι η ατομική πράξη remote fetch–and–add. Η πράξη αυτή

ενεργοποιείται με μια εγγραφή στον ειδικό χώρο διευθύνσεων, με την οποία καθορίζεται η εντολή, η διεύθυνση και ο τελεστέος. Στη συνέχεια ακολουθεί μια ανάγνωση, για να πάρει η διεργασία το αποτέλεσμα. Το κόστος που υπολογίστηκε για μια τέτοια πράξη, είναι 1.75 μs, που τελικά καταλήγει να είναι και το κόστος μιας απομακρυσμένης ανάγνωσης, αφού η εγγραφή είναι σε τοπική διεύθυνση του υπολογιστή και η πράξη της πρόσθεσης υλοποιείται από hardware και κατά συνέπεια, έχουν αμελητέο κόστος.

5.2 Alewife

Το *Alewife* είναι το όνομα ενός ερευνητικού προγράμματος του Πανεπιστημίου M.I.T. και αφορά την σχεδίαση μιας ευρείας κλίμακας παράλληλης μηχανής. Η μηχανή έχει κατανεμημένη μνήμη και η διασύνδεση των κόμβων γίνεται μέσω ενός γρήγορου δικτύου. Η αρχιτεκτονική του *Alewife* προσπαθεί να μειώσει το κόστος της επικοινωνίας με τεχνικές κατάλληλης τοποθέτησης των αρχικών δεδομένων, ενώ παρέχει την δυνατότητα πολύ γρήγορων εναλλαγών των διεργασιών που εκτελούνται σε κάθε κόμβο ώστε να ‘‘κρύβονται’’ οι καθυστερήσεις του δικτύου διασύνδεσης.

Το *Alewife* παρέχει cache coherency μέσω ενός πρωτοκόλλου καταλόγων που ονομάζεται LimitLESS. Το πρωτόκολλο αυτό παρέχει έναν μικρό αριθμό δεικτών, ενώ ο πλήρης κατάλογος είναι αποθηκευμένος στην μνήμη και μπορεί να προσπελαστεί προκαλώντας διακοπή στον επεξεργαστή.



Σχήμα 5.2: Το εσωτερικό ενός κόμβου του *Alewife*

Οι κόμβοι της μηχανής είναι συνδεδεμένοι σε μια τοπολογία πλέγματος (mesh), ενώ κάθε κόμβος, όπως φαίνεται και στο σχήμα 5.2, αποτελείται από έναν ειδικά κατασκευασμένο

επεξεργαστή (Sparcle), το chip ελέγχου του δικτύου, τμήμα της κατανεμημένης μνήμης και τμήμα του κατανεμημένου καταλόγου.

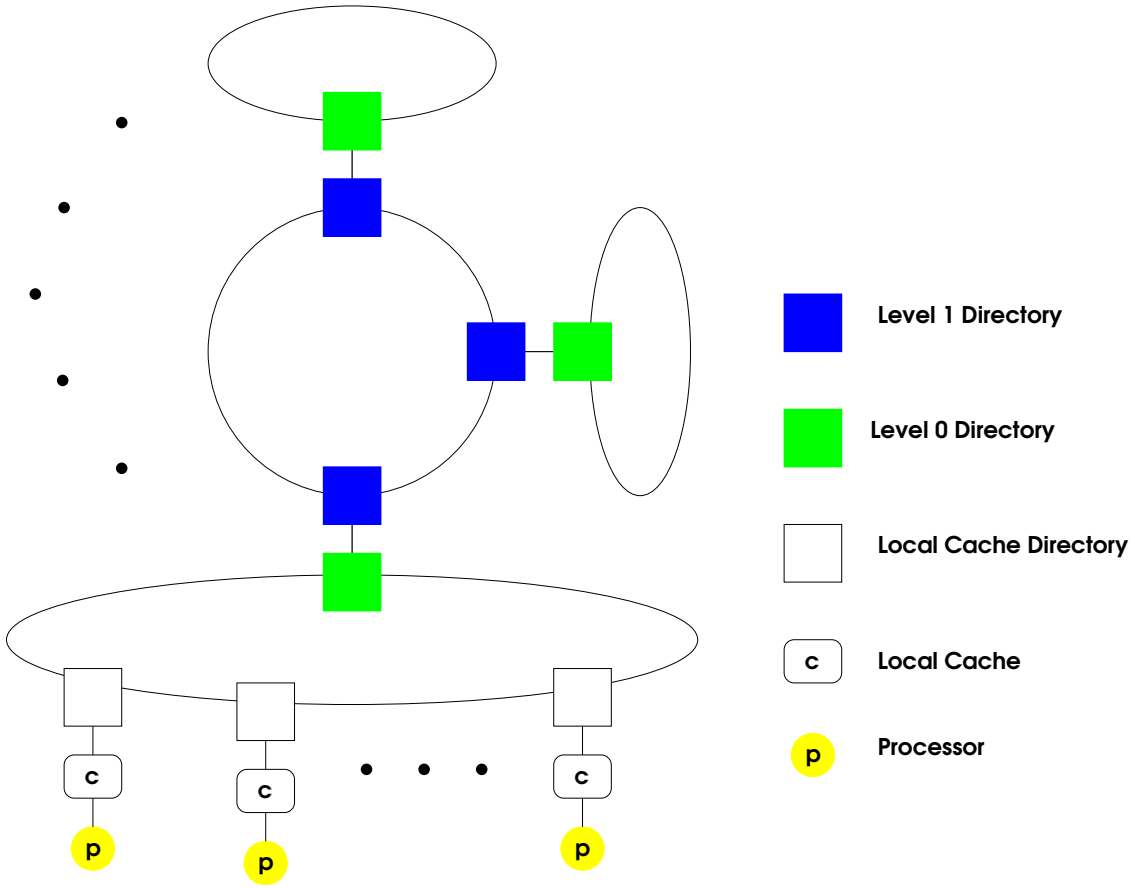
Η αρχιτεκτονική του *Alewife* [18] προτείνει, για τον συγχρονισμό των διεργασιών που τρέχουν στους κόμβους, τις δομές L και J (L−structures, J−structures). Η δομή J είναι μια δομή δεδομένων κατάλληλη για εφαρμογές τύπου παραγωγού−καταναλωτή. Η δομή L είναι ένας πίνακας που επιτρέπει την ατομική προσπέλαση των στοιχείων του. Και για τις δύο αυτές δομές απαιτείται από την μηχανή να διαθέτει ένα επιπλέον bit για κάθε λέξη της μνήμης. Το bit αυτό μπορεί να χρησιμοποιηθεί για να ελέγχει την προσπέλαση στην συγκεκριμένη λέξη της μνήμης. Η αρχιτεκτονική επίσης, ορίζει ειδικές εντολές του επεξεργαστή που θέτουν και ελέγχουν το bit αυτό. Με τον τρόπο αυτό ο συγχρονισμός επιτυγχάνεται εξ ολοκλήρου από το hardware. Ωστόσο ο έλεγχος σε περίπτωση λάθους (π.χ προσπέλαση σε ήδη "κλειδωμένη" λέξη μνήμης) γίνεται από το λογισμικό είτε με επαναληπτικές προσπελάσεις είτε με διακοπή της διεργασίας που θέλει να εκτελέσει τον συγχρονισμό. Πάντως και για τις περιπτώσεις αυτές, η αρχιτεκτονική του Alewife, με την γρήγορη εναλλαγή υποδιεργασιών που διαθέτει, επιτρέπει την γρήγορη εξυπηρέτηση των διακοπών για συγχρονισμούς. (Μια εναλλαγή υποδιεργασίας στον επεξεργαστή Sparcle, κοστίζει περίπου 14 κύκλους). Μια τέτοια υλοποίηση παρέχει πολύ γρήγορες εντολές για κλείδωμα ή συγχρονισμό (της τάξης λίγων κύκλων ρολογιού) ωστόσο, το κόστος είναι αρκετά μεγάλο, δεδομένου ότι θα πρέπει να διαθέσουμε ένα επιπλέον bit για κάθε λέξη της μνήμης και βέβαια να παρέχουμε ειδική υποστήριξη από τον επεξεργαστή, ο οποίος θα πρέπει να έχει ένα διευρυμένο σύνολο εντολών και ειδικές τεχνικές για την εναλλαγή των υποδιεργασιών.

