ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Σχεδίαση και Υλοποίηση Μηχανισμού Μεταφοράς Δεδομένων από Συσκευές Αποθήκευσης σε Δίκτυο Myrinet, Χωρίς τη Μεσολάβηση της Ιεραρχίας Μνήμης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναστάσιος Α. Νάνος

Επιβλέπων: Νεκτάριος Κοζύρης,
Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2006

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Σχεδίαση και Υλοποίηση Μηχανισμού Μεταφοράς Δεδομένων από Συσκευές Αποθήκευσης σε Δίκτυο Myrinet, Χωρίς τη Μεσολάβηση της Ιεραρχίας Μνήμης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναστάσιος Α. Νάνος

Επιβλέπων: Νεκτάριος Κοζύρης,
Επ. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 7^{η} Νοεμβρίου 2006.

.....	.....	.....
Ν. Κοζύρης,	Γ. Παπακωνσταντίνου,	Ν. Παπασπύρου
Επ. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Νοέμβριος 2006.

.....

Αναστάσιος Α. Νάνος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Ηλεκτρονικών Υπολογιστών Ε.Μ.Π

Περίληψη

Η παρούσα εργασία πραγματεύεται τη μελέτη ενός μοντέλου μεταφοράς δεδομένων από συσκευές αποθήκευσης σε δίκτυα Myrinet χωρίς τη μεσολάβηση του επεξεργαστή και της κεντρικής μνήμης του συστήματος που διαθέτει τοπικά τη συσκευή αποθήκευσης. Το μοντέλο αφορά στο σχεδιασμό και στην υλοποίηση μιας εικονικής συσκευής ανάκτησης δεδομένων (πελάτης), μιας εφαρμογής χώρου χρήστη (εξυπηρετητής) καθώς και στη μελέτη και τροποποίηση δομών στον πυρήνα του Linux και στον οδηγό της συσκευής δικτύου Myrinet. Το μοντέλο που σχεδιάστηκε ανταποκρίνεται πλήρως στον αρχικό στόχο και η εφαρμογή εξυπηρέτησης που υλοποιήθηκε σε συνδυασμό με την τροποποίηση του πυρήνα και του οδηγού της συσκευής αποτελεί μια ικανοποιητική εφαρμογή του αρχικού μοντέλου.

Αρχικά παρουσιάζονται οι θεωρητικές βάσεις για τη μελέτη και το σχεδιασμό του μοντέλου. Αναφέρονται οι μέθοδοι πρόσβασης σε συσκευές Εισόδου / Εξόδου στο λειτουργικό σύστημα Linux καθώς και οι βασικές παράμετροι εγκατάστασης συνδέσεων Myrinet για λήψη και αποστολή μηνυμάτων.

Στη συνέχεια αναλύεται ο σχεδιασμός του μοντέλου συγκριτικά με ήδη υπάρχουσες δικτυακές συσκευές ανάκτησης δεδομένων (όπως η NBD) πάνω από TCP/IP, μια αντίστοιχη συσκευή με την NBD πάνω από Myrinet και η gmblock, η συσκευή της εργασίας, που λειτουργεί με δίκτυο Myrinet.

Συγκεκριμένα η εφαρμογή εξυπηρέτησης ανακτά δεδομένα από το μέσο αποθήκευσης απευθείας σε χώρο μνήμης της συσκευής δικτύου, χωρίς να παρεμβαίνει ο πυρήνας (επεξεργαστής / κεντρική μνήμη) στην αντιγραφή. Αυτό επιτυγχάνεται με την απεικόνιση της μοιραζόμενης (SRAM) μνήμης της κάρτας δικτύου από το χώρο φυσικών διευθύνσεων διαύλου στο χώρο εικονικών διευθύνσεων. Με την προσθήκη μιας ακόμα ζώνης μνήμης όπου εισάγονται οι φυσικές διευθύνσεις διαύλου επιτυγχάνεται η ενσωμάτωση των διευθύνσεων συσκευών Εισόδου / Εξόδου στο σύστημα διαχείρισης μνήμης του πυρήνα.

Τέλος δίνεται έμφαση στην υλοποίηση των εφαρμογών (πελάτη – εξυπηρετητή) καθώς και στην τροποποίηση των απαραίτητων δομών στον πυρήνα και στον οδηγό της συσκευής δικτύου Myrinet για την επίτευξη του βελτιστοποιημένου μονοπατιού.

Λέξεις Κλειδιά

Δικτυακή Συσκευή Αποθήκευσης, Συσκευή Block, Δικτυακή Block Συσκευή, Λειτουργικό Σύστημα Linux, Στρωμα διαχείρισης συσκευών Block, GM, LANai, libgm, gmblock, nbd

Abstract

The objective of this study is the design and implementation of a model transfering data from storage devices to Myrinet networks bypassing the memory hierarchy of the system, which hosts the storage device locally.

This model deals with the implementation of a virtual read-only block device (client) and a userspace application, which serves the device (server); however, in order to achieve the optimized data path, several modifications in the Linux Kernel and the Myrinet device driver were introduced. The initial objective has been accomplished: the optimized data path drives data through PCI to the Myrinet NIC from any block device.

First, we study the fundamental theoretical concepts required to design the model. We present methods to access storage devices in the host Operating System (Linux) and methods to install and use a Myrinet link to send and receive messages.

Moreover, the design of the model is analyzed and confronted with existing devices, like nbd over TCP/IP or an nbd over Myrinet. Specifically, the userspace application reads data from the storage device directly through the PCI bus, bypassing the host CPU and the host main memory. This is achieved by mapping the SRAM (Static RAM) of the LANai proccessor (Myrinet NIC) to virtual memory addresses. With the introduction of a memory zone in the Linux Kernel, the physical bus addresses (PCI) are incorporated in the memory management policy of the Linux Kernel.

Finally, we present the implementation of the applications (client – server), as well as the modifications required to achieve the optimized data path (Linux kernel / Myrinet driver).

Keywords

Myrinet, Network Block Device, Network Attached Storage, Storage Area Network, Linux, Linux Block Layer, GM, LANai, libgm, gmblock, nbd

Ευχαριστίες

Η παρούσα εργασία εκπονήθηκε στο Εργαστήριο Υπολογιστικών Συστημάτων του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών του Εθνικού Μετσοβίου Πολυτεχνείου υπό την επίβλεψη του επίκουρου καθηγητή κύριου Νεκτάριου Κοζύρη.

Θα ήθελα να ευχαριστήσω θερμά τον κύριο Κοζύρη για την ευκαιρία που μου έδωσε να ασχοληθώ με το συγκεκριμένο τομέα της επιστήμης των υπολογιστών στο Εργαστήριο Υπολογιστικών Συστημάτων, καθώς και για τις πολύτιμες συμβουλές του καθόλη τη διάρκεια περάτωσης της εργασίας.

Η βοήθεια του υποψήφιου διδάκτορα Βαγγέλη Κούκη ήταν ανεκτίμητη τόσο για τις καίριες παρεμβάσεις του στην υλοποίηση του μοντέλου όσο και για τις τεχνικές γνώσεις που μου μετέδωσε, χωρίς τις οποίες η ολοκλήρωση της εργασίας δε θα ήταν εφικτή.

στον αδερφό μου Παναγιώτη,
στους γονείς μου, Αντώνη και Ελευθερία

Περιεχόμενα

1 Συστήματα Αποθήκευσης **21**

1.1 Συσκευές Αποθήκευσης 22
1.1.1 Σκληρός Δίσκος 22
1.1.2 Συστοιχία Σκληρών Δίσκων (RAID) 23
1.1.2.1 Έλεγχοι λαθών 24
1.2 Είσοδος / Έξοδος 27
1.2.1 Διεπαφή συσκευών Ε/Ε με τις Επεξεργαστικές μονάδες 29
1.2.2 Απευθείας Πρόσβαση στη Μνήμη (Direct Memory Access) 30
1.2.3 Πρότυπα Διαδρόμου Ε/Ε για Συσκευές Αποθήκευσης . . 31
1.2.3.1 ATA 31
1.2.3.2 SCSI 31
1.2.3.3 Μια διαφορετική Προσέγγιση 32
1.3 Τεχνολογίες συσκευών αποθήκευσης 33
1.3.1 Τοπική Αποθήκευση 34
1.3.2 Δικτυακή Αποθήκευση 34
1.3.2.1 NAS 34
1.3.2.2 SAN 35
1.4 Τρόπος Αποθήκευσης Δεδομένων 35

2 Δίκτυα Διασύνδεσης **37**

2.1 Ethernet 38
2.1.1 Τοπολογία και Δομή Δικτύου 39
2.1.2 Ethernet και OSI 40
2.1.3 Δομή του πλαισίου Ethernet 41
2.1.4 Μετάδοση Πλαισίου 42
2.1.5 Η μέθοδος προσπέλασης CSMA/CD 43
2.2 TCP/IP 44
2.2.1 IP 45
2.2.1.1 Δεδομενογράφημα IP 45
2.2.1.2 Διευθυνσιοδότηση IP 47
2.2.1.3 Υποδίκτυα 48

2.2.1.4 Πρωτόκολλο ARP 49
2.2.1.5 Δρομολόγηση IP 50
2.2.2 Transmission Control Protocol 52
2.2.2.1 Συνδέσεις TCP 53
2.2.2.2 Δομή πακέτου TCP 53
2.3 Myrinet 54
2.3.1 Δομή πακέτου 59
2.3.2 Διεπαφές Δικτύου 60
2.3.3 Δρομολόγηση 60
2.3.4 Πρωτόκολλο διεπαφής δικτύου / Το λογισμικό GM . . . 61
2.3.4.1 GM ports / ακροσημεία 62
2.3.4.2 Αρχικοποίηση του GM 65
2.3.4.3 Μνήμη 66
2.3.4.4 Αποστολή 66
2.3.4.5 Λήψη 67
2.3.5 Ένα σενάριο αποστολής 72

3 Συστήματα αποθήκευσης στο Linux 75
3.1 Το Linux 75
3.1.1 Ο πυρήνας του Linux 77
3.1.2 Modules 78
3.1.3 Κλάσεις συσκευών και modules 78
3.2 Χειρισμός Μνήμης 81
3.2.1 Φυσικές διευθύνσεις και σελίδες 82
3.2.2 Μνήμη High και Low 82
3.2.3 Page Tables 83
3.2.4 Virtual Memory Areas (Περιοχές εικονικής μνήμης) . . . 83
3.2.5 Ζώνες Μνήμης (Memory Zones) 84
3.3 Αρχιτεκτονική Εισόδου / Εξόδου 85
3.3.1 I/O ports 85
3.3.2 Συσκευές στο Linux 87
3.3.2.1 Αρχεία συσκευών 87
3.3.3 Οδηγοί Συσκευών (Device Drivers) 88
3.3.4 Direct I/O 90
3.4 Block Device Drivers 91
3.4.1 Βασικές δομές δεδομένων του στρώματος block 92
3.4.2 Block αιτήσεις Ε/Ε 93
3.4.3 Ουρές αιτήσεων 94
3.4.4 Διεπαφή οδηγού συσκευής block 94

4 Σχεδιασμός **99**
4.1 Μια τυπική δικτυακή συσκευή αποθήκευσης 99
4.1.1 Πελάτης 99
4.1.2 Εξυπηρετητής 102
4.1.3 Δίκτυο 106
4.2 Μια δικτυακή συσκευή αποθήκευσης για Myrinet 106
4.2.1 Εξυπηρετητής 106
4.3 Η συσκευή δικτυακής αποθήκευσης gmblock 110
4.3.1 Εξυπηρετητής 110

5 Υλοποίηση **115**
5.1 Εισαγωγή 115
5.1.1 Το module - πελάτης 117
5.1.2 Ο εξυπηρετητής 117
5.1.3 Βασικές Δομές Επικοινωνίας 119
5.2 Ο πελάτης 121
5.2.1 Αρχικοποίηση και Καταχώρηση 124
5.2.2 Χειρισμός Αιτήσεων 125
5.3 Ο Εξυπηρετητής 128
5.4 Υλοποίηση του βελτιστοποιημένου μονοπατιού 131
5.4.1 Αλλαγές στο GM 131
5.4.2 Αλλαγές στον πυρήνα 133

6 Επίλογος **137**
6.1 Σύνοψη 137
6.2 Επεκτάσεις 138

Κατάλογος Σχημάτων

2.1 Μοντέλο OSI 40
2.2 Μια τυπική τοπολογία σε δίκτυο Myrinet 56
2.3 Δομή πακέτου Myrinet 59
2.4 Η κάρτα δικτύου Myrinet 60
2.5 Μια αποστολή σε δίκτυο Myrinet με χρήση του GM 74

3.1 Μία όψη του πυρήνα σε υποσυστήματα 79

4.1 Μια τυπική δικτυακή συσκευή block 100
4.2 Η αρχιτεκτονική του Πελάτη 101
4.3 Ο Πελάτης 102
4.4 Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής nbd (λογικό επίπεδο) 104
4.5 Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής nbd (φυσικό επίπεδο) 105
4.6 Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής nbd πάνω από Myrinet (λογικό επίπεδο) 108
4.7 Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής nbd πάνω από Myrinet (φυσικό επίπεδο) 109
4.8 Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής gmblock (λογικό επίπεδο) 111
4.9 Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής gmblock (φυσικό επίπεδο) 112

5.1 Το μοντέλο που υλοποιήθηκε 116
5.2 Στρώματα χειρισμού της συσκευής gmblock 118
5.3 Η αίτηση gmblock 119
5.4 Η δομή gm_con_t 120
5.5 Οι δομές δεδομένων σύνδεσης και μηνύματος 123
5.6 Εκκίνηση της διεργασίας του module 124
5.7 Αρχικοποίηση της σύνδεσης GM 125

5.8 Αρχικοποίηση και καταχώρηση της συσκευής gmblock 125
5.9 Μετάφραση μιας αίτησης block σε gmblock 127
5.10 Η συνάρτηση χειρισμού μιας ληφθείσας αίτησης της συσκευής gmblock 128
5.11 Ο εξυπηρετητής του οδηγού συσκευής gmblock 130
5.12 Η κάρτα δικτύου Myrinet (με τα DMA engines) 132
5.13 Η απεικόνιση της στατικής μνήμης του Lanai 133
5.14 Ανάγνωση με Ο DIRECT 135
5.15 Οι κλήσεις συστήματος για ανάγνωση με Ο DIRECT 136

Εισαγωγή

Στην παρούσα εργασία παρουσιάζεται ο σχεδιασμός και η υλοποίηση ενός μοντέλου μεταφοράς δεδομένων από συσκευές αποθήκευσης σε δίκτυο Myrinet χωρίς τη μεσολάβηση της ιεραρχίας μνήμης και χωρίς χρήση του επεξεργαστή του συστήματος εξυπηρέτησης.

Στόχος είναι η αποδέσμευση του επεξεργαστή και της κεντρικής μνήμης από τη διαδικασία αντιγραφής των δεδομένων από το μέσο αποθήκευσης (π.χ. σκληρός δίσκος) στην κάρτα δικτύου Myrinet.

Στο Κεφάλαιο 1 παρουσιάζονται συνοπτικά τα συστήματα αποθήκευσης που κυριαρχούν σήμερα. Αναλύονται οι τρόποι πρόσβασης στα μέσα αποθήκευσης (μέθοδοι Εισόδου / Εξόδου) και αναφέρονται συνοπτικά οι εναλλακτικές επιλογές δικτυακών συσκευών αποθήκευσης.

Το Κεφάλαιο 2 περιγράφει τα δίκτυα διασύνδεσης που χρησιμοποιούνται κυρίως για συσκευές δικτυακής αποθήκευσης και αναλύει τα πρότυπα Ethernet, TCP/IP. Δίνεται έμφαση στο δίκτυο Myrinet, λόγω της υψηλής ταχύτητας που διαθέτει το φυσικό στρώμα καθώς και της δομής του αφού πρόκειται για μοντέλο δίκτυου επικοινωνίας χώρου χρήστη (userspace network model)

Στο Κεφάλαιο 3 αναλύεται η δομή του πυρήνα στο λειτουργικό σύστημα Linux, όσον αφορά στα συστήματα αποθήκευσης και στους τρόπους πρόσβασης σε συσκευές Εισόδου / Εξόδου όπως τα μέσα αποθήκευσης και οι συσκευές διεπαφής δικτύου.

Ο σχεδιασμός του μοντέλου μεταφοράς δεδομένων της εργασίας εκτίθεται στο Κεφάλαιο 4. Αρχικά παρουσιάζεται μια συσκευή δικτυακής αποθήκευσης για συμβατικά δίκτυα (TCP/IP), στη συνέχεια για δίκτυο Myrinet όπου επιβάλλεται μια βελτιστοποίηση με παράκαμψη της page cache και τέλος δίνεται το μοντέλο της συσκευής που υλοποιήθηκε. Αναλύονται τα μονοπάτια των δεδομένων σχηματικά για όλες τις περιπτώσεις συσκευών τόσο σε φυσικό όσο και σε λογικό επίπεδο και δίνεται έμφαση στο βελτιστοποιημένο μονοπάτι.

Η υλοποίηση του μοντέλου βελτιστοποιημένης μεταφοράς δεδομένων παρουσιάζεται στο Κεφάλαιο 5. Αναλύονται διεξοδικά τα βασικά σημεία της υλοποίησης και περιγράφεται σχηματικά ο τρόπος με τον οποίο δομήθηκε η υλοποίηση. Δεν κρίθηκε αναγκαία η ενσωμάτωση του κώδικα της υλοποίησης

στο συγκεκριμένο κεφάλαιο.

Τέλος στο Κεφάλαιο 6 δίνεται μια σύνοψη των προηγούμενων κεφαλαίων και πιθανές μελλοντικές επεκτάσεις του συγκεκριμένου μοντέλου.

Κεφάλαιο 1

Συστήματα Αποθήκευσης

Εισαγωγή

Τα συστήματα Εισόδου Εξόδου αποτελούσαν ένα ιστορικά παραμελημένο κομμάτι της αρχιτεκτονικής υπολογιστών. Είναι φανερό πως οι σχεδιαστές επικεντρώνονταν περισσότερο στην ανάπτυξη και εξέλιξη των επεξεργαστών παρά στην βελτιστοποίηση του μηχανισμού εισόδου / εξόδου. Αυτό γίνεται περισσότερο προφανές αν λάβουμε υπόψη πως σαν κοινή αποδεκτή μονάδα μέτρησης της απόδοσης ενός υπολογιστικού συστήματος, θεωρείται μέχρι σήμερα ο χρόνος που ένα πρόγραμμα εκτελείται αμιγώς στον επεξεργαστή, χωρίς να υπολογίζεται ο χρόνος που καταναλώνεται στις μονάδες εισόδου / εξόδου προκειμένου το υπό επεξεργασία πρόγραμμα να διαβάσει τα δεδομένα ή να αποθηκεύσει προσωρινά ή και τελικά αποτελέσματα.

Η παραπάνω άποψη είναι αντιφατική από την ίδια της τη βάση. Ένα υπολογιστικό σύστημα χωρίς μονάδες εισόδου / εξόδου είναι αποκομμένο από κάθε είδους επικοινωνία (με το "περιβάλλον"). Μια πιο αντικειμενική μονάδα μέτρησης της απόδοσης ολόκληρου του συστήματος, είναι ο χρόνος απόκρισης, ο χρόνος δηλαδή που μεσολαβεί από την ανάκτηση των δεδομένων μέχρι την έξοδο των αποτελεσμάτων σε κάποια προσωρινή ή μόνιμη μνήμη.

Σύμφωνα με το νόμο του Amdahl[1], δεν αρκεί να βελτιώνεται συνέχεια η επεξεργαστική δύναμη των υπολογιστικών συστημάτων, πράγμα που εξακολουθεί να γίνεται μέχρι σήμερα, με ρυθμό διπλασιασμού ανά ενάμιση περίπου χρόνο. Ζούμε στην εποχή του αμέτρητου όγκου πληροφοριών, και η ανάγκη γρηγορότερης ανάκτησης και αποθήκευσης είναι επιβεβλημένη. Στον τομέα της επεξεργαστικής ισχύος, ο κορεσμός είναι εμφανής και οι σχεδιαστές αναγκάζο-

[1]Ο Amdahl διαχώρισε τη μεταβολή της απόδοσης μιας μηχανής μέσω μιας βελτίωσης, περιορίζοντας τη βελτίωση στο χρόνο που αυτή έχει επιρροή πάνω στη μηχανή. Δηλαδή αν έχουμε μια μηχανή που εκτελεί δεδομένο πρόγραμμα σε 10 μονάδες χρόνου, και επιβάλλουμε μια βελτίωση στις 3 πρώτες μονάδες χρόνου, ο χρόνος θα μειωθεί κατά 1 μονάδα, χωρίς να επηρεάσει τις υπόλοιπες 7 μονάδες χρόνου.

νται να ενσωματώσουν μελέτες για καλύτερη πρόσβαση σε συσκευές Εισόδου / Εξόδου, ώστε να βελτιώσουν συνολικά την απόδοση των υπολογιστικών συστημάτων είτε σε επίπεδο χρήστη είτε σε απαιτητικές εφαρμογές υπολογισμών μεγάλου όγκου δεδομένων σε επίπεδο επαγγελματικής χρήσης[HP03].

1.1 Συσκευές Αποθήκευσης

Ο όρος συσκευή αποθήκευσης αναφέρεται κυρίως σε σταθερή (non-volatile) αποθήκευση δηλαδή σε συσκευές, όπου τα δεδομένα θα παραμείνουν, όταν αυτές δε θα βρίσκονται σε παροχή ρεύματος. Η απλούστερη κατηγοριοποίηση που μπορεί να γίνει σε τέτοιου τύπου συσκευές, είναι ανάλογα με την τεχνική αποθήκευσης που χρησιμοποιούν καθώς και με την ευκολία ανάκτησης και αποθήκευσης δεδομένων.

Οι συσκευές που χρησιμοποιούνται ευρέως είναι οι μαγνητικές (σκληρός δίσκος, μαλακές δισκέτες, αφαιρούμενες μαγνητικές κασέτες), οι οπτικές (CD/DVD επανεγγράψιμα ή μη) και οι συσκευές σταθερής κατάστασης ημιαγώγιμης μνήμης, γνωστές ως Flash memory / memory card devices. Στην παρούσα ενότητα θα αναφερθούμε, όσο αναλυτικά το επιτρέπει ο σκοπός της εργασίας, στη λειτουργία του σκληρού δίσκου - επεκτάσεων αυτού, χρησιμοποιώντας πολλούς συνδεδεμένους σε συστοιχία.

1.1.1 Σκληρός Δίσκος

Ο σκληρός δίσκος αποτελείται κυρίως από λεπτούς δίσκους, γνωστούς ως platters (συνήθως από 1 έως 12) που γυρνούν γύρω από έναν άξονα με ταχύτητα από 3600 έως 15000 στροφές ανά λεπτό (Revolutions Per Minute). Είναι φτιαγμένοι από ένα μη - μαγνητικό μέταλλο ή από γυαλί και επικαλύπτονται και στις δύο όψεις από ένα λεπτό στρώμα μαγνητικής μεμβράνης. Η επιφάνεια των λεπτών δίσκων χωρίζεται σε ομόκεντρους κύκλους, που ονομάζονται ίχνη (tracks). Συνήθως υπάρχουν 5000 - 50000 ίχνη σε κάθε επιφάνεια. Κάθε ίχνος με τη σειρά του χωρίζεται σε τομείς (sectors) και είναι η μικρότερη δυνατή μονάδα προσπέλασης του δίσκου (512 bytes δεδομένων είναι το πιο συνηθισμένο μέγεθος τομέα). Παλαιότερα, όλα τα ίχνη είχαν τον ίδιο αριθμό τομέων. Σήμερα, αυτός ο αριθμός είναι μεταβλητός αφού τα εξωτερικά ίχνη είναι μεγαλύτερα απ' ότι τα εσωτερικά.

Για την ανάγνωση και την εγγραφή δεδομένων πάνω σε ένα δίσκο, ένας κινούμενος βραχίονας με μια κεφαλή ανάγνωσης / εγγραφής (read/write head) βρίσκεται πάνω από κάθε επιφάνεια. Όλοι οι βραχίονες είναι ενωμένοι σταθερά μεταξύ τους και σε μια δεδομένη στιγμή, όλοι οι βραχίονες "τέμνουν" κατά κάποιο τρόπο κάθε μια από τις επιφάνειες σε ένα συγκεκριμένο ίχνος. Το

σύνολο των ιχνών που βρίσκονται κάτω από τις κεφαλές σε μια δεδομένη στιγμή, ορίζεται ως κύλινδρος.

Για να γίνει προσπέλαση ενός τομέα, ο ελεγκτής δίσκου, μια συσκευή που διαχειρίζεται το (μηχανικό) σύστημα βραχιόνων - λεπτών δίσκων - κρυφής μνήμης προσωρινής αποθήκευσης, αποστέλλει μια εντολή στο βραχίονα να κινηθεί προς το συγκεκριμένο ίχνος. Αυτή η λειτουργία ονομάζεται seek. Όταν βρεθεί ο τομέας κάτω από το βραχίονα, τότε γίνεται η προσπέλαση που ζητήθηκε. Τα δεδομένα (αφού διαβαστούν από την κεφαλή ανάγνωσης / εγγραφής) μεταφέρονται σε προσωρινές (κρυφές) μνήμες (disk cache memory) για να ακολουθήσουν το μονοπάτι προς την κεντρική μνήμη, όπου πρέπει να περάσουν από ελεγκτές διαδρόμου, άλλες κρυφές μνήμες ή και μονοπάτια διαδρόμου (θα αναλυθεί στην επόμενη ενότητα).

1.1.2 Συστοιχία Σκληρών Δίσκων (RAID)

Η συστοιχία σκληρών δίσκων (Redundant Array of Independent / Inexpensive Disks) είναι ένα σύστημα που χρησιμοποιεί σκληρούς δίσκους για να μοιράσει δεδομένα καθώς και πλεονάζουσα πληροφορία σ' αυτούς. Ανάλογα με την "έκδοση" ή το επίπεδο που χρησιμοποιείται, τα πλεονεκτήματα του RAID, συγκριτικά με τους απλούς σκληρούς δίσκους, είναι η ασφάλεια των δεδομένων, η προστασία από λάθη, η συνολική απόδοση ταχύτητας ανάκτησης / εγγραφής (προσπέλασης γενικότερα) ή και ο αποθηκευτικός χώρος.

Στην αρχική του υλοποίηση, το βασικό του πλεονέκτημα ήταν η χρήση απλών, φτηνών δίσκων με παλαιότερη τεχνολογία, σε μια συστοιχία που προσέφερε περισσότερο αποθηκευτικό χώρο, αξιοπιστία, ταχύτητα ή και συνδυασμό αυτών, σε τιμή πολύ χαμηλότερη απ' ό,τι μία συσκευή που θα ενσωμάτωνε όλα αυτά τα χαρακτηριστικά.

Στο πιο απλό του επίπεδο, το RAID ενσωματώνει πολλούς δίσκους σε μία και μόνο λογική μονάδα (logical unit). Έτσι, αντί να εμφανίζονται διαφορετικοί δίσκοι, το λειτουργικό σύστημα αναγνωρίζει μόνο έναν. Συνήθως, το RAID εμφανίζεται σε συστήματα εξυπηρετητών όπου οι δίσκοι είναι πανομοιότυποι και ίδιου μεγέθους. Πρόσφατα, με τις χαμηλές τιμές που εμφανίζουν οι σκληροί δίσκοι και την ενσωμάτωση κάποιων επιλογών RAID στα ολοκληρωμένα κυκλώματα των υπολογιστικών συστημάτων, εμφανίζεται η μέθοδος RAID σε προσωπικούς υπολογιστές αυξημένων απαιτήσεων, όπως για παράδειγμα υπολογιστές για επεξεργασία ήχου ή κινούμενης εικόνας.

Ο ορισμός του RAID εισήγαγε έναν αριθμό από πρωτότυπα "επίπεδα" ή αλλιώς συνδυασμούς δίσκων. Το καθένα από αυτά είχε πλεονεκτήματα και μειονεκτήματα. Με την πάροδο των χρόνων, εμφανίστηκαν διαφορετικές υλοποιήσεις της λογικής RAID. Και βέβαια, υπάρχουν ακόμα περιπτώσεις, όπου η υλοποίηση ενός επιπέδου δεν έχει καμία σχέση με τη λογική του RAID.

Παρόλα αυτά, η ιστορία επιβάλλει πως οποιοσδήποτε συνδυασμός δίσκων γίνεται με σκοπό την αξιοπιστία, την αύξηση του αποθηκευτικού χώρου ή της απόδοσης ανήκει κατά κάποιο τρόπο στο επίπεδο του RAID.

Η συστοιχία σκληρών δίσκων μπορεί να υλοποιηθεί είτε με συγκεκριμένο υλικό είτε με λογισμικό που θα εκτελείται σε συμβατικό υλικό. Επιπρόσθετα, υπάρχουν υβριδικές υλοποιήσεις RAID με λογισμικό και συγκεκριμένο υλικό μαζί. Είναι προφανές ότι το RAID με υλικό είναι πολύ γρηγορότερο, αφού στην αντίθετη περίπτωση θα πρέπει να προστεθεί ακόμα ένα ″επίπεδο″, αυτό της επικοινωνίας λογισμικού και ελεγκτών δίσκων. Στις μέρες μας, υπάρχουν μητρικές πλακέτες για προσωπικούς υπολογιστές με χαμηλό κόστος που περιέχουν ελεγκτές RAID και έτσι οι συστοιχίες σε επίπεδο λογισμικού δε θα παραμείνουν για πολύ ακόμα.

Πριν προχωρήσουμε στην επεξήγηση του κάθε επιπέδου, καλό είναι να αναφέρουμε τον τρόπο με τον οποίο το σύστημα RAID ελέγχει και διορθώνει τυχόν λάθη που δημιουργούνται είτε λόγω του λειτουργικού συστήματος (λάθη λογισμικού δηλαδή) είτε λόγω σφάλματος στο υλικό του σκληρού δίσκου

1.1.2.1 Έλεγχοι λαθών

Ο Έλεγχος λαθών στο RAID, γίνεται κυρίως με block ισοτιμίας (parity blocks). Αν κάποιος από τους δίσκους τεθεί εκτός λειτουργίας και τα δεδομένα δεν μπορούν να προσπελαστούν, ανακτώνται με το συνδυασμό των δεδομένων των δίσκων σε λειτουργία και των block ισοτιμίας.

Τα block ισοτιμίας αποθηκεύονται διαφορετικά σε κάθε επίπεδο, υπολογίζονται όμως με τον ίδιο ακριβώς τρόπο (για τα 3-4-5-6): κάνοντας XOR σε bits, bytes ή blocks.

$$parity_i = d_{i,j} \oplus d_{i,j+1} \oplus ... \oplus d_{i,j+n}.$$

όπου n ο αριθμός των δίσκων.

Ο τρόπος να αναπαραχθεί κάποιο χαμένο δεδομένο είναι να γίνει xor του $parity_i$ με καθένα από τα υπόλοιπα δεδομένα για τη i ″γραμμή″.

Οπότε αν π.χ. χαθεί ο l δίσκος, θα προκύψει από το παρακάτω xor:

$$d_{i,j+l} = d_{i,j} \oplus d_{i,j+1} \oplus ... \oplus d_{i,j+l-1} \oplus d_{i,j+l+1} \oplus ... \oplus d_{i,j+n}.$$

Τα επίπεδα 1 και 2 δε χρησιμοποιούν αυτόν τον τρόπο, ενώ το επίπεδο 0 δε χρησιμοποιεί καθόλου πλεονάζουσα πληροφορία.

RAID 0

Στο συγκεκριμένο επίπεδο, τα δεδομένα μοιράζονται εξίσου σε δύο ή περισσότερους δίσκους χωρίς κάποια επιπλέον πληροφορία. Πρέπει να αναφερθεί

ότι το RAID δεν ήταν μέσα στα ορισμένα από την αρχή "επίπεδα". Συνήθως χρησιμοποιείται, για να αυξήσει την απόδοση ή και για να δημιουργήσει μικρό αριθμό εικονικών δίσκων από ένα μεγάλο αριθμό φυσικών δίσκων. Το μέγεθος του εικονικού (ή των εικονικών) δίσκων που προκύπτει είναι πολλαπλάσιο των δίσκων επί το μέγεθος του μικρότερου δίσκου από αυτούς. Έτσι για παράδειγμα δεν έχει νόημα να χρησιμοποιήσουμε 3 δίσκους σε RAID 0 που θα έχουν μέγεθος 120GB, 140GB, 150GB γιατί το αποτέλεσμα θα είναι ένας εικονικός δίσκος των 360GB.

Η χρήση παραπάνω από δύο δίσκους είναι δυνατή σε αυτό το επίπεδο, όμως καλό είναι να αποφεύγεται, αφού μειώνεται η αξιοπιστία. Ο λόγος είναι, γιατί το σύστημα αρχείων διαμοιράζεται σε όλους τους δίσκους ισάξια, οπότε, όταν ένας από τους δίσκους χαθεί, το σύστημα δεν είναι σε θέση να αντεπεξέλθει σε μια τέτοια απώλεια δεδομένων και το πιθανότερο είναι ότι θα καταστεί αδύνατο να ανακτηθούν τα δεδομένα. Συγκεκριμένα μπορούμε να αναφέρουμε ότι η πιθανότητα να καταστραφεί το σύστημα αρχείων, λόγω σφάλματος υλικού, ενός συστήματος με RAID 0 ισούται με την πιθανότητα να δημιουργηθεί σφάλμα σε ένα δίσκο επί τον αριθμό των δίσκων.

Το μέγεθος ενός block που μοιράζεται μεταξύ των δίσκων είναι πολλαπλάσιο του 512 (του μεγέθους ενός τομέα του σκληρού δίσκου). Αυτό γίνεται, για να μπορούν οι δίσκοι να κάνουν seek ταυτόχρονα και έτσι να αυξάνεται η ταχύτητα αναζήτησης. Η ταχύτητα που μπορεί να επιτευχθεί με το RAID 0 είναι το άθροισμα των ταχυτήτων όλων των δίσκων με μόνο περιορισμό την ταχύτητα του ελεγκτή RAID.

Το RAID 0 χρησιμοποιείται σε περιπτώσεις, όπου η απώλεια δεδομένων είναι σχεδόν ασήμαντη και όπου η ταχύτητα ανάγνωσης είναι κατά πολύ σημαντικότερη από την ταχύτητα εγγραφής. Παραδείγματα για την πρώτη περίπτωση είναι σε συστήματα με παιχνίδια, ή εφαρμογές χωρίς τα δεδομένα ή τα αποτελέσματά τους. Παράδειγμα για τη δεύτερη θα μπορούσε να είναι ένας εξυπηρετητής αρχείων NFS (θα εξηγηθεί στην τελευταία ενότητα), που επιτρέπει μόνο την ανάγνωση και μεγάλο όγκο πληροφοριών.

RAID 1

Στο επίπεδο αυτό, δημιουργείται ένα ακριβές αντίγραφο του πρώτου δίσκου στους υπόλοιπους. Αυτό το επίπεδο είναι χρήσιμο όταν η ταχύτητα / απόδοση ανάγνωσης είναι σημαντικότερη από το μέγεθος του αποθηκευτικού χώρου. Το μέγεθος μιας τέτοιας συστοιχίας μπορεί να είναι ίσο με το μέγεθος του μικρότερου δίσκου. Σ' ένα σύστημα RAID 1 με δύο δίσκους, η πιθανότητα σφάλματος μειώνεται εκθετικά γιατί για ενδεχόμενη απώλεια δεδομένων απαιτείται σφάλμα υλικού και στους δύο δίσκους, που είναι το τετράγωνο της πιθανότητας απώλειας του ενός (υποθέτουμε ότι είναι πανομοιότυποι δίσκοι).

Επίσης, η ταχύτητα ανάγνωσης αυξάνεται γραμμικά με τον αριθμό των δίσκων αφού ο καθένας έχει δικό του ελεγκτή. Έτσι, όταν ζητείται ανάγνωση, μπορεί να γίνει ταυτόχρονα και στους δύο δίσκους.

RAID 2

Το επίπεδο αυτό, διαμοιράζει τα δεδομένα σε επίπεδο bit και χρησιμοποιεί έλεγχο λαθών με τον κώδικα hamming. Οι δίσκοι συγχρονίζονται πλήρως από τον ελεγκτή RAID και μπορούν να αποδώσουν πολύ υψηλές ταχύτητες μεταφοράς δεδομένων. Το επίπεδο αυτό είναι το μόνο που δε χρησιμοποιείται καθόλου.

RAID 3

Στο RAID 3, τα δεδομένα μοιράζονται σε επίπεδο byte με ένα δίσκο αποκλειστικά για έλεγχο ισοτιμίας. Σπάνια απαντάται στην πράξη. ορισμένα από τα μειονεκτήματά του είναι ότι δεν μπορεί να ικανοποιήσει πολλές αιτήσεις για δεδομένα ταυτόχρονα. Αυτό συμβαίνει, γιατί τα δεδομένα μοιράζονται σε όλους τους δίσκους σε επίπεδο byte, οπότε ένα block βρίσκεται διασκορπισμένο σε όλους τους δίσκους και παραμένει εκεί. Έτσι, για να προσπελαστεί το συγκεκριμένο κομμάτι, θα πρέπει να προσπελαστούν όλοι οι δίσκοι.

RAID 4

Το RAID 4, είναι παρόμοιο με το 3 με τη διαφορά ότι τα δεδομένα μοιράζονται σε επίπεδο block, οπότε και δεν περιέχει το μειονέκτημα του 4.

RAID 5

Το RAID 5, θα μπορούσε κανείς να πει ότι είναι βελτίωση των RAID 3,4. Χρησιμοποιεί μοίρασμα δεδομένων σε επίπεδο block και block ισοτιμίας μοιρασμένα επίσης κατά μήκος των δίσκων. Συνήθως υλοποιείται με υλικό που υποστηρίζει πράξεις ισοτιμίας.