5.3 KSR

Η μηχανή KSR είναι μια πολυεπεξεργαστική μηχανή που το κύριο χαρακτηριστικό της είναι η έλλειψη κύριας μνήμης. Ολη η μνήμη του συστήματος, η οποία είναι κοινή για όλο το σύστημα, συμπεριφέρεται σαν ένα ιεραρχικό σύστημα κρυφής μνήμης. Οι κόμβοι της μηχανής είναι συνδεδεμένοι (ανά 32) με τοπολογία δακτυλίου, ενώ είναι δυνατόν η μηχανή να επεκταθεί συνδέοντας πολλούς δακτύλιους σε έναν εξωτερικό δακτύλιο. Η συνδεσμολογία αυτή φαίνεται στο σχήμα 5.3.

Ολες οι προσπελάσεις μνήμης στην μηχανή KSR είναι σειριακά συνεπείς (sequentially consistent) και διατεταγμένες. Το hardware εξασφαλίζει ότι όλοι οι επεξεργαστές έχουν την ίδια εικόνα για το σύνολο της μνήμης.

Σε κάθε δακτύλιο υπάρχουν ειδικοί κόμβοι που διατηρούν καταλόγους για όλες τις υποσελίδες (subpages) που υπάρχουν στον δακτύλιο και προωθούν αιτήσεις για προσπελάσεις



Σχήμα 5.3: Η ιεραρχία των κόμβων της μηχανής KSR−1

σε υποσελίδες που δεν υπάρχουν στον δικό τους δακτύλιο, στον αμέσως επόμενο. Οι χρόνοι προσπέλασης για κάθε ένα από τα διαφορετικά επίπεδα μνήμης είναι σαφώς διαφορετικοί και εξαρτώνται κάθε φορά από τη μέγιστη απόσταση του επεξεργαστή που κάνει την προσπέλαση με αυτόν, που τελικά, θα την εξυπηρετήσει. Οι μέσοι χρόνοι καθυστέρησης (latency) για το κάθε επίπεδο της ιεραρχίας φαίνονται στον πίνακα 5.1. Ο μέσος χρόνος μεταφοράς μιας υποσελίδας από άλλον επεξεργαστή στον ίδιο δακτύλιο είναι της τάξης των 6.7μs (μέτρηση χωρίς άλλο φορτίο στον δακτύλιο).

Η αρχιτεκτονική του KSR παρέχει ειδικές εντολές για αποκλειστική προσπέλαση στις υποσελίδες της μνήμης. Το κόστος της εντολής *gsp*, που υλοποιεί "κλείδωμα" σε hardware, ξεκινά από μερικές δεκάδες microseconds για λίγους επεξεργαστές και ξεπερνά τα 1000 microseconds για περισσότερους από 15 επεξεργαστές.

From:	cycles	capacity
hardware cache	2	256KB
local memory	18	32MB
ring 0	126	1 GB
ring 1	600	34 GB

Πίνακας 5.1: Καθυστερήσεις της μνήμης της μηχανής KSR.

5.4 Meiko CS−2

Η μηχανή Meiko CS−2[27] είναι ένας παράλληλος πολυεπεξεργαστής κατανεμημένης μνήμης. Οι κόμβοι της μηχανής είναι standard SPARC μικροεπεξεργαστές και ο καθένας εκτελεί το λειτουργικό σύστημα Solaris. Για την επικοινωνία, ο κάθε κόμβος έχει έναν επεξεργαστή διασύνδεσης που ονομάζεται ‘‘Elan’’. Το δίκτυο διασύνδεσης αποτελείται από επίπεδα μεταγωγέων τύπου crosspoint, ενώ το εύρος των συνδέσμων (χωρίς το επιπλέον κόστος του πρωτοκόλλου) φτάνει περίπου τα 50 MBytes για κάθε κατεύθυνση. Ο επεξεργαστής διασύνδεσης υλοποιεί τις λειτουργίες απομακρυσμένων εγγραφών και αναγνώσεων και παρέχει την δυνατότητα εκτέλεσης πρωτοκόλλων στον ίδιο τον επεξεργαστή διασύνδεσης και ο προγραμματισμός τους ελέγχεται απευθείας από τον χρήστη. Επίσης, παρέχει εντολές συγχρονισμού, οι οποίες μπορούν να προκαλέσουν διακοπή στον κύριο επεξεργαστή του κόμβου. Ο Elan μπορεί να προσπελαύνεται απευθείας από τις διεργασίες του χρήστη γιατί διαθέτει δικό του πίνακα μετάφρασης ιδεατών σελίδων ο οποίος ενημερώνεται από τον πυρήνα του λειτουργικού συστήματος ταυτόχρονα με τον αντίστοιχο πίνακα του κύριου επεξεργαστή. Για κάθε διεργασία που χρειάζεται επικοινωνία ο επεξεργαστής διασύνδεσης διαθέτει ξεχωριστό σύνολο καταχωρητών για την έκδοση εντολών. Τα παραπάνω δύο χαρακτηριστικά εξασφαλίζουν την αποστολή μηνυμάτων χωρίς παρέμβαση του λειτουργικού συστήματος. Η παραλαβή μηνυμάτων γίνεται πάλι ασύγχρονα απευθείας από το Elan χωρίς παρέμβαση του λειτουργικού συστήματος. Η διεργασία−παραλήπτης ενημερώνεται είτε μέσω μεταβλητής συγχρονισμού είτε με διακοπή από το λειτουργικό σύστημα. Οι σελίδες μνήμης που χρησιμοποιούνται για την ανταλλαγή μηνυμάτων δεν είναι απαραίτητο να είναι κλειδωμένες στην μνήμη καθώς ο Elan έχει την δυνατότητα να προκαλέσει page fault και να φέρει στην κύρια μνήμη την σελίδα.

Η βιβλιοθήκη υποστήριξης “Widget”

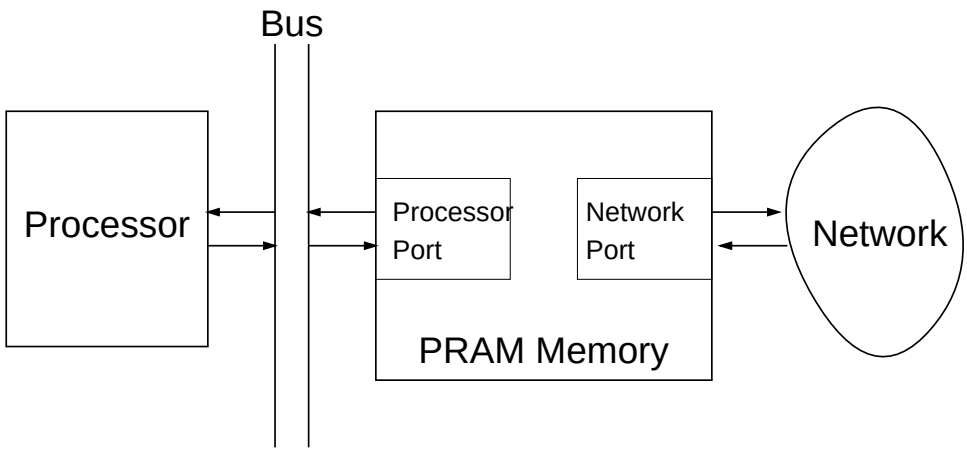
Για την πληρέστερη υποστήριξη του ειδικού επεξεργαστή διασύνδεσης υλοποιήθηκε η βιβλιοθήκη υποστήριξης ‘‘Widget’’ η οποία παρέχει βελτιστοποιημένες συναρτήσεις για την

ανταλλαγή μηνυμάτων παρέχοντας κοινό ιδεατό χώρο μνήμης, συναρτήσεις για υποστήριξη DMA και έλεγχο τερματισμού αποστολής και διασύνδεσης με τις ατομικές πράξεις του Elan.

Ειδικότερα, οι ατομικές πράξεις που υποστηρίζονται, υλοποιούνται απευθείας από την αντίστοιχη υποδιεργασία του Elan την οποία ενεργοποιεί η διεργασία του χρήστη που θέλει να εκτελέσει την ατομική πράξη. Ο τερματισμός της πράξης ελέγχεται με επαναληπτικές προσπελάσεις σε ειδικούς καταχωρητές που καθορίζουν την ολοκλήρωση της πράξης. Από τις μετρήσεις που αναφέρονται στο [27] προκύπτει ότι για την εκτέλεση της ατομικής πράξης *remote atomic swap* απαιτείται χρόνος 50 μs μετρημένος σε αδρανή μηχανή (idle machine), ενώ μετρώντας την ίδια ατομική πράξη καθώς εκτελείται το πρωτόκολλο συγχρονισμού (coherence protocol) που υλοποιήθηκε για τη συγκεκριμένη μηχανή, προκύπτει χρόνος 1230 μς. Η μεγάλη διαφορά στις δύο τιμές δικαιολογείται λόγω συμφόρησης στους μεταγωγείς του δικτύου.