Κάθε φορά που γράφεται ένα block σε ένα σύστημα με RAID 5, ένα block ισοτιμίας γράφεται στην ίδια ″λωρίδα″ (stripe). Τα block ισοτιμίας δε διαβάζονται, παρά μόνο όταν συμβεί ένα σφάλμα και δεν μπορεί να διαβαστεί ένα block από την ίδια λωρίδα. Τότε με XOR (όπως δείξαμε παραπάνω) παράγεται το ζητούμενο block και έτσι η αξιοπιστία του συστήματος είναι φανερά μεγάλη. Επίσης, άλλη μια απόδειξη για την αυξημένη αξιοπιστία είναι ότι, αν χαθεί ένας από τους δίσκους της συστοιχίας, δε χάνεται κανένα block, αφού όλα υπάρχουν στα blocks ισοτιμίας των άλλων. Όσον αφορά δε

στα blocks ισοτιμίας που χάθηκαν, υπάρχουν τα αυθεντικά δεδομένα και πάλι μπορούν να δημιουργηθούν.

΄Ενα σημαντικό μειονέκτημα είναι η περίπτωση να δημιουργηθούν σφάλματα σε δύο δίσκους, οπότε και τα δεδομένα χάνονται ολοκληρωτικά. Αυτή η περίπτωση αντιμετωπίζεται με το επίπεδο 6 που έχει δύο (ή και περισσότερα) block ισοτιμίας ανά λωρίδα. Η ανάλυση του RAID 6 δεν αφορά το στόχο της παρούσης εργασίας.

Εμφωλευμένα επίπεδα RAID

Υπάρχει πιθανότητα να χρησιμοποιηθούν επίπεδα RAID μέσα σε ήδη υπάρχοντα συστήματα RAID. Παραδείγματα αυτών είναι τα:

- RAID 0+1 που είναι ένταξη δύο συστοιχιών δίσκων με RAID 0 σε μια συστοιχία με RAID 1. ΄Ετσι μπορούμε να έχουμε τον αποθηκευτικό χώρο του RAID 0 σε συνδυασμό με την αξιοπιστία του RAID 1.
- RAID 10 που είναι ακριβώς το αντίστροφο του παραπάνω.
- κλπ.

1.2 Είσοδος / ΄Εξοδος

Σ' ένα υπολογιστικό σύστημα, τα διάφορα υποσυστήματα πρέπει να διαθέτουν διεπαφές έτσι, ώστε να επικοινωνούν μεταξύ τους. Για παράδειγμα, Η Κεντρική Μονάδα Επεξεργασίας, πρέπει να επικοινωνεί με την Κεντρική Μνήμη, ή με τις μονάδες Εισόδου Εξόδου. Στις μέρες μας, αυτή η επικοινωνία επιτυγχάνεται κυρίως μέσω της αρχιτεκτονικής *διαδρόμων*. Ο διάδρομος (bus) δρα ως ένας διαμοιραζόμενος σύνδεσμος επικοινωνίας μεταξύ των υποσυστημάτων.

Το σύστημα διαδρόμων χρησιμοποιείται κατά κόρον σήμερα λόγω του χαμηλού κόστους κατασκευής του και της πολλαπλής χρησιμότητάς του -- ένα μοναδικό σύνολο καλωδίων μοιράζεται μεταξύ πολλών συσκευών. Ο προβληματισμός των σχεδιαστών διαδρόμων βέβαια, έγκειται στην ενδεχόμενη συμφόρηση (bottleneck) που μπορεί να δημιουργηθεί και να περιορίσει σημαντικά τη συνολική απόδοση του συστήματος. Ο κυριότερος λόγος που δημιουργείται αυτή η συμφόρηση είναι οι κάθε είδους συσκευές που συνδέονται στον ίδιο διάδρομο, με αποτέλεσμα πολλές φορές η ταχύτητά ροής των δεδομένων σ' αυτόν να περιορίζεται στην ταχύτητα της πιο αργής συνδεδεμένης συσκευής.

Γι αυτό το λόγο οι διάδρομοι χωρίζονται σε δύο βασικές κατηγορίες: τους διαδρόμους Κεντρικής Μονάδας Επεξεργασίας -- Κεντρικής Μνήμης (CPU - Memory bus), και στους διαδρόμους Εισόδου / Εξόδου (I/O bus). Η δεύτερη κατηγορία μπορεί να διαχωριστεί ανάλογα με τον τύπο και τη χρησιμότητα της συσκευής. Πολλές φορές, βέβαια, για να ελαχιστοποιηθεί το κόστος, οι σχεδιαστές προσπαθούν να συνδυάσουν τα βασικά χαρακτηριστικά των υποσυστημάτων και να κατασκευάσουν έναν κοινό διάδρομο για τον Επεξεργαστή, τη Μνήμη και τις συσκευές Εισόδου / Εξόδου ονομάζοντάς τον διάδρομο "ημιορόφου" (mezzazine).

Η επικοινωνία των υποσυστημάτων μέσω του διαδρόμου αναφέρεται σε δεδομένα, διευθύνσεις, εντολές, στον καθορισμό του ποια συσκευή θα διαχειριστεί τις παραπάνω πληροφορίες, καθώς και στο αν η επικοινωνία θα γίνει σύγχρονα ή ασύγχρονα. Έτσι προκύπτει άλλη μια παράμετρος διαχωρισμού των διαδρόμων σε σύγχρονους και μη (synchronoys / asynchronoys bus).

Λόγω της ποικιλίας συσκευών Εισόδου / Εξόδου οι αντίστοιχοι διάδρομοι μπορεί να διαφέρουν στα πρωτόκολλα επικοινωνίας τους. Για να αποφευχθεί μια ενδεχόμενη συμφόρηση στο διάδρομο με τις διάφορες συσκευές, οι σχεδιαστές προσπάθησαν να κατασκευάσουν διαδρόμους επικοινωνίας με την Κεντρική Μνήμη, προσανατολισμένους σε συγκεκριμένου τύπου συσκευές. Για παράδειγμα, οι προσαρμογείς οθόνης (γνωστές ως κάρτες γραφικών) ενώ παλαιότερα χρησιμοποιούσαν τον κοινό διάδρομο για συμβατικές συσκευές (Peripheral Component Interconnect) είχαν μέχρι πρόσφατα διαχωριστεί από τις υπόλοιπες, με το διάδρομο AGP (Accelerated Graphics Peripheral). Σήμερα, γίνεται και πάλι μια προσπάθεια ενοποίησης με την εισαγωγή του πρωτοκόλλου PCI-X (PCI - eXpress).

Οι διάδρομοι διαφορετικών πρωτοκόλλων συνδέονται μεταξύ τους με συσκευές που γνωρίζουν τα πρωτόκολλα επικοινωνίας που αυτοί χρησιμοποιούν. Αυτές οι συσκευές ονομάζονται προσαρμογείς διαδρόμων (bus adapters).

Συνοψίζοντας, σ' ένα κλασικό υπολογιστικό σύστημα σήμερα, μπορούμε να διακρίνουμε τριών ειδών διαδρόμους:

- *Διάδρομοι Επεξεργαστικών Μονάδων - Κεντρικής Μνήμης*
  Ο διάδρομος αυτός συνήθως είναι "μικρός" (δηλαδή αποτελείται από λίγες συσκευές), αρκετά γρήγορος και σχεδιασμένος για συγκεκριμένου τύπου επεξεργαστικές μονάδες και μονάδες μνήμης. Ο σχεδιαστής αυτού του διαδρόμου, γνωρίζει από πριν ποιες συσκευές θα επικοινωνούν μεταξύ τους μέσω αυτού Επίσης, σε αυτό το διάδρομο συνδέονται και οι υπόλοιποι (με βάση μια ιεραρχία), ώστε οι περιφερειακές συσκευές να επικοινωνούν με την κεντρική μνήμη και τις επεξεργαστικές μονάδες.

- *Διάδρομοι Περιφερειακών Συσκευών*

Αυτός ο διάδρομος χρησιμοποιείται για να συνδέσει τους επιμέρους διαδρόμους των περιφερειακών συσκευών με το διάδρομο Επεξεργαστικών Μονάδων - Κεντρικής Μνήμης. Μπορεί να περιέχει συσκευές που δεν είναι γνωστές στο σχεδιαστή του και γι' αυτό είναι επιβεβλημένο ένα γενικό πρωτόκολλο διαδρόμου (όπως π.χ. το PCI).

- *Διάδρομοι Εισόδου Εξόδου*
  Σ' αυτόν το διάδρομο συνδέονται συσκευές ίδιου τύπου με παρόμοιες απαιτήσεις και δυνατότητες σε εύρος επικοινωνίας δεδομένων. Συσκευές αποθήκευσης ίδιας αρχιτεκτονικής συνδέονται μέσω αυτού του διαδρόμου, όπως, για παράδειγμα, IDE (Integrated Drive Electronics) και ο απόγονός του, το ATA (AT-bus Attachment) και όπως SCSI (Small Computer System Interconnect).

Μπορούμε τώρα να αναφερθούμε συνοπτικά στον τρόπο με τον οποίο επιτυγχάνεται η μεταφορά δεδομένων από την Κεντρική Μνήμη στις συσκευές Ε/Ε, δηλαδή στα πιο συνηθισμένα πρωτόκολλα διαδρόμων για συσκευές Ε/Ε.

1.2.1 Διεπαφή συσκευών Ε/Ε με τις Επεξεργαστικές μονάδες

Η PIO (Προγραμματιζόμενη Ε/Ε) είναι ίσως μια από τις πρώτες προσπάθειες ανάπτυξης ενός πρωτοκόλλου επικοινωνίας μεταξύ των διαδρόμων Ε/Ε και των Επεξεργαστικών μονάδων. Ουσιαστικά ο διάδρομος Ε/Ε σε αυτή την περίπτωση συνδέεται με την Κεντρική (ή την κρυφή / γρήγορη) Μνήμη.

Σύμφωνα με τη συνηθισμένη πρακτική, που τμήματά της έχουν υιοθετηθεί μέχρι και σήμερα, κομμάτι του χώρου διευθύνσεων της μνήμης αντιστοιχίζεται με μια συσκευή Ε/Ε. Ανάγνωση και εγγραφή σ' αυτό το κομμάτι της μνήμης προκαλεί μεταφορά δεδομένων, καθώς και τον έλεγχο της συσκευής· επομένως, πρόσβαση και προσπέλαση της μνήμης στο συγκεκριμένο κομμάτι της, αντιστοιχίζεται σε πρόσβαση στη συσκευή. Αυτή η μέθοδος αποκαλείται memory mapped I/O.

Η εναλλακτική αυτής της μεθόδου είναι η χρησιμοποίηση εντολών Ε/Ε από τις ίδιες τις Επεξεργαστικές Μονάδες για μεταφορά δεδομένων ή έλεγχο της συσκευής. Αυτές οι πρακτικές βέβαια είναι απαιτητικές σε επεξεργαστική ισχύ, όχι τόσο σε ποσότητα, αλλά σε χρόνο. Έτσι σε περίπτωση που χρησιμοποιούνται μεγάλες μεταφορές δεδομένων (όπως για παράδειγμα σε συσκευές αποθήκευσης) η επεξεργαστική ισχύς που απαιτείται είναι απαγορευτικός αριθμός για οποιαδήποτε εκτέλεση άλλης λειτουργίας.

Για την αντιμετώπιση του παραπάνω προβλήματος που παρουσιάζει η PIO, αναπτύχθηκε μια τεχνική που επιτυγχάνει "Απευθείας Πρόσβαση στη Μνήμη"

(Direct Memory Access).

1.2.2 Απευθείας Πρόσβαση στη Μνήμη (Direct Memory Access)

Η Απευθείας Προσπέλαση Μνήμης (Direct Memory Access ή DMA) επιτρέπει κάποιο υποσύστημα μέσα στον υπολογιστή να προσπελάσει την Κύρια Μνήμη για ανάγνωση ή εγγραφή, ανεξάρτητα από τις Επεξεργαστικές Μονάδες. Πολλές συσκευές χρησιμοποιούν αυτή τη μέθοδο, όπως συσκευές αποθήκευσης, προσαρμογείς οθόνης, δικτύου ή ήχου. Τα συστήματα που διαθέτουν ″κανάλια″ DMA μπορούν να μεταφέρουν δεδομένα γρηγορότερα απ' ό,τι αυτά που δεν έχουν. Χρησιμεύουν για αντίγραφα ασφαλείας και εφαρμογές πραγματικού χρόνου (real-time applications).

Η DMA, είναι ένα βασικό χαρακτηριστικό των σημερινών υπολογιστικών συστημάτων, αφού επιτρέπει μεταφορά δεδομένων, χωρίς να επιβάλει φόρτο στην επεξεργαστική μονάδα. Αν δεν υπήρχε, η επεξεργαστική μονάδα θα ευθυνόταν αποκλειστικά για την αντιγραφή κάθε block δεδομένων από την πηγή στον προορισμό. Είναι προφανές ότι αυτή η αντιγραφή είναι πιο αργή, αφού η πρόσβαση σε συσκευές Ε/Ε πάνω από ένα διάδρομο επικοινωνίας είναι αρκετά αργή, συγκριτικά με αντίστοιχη αντιγραφή από την Κύρια Μνήμη σε μια επεξεργαστική μονάδα. Έτσι, θα ήταν αδύνατο να ξεκινήσει εκ νέου μια διεργασία πρόσβασης σε μια συσκευή Ε/Ε, αν δεν είχε ολοκληρωθεί η προηγούμενη.

Μια μεταφορά με DMA ουσιαστικά, αντιγράφει ένα block μνήμης από μια συσκευή σε μια άλλη. Ενώ η επεξεργαστική μονάδα αρχικοποιεί τη μεταφορά, δεν την πραγματοποιεί. Επιπλέον, για την ″third-party″ DMA, όπως συνήθως χρησιμοποιείται σήμερα, από τους διαδρόμους Ε/Ε, η μεταφορά γίνεται από έναν ελεγκτή DMA που είναι ουσιαστικά μέρος του βασικού ολοκληρωμένου της μητρικής πλακέτας του συστήματος. Επίσης, αναπτύχθηκε και η Bus-mastering DMA που επιτρέπει σε μια συσκευή να πάρει εξ ολοκλήρου τον έλεγχο του διαδρόμου και να κάνει τη μεταφορά η ίδια. Αυτό συμβαίνει στο διάδρομο PCI.

Μια τυπική χρήση της DMA είναι η αντιγραφή ενός block μνήμης από την Κύρια (Κεντρική Μνήμη) σε κάποιον απομονωτή συσκευής ή και το αντίστροφο. Μια τέτοια χρήση δεν καθυστερεί την επεξεργαστική μονάδα και ως εκ τούτου μπορεί να εκτελέσει άλλες λειτουργίες.

Οι DMA μεταφορές δεδομένων, είναι δομικά στοιχεία των ενσωματωμένων συστημάτων που είναι απαιτητικά σε υψηλή απόδοση. Επίσης, είναι βασικό στοιχείο των zero-copy υλοποιήσεων οδηγών συσκευών, της δρομολόγησης δικτύων, της αναπαραγωγής μουσικής και βίντεο.

Το υλικό της DMA είναι ένας εξειδικευμένος επεξεργαστής (DMA engine) που σε συνδυασμό με τους ελεγκτές DMA μπορούν να αυξήσουν, έστω και αρκετά περιορισμένα, την πολυεπεξεργαστική ισχύ του συστήματος.

1.2.3 Πρότυπα Διαδρόμου Ε/Ε για Συσκευές Αποθήκευσης

Τα συνηθέστερα πρωτόκολλα επικοινωνίας ενός υπολογιστικού συστήματος με συσκευές αποθήκευσης, είναι το ATA και το SCSI.

1.2.3.1 ATA

Το πρότυπο ATA (AT-bus attachment), γνωστό και ως IDE[2] (Integrated Drive Electronics), ήταν για πολλά χρόνια, με αλλεπάλληλες αλλά όχι κατά βάση βελτιώσεις, το κοινό πρότυπο συσκευών αποθήκευσης στους προσωπικούς υπολογιστές. Σήμερα απαντάται μόνο σε περιπτώσεις εσωτερικών σκληρών δίσκων, μέσα στο ίδιο το υπολογιστικό σύστημα, λόγω της αδυναμίας μεταφοράς δεδομένων με καλώδια μεγαλύτερα των 50 εκατοστών. Είναι το φθηνότερο και το πιο στατικό πρωτόκολλο μεταφοράς δεδομένων από συσκευές αποθήκευσης αφού εδώ και αρκετά χρόνια, οι βελτιώσεις που έγιναν σ' αυτό δεν έδωσαν παρά μια μικρή ώθηση στην ταχύτητα ροής των δεδομένων. Έχει εύρος 16 bits και μπορούν στο ίδιο καλώδιο να ενωθούν μέχρι δύο συσκευές αποθήκευσης (είτε σκληροί δίσκοι, είτε cd(-rw)/dvd(-rw)). Οι δύο συσκευές αυτές, χαρακτηρίζονται ως συσκευή 0 (master) και συσκευή 1 (slave). Ουσιαστικά όμως καμία από τις δύο δεν μπορεί να θεωρηθεί ως master, αφού, αν η slave είναι κατειλημμένη, η συσκευή master δεν μπορεί να εκτελέσει μια εντολή, αν η slave δεν ολοκληρώσει. Έτσι ουσιαστικά και οι δύο συσκευές είναι slaves στον οδηγό συσκευής του εκάστοτε λειτουργικού συστήματος.

Το πρότυπο ATA χρησιμοποιεί παράλληλη μεταφορά δεδομένων σε αντίθεση με τον απόγονό του που στις μέρες μας ανθεί, το SATA (Serial ATA). Έτσι για το διαχωρισμό των προτύπων το ATA μετονομάστηκε σε PATA (Parallel ATA).

1.2.3.2 SCSI

Το SCSI (Small Computer System Interface), είναι ένα πρωτόκολλο επικοινωνίας και ένα σύνολο εντολών για συσκευές (ως επί το πλείστον) αποθήκευσης σε εσωτερικούς ή εξωτερικούς διαδρόμους. Το ακρωνύμιο SCSI είναι συνυφασμένο με μεγάλες υπολογιστικές μονάδες και συσκευές αποθήκευσης. Δεν απαντάται τόσο συχνά σε προσωπικούς υπολογιστές όσο σε εξυπηρετητές και συστοιχίες δίσκων RAID.

[2] ο ελεγκτής του δίσκου βρίσκεται στον ίδιο το δίσκο και όχι στο διάδρομο

Η μεταφορά των δεδομένων, όπως και στο ATA, γίνεται παράλληλα. Ωστόσο, σε αντίθεση μ' αυτό, εξελίχθηκε πολύ περισσότερο. Ουσιαστικά δεν αποτελεί ένα πρότυπο, αλλά ένα σύνολο προτύπων συνυφασμένων με τη βασική του έννοια. Το SCSI άνθησε τη δεκαετία του 90 και κυριάρχησε την αγορά των εξυπηρετητών για επαγγελματική χρήση.