5.5 PRAM

Το PRAM [20, 29] είναι μια αρχιτεκτονική κοινόχρηστης μνήμης που αναπτύχθηκε και υλοποιήθηκε στο πανεπιστήμιο Princeton και επιτρέπει την αποδοτική διασύνδεση σταθμών εργασίας παρέχοντας υψηλές ταχύτητες επικοινωνίας και μικρές καθυστερήσεις. Οι βασικές εντολές που υποστηρίζει η αρχιτεκτονική του PRAM είναι οι εγγραφές και οι αναγνώσεις στην κοινή μνήμη του δικτύου. Κάθε κόμβος που θέλει να μοιραστεί μια σελίδα με έναν άλλο διατηρεί ένα αντίγραφο της διαμοιραζόμενης σελίδας. Η αρχιτεκτονική δεν επιβάλλει



Σχήμα 5.4: Διασύνδεση του PRAM με τον δίαυλο του υπολογιστή και το δίκτυο.

κάποιο συγκεκριμένο μοντέλο συνέπειας (consistency) για την μνήμη του δικτύου. Το δίκτυο διασύνδεσης αποτελείται από μεταγωγείς οι οποίοι μπορούν να συνδέονται και μεταξύ τους

επιτρέποντας οποιαδήποτε τοπολογία δικτύου.

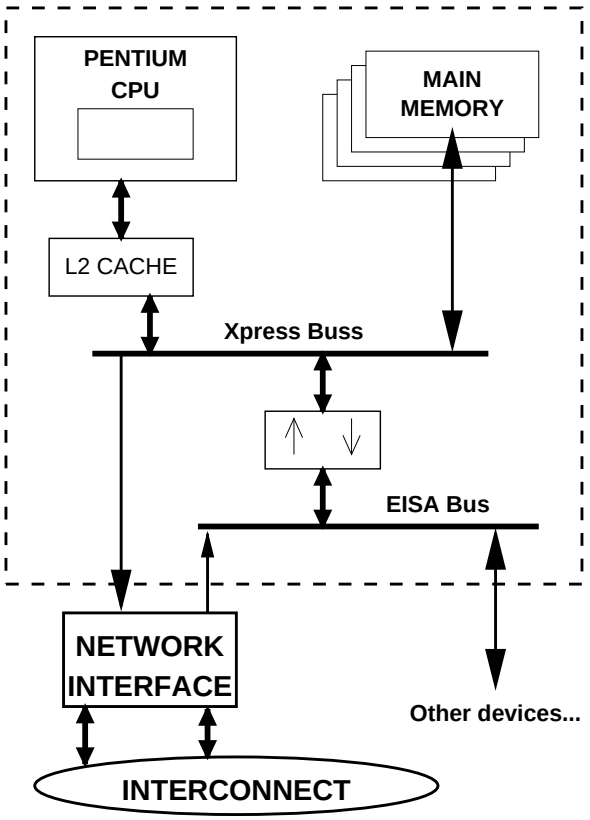
Η μνήμη που χρησιμοποιείται για τις διαμοιραζόμενες σελίδες μπορεί να προσπελαύνεται ταυτόχρονα από τον επεξεργαστή και από το υποσύστημα διασύνδεσης μέσω δίπορτης μνήμης (dual ported memory), όπως φαίνεται και στο σχήμα 5.4. Ολες οι προσπελάσεις σε διευθύνσεις της κοινής μνήμης για τον επεξεργαστή είναι προσπελάσεις στην τοπική του μνήμη. Η τεχνική της απεικόνισης ιδεατής μνήμης επιτρέπει στις διεργασίες του χρήστη να επικοινωνούν χωρίς την παρεμβολή του λειτουργικού συστήματος. Το υποσύστημα διασύνδεσης αναλαμβάνει να μεταφέρει στους άλλους κόμβους τις εγγραφές στο τοπικό αντίγραφο. Οι αναγνώσεις εξυπηρετούνται πάντα από το τοπικό αντίγραφο.

Το PRAM δεν έχει ειδικούς μηχανισμούς για την έκδοση ατομικών πράξεων, ωστόσο παρέχει την δυνατότητα πρόκλησης απομακρυσμένης διακοπής (remote interrupt) εκτελώντας εγγραφές σε ειδικές διευθύνσεις της κοινής μνήμης. Η τεχνική αυτή επιτρέπει να ζητηθεί να γίνει μια λειτουργία σε έναν άλλο κόμβο του δικτύου με μόνο μια τοπική εγγραφή. Με την βοήθεια αυτής της τεχνικής μπορούν να υλοποιηθούν ατομικές πράξεις με κατάλληλο λογισμικό.

Το πρωτότυπο PRAM δεν υποστηρίζει σελιδοποίηση της κοινής μνήμης και κατά συνέπεια, όλες οι εγγραφές στην κοινή μνήμη μεταφέρονται σε όλους τους συνδεδεμένους κόμβους. Ο μέγιστος ρυθμό μεταφοράς είναι 24MBits/sec (ανάμεσα σε διεργασίες του χρήστη), ενώ ο χρόνος για την ολοκλήρωση μιας εγγραφής για δίκτυο μέχρι 10 κόμβων είναι της τάξης του 1 μsec. Με την τεχνική των απομακρυσμένων διακοπών είναι δυνατή η υλοποίηση απομακρυσμένων κλήσεων συναρτήσεων (Remote Procedure Calls). Ο χρόνος για την εκτέλεση μιας κενής απομακρυσμένης κλήσης (null RPC) είναι της τάξης των 200 μsecs.

5.6 SHRIMP

Το SHRIMP [6] [5] είναι ένα σύστημα διασύνδεσης με γρήγορα δίκτυα επικοινωνίας για υπάρχοντα PCs και σταθμούς εργασίας και κατασκευάζεται στο πανεπιστήμιο Princeton. Και το SHRIMP χρησιμοποιεί την τεχνική απεικόνισης ιδεατής μνήμης για να παρακάμψει το λειτουργικό σύστημα και να επιτρέψει απευθείας επικοινωνία με τις διεργασίες του χρήστη. Παρέχει στον χρήστη μηχανισμούς για γρήγορη μεταφορά block δεδομένων και η βασική λειτουργία του είναι η αποστολή μηνυμάτων με ελάχιστο κόστος. Το σύστημα διασύνδεσης του SHRIMP επιτρέπει στον παραλήπτη να έχει στο χώρο διευθύνσεων του τις σελίδες μνήμης που χρησιμοποιεί ο αποστολέας, έτσι ώστε η αποστολή μηνυμάτων να γίνεται με απλές εντολές *store* που το σύστημα διασύνδεσης καταλαβαίνει και αναλαμβάνει να μεταφέρει στον παραλήπτη.



Σχήμα 5.5: Διασύνδεση του SHRIMP με τον δίαυλο του υπολογιστή και το δίκτυο.

Η μεταφορά των δεδομένων μπορεί να γίνεται αυτόματα μετά από κάθε εντολή *store* ή μαζικά (DMA), όταν ο χρήστης το καθορίσει με την εντολή *send*. Οπως φαίνεται και στο σχήμα 5.5, το υποσύστημα διασύνδεσης του SHRIMP διαβάζει κατευθείαν τον δίαυλο του επεξεργαστή με την μνήμη και μπορεί να ανιχνεύει εντολές *store* για τις διευθύνσεις για τις οποίες πρέπει να μεταφέρει τα δεδομένα.

Η αρχιτεκτονική του SHRIMP απαιτεί οι σελίδες μνήμης στις οποίες γίνονται οι αποστολές μηνυμάτων να είναι κλειδωμένες στην κύρια μνήμη του υπολογιστή και για τις σελίδες μνήμης που πρέπει να αποστέλλονται από το δίκτυο θα πρέπει να χρησιμοποιείται η πολιτική write–through από την κρυφή μνήμη. Για την προστασία της μνήμης των διεργασιών χρησιμοποιείται η προστασία που παρέχει το λειτουργικό σύστημα για την ιδεατή μνήμη. Επίσης για τον έλεγχο του υποσυστήματος διασύνδεσης, η αρχιτεκτονική του SHRIMP ορίζει τον χώρο διευθύνσεων εντολών. Ο χώρος αυτός έχει μέγεθος ίσο με το μέγεθος της διαθέσιμης φυσικής μνήμης ενώ δεσμεύεται και αντίστοιχο τμήμα των ιδεατών διευθύνσεων. Επειδή ο χώρος διευθύνσεων του δικτύου δεν είναι ενιαίος αλλά ξεχωριστός για κάθε υπολογιστή, το SHRIMP, για να

υποστηρίζει ανταλλαγή μηνυμάτων σε επίπεδο χρήστη, πρέπει να διατηρεί έναν πίνακα αντιστοίχισης ιδεατών σελίδων σε φυσικές, για κάθε τοπική ιδεατή σελίδα που πρέπει να μεταφέρεται μέσω του δικτύου. Το λειτουργικό σύστημα είναι υπεύθυνο να δίνει σε κάθε διεργασία, που έχει στο χώρο διευθύνσεων της κάποια σελίδα μνήμης, και την αντίστοιχη σελίδα εντολών μέσω των οποίων ενεργοποιούνται οι μαζικές μεταφορές δεδομένων. Η αρχική ανάθεση μνήμης και η αντιστοίχιση ιδεατών διευθύνσεων σε φυσικές γίνεται με κλήση συστήματος. Η παραλαβή ενός μηνύματος μπορεί να γίνει είτε με διακοπή που θα προκληθεί από το υποσύστημα διασύνδεσης για να ενημερωθεί η διεργασία–παραλήπτης είτε με ευθύνη της διεργασίας–παραλήπτη, η οποία θα πρέπει να ελέγχει κάποια προκαθορισμένη διεύθυνση σημαίας που θα καθορίζει το τέλος μηνύματος.

5.7 Hamlyn

Το Hamlyn [7] είναι μια αρχιτεκτονική για γρήγορα συστήματα διασύνδεσης επεξεργαστών που μπορεί να χρησιμοποιηθεί σε πολυεπεξεργαστικά συστήματα ή ομάδες κοντινών (closely–clustered) υπολογιστών και αναπτύχθηκε από την εταιρία Hewlett–Packard. Το Hamlyn στηρίζεται σε μια γρήγορη υλοποίηση για μεταφορά μηνυμάτων που υλοποιείται με DMA.

Κάθε διεργασία διαθέτει ένα σύνολο από καταχωρητές για να μπορεί να ξεκινήσει ένα DMA και μια ουρά όπου μπορεί να αποθηκεύει μικρά μηνύματα για αποστολή με απλές εντολές *store* (μέχρι 64 μηνύματα). Μεγάλα μηνύματα μπορούν επίσης να αποσταλούν καθορίζοντας έναν δείκτη στην αρχική διεύθυνση και το μέγεθος τους. Οι διευθύνσεις των καταχωρητών και της ουράς αντιστοιχίζονται στον χώρο διευθύνσεων της διεργασίας ώστε να μην είναι απαραίτητη η παρέμβαση του λειτουργικού συστήματος. Για να αποσταλεί ένα μήνυμα θα πρέπει νωρίτερα, ο παραλήπτης να έχει δεσμεύσει μια περιοχή μνήμης όπου θα αποθηκευτεί και την οποία θα πρέπει να γνωρίζει ο αποστολέας. Η μνήμη δεσμεύεται από την μνήμη του υπολογιστή. Με το τρόπο αυτό εξασφαλίζεται ότι τα μηνύματα θα έχουν προδεσμευμένο χώρο στον οποίο θα γράφονται και ταυτόχρονα δεν υπάρχει ανάγκη για αντιγραφή των μηνυμάτων από ενταμιευτές. Η μεταφορά των μηνυμάτων γίνεται απευθείας από την μνήμη του αποστολέα στην μνήμη του δέκτη.

Το Hamlyn παρέχει μηχανισμούς προστασίας ώστε να εξασφαλίζεται η προστασία της μνήμης (κλειδιά για κάθε πακέτο αποστολής). Για την παραλαβή του μηνύματος, στην μνήμη του παραλήπτη διατηρείται μια ουρά ενημέρωσης παραλαβής (message notification queue). Οταν έρθει ένα μήνυμα, το Hamlyn ενημερώνει την ουρά αυτή, ενώ αν η διεργασία δεν εκτελείται εκείνη τη στιγμή γίνεται διακοπή για να ενημερωθεί η διεργασία για την παραλαβή του μηνύματος.

Βιβλιοθήκη υποστήριξης Rats

Χρησιμοποιώντας την μοναδική λειτουργία αποστολής μηνυμάτων υλοποιήθηκε μια βιβλιοθήκη υποστήριξης για το *Hamlyn*. Οι πρωταρχικές λειτουργίες που παρέχει η βιβλιοθήκη είναι η αποστολή (sendmsg) και η παραλαβή (get_notification_record) μηνύματος. Με βάση αυτές τις πρωταρχικές λειτουργίες παρέχεται η υλοποίηση διαφορετικών πρωτοκόλλων/υπηρεσιών. Ετσι παρέχεται για παράδειγμα, πρωτόκολλο που είναι παρόμοιο με το πρωτόκολλο UDP στα IP δίκτυα (datagram) και πρωτόκολλο που εξασφαλίζει ασφαλή μεταφορά μηνυμάτων από τον αποστολέα στον παραλήπτη (streams). Ακόμη, μέσω της βιβλιοθήκης, υποστηρίζονται οι λειτουργίες απομακρυσμένων εγγραφών και αναγνώσεων.

Στις μετρήσεις που αναφέρονται στο [7] φαίνεται ότι ο συνολικός χρόνος για την αποστολή και την παραλαβή ενός μηνύματος μεγέθους 16 bytes απαιτούνται 8.8μs ενώ για την συγκεκριμένη πράξη, η βιβλιοθήκη Rats έχει κόστος 1.6μs. Ο ρυθμός μετάδοσης που προκύπτει (χρησιμοποιώντας πακέτα μεγέθους 16 bytes) είναι 12.1MBytes/s.

5.8 Memory Channel

Το Memory Channel [12] είναι η πρόταση της εταιρίας Digital για τις παράλληλες μηχανές της επόμενης γενιάς. Πρόκειται για μια αρχιτεκτονική που αφορά την διασύνδεση υπολογιστών χρησιμοποιώντας τον δίαυλο PCI και κατά συνέπεια, η μνήμη του συστήματος βρίσκεται στον χώρο διευθύνσεων εισόδου/εξόδου. Οι κόμβοι του δικτύου μπορεί να είναι σταθμοί εργασίας Digital που υποστηρίζουν τον δίαυλο PCI.

Η αρχιτεκτονική του Memory Channel βασίζεται και αυτή στην τεχνική της αντιστοίχισης της μνήμης του δικτύου στον χώρο των ιδεατών διευθύνσεων κάθε κόμβου του δικτύου. Η βασική εντολή που υποστηρίζεται είναι (όπως και στον Τηλέγραφο) οι απομακρυσμένες εγγραφές οι οποίες μπορούν να εκτελούνται απευθείας χωρίς την παρέμβαση του λειτουργικού συστήματος. Το κόστος εκτέλεσης μιας απομακρυσμένης εγγραφής είναι περίπου 5μs, ενώ το bandwidth της επικοινωνίας μεταξύ δύο κόμβων είναι περίπου 100 MBytes/sec.

5.9 Σύγκριση με τον Τηλέγραφο I

Η παρουσίαση των συστημάτων στις προηγούμενες ενότητες έγινε σε μια προσπάθεια να κατατάξουμε την λειτουργικότητα και απόδοση της αρχιτεκτονικής του Τηλέγραφου (και του πρωτοτύπου του Τηλέγραφου I) μέσα στον χώρο των παράλληλων αρχιτεκτονικών είτε αυτές αφορούν μεγάλες πολυεπεξεργαστικές μηχανές (Flash, Alewife, KSR) είτε πρόκειται για αρχιτεκτονικές με βάση δίκτυα υπολογιστών όπως είναι το PRAM, το SHRIMP, το Memory

Channel και το Hamlyn.