Τα διάφορα πρότυπα που βασίζονται στο SCSI δημιούργησαν ένα βασικό πρόβλημα στην επικράτησή του, αφού δεν υπήρχε συγκεκριμένος τύπος σύνδεσης μεταξύ των συσκευών που κατασκεύαζαν διαφορετικές εταιρίες.

Κάθε συσκευή στο διάδρομο αναγνωρίζεται με ένα τουλάχιστον LUN (Logical Unit Number). Μια συσκευή αποθήκευσης αποτελείται από έναν αριθμό από logical blocks που αναφέρονται ως LBA (Logical Block Address)

Το σύνολο των εντολών που υποστηρίζονται από έναν ελεγκτή ή μια συσκευή scsi ποικίλλει ανάλογα με τον τύπο τους. Μπορούμε να διαχωρίσουμε τις περίπου 60 εντολές scsi σε N (Non-data), W (writing data), R (reading data), B (bidirectional).

1.2.3.3 Μια διαφορετική Προσέγγιση

Τα πρότυπα ATA και SCSI παρ όλες τις χαρακτηριστικές διαφορές τους, έχουν ένα βασικό κοινό στοιχείο: υλοποιούν παράλληλη μεταφορά δεδομένων σε μοιραζόμενο διάδρομο. Αυτό έχει ως αποτέλεσμα την απαίτηση για συγχρονισμό μεταξύ των συσκευών στον εκάστοτε διάδρομο. Παρόλο που μεταφέρονται παραπάνω από 1 bit ανά μετάδοση, η παράλληλη μεταφορά είναι πιο αργή από μια πολύ μικρότερου μήκους σειριακή μεταφορά. Έτσι, οι σχεδιαστές κατασκεύασαν την εξέλιξη των προτύπων ATA και SCSI το SATA και τα SSA, FC-AL, Serial Attached SCSI (SAS), αντίστοιχα.

Στις μέρες μας το SATA αποτελεί κοινό πρότυπο για συσκευές αποθήκευσης προσωπικών υπολογιστών, ενώ τα αντίστοιχα SSA, FC-AL, SAS, όπως θα περίμενε κανείς από την πρόσφατη ιστορία, απαντώνται σε εξυπηρετητές δικτύων απαιτητικούς σε πολύ γρήγορη μεταφορά δεδομένων και αρκετά αξιόπιστη.

Το πρότυπο SATA, αφήνει πίσω τον κλασικό διάδρομο αφέντη / σκλάβου (master / slave bus) δίνοντας στην κάθε συσκευή αποκλειστικό καλώδιο και εύρος ζώνης μεταφοράς δεδομένων. Παρόλο που χρειάζεται περισσότερους ελεγκτές, όταν πρωτοεμφανίστηκε το πρότυπο, δεν αποτελούσε βασικό μειονέκτημα. Επίσης, συσκευές SATA μπορούν να συνδεθούν σε ελεγκτές SAS (Serial Attached SCSI) και να επικοινωνήσουν στο ίδιο φυσικό καλώδιο, όπως και οι συσκευές SAS.

Το SAS, όπως προαναφέρθηκε, είναι ένα σειριακό πρωτόκολλο επικοινωνίας για συσκευές αποθήκευσης. Σχεδιάστηκε με στόχο την επαγγελματική αγορά, για να αντικαταστήσει το SCSI, που μέχρι τώρα μεσουρανούσε. Επι-

τρέπει μεγαλύτερη ταχύτητα από οτιδήποτε παρόμοιο και είναι συμβατό προς τα πίσω με το SATA. Παρ όλη τη βασική του διαφορά με το SCSI (σειριακή vs παράλληλη μεταφορά), χρησιμοποιεί εντολές SCSI για επικοινωνία με συσκευές.

Αποτελείται από τα εξής:

- Initiator (ουσιαστικά το σύστημα που θα φιλοξενήσει τις συσκευές)
- Expanders
- Targets (συνήθως πρόκειται για συστοιχίες δίσκων)

Το σύνολο της συσκευών που υλοποιούν ένα SAS σύστημα, ονομάζεται SAS Domain και μπορεί να περιέχει μέχρι 16256 συσκευές, αριθμό εξωφρενικό για τις μέχρι σήμερα προδιαγραφές παρόμοιων συστημάτων. Οι ταχύτητες μεταφοράς αγγίζουν τα 3 Gbit/s και αναμένεται να φτάσουν τα 10 Gbit/s μέχρι το 2010.

Η τοπολογία ενός SAS domain, ακολουθεί πρότυπα δικτύου υπολογιστών. Οι Initiators συνδέονται στους Expanders. Οι Expanders (Fanout ή Edge) ενώνονται σε οποιοδήποτε συνδυασμό με Initiators, Expanders ή Targets, με δύο εξαιρέσεις: πρώτον, μπορεί να υπάρχει μόνο ένας Fanout Expander σε ένα domain, δεύτερον, κάθε Edge Expander μπορεί να ενωθεί μόνο σε έναν άλλο expander, με δεδομένο ότι το σύνολο συσκευών που μπορούν να ενωθούν στον καθένα είναι 128.

1.3 Τεχνολογίες συσκευών αποθήκευσης

Το μειωμένο εύρος των παραπάνω διαδρόμων επικοινωνίας σε συνδυασμό με την όλο και αυξανόμενη ανάπτυξη των τεχνολογιών δικτύων διασύνδεσης και την επιρροή των συστημάτων παράλληλης επεξεργασίας οδήγησε τους σχεδιαστές να προτείνουν νέες λύσεις για συστήματα αποθήκευσης απαιτητικά σε ταχύτητες ανάκτησης και εγγραφής δεδομένων. Έτσι, στις μέρες μας, επικρατούν ταυτόχρονα χωρίς να αλληλοαναιρούνται, τρεις κυρίως μέθοδοι αποθήκευσης δεδομένων:

- *Direct Attached Storage (Άμεσα συνδεδεμένη αποθήκευση)*
- *Network Attached Storage (Δικτυακά συνδεδεμένη αποθήκευση)*
- *Storage Area Network (Δίκτυο μοιραζόμενης Αποθήκευσης)*

1.3.1 Τοπική Αποθήκευση

Η τοπική αποθήκευση αναφέρεται σε ό,τι αναλύθηκε παραπάνω. Αποτελεί τον πιο συνηθισμένο τρόπο αποθήκευσης δεδομένων για προσωπικούς υπολογιστές ή και μικρής εμβέλειας επαγγελματικής χρήσης συστήματα. Η απόδοση σε ταχύτητα, αξιοπιστία και επεκτασιμότητα μειώνεται εκθετικά με την αύξηση των συσκευών ή του αποθηκευτικού χώρου.

1.3.2 Δικτυακή Αποθήκευση

Η δικτυακή αποθήκευση στις μέρες μας αποκτά διαφορετικό χαρακτήρα με την ανάπτυξη σταθερών, αξιόπιστων αλλά και γρήγορων τεχνολογιών δικτύου.

1.3.2.1 NAS

Τα πρότυπα δικτύων είναι ισχυρά πρότυπα που προκύπτουν από αναλύσεις συστημάτων. Τα δύο βασικότερα που έδωσαν ώθηση στην ανάπτυξη του προτύπου NAS είναι τα εξής:

- Network File System (NFS)
  Αναπτύχθηκε από τη Sun Microsystems και αποτελεί το βασικό πρότυπο για πρόσβαση σε δικτυακά δεδομένα σε UNIX συστήματα.

- Common Internet File System (CIFS)
  Αναπτύχθηκε από την IBM και τη Microsoft και αποτελεί το πρότυπο για τα λειτουργικά συστήματα Windows.

Από τα παραπάνω προκύτπει ότι οι συσκευές αποθήκευσης που προσφέρουν δεδομένα δικτυακά, είναι πλέον ευκολότερο να συνδεθούν και να τις διαχειριστεί κάποιος από ό,τι συσκευές Τοπικής Αποθήκευσης παρόμοιου μεγέθους.

Εξάλλου, τα τελευταία χρόνια, οι τεχνολογίες δικτύου όχι μόνο "έφτασαν" σε ταχύτητα τις τεχνολογίες διαδρόμων Ε/Ε και συσκευών αποθήκευσης αλλά και τις ξεπέρασαν. Έτσι, η συμφόρηση πέρασε από το δίκτυο στον εξυπηρετητή και στη συνδεδεμένη σ' αυτόν συσκευή αποθήκευσης.

Με στόχο τη μείωση του κόστους, οι εταιρίες προσανατολίζονται περισσότερο στη λύση δικτυακής αποθήκευσης αφού:

- Τα σταθερά πρότυπα του NAS επιτρέπουν ευκολότερη εγκατάσταση και χαμηλότερο κόστος διαχείρισης και συντήρησης.

- Η αυξανόμενη ταχύτητα των δικτύων μπορεί να απαλείψει το κενό απόδοσης που υπήρχε μεταξύ NAS και DAS για πολλές εφαρμογές

- Η πραγματική διάθεση δεδομένων μέσω δικτύου σε ετερογενείς πελάτες γίνεται με το NAS και όχι με το DAS.
- Η σημερινή τάση απαιτεί την "επανακεντρικοποίηση"(recentralize) της αποθήκευσης για τη μείωση του κόστους διαχείρισης.

Ορισμένα από τα πλεονεκτήματα του NAS είναι: αντίγραφα ασφαλείας μέσω δικτύου, μοίρασμα / συγκέντρωση δεδομένων, εύκολη διαχείριση των πηγών αποθήκευσης, κοινά δεδομένα μεταξύ ετερογενών εξυπηρετητών και συσκευών αποθήκευσης.

1.3.2.2 SAN

Οι εταιρίες βιάστηκαν να καρπωθούν τα πλεονεκτήματα του NAS χωρίς όμως να χάσουν την πελατεία τους και έτσι κατασκεύασαν συσκευές με εξειδικευμένο υλικό που χρησιμοποιεί τα πρότυπα του NAS. Αυτές οι συσκευές ονομάζονται SAN (Storage Area Networks). Τα SANs δεν έχουν σαφώς ορισμένα πρότυπα.

Αντί λοιπόν να τοποθετηθούν οι συσκευές αποθήκευσης στο δίκτυο, παρεμβάλλεται το δίκτυο μεταξύ των εξυπηρετητών και των συσκευών αυτών με αποτέλεσμα να υφίστανται την καθυστέρηση του δικτύου σε συνδυασμό με την καθυστέρηση των διαδρόμων του DAS. Το εγχείρημα αυτό αποσκοπούσε στο να καλυφθεί το κενό που δημιούργησε το NAS αφού δεν μπορούσαν πολλά συστήματα να προσπελάσουν τη συγκεκριμένη συσκευή αποθήκευσης.

1.4 Τρόπος Αποθήκευσης Δεδομένων

Ο τρόπος αποθήκευσης δεδομένων σε συσκευές για αυτό το σκοπό, γίνεται με τον ορισμό κάποιας αφηρημένης δομής δεδομένων (ADT) έτσι, ώστε η ανάκτηση αυτών να είναι εφικτή. Τέτοιες δομές μπορεί να είναι ένα Σύστημα Αρχείων Δίσκου (Disk File System), μια Σχεσιακή Βάση δεδομένων (Relational Database) ή και συνδυασμός αυτών, δηλαδή ένα σύστημα αρχείων με βάση δεδομένων (Database File System).

Το πώς οργανώνονται τα δεδομένα στη συσκευή αποθήκευσης δεν απασχολεί το χρήστη, αφού μέσω του ενδιάμεσου αυτού επιπέδου (του συστήματος αρχείων, της βάσης δεδομένων κοκ.) μπορεί να ανακτήσει ή να αποθηκεύσει δεδομένα, ανεξάρτητα.

Τα συστήματα αρχείων ποικίλλουν ανάλογα με το στόχο που πρέπει να επιτελέσουν. Συνήθως τα δεδομένα οργανώνονται σε αρχεία που αποτελούν τη μικρότερη δυνατή δομή, και σε φακέλους. Οι φάκελοι περιέχουν με τη σειρά τους, είτε περισσότερους φακέλους είτε αρχεία. Έτσι δομείται μια

ιεραρχική οργάνωση των δεδομένων με βάση το πώς θα επιλέξει ο χρήστης, ο οποίος είναι υπεύθυνος για την κατανομή των αρχείων ή των φακέλων. Βασικό ρόλο στον καθορισμό του συστήματος αρχείων που θα χρησιμοποιηθεί σε μια συσκευή αποθήκευσης έχει και το λειτουργικό σύστημα με το οποίο ο χρήστης αλληλεπιδρά με τη συσκευή αυτή. Έτσι αν για παράδειγμα η συσκευή αποθήκευσης είναι μια κάρτα ημιαγώγιμης μνήμης περιορισμένου χώρου σε μια ψηφιακή φωτογραφική μηχανή, οι φάκελοι είναι μάλλον περιττοί.

Οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν πίνακες και σχέσεις μεταξύ αυτών, για να αναπαραστήσουν δεδομένα. Πολλές φορές μπορεί σε ένα χαμηλότερο επίπεδο να οργανώνουν τα δεδομένα τους σε συστήματα αρχείων, για να είναι συμβατές με ένα λειτουργικό σύστημα. Ο χρήστης αλληλεπιδρά με τέτοιου τύπου δομές με μια γλώσσα δομημένων ερωτήσεων, την SQL (Structured Query Language).

Ορισμένες δομές δεδομένων εκτός από τα δεδομένα του χρήστη, αποθηκεύουν και δεδομένα που αφορούν στον τρόπο οργάνωσης των παραπάνω. Έτσι πχ οι βάσεις δεδομένων αποθηκεύουν τις σχέσεις που έχουν οι πίνακες δεδομένων μεταξύ τους, τα συστήματα αρχείων αποθηκεύουν δεδομένα για γρηγορότερη ανάκτηση αρχείων ή και για ασφάλεια κλπ. Αυτά τα παραπάνω δεδομένα πληροφορίας ονομάζονται metadata.

Κεφάλαιο **2**

Δίκτυα Διασύνδεσης

Τα δίκτυα διασύνδεσης υπολογιστικών συστημάτων άρχισαν να αναπτύσσονται σχεδόν από τη δημιουργία του πρώτου υπολογιστή. Ιστορικά αναφέρονται ως προσπάθεια διαφανούς και σίγουρης επικοινωνίας σε περιόδους πολέμου, όμως γρήγορα γιγαντώθηκαν σε ακαδημαϊκό επίπεδο και στις αρχές της περασμένης δεκαετίας πέρασαν στο ευρύ κοινό με τη δημιουργία του Internet και της δημοφιλούς υπηρεσίας του, τον Παγκόσμιο Ιστό (World Wide Web).

Ουσιαστικά το Internet είναι η σύνδεση όλων των επιμέρους δικτύων υπολογιστικών συστημάτων που υπάρχουν ανά τον κόσμο. Στην παρούσα εργασία θα επικεντρωθούμε περισσότερο στα τοπικά δίκτυα υπολογιστών και πιο συγκεκριμένα στους τρόπους διασύνδεσης συστοιχίας υπολογιστών.

Η συστοιχία υπολογιστών (computer cluster) είναι ένα σύνολο συνδεδεμένων υπολογιστικών συστημάτων που λειτουργούν ενωμένα έτσι, ώστε ορισμένες φορές να μπορούν να εμφανιστούν σαν ένας υπολογιστής. Τις περισσότερες φορές οι συστοιχίες συνδέονται μεταξύ τους με τοπικά δίκτυα διασύνδεσης και έχουν στόχο τη βελτίωση της απόδοσης και της αξιοπιστίας που προσφέρουν σε σύγκριση με ένα υπολογιστικό σύστημα παρόμοιας δυναμικής. Χαρακτηριστικό τους είναι το χαμηλό κόστος τους, συγκριτικά με το ισοδύναμο αυτόνομο σύστημα.

Τα δίκτυα μπορούν να χωριστούν ανάλογα με το μέγεθός τους σε τρεις κατηγορίες:

- Τοπικά Δίκτυα (Local Area Networks).
  Η διάμετρός τους δεν ξεπερνά τα μερικά χιλιόμετρα. Ο ρυθμός μετάδοσης είναι τουλάχιστον της τάξης των μερικών δεκάδων Mbps ή Gbps και ανήκουν πλήρως σε ένα οργανισμό.

- Δίκτυα Ευρείας Περιοχής (Wide Area Networks)
  Αυτά τυπικά συνδέουν ολόκληρες χώρες, έχουν ρυθμό μετάδοσης της

τάξης των μερικών Mbps, και ανήκουν σε πολλούς οργανισμούς. Παράδειγμα τέτοιων δικτύων είναι το Internet.

- Μητροπολιτικά Δίκτυα (Metropolitan Area Networks).
  Ανάμεσα στα LANs και WANs βρίσκονται τα MANs. Είναι δίκτυα που καλύπτουν μια ολόκληρη πόλη και έχουν ταχύτητες μετάδοσης, ανάλογες μ' αυτές των LANs. Ανάλογο MAN είναι το δίκτυο της καλωδιακής τηλεόρασης (Cable Television).

Στις ενότητες που ακολουθούν θα αναφερθούν συνοπτικά τα πρωτόκολλα επικοινωνίας TCP/IP και Ethernet, λόγω της ευρείας διάδοσής τους και θα αντιπαρατεθούν με το δίκτυο Myrinet και τον τρόπο λειτουργίας του.

2.1 Ethernet

Το Ethernet είναι ένα σύνολο πρωτοκόλλων για Τοπικά Δίκτυα. Σ' αυτό, ορίζονται πρότυπα καλωδίωσης και σηματοδότησης για το φυσικό επίπεδο και σαφείς τρόποι διευθυνσιοδότησης. Από την αρχή της περασμένης δεκαετίας μέχρι σήμερα, αντικατέστησε τεχνολογίες όπως Token Ring, FDDI, ARCNET και εδραιώθηκε στο χώρο με τη μορφή που το απαντάμε: σε τοπολογία αστέρα και με συνεστραμμένα ζεύγη καλωδίων, και όχι με ομοαξονικό καλώδιο.

Ο όρος Ethernet απαντάται στην οικογένεια των προϊόντων για τοπικά δίκτυα και καλύπτεται από το πρότυπο IEEE 802.3 που ορίζει αυτό που είναι κοινώς γνωστό ως πρωτόκολλο CSMA/CD. Οι ταχύτητες που μπορεί κανείς να συναντήσει σε δίκτυα με Ethernet είναι από μερικές δεκάδες Mbps μέχρι κάποιες δεκάδες Gbps.

Το Ethernet αντικατέστησε υπάρχουσες τεχνολογίες και εδραιώθηκε στην αγορά, λόγω συγκεκριμένων χαρακτηριστικών:

- Είναι κατανοητό, εύκολο στην υλοποίηση, στην εγκατάσταση και στη διαχείριση.
- Συστήνει δίκτυα χαμηλού κόστους
- Δίνει δυνατότητα για μεγάλου εύρους τοπολογία.
- Εγγυάται την επιτυχή σύνδεση και λειτουργία προϊόντων ανεξάρτητα από τον κατασκευαστή.