Αν και ο Τηλέγραφος ανήκει στην δεύτερη κατηγορία των παράλληλων αρχιτεκτονικών, ωστόσο μια σύγκριση με τις αρχιτεκτονικές των πολυεπεξεργαστικών μηχανών έχει ενδιαφέρον, καθώς τελικά, και οι δύο διαφορετικές προσεγγίσεις αφορούν το ίδιο πρόβλημα, *την παροχή δηλαδή, όλο και μεγαλύτερης υπολογιστικής ισχύς στον τελικό χρήστη* και φυσικά έχει μεγάλη σημασία η σχέση κόστους/απόδοσης για τον συγκεκριμένο στόχο.

Για τις περισσότερες αρχιτεκτονικές, οι διεργασίες του χρήστη προσπελαύνουν το υποσύστημα διασύνδεσης απευθείας, με μόνη προστασία αυτή που παρέχει το λειτουργικό σύστημα με την μετάφραση των ιδεατών διευθύνσεων σε φυσικές. Η τεχνική αυτή παρέχει τον ίδιο βαθμό προστασίας για όλες τις προσπελάσεις μνήμης στο σύστημα και ταυτόχρονα έχει και το ελάχιστο κόστος. Ωστόσο, άλλες αρχιτεκτονικές χρησιμοποιούν τους πίνακες μετάφρασης του επεξεργαστή (Τηλέγραφος, PRAM, Shrimp, Hamlyn), ενώ όσες έχουν ειδικούς επεξεργαστές για το υποσύστημα διασύνδεσης (Flash, Meiko CS−2) χρησιμοποιούν επιπλέον πίνακες μετάφρασης με συνέπεια το λειτουργικό σύστημα να χρειάζεται να ενημερώνει και αυτούς.

Διαφορές παρατηρούμε και για το προγραμματιστικό μοντέλο που προωθούν οι διαφορετικές αρχιτεκτονικές. Μερικές από αυτές παρέχουν μηχανισμούς υποστήριξης και για το μοντέλο κοινόχρηστης μνήμης αλλά και για το μοντέλο ανταλλαγής μηνυμάτων. Ο Τηλέγραφος, το Flash, το Meiko CS−2, το PRAM και το Memory Channel ανήκουν στην παραπάνω κατηγορία. Αντίθετα το Alewife και η KSR παρέχουν μηχανισμούς που υποστηρίζουν περισσότερο, το μοντέλο κοινόχρηστης μνήμης. Οι αρχιτεκτονικές των Hamlyn, Shrimp προωθούν το μοντέλο ανταλλαγής μηνυμάτων.

Ο Τηλέγραφος και το PRAM, με την δυνατότητα των γρήγορων τοπικών εγγραφών και την ταυτόχρονη υποστήριξη αντιγράφων, παρέχουν μια γρήγορη και σχετικά φθηνή υλοποίηση κοινόχρηστης μνήμης, χωρίς ωστόσο να παρέχουν μηχανισμούς που θα διατηρούν την μνήμη συμβατή, αφήνοντας αυτή τη λειτουργία να την χειριστεί το λογισμικό. Αντίθετα το Alewife και το Flash παρέχουν μηχανισμούς καταλόγου για να διατηρούν συμβατές τις κρυφές μνήμες, οι οποίοι όμως επιβαρύνουν το κόστος της αρχιτεκτονικής.

Για την γρήγορη ανταλλαγή μηνυμάτων ο Τηλέγραφος (όπως και το Memory Channel) επιτρέπει την εκτέλεση απομακρυσμένων εγγραφών με επίσης μικρό κόστος, ενώ για την έκδοση τους δεν χρειάζεται παρά μόνο μια εντολή store του επεξεργαστή. Η αρχιτεκτονική του PRAM και του SHRIMP επιτρέπουν μόνο τοπικές εγγραφές τις οποίες και αυτόματα μεταφέρουν στον μακρυνό κόμβο. Η αρχιτεκτονική του Hamlyn αντίθετα παρέχει έναν μηχανισμό DMA με βελτιστοποιήσεις για την περίπτωση που το μήνυμα είναι μικρό, ενώ παράλληλα επιτρέπει την μεταφορά μηνυμάτων χωρίς να χρειάζεται το δίκτυο διασύνδεσης να εγγυάται την σειρά μεταφοράς των πακέτων (κάτι που εγγυάται το δίκτυο διασύνδεσης του

Τηλέγραφου). Ο μηχανισμός DMA έχει πολύ καλή απόδοση για τις περιπτώσεις που θέλουμε να στείλουμε μεγάλα μηνύματα (π.χ., μεγέθους σελίδας μνήμης). Μηχανισμό DMA παρέχει και η αρχιτεκτονική του SHRIMP. Ο Τηλέγραφος I δεν παρέχει μηχανισμό DMA και έτσι, για την αποστολή μεγάλων μηνυμάτων θα πρέπει να κάνει μεγάλο αριθμό απομακρυσμένων εγγραφών. Αυτό αυξάνει το κόστος μετάδοσης γιατί κάθε πακέτο απομακρυσμένης εγγραφής έχει μέγεθος 9 bytes από τα οποία τα 4 μόνο είναι τα πραγματικά δεδομένα.

Ο Τηλέγραφος όμως παρέχει απομακρυσμένες αναγνώσεις που επιτρέπουν την επιλεκτική προσπέλαση δεδομένων σε μακρυνούς κόμβους χωρίς να χρειάζεται να διατηρεί τοπικό αντίγραφο. Απομακρυσμένες αναγνώσεις παρέχει και το Memory Channel. Αντίθετα το SHRIMP πρέπει να διατηρεί αντίγραφο σε κάθε κόμβο που πρέπει να προσπελαύνει τα κοινά δεδομένα με αποτέλεσμα μεγαλύτερη σπατάλη μνήμης. Η αρχιτεκτονική του Hamlyn παρέχει και αυτή απομακρυσμένες αναγνώσεις, με την βοήθεια όμως της βιβλιοθήκης υποστήριξης.

Αρχιτεκτονική	Χρόνος
Flash	$\approx 1.75\mu s$
Alewife	$< 0.5\mu s$
KSR−I	$\approx 3\mu s$
Meiko CS−2	$> 50\mu s$
PRAM	$\approx 200\mu s$
Hamlyn	$\approx 8.8\mu s$
Memory Channel	$\approx 5.0\mu s$
Τηλέγραφος I	$\approx 8.2\mu s$

Πίνακας 5.2: Χρόνος εκτέλεσης ατομικών πράξεων στις διάφορες αρχιτεκτονικές.

Συγκρίνοντας τον τρόπο με τον οποίο οι διάφορες αρχιτεκτονικές παρέχουν ατομικές πράξεις παρατηρούμε ότι οι αρχιτεκτονικές των Flash, Alewife και Meiko CS−2 παρέχουν ειδική υποστήριξη από το hardware, είτε με την μορφή ειδικού επεξεργαστή για το σύστημα διασύνδεσης (Flash, Meiko CS−2), είτε με επιπλέον μνήμη του συστήματος (Alewife). Το κόστος και των δύο προσεγγίσεων θεωρείται μεγάλο, δεδομένου του κόστους που εισάγει η σχεδίαση και κατασκευή ενός επεξεργαστή αλλά και του κόστους της επιπλέον μνήμης που απαιτεί η λύση του Alewife. Ο Τηλέγραφος παρέχει ατομικές πράξεις απευθείας σε hardware αποφεύγοντας όμως, το κόστος κατασκευής ενός ειδικού επεξεργαστή ή την χρήση επιπλέον μνήμης. Το PRAM παρέχει την υποστήριξη για την υλοποίηση ατομικών πράξεων με λογισμικό και η μέτρηση έγινε χρησιμοποιώντας υπολογιστές με τον επεξεργαστή Intel 80286. Για την αρχιτεκτονική της μηχανής KSR, η ατομική προσπέλαση σε κάποια δεδομένα απαιτεί ο κόμβος που θα εκτελέσει την πράξη να έχει το μοναδικό αντίτυπο της υποσελίδας

στην οποία βρίσκεται η ατομική μεταβλητή. Για να εξασφαλιστεί κάτι τέτοιο θα πρέπει πρώτα, να στείλει μηνύματα στους κόμβους που διαθέτουν αντίγραφα για να τα ακυρώσει. Αντίθετα για τον Τηλέγραφο, οι ατομικές πράξεις εκτελούνται σε μεταβλητές που βρίσκονται ήδη σε σελίδες που δεν διαθέτουν αντίγραφα, χωρίς να υπάρχει η ανάγκη για ακυρώσεις αντιγράφων.

Στον πίνακα 5.2 φαίνονται οι χρόνοι εκτέλεσης μιας ατομικής πράξης για τις διάφορες αρχιτεκτονικές. Αν και οι χρόνοι αυτοί για τις περισσότερες αρχιτεκτονικές είναι εκτίμηση του πραγματικού χρόνου (είτε μετρήσεις σε μη ενεργό (idle) δίκτυο), ωστόσο είναι ενδεικτικοί της απόδοσης της κάθε αρχιτεκτονικής. Ο χρόνος που αναφέρεται για το PRAM είναι ο χρόνος εκτέλεσης μιας κενής απομακρυσμένης κλήσης συνάρτησης (null RPC). Οι χρόνοι για τις μηχανές με ειδικούς επεξεργαστές είναι μικροί με εξαίρεση την μηχανή Meiko CS−2, για την οποία προκύπτει πολύ μεγάλο κόστος εκτέλεσης, ενώ η ατομική πράξη που μετρήθηκε είναι ένα $fetch_and_store$. Ο χρόνος που αναφέρεται στην μηχανή KSR αφορά την εκτέλεση μιας πράξης lock χωρίς να υπάρχει η ανάγκη για ακύρωση αντιγράφων. Αν πρέπει να γίνουν ακυρώσεις αντιγράφων ο χρόνος εκτέλεσης της πράξης ξεπερνά τα 50μs. Αντίθετα ο Τηλέγραφος έχει κόστος της τάξης των λίγων microseconds το οποίο, κατά κύριο λόγο, οφείλεται στην καθυστέρηση του δικτύου. Η μέτρηση που αναφέρεται για το Hamlyn δεν είναι το κόστος ατομικής πράξης αλλά είναι το κόστος μεταφοράς του μικρότερου δυνατού πακέτου στο δίκτυο (αποστολή και παραλαβή).

Ανάλογα με τον τρόπο με τον οποίο ενεργοποιούνται οι ατομικές πράξεις ή οι μακροεντολές που υποστηρίζει το υποσύστημα διασύνδεσης, διακρίνουμε τις αρχιτεκτονικές σε αυτές που παρέχουν ένα μόνο σύνολο από καταχωρητές για την ενεργοποίηση των εντολών και σε αυτές που παρέχουν ξεχωριστό σύνολο καταχωρητών για κάθε διεργασία του κόμβου. Ο Τηλέγραφος Ι παρέχει μόνο ένα σύνολο καταχωρητών, σε αντίθεση με τα Hamlyn, Flash και Meiko CS−2, όπου η κάθε διεργασία διαθέτει δικό της σύνολο καταχωρητών ενεργοποίησης ατομικών πράξεων/μακροεντολών. Η αρχιτεκτονική του SHRIMP δεν παρέχει ξεχωριστό σύνολο καταχωρητών για κάθε διεργασία, αλλά υλοποιεί μια διαφορετική τεχνική χρησιμοποιώντας επιπλέον μη υπαρκτές φυσικές διευθύνσεις για κάθε υπαρκτή φυσική διεύθυνση του συστήματος. Προσπελαύνοντας τις μη υπαρκτές διευθύνσεις ενεργοποιούνται οι αντίστοιχες μακροεντολές.

Η ύπαρξη ξεχωριστών συνόλων καταχωρητών (αλλά και η λύση που ακολουθήθηκε από το SHRIMP) απλουστεύει το λογισμικό υποστήριξης, αφού δεν υπάρχει λόγος υποστήριξης αδιάλλειπτης εκτέλεσης τμημάτων κώδικα, αλλά επιβαρύνει το hardware που θα πρέπει να διαθέτει επιπλέον καταχωρητές για κάθε διεργασία που θα εκτελείται από το σύστημα (ενώ η λύση που ακολουθήθηκε από το SHRIMP μειώνει το μέγεθος των διαθέσιμων φυσικών διευθύνσεων για το σύστημα).

Κεφάλαιο 6

Συμπεράσματα–Επεκτάσεις

Η εργασία αυτή ασχολείται με την ανάπτυξη λογισμικού για την υποστήριξη συστημάτων διασύνδεσης υπολογιστών με δίκτυα υψηλών ταχυτήτων. Σαν πλατφόρμα υλοποίησης χρησιμοποιήθηκε η αρχιτεκτονική του Τηλέγραφου και συγκεκριμένα το πρωτότυπο του Τηλέγραφου Ι που αποτελεί και την πρώτη υλοποίηση της αρχιτεκτονικής αυτής. Περιγράφηκαν οι βασικές λειτουργίες του πρωτοτύπου και οι αναγκαίες επεμβάσεις στο λειτουργικό σύστημα για την υποστήριξή τους. Επίσης έγινε μια συγκριτική παρουσίαση της αρχιτεκτονικής του Τηλέγραφου και άλλων αρχιτεκτονικών μέσα από την οποία διαγράφεται η μελλοντική πορεία των πολυεπεξεργαστικών συστημάτων. Ο τελικός στόχος παραμένει η μέγιστη αξιοποίηση των διαθέσιμων υπολογιστικών συστημάτων.

Η αρχιτεκτονική του Τηλέγραφου, παρέχοντας λειτουργίες που υποστηρίζουν και τα δύο βασικά προγραμματιστικά μοντέλα στηρίζει τον παραπάνω στόχο. Η μέγιστη δυνατή αξιοποίηση δεν είναι δυνατόν να προκύψει υποστηρίζοντας μόνο μία από τις δύο τεχνικές αλλά θα πρέπει ο προγραμματιστής να έχει την δυνατότητα να επιλέγει δυναμικά τον τρόπο με τον οποίο θα χρησιμοποιεί την διαθέσιμη κοινή μνήμη του δικτύου. Οι εφαρμογές που υλοποιήθηκαν χρησιμοποιώντας τον προσομοιωτή αναδεικνύουν αυτό το χαρακτηριστικό της αρχιτεκτονικής του Τηλέγραφου. Το σύστημα μπορεί πολύ εύκολα να υποστηρίξει περιβάλλον κοινόχρηστης μνήμης με την υποστήριξη των πολλαπλών αντιγράφων που παρέχει, αλλά εξίσου εύκολα μπορεί να χρησιμοποιηθεί και για εφαρμογές που χρειάζονται μόνο να ανταλλάσουν μηνύματα (πρόβλημα παραγωγού–καταναλωτή, remote paging) αξιοποιώντας τις τηλε–εγγραφές.

Από την συγκριτική μελέτη των διαφορετικών αρχιτεκτονικών προκύπτει ξεκάθαρα ότι μπορούμε να πάρουμε την μέγιστη απόδοση του συστήματος μόνο αν μειωθεί στο ελάχιστο η παρέμβαση του λειτουργικού συστήματος. Οι παραδοσιακές αρχιτεκτονικές ανταλλαγής μηνυμάτων χρησιμοποιούν κλήσεις συστήματος και αντιγράφουν τα μεταφερόμενα δεδομένα

από τον χώρο διευθύνσεων του χρήστη στον χώρο του πυρήνα για να ακολουθήσει άλλη μια αντιγραφή στον μακρυνό κόμβο πριν την παραλαβή του μηνύματος. Ακόμη, για την παραλαβή του μηνύματος πρέπει να γίνει διακοπή από το λειτουργικό σύστημα, για να ενημερωθεί η διεργασία που περιμένει το μήνυμα. Το επιπλέον κόστος του λογισμικού για να υλοποιήσει ένα τέτοιο πρωτόκολλο είναι πολύ μεγάλο και προκύπτει να είναι πολύ μεγαλύτερο από τον χρόνο που χρειάζεται το δίκτυο μεταφοράς για να μεταφέρει το μήνυμα. Το πρόβλημα αυτό το αντιμετωπίζουν όλες οι αρχιτεκτονικές εκμεταλλευόμενες την προστασία που παρέχει η ιδεατή μνήμη που υποστηρίζεται από όλα σχεδόν τα λειτουργικά συστήματα.

Στην εργασία αυτή, η προσπάθεια ήταν να αποφευχθεί οποιαδήποτε παρεμβολή του λειτουργικού συστήματος για την υποστήριξη του Τηλέγραφου η οποία θα επιβάρυνε σημαντικά το κόστος εκτέλεσης των λειτουργιών του. Οι ατομικές πράξεις υποστηρίζονται μέσω του κώδικα Pal και παρουσιάζεται μια διαφορετική προσέγγιση για να παρέχεται άμεση υποστήριξη από το ίδιο το λειτουργικό σύστημα.