Το Ethernet αρχικά αναπτύχθηκε σαν ένα πειραματικό δίκτυο με ομοαξονικό καλώδιο τη δεκαετία του '70 από την XEROX με ταχύτητα 3Mbps χρησιμοποιώντας το πρωτόκολλο CSMA/CD (Carrier Sense Multiple Access /

Collision Detect) με περιστασιακές ανάγκες σε κίνηση φορτίου. Η επιτυχία αυτού του έργου, οδήγησε στη δημιουργία του πρώτου 10Mbps προτύπου Ethernet από τις DEC, Intel, Xerox.

Τα τοπικά δίκτυα με Ethernet αποτελούνται από κόμβους δικτύου και μέσα διασύνδεσης. Οι κόμβοι διακρίνονται σε δύο μεγάλες κατηγορίες:

- Τον Τερματικό Εξοπλισμό (DTE), που είναι η πηγή των δεδομένων που κινούνται στο δίκτυο. Σταθμοί εργασίας, προσωπικοί υπολογιστές, εξυπηρετητές αρχείων ή εκτυπωτών που αναφέρονται και ως τερματικοί σταθμοί.

- Τον Εξοπλισμό Διασύνδεσης (DCE), που είναι οι συσκευές που λαμβάνουν και μεταδίδουν / μεταφέρουν πακέτα κατά μήκος του δικτύου. Μπορούν είτε να είναι αυτόνομες συσκευές όπως επαναλήπτες (repeaters), μεταγωγείς (switches), ή πλήμνες (hub) είτε διεπαφές επικοινωνίας όπως διεπαφές δικτύου και modems (διαποδιαμορφωτής).

Το μέσο μετάδοσης στο Ethernet, όπως έχει διαμορφωθεί σήμερα, διακρίνεται σε δύο κατηγορίες χάλκινου καλωδίου: το καλώδιο αθωράκιστων συνεστραμμένων ζευγών (UTP) και το αντίστοιχο θωρακισμένο (STP). Επιπρόσθετα, σε κάποιες περιπτώσεις, χρησιμοποιείται και το καλώδιο οπτικής ίνας.

2.1.1 Τοπολογία και Δομή Δικτύου

Τα τοπικά δίκτυα (LAN) υφίστανται σε διάφορες τοπολογίες αλλά ανεξάρτητα από το μέγεθος και την πολυπλοκότητά τους, διακρίνονται σε τρεις βασικές δομές διασύνδεσης.

Η απλούστερη, είναι η σύνδεση από σημείο προς σημείο. Σ' αυτόν τον τρόπο σύνδεσης, εμπλέκονται δύο μόνο συσκευές και μπορεί να γίνει οποιοσδήποτε συνδυασμός. Το καλώδιο διασύνδεσης σε αυτή την περίπτωση είναι ένας "σύνδεσμος δικτύου". Το μέγιστο μήκος καλωδίου εξαρτάται από τον τύπο του και από τον τρόπο μετάδοσης.

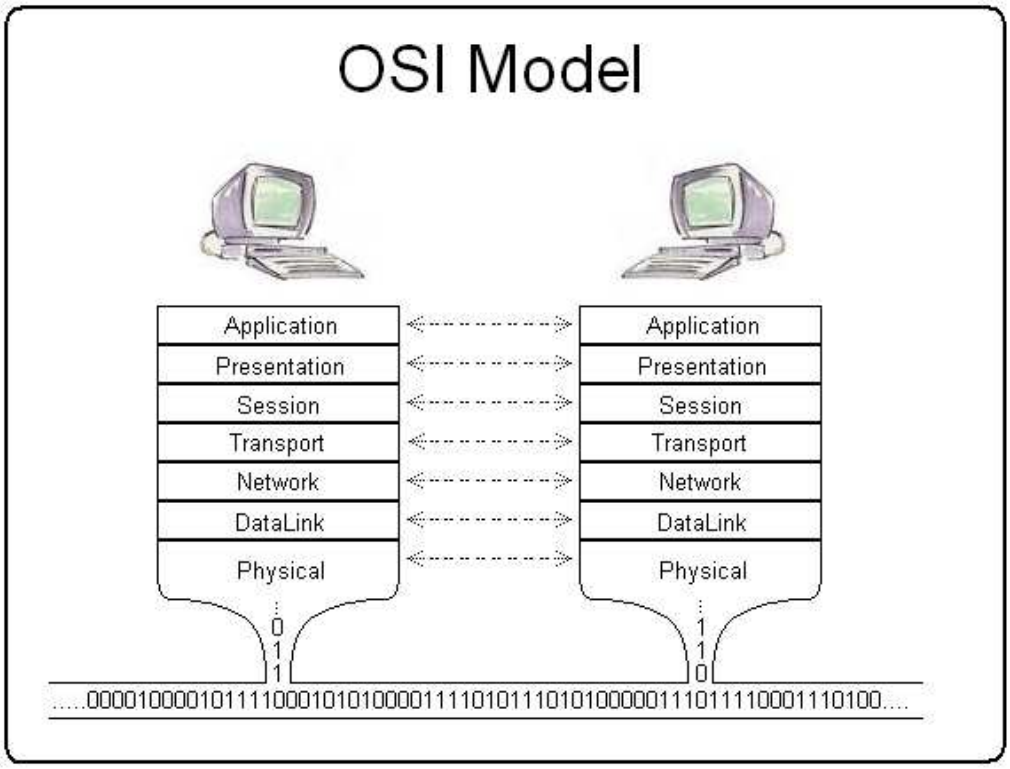
Τα δίκτυα Ethernet στην αρχή της ανάπτυξής τους, υλοποιούνταν με μια δομή ομοαξονικού διαδρόμου. Το μήκος του καλωδίου που αποτελούσε τον κεντρικό διάδρομο δεν μπορούσε να ξεπεράσει τα 500m και μπορούσαν να συνδεθούν πάνω σε αυτό μέχρι 100 σταθμοί. Επιμέρους δίκτυα, μπορούσαν να συνδεθούν μεταξύ τους με επαναλήπτες με την προϋπόθεση ότι δεν υπάρχουν εναλλακτικά μονοπάτια μεταξύ σταθμών και ότι ο αριθμός τους δεν ξεπερνά τα 1024 DTE. Το μέγιστο μήκος ενός μονοπατιού μεταξύ δύο τερματικών σταθμών ήταν επίσης ένα μέγεθος πεπερασμένο.

Παρόλο που σήμερα δεν πρόκειται να συστηθεί δίκτυο με τέτοια τοπολογία, υπάρχουν ακόμα περιπτώσεις όπου τα συναντά κανείς.

Από τις αρχές της δεκαετίας του 1990, η επιλογή της τοπολογίας δικτύου Ethernet ήταν η συνδεσμολογία αστέρα. Η κεντρική δικτυακή μονάδα είναι είτε ένας πολύθυρος επαναλήπτης (multiport repeater), είτε ένας μεταγωγέας δικτύου (network switch). Όλες οι συνδέσεις σε ένα δίκτυο με τοπολογία αστέρα υλοποιούνται με συνδέσμους σημείο - προς - σημείο είτε με καλώδιο συνεστραμμένων ζευγών είτε με οπτικό καλώδιο.

2.1.2 Ethernet και OSI

Το πρότυπο 802.3 περιλαμβάνει στρώματα που αντιστοιχούν σε αυτά του μοντέλου αναφοράς OSI. Όπως και σε άλλα πρωτόκολλα της σειράς 802, το στρώμα ζεύξης δεδομένων διακρίνεται στα εξής:



Σχήμα 2.1: Μοντέλο OSI

- υπό-στρωμα ελέγχου προσπέλασης στο μέσο μετάδοσης (Medium Access Control)

Στα LAN το παρόν υπό-στρωμα είναι ιδιαίτερα σπουδαίο, αφού ως βάση της επικοινωνίας χρησιμοποιείται σχεδόν καθ' ολοκληρίαν δίαυλος πολλαπλής προσπέλασης.

- υπό-στρωμα MAC για πελάτη που μπορεί να είναι:
  - ⋆ υπόστρωμα ελέγχου λογικής σύνδεσης (Logical Link Control) σε περίπτωση που η μονάδα σύνδεσης είναι τερματικός σταθμός. Αυτό το υπόστρωμα είναι η διεπαφή μεταξύ του υποστρώματος MAC και των ανώτερων στρωμάτων της στοίβας του πρωτοκόλλου του τερματικού σταθμού.
  - ⋆ Γέφυρα, αν η μονάδα σύνδεσης είναι DCE.
    Οι γέφυρες προσφέρουν διεπαφές μεταξύ τοπικών δικτύων που χρησιμοποιούν είτε το ίδιο πρωτόκολλο είτε διαφορετικό. Το πρότυπό τους ορίζεται στο 802.1.

Οι προδιαγραφές για το LLC και τις γέφυρες είναι κοινές για όλα τα IEEE 802 πρωτόκολλα οπότε η συμβατότητα δικτύου αποτελεί ευθύνη του κάθε πρωτοκόλλου.

Το στρώμα MAC ελέγχει την πρόσβαση του κόμβου στο μέσο μετάδοσης και είναι συγκεκριμένο σε κάθε πρωτόκολλο. Όλα τα 802.3 MAC πρέπει να διαθέτουν τις ίδιες βασικές προδιαγραφές, ανεξάρτητα αν υλοποιούν ή όχι κάποια προαιρετική επέκταση. Κοινό τους σημείο πρέπει να είναι οπωσδήποτε ο ίδιος ρυθμός μετάδοσης.

Το φυσικό στρώμα του 802.3 εξαρτάται από το ρυθμό μετάδοσης δεδομένων, την κωδικοποίηση σήματος και τον τύπο σύνδεσης των δύο κόμβων. Για παράδειγμα το Gigabit Ethernet λειτουργεί είτε με καλώδιο συνεστραμμένων ζευγών είτε με καλώδιο οπτικών ινών, αλλά κάθε τύπος καλωδίωσης ή κωδικοποίησης σήματος απαιτεί διαφορετική υλοποίηση του φυσικού στρώματος.

Το υπόστρωμα MAC έχει δύο βασικές "ευθύνες":

- την ενθυλάκωση δεδομένων, συμπεριλαμβανομένης και της συναρμολόγησης (σύνοψης) των πλαισίων πριν από τη μετάδοση, και της ανίχνευσης σφαλμάτων κατά τη διάρκεια και μετά το πέρας της λήψης.
- Τον καθορισμό της πρόσβασης στο μέσο, με εκκίνηση της μετάδοσης και ανάκαμψη από σφάλματα μετάδοσης.

2.1.3 Δομή του πλαισίου Ethernet

Το πρότυπο 802.3 ορίζει μια βασική μορφή του πλαισίου δεδομένων που είναι απαραίτητο σε όλες τις υλοποιήσεις MAC καθώς και κάποιες επεκτάσεις του. Στη βασική του μορφή αποτελείται από επτά πεδία:

- Preamble (PRE). 7 bytes. Αποτελεί μια ακολουθία 0 και 1 που ενημερώνει τους σταθμούς - παραλήπτες ότι φτάνει ένα πλαίσιο και είναι ένας τρόπος συγχρονισμού των φυσικών στρωμάτων με τη ροή των εισερχόμενων bit.

- Start-of-frame delimiter (SOF). 1 byte. Είναι μια ακολουθία 0 και 1 που τελειώνει με δύο συνεχόμενα 1 για να αναγνωρίζεται το πλέον σημαντικό bit του πεδίο DA.

- Destination Address (DA). 6 bytes. Το πεδίο αυτό καθορίζει ποιοι σταθμοί πρέπει να λάβουν το πλαίσιο. Το πλέον σημαντικό bit του πεδίου καθορίζει τον τύπο της διεύθυνσης. Αν είναι μεμονωμένη τότε είναι 0 αλλιώς αν είναι ομαδική, είναι 1. Το δεύτερο σημαντικότερο bit καθορίζει το αν είναι globally ή locally administered. Τα υπόλοιπα 46 είναι μια τιμή μοναδική που αναγνωρίζει ένα σταθμό, ένα σύνολο σταθμών ή όλους τους σταθμούς στο δίκτυο.

- Source Addresses (SA). 6 bytes.

- Length/Type. 2 bytes. Το πεδίο αυτό δείχνει είτε τον αριθμό των bytes δεδομένων του στρώματος MAC-πελάτη που περιέχονται στο πεδίο data του πλαισίου, είτε τον τύπο του πλαισίου (αν αποτελεί επέκταση).

- Data. n bytes, n‹=1500. Αν το μέγεθός τους είναι μικρότερο από 46 bytes, τότε προστίθεται ένα "γεμισμα" (pad) έτσι, ώστε να φτάσουν το ελάχιστο μέγεθος.

- Frame Check Sequence (FCS). 4 bytes. Περιέχει έναν 32-bit κυκλικό κώδικα πλεονασμού που υπολογίζεται από το στρώμα MAC του αποστολέα και επανυπολογίζεται από το στρώμα MAC του παραλήπτη για έλεγχο κατεστραμμένων πλαισίων. Υπολογίζεται από τα πεδία DA, SA, Length/Type και Data.

2.1.4 Μετάδοση Πλαισίου

Όταν το στρώμα MAC ενός τερματικού σταθμού λάβει μια αίτηση για μετάδοση πλαισίου, με τη διεύθυνση και τα δεδομένα πληροφορίας από το υπόστρωμα LLC, το MAC ξεκινά την ακολουθία μετάδοσης μεταφέροντας τις πληροφορίες του LLC στο πλαίσιο απομονωτή του MAC (MAC frame buffer).

- τοποθετούνται τα πεδία PRE και SOF

- τοποθετούνται τα πεδία DA και SA

- καταμετρώνται τα δεδομένα LLC και τοποθετείται ο αριθμός στο πεδίο Length/Type.
- τοποθετούνται τα bytes δεδομένα LLC στο πεδίο Data. Αν χρειαστεί "γεμίζονται" μέχρι να φτάσουν τα 46 bytes.
- υπολογίζεται η τιμή του FCS με τα πεδία DA, SA, Length/Type και Data και τοποθετείται μετά το τέλος του πεδίου Data.

Αφού συναρμολογηθεί το πλαίσιο, η πραγματική μετάδοση εξαρτάται από το αν το MAC λειτουργεί σε μονή ή διπλή κατεύθυνση.

Το πρότυπο 802.3 απαιτεί όλα τα στρώματα MAC να λειτουργούν σε μονή κατεύθυνση (half-duplex) έτσι, ώστε το στρώμα να μπορεί είτε να μεταδώσει είτε να λάβει ένα πλαίσιο χωρίς να επιτελεί τις λειτουργίες αυτές ταυτόχρονα.

2.1.5 Η μέθοδος προσπέλασης CSMA/CD

Το πρωτόκολλο CSMA/CD αναπτύχθηκε σαν μια μορφή διευθέτησης του κοινού μέσου που μοιράζονται δύο ή περισσότεροι σταθμοί σε ένα περιβάλλον χωρίς μεταγωγή. Οι κανόνες του πρωτοκόλλου συνοψίζονται από το ακρονύμιο:

- Carrier Sense.
  Κάθε σταθμός παρακολουθεί την κίνηση στο μέσο, για να διαπιστώσει πότε υπάρχουν κενά στη μετάδοση πλαισίων.
- Multiple Access.
  Οι σταθμοί μπορεί να ξεκινήσουν τη μετάδοση οποιαδήποτε στιγμή αντιληφθούν πως δεν υπάρχει κίνηση στο δίκτυο.
- Collision detect.
  Αν δύο ή περισσότεροι σταθμοί στο ίδιο δίκτυο (collision domain) ξεκινήσουν να μεταδίδουν περίπου την ίδια στιγμή, η ροή των bits του ενός και του άλλου θα συγκρουστούν, με αποτέλεσμα και οι δύο μεταδόσεις να είναι μη αναγνώσιμες. Αν συμβεί αυτό, κάθε σταθμός μετάδοσης, πρέπει να είναι ικανός να αντιληφθεί ότι έχει συμβεί μια σύγκρουση, πριν να τελειώσει η μετάδοση των πλαισίων. Πρέπει να σταματήσει τη μετάδοση, μόλις αντιληφθεί τη σύγκρουση και να περιμένει ένα χρονικό διάστημα, πριν προσπαθήσει να ξαναστείλει το πλαίσιο.

Η χειρότερη περίπτωση συμβαίνει, όταν οι δύο πιο μακρινοί σταθμοί στο δίκτυο θέλουν να στείλουν ένα πλαίσιο και ο δεύτερος ξεκινά να μεταδίδει λίγο πριν να φτάσει σε αυτόν το πλαίσιο από τον πρώτο σταθμό. Η σύγκρουση θα ανιχνευτεί σχεδόν αμέσως από τον δεύτερο σταθμό, αλλά δε θα ανιχνευθεί από

τον πρώτο, μέχρι να φτάσει στον τελευταίο το "χαλασμένο" σήμα. Ο μέγιστος χρόνος που χρειάζεται για να ανιχνευτεί μια σύγκρουση είναι περίπου ίσος με το διπλάσιο του χρόνου που απαιτείται να φτάσει το σήμα από τη μία άκρη του δικτύου στην άλλη και ορίζεται ως παράθυρο σύγκρουσης ή slot time.

Αυτό σημαίνει ότι το ελάχιστο μέγεθος πλαισίου και η μεγαλύτερη "διάμετρος" σύγκρουσης είναι ευθέως ανάλογα του slot time, που σημαίνει ότι το slot time και η διάμετρος σύγκρουσης είναι τόσο μεγαλύτερα όσο πιο μεγαλύτερα είναι τα ελάχιστα μεγέθη πλαισίων. Έτσι μπορεί να καθοριστεί το μέγιστο θεωρητικό μήκος καλωδίωσης για 10Mbps στα 2500m και για 100Mbps στα 200μ. Όσον αφορά το Gigabit Ethernet, οι σχεδιαστές διάλεξαν να αυξήσουν το ελάχιστο μέγεθος πλαισίου, γεμίζοντάς το με bytes, που αφαιρούνται όταν φτάσει στον προορισμό του. Έτσι, ακόμα και το Gigabit παραμένει στις προδιαγραφές για απόσταση που έχει το Fast Ethernet, δηλαδή αυτό των 100Mbps, στα 100m.

2.2 TCP/IP

Τα πρωτόκολλα του Internet είναι τα πλέον διαδεδομένα ανοικτά (non-proprietary) πρωτόκολλα γιατί μπορούν να χρησιμοποιηθούν για οποιοδήποτε σύνολο συνδεδεμένων δικτύων και είναι κατάλληλα και για Τοπικά Δίκτυα και για Δίκτυα ευρείας Περιοχής. Τα πρωτόκολλα του Internet είναι μια οικογένεια πρωτοκόλλων με πιο γνωστά το TCP (Transmision Control Program) και το IP[Tan02] (Internet Protocol). Στην οικογένεια αυτή ορίζονται επίσης κοινές εφαρμογές όπως το ηλεκτρονικό ταχυδρομείο, η μεταφορά αρχείων κλπ, αλλά δεν αφορούν την παρούσα εργασία.