6.1 Μελλοντική εργασία

Αν και η εργασία εξαρτάται σε μεγάλο βαθμό από την συγκεκριμένη αρχιτεκτονική του Τηλέγραφου, υπάρχουν γενικότερα θέματα τα οποία μπορούν να μελετηθούν και δεν αφορούν αποκλειστικά την αρχιτεκτονική του Τηλέγραφου και την υλοποίηση της:

- ο προσομοιωτής που υλοποιήθηκε αν και αρχικά δεν είχε στόχο την εκτέλεση παράλληλων εφαρμογών μπορεί να επεκταθεί ώστε να επιτρέπει απευθείας την εκτέλεση εφαρμογών (προσομοίωση σε επίπεδο γλώσσας μηχανής). Βέβαια, υπάρχουν αρκετά τεχνικά προβλήματα που αφορούν τον τρόπο με τον οποίο θα εξυπηρετούνται οι εντολές *load* και *store* από τον προσομοιωτή αλλά είναι δυνατόν να ξεπεραστούν με ειδικές παρεμβάσεις στον διαχειριστή διακοπών (interrupt handler) του λειτουργικού συστήματος και εκτελώντας τα προγράμματα με απενεργοποιημένη την δυνατότητα του επεξεργαστή να εκτελεί δύο εντολές ταυτόχρονα[1] (dual issue)·

- η εκτέλεση πραγματικών παράλληλων εφαρμογών μπορεί να οδηγήσει σε βαθύτερη μελέτη της συμπεριφοράς τους. Ετσι, για παράδειγμα, μπορεί να ελεγχθεί η συχνότητα εκτέλεσης των ατομικών πράξεων και να διερευνηθεί το κόστος τους σε σύγκριση με το συνολικό κόστος εκτέλεσης των εφαρμογών. Από μια τέτοια μελέτη θα μπορούν να προκύψουν συμπεράσματα για το αν οι ατομικές πράξεις και οι εντολές συγχρονισμού αξίζει να γίνονται απευθείας από το hardware ή όχι·

[1] Ο επεξεργαστής Alpha–AXP έχει τέτοια δυνατότητα.

- ακόμη, είναι ενδιαφέρον να υλοποιηθεί ένας κατανεμημένος διαχειριστής της μνήμης του Τηλέγραφου που θα λαμβάνει υπ'όψη του τους μετρητές προσπελάσεων που διατηρεί ο Τηλέγραφος. Ετσι, θα μπορούν να μελετηθούν αλγόριθμοι για την αρχική τοποθέτηση των δεδομένων των εφαρμογών, αλλά και για την δυναμική επανατοποθέτηση τους, ώστε να επιτυγχάνεται η βέλτιστη προσπέλαση των δεδομένων σε κάθε χρονική στιγμή.

Παράρτημα Α

Κώδικας Pal

Α.1 Γενική περιγραφή

Ο χώρος διευθύνσεων του κώδικα Pal είναι προκαθορισμένος από την αρχιτεκτονική του επεξεργαστή Alpha. Τα δύο πρώτα MegaBytes της κύριας μνήμης του υπολογιστή είναι δεσμευμένα και χρησιμοποιούνται από το σύστημα για τον κώδικα Pal και για τον κώδικα εκκίνησης/διαγνωστικών ελέγχων του συστήματος.

Στον χώρο αυτό, των 2 MegaBytes, υπάρχουν οι σταθερές θέσεις (slots) του κώδικα Pal αλλά και ο επιπλέον διαθέσιμος χώρος (scratch area) για τον κώδικα Pal. Ο κώδικας Pal έχει προκαθορισμένες διευθύνσεις στις οποίες μπορεί να βρίσκεται, τουλάχιστον για όσα τμήματα του θα μπορούν να εκτελούνται και μέσω της assembly εντολής *call_pal*. Τα τμήματα του, τα οποία δεν μπορούν να καλεστούν μέσω της εντολής *call_pal* βρίσκονται και αυτά στον αρχικό χώρο των 2 MegaBytes αλλά δεν καταλαμβάνουν κάποιες από τις προκαθορισμένες θέσεις.

Οι ελεύθερες θέσεις του κώδικα Pal προέκυψαν θεωρώντας ως αχρησιμοποίητες τις θέσεις που ο μεταφραστής assembly του συστήματος δεν έχει δώσει κωδικό όνομα. Πρακτικά αυτό επιβεβαιώθηκε και με την διαπίστωση ότι δεν άλλαζε η λειτουργία του συστήματος.

Α.2 Προγραμματισμός σε κώδικα Pal

Ο κώδικας Pal αποτελείται από το σύνολο των εντολών του επεξεργαστή που μπορούν να χρησιμοποιούν τα προγράμματα των χρηστών, επαυξημένο με ειδικές εντολές που επιτρέπουν την προσπέλαση στους ειδικούς εσωτερικούς καταχωρητές του επεξεργαστή και οι οποίες εκτελούνται στο ειδικό περιβάλλον εκτέλεσης, όπως αυτό περιγράφηκε στην υποενότητα 2.4.1.

Για τον προγραμματισμό σε κώδικα Pal χρησιμοποιήθηκε ο μεταφραστής assembly του συστήματος, όπως παρέχεται με το λειτουργικό σύστημα Digital Unix, και ένας επιπλέον

μεταφραστής (HAL assembler), ο οποίος διατίθεται ελεύθερα και μεταφράζει απευθείας μακρο−εντολές τύπου Vax (στις οποίες συμπεριλαμβάνονται και οι επιπλέον εντολές που υπάρχουν στον περιβάλλον Pal) σε γλώσσα μηχανής για τον επεξεργαστή Alpha.

Για τις προσπελάσεις στους ειδικούς καταχωρητές ακολουθήθηκαν οι περιορισμοί που αναφέρονται στο Hardware Reference Manual για τον επεξεργαστή [9]. Η εγγραφή νέου κώδικα στις προκαθορισμένες διευθύνσεις γίνεται κατά την εκκίνηση του συστήματος μέσω του προγράμματος *osf_boot* το οποίο είναι υπεύθυνο για να ξεκινήσει την εκτέλεση του πυρήνα του λειτουργικού συστήματος Mach. Κατά την διάρκεια εκτέλεσης του προγράμματος *osf_boot* έχουμε την δυνατότητα να γράψουμε τον δικό μας κώδικα Pal χρησιμοποιώντας απευθείας, φυσικές διευθύνσεις μνήμης. Ο κώδικας μας βρίσκεται στο τμήμα εντολών (text segment) του προγράμματος από όπου και αντιγράφεται στις προκαθορισμένες θέσεις.

Παράδειγμα

Το παρακάτω είναι ο κώδικας Pal που τοποθετήθηκε στο 51ο slot και μας επιτρέπει να εκτελέσουμε κώδικα Pal από σελίδες μνήμης του πυρήνα. Ο προγραμματιστής που θα καλέσει αυτή του ρουτίνα του κώδικα Pal θα πρέπει να δώσει σαν όρισμα στον καταχωρητή *a0* την ιδεατή διευθύνση του πυρήνα, στην οποία υπάρχει ο κώδικας Pal που θα εκτελεστεί. Για τις ιδεατές διευθύνσεις του πυρήνα, οι αντίστοιχες φυσικές προκύπτουν με αποκοπή μερικών σημαντικών ψηφίων (εξαρτάται από το μέγεθος των φυσικών διευθύνσεων της υλοποίησης).

```
.text
.long 0x47ff041f  /* No op */
.long 0x66310024  /* Keep the old exc_addr in r17 */

.long 0x46003410  /* bis r16,#1,r16  Stay in PALMODE !!! */
.long 0x47ff041f

.long 0x76100024  /* hw_mtpr a0, exc_addr */
.long 0x47ff041f

.long 0x47ff041f
.long 0x47ff041f

.long 0x47ff041f
.long 0x47ff041f
```

```
    .long 0x7bff8000  /* hw_rei ...hopefully we are still in Palmode */
    .long 0x47ff041f

    .long 0x47ff041f
    .long 0x47ff041f

    .long 0x47ff041f
    .long 0x47ff041f
```

Ο κώδικας χρησιμοποιεί τον καταχωρητή *r17* για να κρατήσει την αρχική διεύθυνση επιστροφής και στην συνέχεια αντιγράφει στον ειδικό εσωτερικό καταχωρητή την νέα διεύθυνση στην οποία θα συνεχιστεί η εκτέλεση εντολών. Η μετάβαση γίνεται με την εντολή *hw_rei*. Για να επιστρέψουμε από το περιβάλλον Pal στον κανονικό περιβάλλον εκτέλεσης πρέπει πρώτα να αντιγράψουμε τον καταχωρητή *r17* στον εσωτερικό καταχωρητή *exc_addr* και στην συνέχεια να εκτελέσουμε την εντολή επιστροφής από το περιβάλλον εκτέλεσης κώδικα Pal. Η ακολουθία των εντολών είναι η ακόλουθη:

```
hw_mtpr r17, exc_addr /* .long 0x76310024 */
nop                   /* .long 0x47FF041F */

nop
nop

nop
nop

nop                   /* .long 0x47FF041F */
hw_rei                /* .long 0x7BFF8000 */
```

Ο αριθμός των ενδιάμεσων κενών εντολών (no−ops) που πρέπει να υπάρχουν μεταξύ μιας εγγραφής στους εσωτερικούς καταχωρητές του επεξεργαστή (εντολή *hw_mtpr*) και μιας εντολής ανάγνωσης από τον ίδιο καταχωρητή (εντολές *hw_rei,hw_mfpr*) καθορίζεται από την υλοποίηση του επεξεργαστή.