Τα πρωτόκολλα αυτά αναπτύχθηκαν στα μέσα της δεκαετίας του 1970 όπου το Υπουργείο Άμυνας των Ηνωμένων Πολιτειών ενδιαφέρθηκε να εγκαταστήσει ένα δίκτυο μεταγωγής πακέτων που θα καθιστούσε εύκολη την επικοινωνία μεταξύ διαφορετικού τύπου υπολογιστικών συστημάτων σε ιδρύματα έρευνας. Με το στόχο της ετερογενούς συνδεσιμότητας, το DARPA (Defense Advanced Researh Projects Agency), χρηματοδότησε την έρευνα για το πρωτόκολλο του Internet (IP) που αναπτύχθηκε στα τέλη της δεκαετίας του 1970, από το πανεπιστήμιο του Standford και τις εταιρίες Bolk, Beranek και Newman (BBN).

Το TCP/IP ενσωματώθηκε αργότερα με την εμφάνιση του BSD (Berkeley Software Distribution) UNIX και ήταν η βάση να δημιουργηθεί το Internet και το World Wide Web (WWW).

Εκτενή αναφορά για όλα τα πρωτόκολλα του Internet μπορεί κανείς να βρει στα RFCs (Request For Comments), τεχνικές αναφορές που δημοσιεύονται και αναλύονται από ολόκληρη την κοινότητα του Internet.

2.2.1 IP

Το IP[Cis] είναι ένα πρωτόκολλο του στρώματος δικτύου και περιέχει πληροφορίες διευθυνσιοδότησης και ελέγχου που επιτρέπουν στα πακέτα να δρομολογούνται. Μαζί με το TCP αποτελούν την καρδιά του Internet. Οι ευθύνες του IP είναι δύο: η χωρίς απώλεια, καλύτερη δυνατή παράδοση δεδομενογραφημάτων μέσα σε ένα διαδίκτυο (internetwork) και η δυνατότητα κατάτμησης και επανασυναρμολόγησης των δεδομενογραφημάτων αυτών, έτσι, ώστε να υποστηρίζονται σύνδεσμοι δεδομένων με διαφορετικές τιμές των μέγιστων μονάδων μεταφοράς (MTU size).

2.2.1.1 Δεδομενογράφημα IP

Το δεδομενογράφημα IP αποτελείται από δύο τμήματα: επικεφαλίδα και κείμενο. Η επικεφαλίδα έχει ένα προκαθορισμένο τμήμα των 20 bytes και ένα προαιρετικό τμήμα μεταβλητού μήκους. Μεταδίδεται σε σειρά μεγάλου άκρου (big endian). Τα πεδία του δεδομενογραφήματος IP είναι:

- Version.
  Κρατά την πληροφορία της έκδοσης στην οποία ανήκει το πρωτόκολλο του δεδομενογραφήματος. Έτσι γίνεται εφικτό το να κρατά μήνες ή και χρόνια η μετάβαση μεταξύ εκδόσεων, με κάποια συστήματα να χρησιμοποιούν την παλιά, και άλλα την καινούρια.

- IP Header Length (IHL).
  Επειδή το μέγεθος της επικεφαλίδας δεν είναι σταθερό, το παρόν πεδίο δείχνει πόσο μεγάλη είναι σε λέξεις των 32-bit. Η ελάχιστη τιμή είναι 5 που δείχνει ότι δεν υπάρχει προαιρετικό τμήμα. Η μέγιστη τιμή του πεδίου αυτού είναι 15 που περιορίζει την επικεφαλίδα στα 60 bytes και έτσι το προαιρετικό τμήμα στα 40 bytes.

- Type-Of-Service.
  Επιτρέπει στον host να πει στο υποδίκτυο το είδος της υπηρεσίας που επιθυμεί. Υπάρχουν διάφοροι συνδυασμοί αξιοπιστίας και ταχύτητας. Στην ψηφιακή φωνή, η γρήγορη παράδοση είναι σπουδαιότερη από την ορθή παράδοση. Στη μεταφορά αρχείων, η μετάδοση χωρίς λάθη είναι σπουδαιότερη από την ταχεία μετάδοση. Περιέχει ένα πεδίο Precedence (προβάδισμα) των 3-bit, τρεις σημαίες, D, T, R και δύο αχρησιμοποίητα bit. Το πεδίο Precedence είναι μια προτεραιότητα από το 0 μέχρι το 7. Τα τρία bit σημαίες επιτρέπουν στον host να καθορίζει αυτό που τον ενδιαφέρει περισσότερο από το σύνολο { Καθυστέρηση - Delay, Διέλευση - Throughput, Αξιοπιστία - Reliability }. Θεωρητικά, τα πεδία

αυτά επιτρέπουν στους δρομολογητές να επιλέξουν μεταξύ ζεύξεων στο φυσικό επίπεδο. Πρακτικά, το πεδίο αυτό δε χρησιμοποιείται σήμερα.

- Total Length.
  Περιλαμβάνει τα πάντα που βρίσκονται μέσα στο δεδομενογράφημα, τόσο στην επικεφαλίδα, όσο και στα δεδομένα. Το μέγιστο μήκος είναι 65.535 bytes.

- Identification.
  Το παρόν πεδίο χρειάζεται για να επιτρέπει στον host προορισμού να καθορίσει σε ποιο δεδομενογράφημα ανήκει το νεοφερμένο τεμάχιο. Όλα τα τεμάχια ενός δεδομενογραφήματος περιέχουν την ίδια τιμή σε αυτό το πεδίο.

- Flags
  Περιέχει ένα πεδίο 3-bit, εκ των οποίων τα 2 λιγότερο σημαντικά ελέγχουν την κατάτμηση. Το λιγότερο σημαντικό ορίζει αν μπορεί να κομματιαστεί το πακέτο. Το μεσαίο ορίζει αν είναι το τελευταίο κομμάτι σε μια σειρά κομματιασμένα πακέτα. Το πλέον σημαντικό bit δε χρησιμοποιείται.

- Fragment Offset.
  Το πεδίο αυτό, πληροφορεί για το σε ποιο σημείο του τρέχοντος δεδομενογραφήματος ανήκει το τεμάχιο αυτό. Όλα τα τεμάχια ενός δεδομενογραφήματος εκτός από το τελευταίο πρέπει να είναι πολλαπλάσια των 8 bytes που αποτελεί τη στοιχειώδη μονάδα τεμαχίου. Εφόσον υπάρχουν 13 bit, το μέγιστο δεδομενογράφημα αποτελείται από 8192 τεμάχια με αποτέλεσμα ένα μέγιστο μήκος 65536 bytes που είναι μεγαλύτερο κατά ένα από το πεδίο Total Length.

- Time-to-Live.
  Περιέχει ένα μετρητή που σταδιακά μειώνεται μέχρι να γίνει 0. Σ' αυτό το σημείο, το δεδομενογράφημα απορρίπτεται. Αυτό το χαρακτηριστικό εμποδίζει τα δεδομενογραφήματα να περιφέρονται αιωνίως.

- Protocol.
  Δείχνει σε ποια διαδικασία θα μεταφερθεί το δεδομενογράφημα, δηλαδή ποιο πρωτόκολλο ανώτερης βαθμίδας θα χρησιμοποιήσει το πακέτο, αφού τελειώσει η επεξεργασία του από το IP.

- Header Checksum.
  Βοηθά στην ακεραιότητα της επικεφαλίδας του IP. Είναι χρήσιμο για την ανίχνευση λαθών που δημιουργούνται από χαλασμένες λέξεις μνήμης

μέσα σε έναν δρομολογητή. Πρέπει να υπολογίζεται ξανά σε κάθε βήμα επειδή πάντοτε αλλάζει τουλάχιστον ένα πεδίο (πχ το Time-to-Live), αλλά μπορούν να εφαρμοστούν διάφορες μέθοδοι, ώστε να επιταχυνθεί η διαδικασία.

- Source Address.
  Δείχνει τον κόμβο-αποστολέα.

- Destination Address.
  Δείχνει τον κόμβο-παραλήπτη.

- Options.
  Σχεδιάστηκε για να προσφέρει μια διέξοδο ώστε οι επόμενες εκδόσεις του πρωτοκόλλου να περιλαμβάνουν πληροφορίες που σπάνια είναι αναγκαίες. Τα προαιρετικά τμήματα έχουν μεταβλητό μήκος. Καθένα ξεκινά με κωδικό του 1 byte που καθορίζει το συγκεκριμένο είδος προαιρετικού πεδίου. Μερικά ακολουθούνται από ένα πεδίο μήκους του 1 byte και μετά από ένα ή περισσότερα byte δεδομένων. Το πεδίο αυτό συμπληρώνεται μέχρι να γίνει πολλαπλάσιο των 4 bytes. Οι επιλογές που δεν υποστηρίζονται από όλους τους δρομολογητές είναι: ασφάλεια (security), αυστηρή δρομολόγηση πηγής (strict source routing), χαλαρή δρομολόγηση πηγής (loose source routing), καταγραφή διαδρομής (record route), χρονική σφραγίδα (timestamp).

- Data. Περιέχει πληροφορίες που χρησιμοποιούνται από τα ανώτερα επίπεδα.

2.2.1.2 Διευθυνσιοδότηση IP

Κάθε host και κάθε δρομολογητής στο Internet έχει μια διεύθυνση IP που κωδικοποιεί τον αριθμό δικτύου του καθώς και τον αριθμό του host. Ο συνδυασμός τους είναι μοναδικός· δεν υπάρχει περίπτωση δύο μηχανήματα να έχουν την ίδια διεύθυνση IP. Όλες οι διευθύνσεις IP έχουν μήκος 32-bit και χρησιμοποιούνται στα πεδία Source Address και Destination Address των πακέτων IP. Οι δομές που χρησιμοποιούνται για τις διευθύνσεις IP φαίνονται στο σχήμα. Οι μηχανές που είναι συνδεδεμένες σε πολλά δίκτυα έχουν διαφορετική διεύθυνση IP ανά δίκτυο.

Οι διευθύνσεις των δικτύων, που είναι 32-bit αριθμοί, είναι συνήθως γραμμένες σε δεκαδικό συμβολισμό με τελείες (dotted decimal notation). Σε αυτή τη μορφή, καθένα από τα 4 bytes είναι γραμμένο ως δεκαδικός από το 0 μέχρι το 255. Οι τιμές 0 και -1 έχουν ειδική σημασία. Η τιμή 0 δηλώνει το ίδιο

δίκτυο ή τον ίδιο host. Η τιμή -1 χρησιμοποιείται ως διεύθυνση εκπομπής για να δηλώνει όλους τους host στο δίκτυο που υποδεικνύει.

Η IP διεύθυνση 0.0.0.0 χρησιμοποιείται από τους host κατά τη διάρκεια της εκκίνησής τους, αλλά δεν επαναχρησιμοποιείται μετά από αυτήν. Οι διευθύνσεις με αριθμό δικτύου 0 αναφέρονται στο τρέχον δίκτυο. Αυτές οι διευθύνσεις επιτρέπουν στα μηχανήματα να αναφέρονται στο δικό τους δίκτυο, χωρίς να γνωρίζουν τον αριθμό του (πρέπει όμως να γνωρίζουν την κατηγορία του ώστε να ξέρουν πόσα 0 να συμπεριλάβουν). Η διεύθυνση που αποτελείται μόνο από 1 επιτρέπει την εκπομπή στο τοπικό δίκτυο. ΟΙ διευθύνσεις με κανονικό αριθμό δικτύου και 1 στο πεδίο του αριθμού host επιτρέπουν στις μηχανές να κοινοποιούν πακέτα εκπομπής σε απομακρυσμένα δίκτυα LAN οπουδήποτε στο Internet. Τέλος όλες οι διευθύνσεις της μορφής 127. έχουν δεσμευτεί για έλεγχο με βρόχο επιστροφής (loopback). Τα πακέτα που στέλνονται σε αυτή τη διεύθυνση δεν εξάγονται στο καλώδιο. Επεξεργάζονται τοπικά και αντιμετωπίζονται ως εισερχόμενα πακέτα. Αυτό επιτρέπει την αποστολή πακέτων στο τοπικό δίκτυο χωρίς να γνωρίζει ο αποστολέας τον αριθμό του. Αυτή η ιδιότητα χρησιμοποιείται επίσης για τη διόρθωση (εκσφαλμάτωση / debugging) του λογισμικού του δικτύου.

2.2.1.3 Υποδίκτυα

Όπως έχουμε δει, όλοι οι host σ' ένα δίκτυο πρέπει να έχουν τον ίδιο αριθμό δικτύου. Καθώς τα δίκτυα μεγαλώνουν, αυτή η ιδιότητα της διευθυνσιοδότησης IP είναι πιθανό να προκαλέσει προβλήματα κυρίως όσον αφορά στη διαχείρισή τους. Κάθε φορά που εγκαθίσταται ένα νέο δίκτυο, ο διαχειριστής του πρέπει να επικοινωνήσει με το NIC (Network Infomation Center), για να πάρει νέο αριθμό δικτύου. Κατόπιν αυτός ο αριθμός πρέπει να ανακοινωθεί σε όλο τον κόσμο. Επιπλέον, η μετακίνηση ενός μηχανήματος από ένα LAN σε κάποιο άλλο απαιτεί αλλαγή της IP διεύθυνσης του, η οποία με τη σειρά της απαιτεί την αλλαγή των αρχείων της διάρθρωσης και επίσης την ανακοίνωση της νέας διεύθυνσης IP σε όλο το δίκτυο.

Η λύση αυτού του προβλήματος είναι το να επιτραπεί ο διαμελισμός του δικτύου σε πολλά μέρη για εσωτερική χρήση, με το δίκτυο όμως να συμπεριφέρεται σαν ένα και μοναδικό προς όλο τον υπόλοιπο κόσμο. Στην ορολογία του Internet αυτά τα μέρη καλούνται υποδίκτυα (subnets). Έξω από το δίκτυο ο διαχωρισμός σε επιμέρους υποδίκτυα δεν είναι ορατός· έτσι η εκχώρηση ενός νέου υποδικτύου δεν απαιτεί την επικοινωνία με το NIC, ούτε την αλλαγή εξωτερικών βάσεων δεδομένων.

Η διεύθυνση υποδικτύου (subnet address) κατασκευάζεται ως εξής: δανείζεται bits από το πεδίο του host και τα αναδεικνύει ως πεδίο υποδικτύου. Ο αριθμός των bits ποικίλλει και καθορίζεται από τη μάσκα υποδικτύου (subnet

mask) που έχει τη ίδια μορφή και τεχνική αναπαράσταση με τη διεύθυνση IP. Η μάσκα υποδικτύου βέβαια, έχει δυαδικά 1 σε όλα τα bits που καθορίζουν το δίκτυο και το υποδίκτυο και δυαδικά 0 στο πεδίο που καθορίζει το host. Τα bits της μάσκας υποδικτύου πρέπει να "έρχονται" από τα αριστερά του πεδίου host.

Ο δρομολογητής εκτελεί μια σειρά διεργασιών για να εξαγάγει τη διεύθυνση δικτύου (ή υποδικτύου). Κατ αρχήν, ανακτά το πεδίο Destination Address από το εισερχόμενο πακέτο IP και με δεδομένη τη μάσκα υποδικτύου, εκτελεί ένα λογικό ΚΑΙ, για να εξαγάγει τον αριθμό δικτύου. Αυτό έχει ως αποτέλεσμα να αφαιρεθεί το πεδίο της IP διεύθυνσης που αντιστοιχεί στον host ενώ το πεδίο του δικτύου υφίσταται ακόμα. Στη συνέχεια, ελέγχει τον αριθμό δικτύου της διεύθυνσης προορισμού και το ταιριάζει με μια εξερχόμενη διεπαφή. Τέλος, προωθεί το πλαίσιο στην IP διεύθυνση.

2.2.1.4 Πρωτόκολλο ARP

Παρόλο που κάθε μηχάνημα στο Internet έχει μία (ή περισσότερες) διευθύνσεις IP, αυτές δεν μπορούν να χρησιμοποιηθούν για αποστολή πακέτων, αφού το υλικό του στρώματος ζεύξης δεδομένων δεν μπορεί να καταλάβει τις διευθύνσεις Internet. Σήμερα η πλειοψηφία των host είναι συνδεδεμένη σε LAN μέσω μιας κάρτας που αντιλαμβάνεται μόνο διευθύνσεις LAN.

Μέσω εκπομπής ενός πακέτου ερωτήματος για μια συγκεκριμένη IP διεύθυνση, ο αποστολέας ενός πακέτου μπορεί να λάβει απάντηση μόνο από τον παραλήπτη που έχει τη συγκεκριμένη IP διεύθυνση. Η απάντηση στο ερώτημα θα είναι η διεύθυνση στρώματος MAC του παραλήπτη και έτσι ο αποστολέας μπορεί να στείλει το πακέτο. Το πρωτόκολλο μέσω του οποίου διατυπώνεται αυτή η ερώτηση είναι το ARP (Address Resolution Protocol / Πρωτόκολλο Επίλυσης Διευθύνσεων). Σχεδόν κάθε μηχανή συνδεδεμένη στο Internet το χρησιμοποιεί.

Μετά τη λήψη της φυσικής διεύθυνσης (MAC Address), οι μηχανές κατασκευάζουν έναν προσωρινό πίνακα ARP για να αποθηκεύσουν τις αντιστοιχίες IP-MAC διευθύνσεων χωρίς να εκπέμπουν ARP πακέτα, όταν θελήσουν να επικοινωνήσουν εκ νέου με μια μηχανή. Επιπρόσθετα, το RARP (Reverse Address Resolution Protocol) χρησιμοποιείται, για να αντιστοιχίσει MAC διευθύνσεις σε IP. Το RARP, που είναι το λογικό αντίστροφο του ARP, μπορεί να χρησιμοποιηθεί σε σταθμούς εργασίας χωρίς μόνιμη αποθήκευση, εφόσον στη διαδικασία της εκκίνησης αγνοούν τις διευθύνσεις IP. Το RARP βασίζεται σε κάποιον εξυπηρετητή με αποθηκευμένο τον πίνακα αντιστοιχίας MAC-IP διευθύνσεων.

2.2.1.5 Δρομολόγηση IP

Κάθε δρομολογητής σε ένα υποδίκτυο έχει έναν πίνακα που περιέχει κάποιον αριθμό διευθύνσεων IP (host) και την αντίστοιχη διεπαφή, για να φτάσουν τα πακέτα στους hosts. Επιπρόσθετα, περιέχει και έναν αριθμό από διευθύνσεις IP που αντιστοιχίζονται σε άλλους δρομολογητές σε διαφορετικά υποδίκτυα. έτσι κάθε φορά που φτάνει ένα πακέτο αναζητείται η διεύθυνση προορισμού του στον πίνακα δρομολόγησης. Αν το πακέτο προορίζεται για κάποιο απομακρυσμένο δίκτυο, προωθείται στον επόμενο δρομολογητή στη διεπαφή που καθορίζεται από τον πίνακα. Αν πρόκειται για τοπικό host στέλνεται κατευθείαν στον προορισμό του. Αν δεν είναι γνωστό το δίκτυο, το πακέτο προωθείται σε έναν προκαθορισμένο δρομολογητή που έχει εκτενέστερους πίνακες (default gateway). Ο αλγόριθμος αυτός έχει ως αποτέλεσμα κάθε δρομολογητής να καταγράφει μόνο τα άλλα δίκτυα και τους τοπικούς host και όχι ζευγάρια δίκτυο - host με αποτέλεσμα τη μείωση του μεγέθους του πίνακα δρομολόγησης.

Όταν έχουμε υποδίκτυα οι πίνακες δρομολόγησης τροποποιούνται με την προσθήκη καταχωρήσεων της μορφής: (αυτό το δίκτυο, υποδίκτυο, 0), (αυτό το δίκτυο, αυτό το υποδίκτυο, host). Κατά αυτόν τον τρόπο, κάθε δρομολογητής του υποδικτύου Κ γνωρίζει το πώς να πάει σε όλα τα υπόλοιπα υποδίκτυα και επίσης το πώς να πάει σε οποιονδήποτε host του υποδικτύου Κ. Δεν είναι απαραίτητο να γνωρίζει λεπτομέρειες για τους host των άλλων υποδικτύων. Στην πραγματικότητα, η μόνη μετατροπή που χρειάζεται είναι ο κάθε δρομολογητής να εκτελέσει το λογικό ΚΑΙ με τη μάσκα υποδικτύου του δικτύου, ώστε να απαλλαγεί από τον αριθμό του host και να ψάξει στους πίνακές του για τη διεύθυνση που θα προκύψει (αφού καθορίσει την κατηγορία του δικτύου).

Οι συσκευές δρομολόγησης στο Internet παραδοσιακά λέγονταν πύλες (gateways). Στη σημερινή ορολογία βέβαια, ο όρος πύλη αναφέρεται συγκεκριμένα σε μια συσκευή που εφαρμόζει τη μετάφραση πρωτοκόλλου στρώματος εφαρμογής μεταξύ συσκευών. Εσωτερικές πύλες αναφέρονται σε συσκευές που εκτελούν αυτές τις εφαρμογές πρωτοκόλλων μεταξύ μηχανών ή δικτύων κάτω από τον ίδιο διαχειριστικό έλεγχο ή επίβλεψη, όπως ένα εσωτερικό δίκτυο μιας εταιρίας. Ένα τέτοιο δίκτυο είναι ένα Αυτόνομο Σύστημα. Εξωτερικές πύλες εκτελούν αυτές τις εφαρμογές μεταξύ διαφορετικών, ανεξάρτητων δικτύων.

Οι δρομολογητές στο Internet είναι ιεραρχικά οργανωμένοι. Αυτοί που χρησιμοποιούνται για ανταλλαγή πληροφοριών στο ίδιο αυτόνομο σύστημα λέγονται εσωτερικοί δρομολογητές, και χρησιμοποιούν διάφορα πρωτόκολλα εσωτερικών πυλών IGP (Interior Gateway Protocols), για να επιτύχουν αυτό το στόχο. Το Routing Information Protocol (RIP) είναι ένα παράδειγμα ενός IGP.

Οι δρομολογητές που μεταφέρουν πληροφορίες μεταξύ αυτόνομων συστημάτων λέγονται εξωτερικοί δρομολογητές και χρησιμοποιούν ένα πρωτόκολλο

εξωτερικών πυλών, όπως το BGP (Border Gateway Protocol) για την ανταλλαγή πληροφοριών.

Τα πρωτόκολλα δρομολόγησης IP είναι δυναμικά και καλούν τα μονοπάτια (routes) να υπολογίζονται σε τακτά χρονικά διαστήματα από κάποιο λογισμικό σε μια συσκευή δρομολόγησης. Αυτό αντιτίθεται στην στατική δρομολόγηση όπου οι δρομολογητές εγκαθίστανται από τους διαχειριστές δικτύων και δεν αλλάζουν, εκτός αν οι διαχειριστές το επιθυμούν.

΄Ενας πίνακας δρομολόγησης IP, που αποτελείται από ζεύγη destination address / next hop, χρησιμοποιείται, για να ενεργοποιήσει τη δυναμική δρομολόγηση. Η δρομολόγηση IP ορίζει ότι τα δεδομενογραφήματα IP θα ταξιδεύουν κατά μήκος των δικτύων με ένα βήμα τη φορά. Σε μια κατάσταση της διαδρομής, δεν είναι γνωστό ολόκληρο το μονοπάτι. Αντ' αυτού, σε κάθε "στάση" ο επόμενος προορισμός υπολογίζεται ταιριάζοντας τη διεύθυνση προορισμού μέσα στο δεδομενογράφημα με μια εγγραφή στον πίνακα δρομολόγησης του κόμβου.

Η συνεισφορά κάθε κόμβου στη διαδικασία δρομολόγησης περιορίζεται στην προώθηση πακέτων βασισμένη σε εσωτερικές πληροφορίες. Οι κόμβοι δεν παρακολουθούν τα πακέτα για το αν θα φτάσουν στον τελικό προορισμό τους, ούτε το IP προσφέρει έναν τρόπο ενημέρωσης του αποστολέα σε περιπτώσεις ανωμαλιών δρομολόγησης. Ο έλεγχος αυτός αφήνεται σε ένα άλλο πρωτόκολλο, το ICMP.

Όταν συμβεί κάτι αναπάντεχο, το γεγονός αναφέρεται από το πρωτόκολλο μηνυμάτων ελέγχου του Internet, το ICMP (Internet Control Message Protocol), που χρησιμοποιείται επίσης για τη διεξαγωγή δοκιμών στο Internet. Ορίζονται περίπου 12 τύποι μηνυμάτων ICMP. Οι περισσότερο σημαντικοί φαίνονται στον πίνακα. Κάθε τύπος μηνύματος ICMP ενσωματώνεται σε ένα πακέτο IP.

Τύπος μηνύματος	**Περιγραφή**
Destination Unreachable	Το πακέτο δεν μπορούσε να παραδοθεί
Time exceeded	Το πεδίο διάρκειας ζωής έφτασε το 0
Parameter Problem	Άκυρο πεδίο επικεφαλίδας
Source quench	Πακέτο Φραγής
Redirect	Ανακατασκεύασε τον πίνακα δρομολόγησης
Echo Request	Ρώτα μια μηχανή αν είναι συνδεδεμένη στο δίκτυο
Echo Reply	Θετική απάντηση στην παραπάνω ερώτηση
Timestamp Request	Όπως το Echo Request αλλά με χρονική σφραγίδα
Timestamp Reply	Όπως το Echo Reply αλλά με χρονική σφραγίδα

Πίνακας 2.1: Μηνύματα ICMP

2.2.2 Transmission Control Protocol

Το πρωτόκολλο TCP εξασφαλίζει αξιόπιστη επικοινωνία σε περιβάλλον IP. Αντιστοιχίζεται στο στρώμα μεταφοράς (επίπεδο 4) του μοντέλου αναφοράς OSI. Μερικές από τις υπηρεσίες που προσφέρει είναι οι ακόλουθες: μεταφορά ρεύματος δεδομένων, αξιοπιστία, αποτελεσματικό έλεγχο ροής, λειτουργία διπλής κατεύθυνσης (full-duplex operation) και πολύπλεξη.

Με τη μεταφορά ρεύματος δεδομένων, το TCP αποδίδει ένα ρεύμα από bytes που αναγνωρίζονται με αύξοντες αριθμούς (μια ακολουθία αριθμών). Αυτή η υπηρεσία ευνοεί τις εφαρμογές, καθώς δε χρειάζεται να τεμαχίσουν τα δεδομένα σε blocks πριν τα παραδώσουν στο TCP. Αντιθέτως το TCP ομαδοποιεί τα bytes σε τμήματα και τα περνά με τη σειρά του στο IP.

Κάθε byte σε μια σύνδεση TCP έχει το δικό του αύξοντα αριθμό των 32-bit. Για έναν host που μεταδίδει σε πλήρη ταχύτητα σε δίκτυο LAN των 10Mbps, οι αύξοντες αριθμοί, θεωρητικά, θα αναδιπλωθούν σε μία ώρα περίπου, αλλά στην πράξη θα πάρει πολύ περισσότερο. Οι αύξοντες αριθμοί χρησιμοποιούνται τόσο για τις επαληθεύσεις, όσο και το μηχανισμό παραθύρου, ο οποίος χρησιμοποιεί ξεχωριστά πεδία επικεφαλίδας των 32-bit.

Οι αποστέλλουσες και λαμβάνουσες ποσότητες TCP ανταλλάσσουν δεδομένα με τη μορφή τεμαχίων. Το τεμάχιο (segment) αποτελείται από μια σταθερή επικεφαλίδα των 20 byte (συν ένα προαιρετικό κομμάτι), ακολουθούμενη από 0 ή περισσότερα bytes δεδομένων. Το λογισμικό του TCP αποφασίζει για το μέγεθος των τεμαχίων. Μπορεί να συσσωρεύσει δεδομένα από πολλές εντολές εγγραφής (write) σε ένα τεμάχιο ή να διαχωρίσει τα δεδομένα μίας εγγραφής σε πολλαπλά τεμάχια. Δύο περιορισμοί υπαγορεύουν το μέγεθος του τεμαχίου. Πρώτον, κάθε τεμάχιο, συμπεριλαμβανομένης της επικεφαλίδας TCP, πρέπει να προχωρά στο ωφέλιμο φορτίο IP των 65.535 bytes. Δεύτερον, κάθε δίκτυο έχει μία μέγιστη μονάδα μεταφοράς MTU (Maximum Transfer Unit) και κάθε τεμάχιο πρέπει να χωράει στην MTU. Στην πράξη, η MTU είναι εν γένει λίγες χιλιάδες bytes και έτσι ορίζει το άνω όριο του μεγέθους τεμαχίου. Εάν ένα τεμάχιο περάσει μέσα από μία σειρά δικτύων χωρίς να διασπαστεί και κατόπιν συναντήσει ένα δίκτυο που έχει μικρότερη MTU από το τεμάχιο, ο δρομολογητής που βρίσκεται στο όριο, κομματιάζει το τεμάχιο σε δύο ή περισσότερα τεμάχια.

΄Ενα τεμάχιο που είναι πολύ μεγάλο για το δίκτυο το οποίο πρέπει να διασχίσει κομματιάζεται σε πολλά τεμάχια από το δρομολογητή. Το κάθε νέο τεμάχιο αποκτά τις δικές του επικεφαλίδες TCP και IP και έτσι ο τεμαχισμός από τους δρομολογητές αυξάνει τη συνολική επιβάρυνση (αφού κάθε πρόσθετο τεμάχιο προσθέτει 40 επιπλέον bytes πληροφοριών επικεφαλίδας).

Το βασικό πρωτόκολλο που χρησιμοποιείται από τις οντότητες TCP είναι το πρωτόκολλο του ολισθαίνοντος παραθύρου (sliding window protocol). Όταν

ο αποστολέας μεταδίδει ένα τεμάχιο, εκκινεί επίσης ένα χρονομετρητή. Όταν το τεμάχιο φτάσει στον προορισμό του, η λαμβάνουσα οντότητα TCP επιστρέφει ένα τεμάχιο (με δεδομένα, άν αυτά υπάρχουν, αλλιώς χωρίς δεδομένα) που μεταφέρει αριθμό επαλήθευσης (acknowlegment number) ίσο με τον επόμενο αύξοντα αριθμό που περιμένει να δεχτεί. Αν το χρονόμετρο του αποστολέα λήξει πριν λάβει την επαλήθευση, τότε ο αποστολέας μεταδίδει το τεμάχιο ξανα. Αυτή η διαδικασία αυξάνει την αξιοπιστία του πρωτοκόλλου TCP, αφού σε περίπτωση απόρριψης ενός πακέτου, αυτό θα επαναμεταδοθεί.

2.2.2.1 Συνδέσεις TCP

Για αξιόπιστες υπηρεσίες μεταφοράς οι TCP hosts πρέπει να εγκαταστήσουν μια σύνδεση επικοινωνίας μεταξύ τους. Αυτή η σύνδεση επιτυγχάνεται με έναν μηχανισμό "τριμερούς χειραψίας". Ο μηχανισμός αυτός, συγχρονίζει τα δύο άκρα μιας σύνδεσης επιβάλλοντας και στις δύο πλευρές να συμφωνήσουν στους αρχικούς αύξοντες αριθμούς.

Η "τριμερής χειραψία", εγγυάται ότι και οι δύο πλευρές είναι έτοιμες να μεταδώσουν δεδομένα καθώς και ότι η απέναντι πλευρά είναι επίσης έτοιμη να μεταδώσει. Αυτό είναι απαραίτητο για να μη μεταδίδονται ή επαναμεταδίδονται πακέτα κατά τη διάρκεια εγκατάστασης σύνδεσης ή μετά τον τερματισμό της σύνδεσης.

Κάθε host διαλέγει τυχαία έναν αύξοντα αριθμό που χρησιμοποιείται για να εντοπίσει bytes μέσα στο ρεύμα δεδομένων που στέλνει ή λαμβάνει. Έπειτα, η "τριμερής χειραψία" υλοποιείται με τον ακόλουθο τρόπο:

Το πρώτο host εκκινεί μια σύνδεση στέλνοντας ένα πακέτο με τον αρχικό αύξοντα αριθμό (X) και ένα SYN bit το οποίο δείχνει μια αίτηση για σύνδεση. Το δεύτερο host λαμβάνει το SYN, καταγράφει τον αύξοντα αριθμό (X) και απαντά επαληθεύοντας το SYN (με ένα ACK = X + 1). Επίσης, στέλνει και το δικό του αύξοντα αριθμό (SEQ = Y). Η τεχνική αυτή λέγεται forward acknowledgment. Το πρώτο host τότε, επαληθεύει όλα τα bytes που έχει λάβει από το δεύτερο host, με ένα forward acknowledgment που δείχνει το επόμενο byte που το πρώτο host περιμένει να λάβει (ACK = Y + 1). Η μεταφορά δεδομένων μπορεί τώρα να αρχίσει.

2.2.2.2 Δομή πακέτου TCP

Το πακέτο TCP αποτελείται από τα εξής πεδία:

- Source Port / Destination Port. Είναι οι ταυτότητες των τοπικών ακραίων σημείων της σύνδεσης. Οι αριθμοί των υποδοχών αφετηρίας και προορισμού από κοινού προσδιορίζουν μονοσήμαντα τη σύνδεση.

- Sequence Number / Acknowledgment Number. Εκτελούν τις συνήθεις λειτουργίες τους. Και τα δύο πεδία έχουν μέγεθος 32-bit επειδή στο συρμό TCP κάθε byte δεδομένων είναι αριθμημένο.

- Data Offset. Καθορίζει πόσες λέξεις των 32-bit περιέχονται στην επικεφαλίδα TCP.

- Reserved. Ένα πεδίο 6 bit που δε χρησιμοποιείται.

- Flags. Περιέχει μια σειρά από πληροφορίες ελέγχου συμπεριλαμβανομένων και των SYN και ACK bits που χρησιμοποιούνται για την εκγατάσταση σύνδεσης καθώς και του FIN για τον τερματισμό της σύνδεσης.

- Window. Καθορίζει το πόσα bytes μπορούν να σταλούν ξεκινώντας από το byte για το οποίο έγινε η επαλήθευση.

- Checksum. Ελέγχει την επικεφαλίδα, τα δεδομένα και τη νοητή ψευδοεπικεφαλίδα.

- Urgent Pointer. Χρησιμοποιείται για να δείξει την απόσταση (offset) σε bytes από τον τρέχοντα αύξοντα αριθμό, στην οποία βρίσκονται τα επείγοντα δεδομένα.

- Options. Το πεδίο αυτό σχεδιάστηκε ως ένας τρόπος να προστεθούν επιπλέον δυνατότητες που δεν καλύπτονται από την κανονική επικεφαλίδα. Η πιο σπουδαία επιλογή είναι αυτή που επιτρέπει σε κάθε host να ορίσει το μέγιστο ωφέλιμο φορτίο TCP που είναι διατεθειμένος να δεχτεί.

- Data. Περιέχει προαιρετικά δεδομένα ανάλογα με το πεδίο Options.

2.3 Myrinet

Εισαγωγή

Η ραγδαία ανάπτυξη τοπικών δικτύων υψηλής ταχύτητας μετέφερε το bottleneck της επικοινωνίας με τοπικά δίκτυα από το περιορισμένο εύρος ζώνης των υλικών δικτύωσης φυσικού στρώματος, στο μονοπάτι λογισμικού που ακολουθούν τα πακέτα / μηνύματα στα άκρα λήψης και αποστολής. Πιο συκγκριμένα, σε μια τυπική αρχιτεκτονική UNIX δικτύου το μονοπάτι των μηνυμάτων μέσα από τον πυρήνα εμπεριέχει αρκετές αντιγραφές και περνά από πολυάριθμα στρώματα αφαίρεσης μεταξύ του οδηγού της συσκευής δικτύωσης

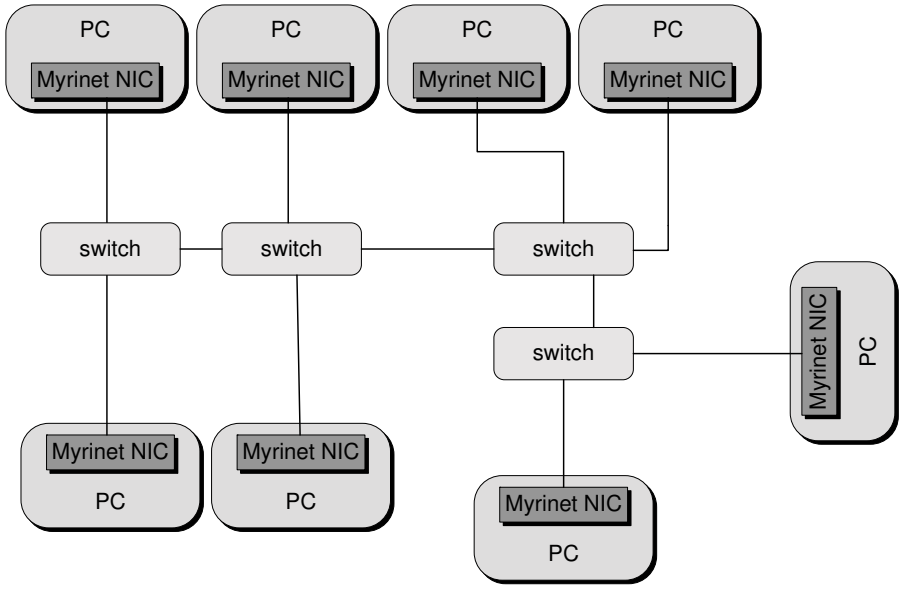
και της εφαρμογής χώρου χρήστη. Σαν αποτέλεσμα, η επιβάρυνση που προστίθεται περιορίζει το μέγιστο εύρος ζώνης επικοινωνίας και προκαλεί υψηλό latency στα μηνύματα που μεταφέρονται από άκρο σε άκρο. Σε ενδεχόμενη αναβάθμιση από δίκτυο Ethernet σε δίκτυο υψηλότερου εύρους ζώνης, ο χρήστης δε θα παρατηρήσει μια αντίστοιχη αύξηση της ταχύτητας επικοινωνίας με την αύξηση του προσφερόμενου εύρους ζώνης. Μια τέτοια λύση (αναβάθμισης του υλικού δικτύωσης) φαίνεται να προκαλεί περισσότερο την προσοχή της αγοράς, αφού είναι κοινός τόπος να ενδιφέρει περισσότερο η ταχύτητα μετάδοσης μεγάλων ποσών δεδομένων παρά το latency ανά μήνυμα / πακέτο. Ενώ αυτό το γεγονός ισχύει (για παράδειγμα σε εφαρμογές αναπαραγωγής video), οι περισσότερες εφαρμογές δικτύου χρησιμοποιούν μικρού μεγέθους μηνύματα και βασίζονται σημαντικά σε γρήγορη έκδοση και εξυπηρέτηση αιτήσεων. Η όλο και αυξανόμενη χρήση κατανεμημένης μοιραζόμενης μνήμης, απομακρυσμένων κλήσεων διαδικασιών (rpc), καθως και κατανεμημένων συνεργαζόμενων γρήγορων («κρυφών») μνημών, θα αυξήσει τη σπουδαιότητα του χαμηλού latency και του υψηλού εύρους ζώνης στο βαθμό του latency.