Με αυτόν τον τρόπο επιτυγχάνεται η εκτέλεση περισσοτέρων από 16 εντολών (μέχρι 2048, που μπορούν να χωρέσουν σε μια σελίδα μνήμης[1]).

Οι σελίδες μνήμης του πυρήνα που έχουν τον κώδικα Pal θα πρέπει να είναι μόνιμα ‘‘κλειδωμένες’’ (pinned) στην μνήμη του συστήματος.

[1]Αυτό δεν είναι περιορισμός της συγκεκριμένης τεχνικής αλλά δεν υπήρχε λόγος να επιτρέπονται περισσότερες εντολές.

Βιβλιογραφία

[1] *1983/84 Components Data Book*, chapter Z8530 SCC Serial Communications Controller, Product Specification, pages 409−−429.
Zilog, Campbell, CA 95008, September 1983.

[2] Advanced Micro Devices, Inc., Sunnyvale, California.
Serial Communications Controller AmZ8030/8530, 1986.

[3] Anant Agarwal, David Chaiken, Godfrey D'Souza, Kirk Johnson, David Kranz, John Kubtiatowicz, Kiyoshi Kurihara, Beng−Hong Lim, Gino Maa, Dan Nussbaum, Mike Parkin, and Donald Yeung.
The MIT Alewife Machine: A Large−Scale Distributed Memory Multiprocessor.
In *Proceedings of the Workshop on Scalable Shared−Memory Multiprocessors*, Seattle, June 1990. Kluwer Academic Publishers.

[4] Thomas E. Anderson, David E. Culler, David A. Patterson, and the NOW team.
A Case for NOW (Networks of Workstations).
IEEE MICRO, February 1995.

[5] Matthias A. Blumrich, Cezary Dubnicki, Edward W. Felten, Kai Li, and Malena R. Mesarina.
Two Virtual Memory Mapped Network Interface Designs.
In *The Hot Interconnects II Symposium Record*, pages 134−−142, August 1994.

[6] Matthias A. Blumrich, Kai Li, Richard Alpert, Cezary Dubnicki, Edward W. Felten, and Jonathan Sandberg.
Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer.
In *Proceedings of the Twenty−First International Symposium on Computer Architecture*, pages 142−−153, April 1994.

[7] Greg Buzzard, David Jacobson, Scott Marovich, and John Wilkes.
Hamlyn: a high−performance network interface with sender−based memory management.
In *Proceedings of the Hot Interconnects III Symposium*, August 1995.

[8] Digital Equipment Corporation, Maynard,Massachusetts.
DEC OSF/1 Assembly Language Programmer's Guide.

[9] Digital Equipment Corporation, Maynard,Massachusetts.
DECchip 21064−AA Microprocessor Hardware Reference Manual.

[10] Digital Equipment Corporation, Maynard,Massachusetts.
TURBOchannel Hardware Specification.

[11] George Dramitinos.
Porting the Mach Operating System on a Pelican.
Senior Thesis, University of Crete, May 1994.

[12] R. Gillet.
Memory Channel.
In *Proceedings of the Hot Interconnects III Symposium*, August 1995.

[13] David Golub, Randall Dean, Alessandro Forin, and Richard Rashid.
Unix as an Application Program.
In *Proceedings of the USENIX Summer Conference*, June 1990.

[14] John Heinlein, Kourosh Gharachorloo, Scott Dresser, and Anoop Gupta.
Integration of Message Passing and Shared Memory in the Stanford FLASH Multiprocessor.
In *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, October 1994.

[15] J. L. Hennessy and D. A. Patterson.
Computer Architecture: A Quantitative Approach.
Morgan Kaufmann Publishers, Inc., 1990.

[16] Jan Elder, Jim Lipkis, Edith Schonberg.
Process Management for Highly Parallel Unix Systems.
In *Proceedings of the USENIX Workshop on UNIX and Supercomputers*, pages 1−−17, September 1988.

[17] Manolis Katevenis.
Telegraphos: High−Speed Communication Architecture for Parallel and Distributed Computer Systems.
Technical Report 123, ICS−FORTH, May 1994.
ftp://ftp.csi.forth.gr/tech−reports/1994/1994.TR123.Telegraphos.ps.Z.

[18] David Kranz, Beng−Hong Lim, Donald Yeung, and Anant Agarwal.
Low−Cost Support for Fine−Grain Synchronization in Multiprocessors.

[19] J. Kuskin, D. Ofelt, M. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. Hennessy.
The FLASH Multiprocessor.
In *Proc. 21−th International Symposium on Comp. Arch.*, pages 302−−313, Chicago, IL, April 1994.

[20] R. J. Lipton and D. N. Serpanos.
Uniform−cost Communication in scalable multiprocessors.
In *Proceedings of the International Conference on Parallel Processing*, pages I429−−I432, August 1990.

[21] Keith Loepere, editor.
Mach 3 Kernel Interfaces.
Open Software Foundation and Carnegie Mellon University, 2.2 edition, July 1992.

[22] E. P. Markatos, Manolis Katevenis, George Kalokerinos, and Dimitrios Serpanos.
An Efficient Processor−Network Interface for Local Area Multiprocessors.
In *Proceedings of the 4th International Workshop on SCI−based High−Performance Low−Cost Computing*, pages 23−−32, October 1995.
http://www.ics.forth.gr/ markatos/papers/markatos/scizzl4.ps.gz.

[23] Evangelos P. Markatos and George Dramitinos.
Implementation of a Reliable Remote Memory Pager.
In *USENIX 1996 Technical Conference*, San Diego, CA., January 1996.

[24] Evangelos P. Markatos, George Dramitinos, and Kosmas Papachristos.
Implementation and Evaluation of a Remote Memory Pager.
Technical Report 129, Institute of Computer Science, FORTH, March 1995.
ftp://ftp.csi.forth.gr/tech−reports/1995/1995.TR129.remote_memory_paging.ps.gz.

[25] Evangelos P. Markatos and Manolis G.H. Katevenis.
Telegraphos: High−Performance Networking for Parallel Processing on Workstation Clusters.
In *Proceedings of the Second International Symposium on High−Performance Computer Architecture, (HPCA)*, San Jose, CA, USA, February 1996.

[26] John M. Mellor−Crummey and Michael L. Scott.
Algorithms for Scalable Synchronization on Shared−Memory Multiprocessors.
TOCS, 9(1):21−−65, February 1991.

[27] Thomas Pfenning.
Shared Memory Programming on the Meiko CS−2.

Technical Report 94.180, University of Koln, 1994.
Submitted to HPCN Europe 1995, Mainland.

[28] Richard Rashid, Robert Baron, Alessandro Forin, David Golub, Michael Jones, Daniel Julin, Douglas Orr, and Richard Sanzi.
Mach: A Foundation for Open Systems.
In *Proceedings of the second WorkShop on Workstation Operating Systems*, September 1989.

[29] Dimitrios N. Serpanos and Richard J. Lipton.
The Design of a High−Speed Network.
In *Third IFIP WG 6.4 Conference on High Speed Networking*, pages 215−−224, March 1991.

[30] Richard L. Sites, editor.
Alpha Architecture Reference Manual.
Digital Press, 1992.

[31] T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser.
Active Messages: a Mechanism for Integrated Communication and Computation.
In *Proc. 19−th International Symposium on Comp. Arch.*, pages 256−−266, Gold Coast, Australia, May 1992.