Μια από τις πιο σπουδαίες τεχνικές για βελτίωση τόσο της απόδοσης όσο και της ευελιξίας του στρώματος δικτύου σε μηχανές σταθμούς εργασίας είναι η επανατοποθέτηση μερών της επεξεργασίας των πρωτοκόλων επικοινωνίας στο χώρο χρήστη.

Το Myrinet[BCF+95] είναι ένα δίκτυο επικοινωνίας μεταγωγής πακέτων υψηλής απόδοσης που χρησιμοποιείται ευρέως για τη σύνδεση υπολογιστικών συστημάτων σε συστοιχίες. Οι συστοιχίες υπολογιστών είναι ένα μοντέλο χαμηλού κόστους που έχουν υψηλή απόδοση, μοιράζοντας ζητούμενους υπολογισμούς σε ένα σύνολο συστημάτων χαμηλού κόστους και υψηλής διαθεσιμότητας (high availability), εντοπίζοντας σφάλματα και αποκόπτοντας εσφαλμένα μονοπάτια επικοινωνίας.

Παρόλο που τα δίκτυα υψηλών ταχυτήτων (Gigabit) είναι σήμερα ευρέως διαδεδομένα σε συστοιχίες υπολογιστών, οι επιδόσεις των εφαρμογών δεν είναι τόσο καλές όσο θα περίμενε κανείς. Το γεγονός αυτό οφείλεται τόσο στην υλοποίηση παραδοσιακών και πολύπλοκων πρωτοκόλλων επικοινωνίας (π.χ. TCP/IP) όσο και στην υλοποίηση των παραπάνω πρωτοκόλλων μέσα στο λειτουργικό σύστημα. Τα παραδοσιακά πρωτόκολλα επικοινωνίας αυξάνουν το software overhead επιβάλλοντας μια πολύπλοκη «στοίβα». Από την άλλη πλευρά, η παρέμβαση του λειτουργικού συστήματος αυξάνει ακόμα περισσότερο το overhead και την καθυστέρηση (latency) με αντιγραφές σε απομονωτές του πυρήνα (kernel buffers), κλήσεις συστήματος (system calls), μεταγωγές περιεχομένου (context switching) και χειρισμό διακοπών (interrupt handling). Η μεσολάβηση λοιπόν του λειτουργικού συστήματος οδηγεί σε χαμηλό bandwidth, μεγάλο latency, και επιπλέον σε δύσκολα επεκτάσιμες λύσεις.

Το Myrinet είναι μια μορφή δικτύωσης βασισμένη στην αρχιτεκτονική MPP



Σχήμα 2.2: Μια τυπική τοπολογία σε δίκτυο Myrinet

(Massive Parallel Processors) για επικοινωνία και μεταγωγή πακέτων. Μπορεί κανείς να το φανταστεί σαν ένα δίκτυο MPP που μπορεί να έχει έκταση ενός πανεπιστημιακού συγκροτήματος, παρά σαν ένα τηλεπικοινωνιακό δίκτυο ευρείας περιοχής. Τα βασικά του σχεδιαστικά χαρακτηριστικά είναι η χαμηλή καθυστέρηση (low-latency), ο έλεγχος ροής και σφαλμάτων (flow control), το packet framing, καθώς και η απευθείας σύνδεση εφαρμογών χώρου χρήστη με το δίκτυο. Όλα αυτά, συντελούν στη βελτίωση της απόδοσης του κατανεμημένου υπολογισμού, της μεταφοράς εικόνων και γενικότερα όλων των εφαρμογών που απαιτούν επικοινωνία χαμηλής καθυστέρησης.

Η καινοτομία του Myrinet αφορά στον τρόπο επικοινωνίας˙ το πέρασμα μηνυμάτων μεταξύ μικρών αλλά και αυτόνομων κόμβων. Τα δεδομένα, όπως και στα μέχρι τώρα δίκτυα, μεταφέρονται με πακέτα (packets). Κάθε κόμβος μπορεί να στείλει ένα πακέτο σε έναν άλλο κόμβο. Το πακέτο αυτό περιέχει μια ακολουθία από bytes με επικεφαλίδα που καθορίζει τη δρομολόγηση. Τα κυκλώματα δρομολόγησης εξετάζουν την επικεφαλίδα για να οδηγήσουν τα πακέτα μέσα στο δίκτυο. ΄Ενα κομμάτι των δεδομένων που ακολουθεί την επικεφαλίδα χωρίς σταθερό μήκος, περιέχει τα bytes προς μεταφορά. Η ″ουρά″ (trailer ή tail) του πακέτου το ″τερματίζει″ και περιέχει έναν ″αθροιστικό

έλεγχο" (checksum).

Το μεταβλητό μέγεθος των πακέτων, τα όρια μεγέθους των απομονωτών καθώς και η "δικαιοσύνη" με την οποία τα αντιμετωπίζουν οι "δρομολογητές" στο δίκτυο, έχει ως αποτέλεσμα να περιορίζει το μέγεθος των πακέτων. Αν ένα μήνυμα λοιπόν, δεν μπορεί να περιληφθεί σε ένα πακέτο, ο αποστολέας το τεμαχίζει και επανενώνεται στον προορισμό.

Σε αντίθεση με τα συμβατικά τοπικά δίκτυα, τα χαρακτηριστικά που κάνουν την επικοινωνία με μηνύματα να υπερέχει, είναι οι εξαιρετικά υψηλές ταχύτητες μεταφοράς, τα πολύ χαμηλά ποσοστά σφαλμάτων και ο έλεγχος ροής σε κάθε σύνδεσμο επικοινωνίας.

- Υψηλές ταχύτητες
  Οι ταχύτητες των καναλιών μεταφοράς δεδομένων σήμερα ποικίλλουν από μερικές εκατοντάδες μέχρι κάποιες χιλιάδες mbps. Τα δίκτυα επικοινωνίας με μηνύματα, οργανώνουν τα αυτόνομα κανάλια σε "συνδέσμους" διπλής κατεύθυνσης (full-duplex pairs ή links) σε αντίθεση με τα συμβατικά δίκτυα (π.χ. Ethernet) που μπορούν να μεταφέρουν πακέτα μονομερώς κάθε φορά.

- Τοπολογία
  ΄Ενα σημαντικό πλεονέκτημα των δικτύων επικοινωνίας με μηνύματα είναι και η τοπολογία τους. Τυπικά κυκλώματα δρομολόγησης συνδεδεμένα με "συνδέσμους" σε μια μαθηματικά κανονικοποιημένη τοπολογία συνθέτουν το δίκτυο. Μετά τους υπερκύβους (hypercubes), υιοθετήθηκαν τοπολογίες χαμηλών διαστάσεων όπως ένας δισδιάστατος βρόχος (2-D mesh). Στο σχήμα, τα πακέτα που εισάγονται από τα αριστερά και εξάγονται από τα δεξιά των "δρομολογητών" είναι τα πακέτα επικοινωνίας. Μόλις ένας κόμβος αποκωδικοποιήσει την επικεφαλίδα του πακέτου, το προωθεί στον επόμενο κόμβο - προορισμό του, αν το κανάλι είναι ελεύθερο. Σε αντίθετη περίπτωση, το πακέτο μπλοκάρεται, μέχρι να ελευθερωθεί ο επόμενος κόμβος.

  Αυτή η μέθοδος δρομολόγησης αφαιρεί πιθανότητες αδιεξόδου λόγω ατέρμονων κυκλικών εξαρτήσεων. Αντίθετα με τις συμβατικές δικτυακές τεχνολογίες, όπου τα δεδομένα μοιράζονται το μοναδικό φυσικό μέσο δρομολόγησης, ο δισδιάστατος βρόχος είναι επεκτάσιμος. Η συνολική χωρητικότητα (network capacity) αυξάνει με τον αριθμό των κόμβων επειδή πολλά πακέτα μπορούν να βρίσκονται στο βρόχο ταυτόχρονα, ακολουθώντας διαφορετικά μονοπάτια.

- ΄Ελεγχος σφαλμάτων

Όσο τα δίκτυα επικοινωνίας με μηνύματα λειτουργούν βασισμένα στην τεχνολογία ενός αντίστοιχου υπολογιστή πολλών επεξεργαστικών μονάδων που επικοινωνούν με μηνύματα, τα λάθη στα bit ή σε χαμένα πακέτα είναι εξαιρετικά σπάνια. Ακόμα πιο σπάνια είναι και τα σφάλματα στην επικοινωνία που δεν ανιχνεύονται. Η υψηλή αξιοπιστία που προσφέρουν αυτού του είδους τα δίκτυα σε αντίθεση με την κλασική υπόθεση ότι τα συμβατικά δίκτυα είναι αναξιόπιστα, έχει πολλές συνέπειες (πολλαπλή σημασία). Αν το λογισμικό ενός υπολογιστικού συστήματος που λειτουργεί με μηνύματα μπορούσε να υποθέσει σφάλματα επικοινωνίας στο φυσικό επίπεδο, τότε, θα χρειαζόταν περισσότερο πολύπλοκα επικοινωνιακά πρωτόκολλα προκειμένου να είναι η επικοινωνία από άκρο σε άκρο αξιόπιστη. Επίσης, αυτά τα πρωτόκολλα θα χρειάζονταν επιπρόσθετο χώρο, για να αποθηκεύονται προσωρινά τα σταλθέντα πακέτα. Σε τέτοιου ρυθμού επικοινωνία (MPP communication rate) τέτοια πρωτόκολλα θα προσέθεταν σημαντικό overhead για το κάθε πακέτο.

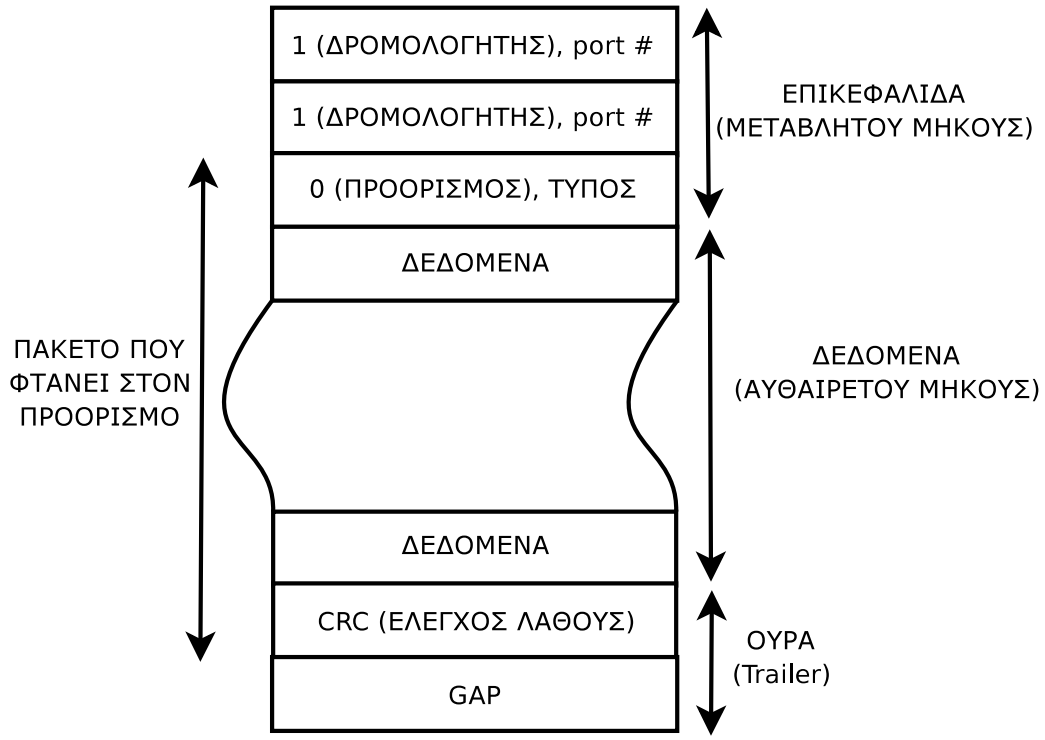
- ΄Εξυπνη δρομολόγηση

Σ' ένα περιβάλλον αξιόπιστης επικοινωνίας, τα κυκλώματα δρομολόγησης χρησιμοποιούν έναν "επιθετικό" τρόπο οδήγησης των πακέτων στο δίκτυο. Τυπικοί "store-and-forward" απομονωτές αποθηκεύουν ολόκληρο το πακέτο και ελέγχουν το checksum σε κάθε ενδιάμεσο κόμβο, πριν στείλουν το πακέτο στο απαραίτητο κανάλι. Σε κάθε περίπτωση, ο κόμβος αποκωδικοποιεί την επικεφαλίδα και αμέσως προωθεί το πακέτο στον προορισμό του. Αν το κανάλι προορισμού είναι κατειλημμένο, τότε ο κόμβος το κρατά αποθηκευμένο και το στέλνει σε πρώτη ευκαιρία.

Αυτή η έξυπνη δρομολόγηση ονομάζεται "cut-through routing".

- ΄Ελεγχος ροής δεδομένων

Άν το απαιτούμενο εξερχόμενο κανάλι είναι ήδη σε λειτουργία, ένα πακέτο "store-and-forward" πρέπει να μείνει στην ουρά του κυκλώματος δρομολόγησης ή του κόμβου, εάν αυτό έχει διαθέσιμη μνήμη. Αν το κανάλι είναι κατειλημμένο, τότε το κύκλωμα δρομολόγησης έχει δυνατότητα να το "μπλοκάρει" έτσι, ώστε να μη χρειάζεται "packet-buffering". Αυτό βέβαια έχει ως προαπαιτούμενο κάθε "σύνδεσμος" να διαθέτει έλεγχο ροής.



Σχήμα 2.3: Δομή πακέτου Myrinet

2.3.1 Δομή πακέτου

Όταν ένα πακέτο φτάνει στον δρομολογητή, το πρώτο byte της επικεφαλίδας καθορίζει την πόρτα εξόδου πριν διαγραφεί. Όταν ένα πακέτο φτάνει σ' έναν κόμβο, το byte που έχει μείνει πρώτο στην επικεφαλίδα, αναγνωρίζει τον τύπο του πακέτου. Το πλέον σημαντικό bit του κάθε byte της επικεφαλίδας, διαχωρίζει τα ″προς δρομολογητές″ και τα ″προς κόμβο″ bytes. Η πληροφορία αυτή, θα μπορούσε να είναι πλεονάζουσα. Αυτό θα ίσχυε, αν όλα τα πακέτα ″ήξεραν″ την τοπολογία του δικτύου. Όμως, αυτή η τεχνική βοηθά στην εποπτεία λαθών που οδηγούν σε εσφαλμένη δρομολόγηση, σε περιπτώσεις όπου ένα πακέτο με πρώτο byte ″προς κόμβο″ φτάνει σε δρομολογητή, ή το αντίστροφο, ένα πακέτο με πρώτο byte ″προς δρομολογητή″ φτάνει σε κόμβο.

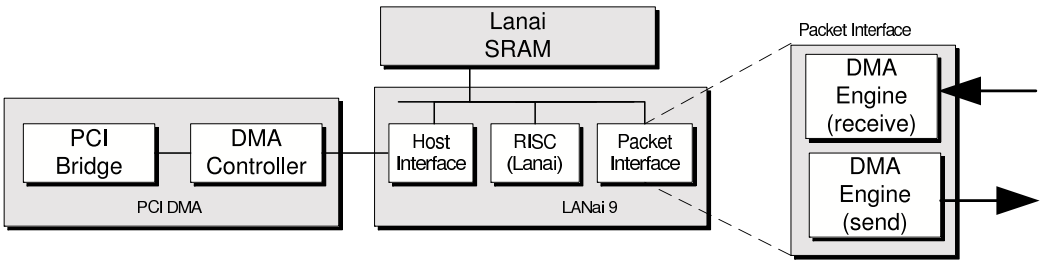
Το κομμάτι των δεδομένων του πακέτου είναι αυθαίρετου μήκους. Έτσι, μπορεί να περιέχει οποιονδήποτε τύπο πακέτου όπως π.χ. ένα πακέτο IP χωρίς κάποιο επίπεδο προσαρμογής (adaption layer).

Η «ουρά» του πακέτου (trailer) αποτελείται από έναν 8-bit Cyclic-Redundancy-Check (CRC) χαρακτήρα που υπολογίζεται με βάση ολόκληρο το πακέτο. Επειδή η επικεφαλίδα μεταβάλλεται σε κάθε ″hop″, ο χαρακτήρας αυτός υπο-

λογίζεται εκ νέου σε κάθε "σύνδεσμο". Αν ο CRC είναι εσφαλμένος όταν εισέρχεται σε έναν δρομολογητή, τα λανθασμένα bits στην έξοδό του απ' αυτόν, θα είναι ακριβώς τα ίδια. Έτσι στον προορισμό, μπορεί εύκολα να βρεθεί ένα λάθος στη δρομολόγηση ή σε κάποιο σύνδεσμο.

2.3.2 Διεπαφές Δικτύου

Η διεπαφή δικτύου του Myrinet βασίζεται σε ένα μικροεπεξεργαστή, το LANai. Προσφέρει μια υψηλής απόδοσης διεπαφή μεταξύ ενός υβριδικού διαδρόμου (E-BUS) και ενός συνδέσμου Myrinet.



Σχήμα 2.4: Η κάρτα δικτύου Myrinet

Η στατική RAM αποθηκεύει το MCP (Myrinet Control Program) και τα πακέτα. Σε μια περίοδο (clock cycle) η πρόσβαση σ' αυτή τη μνήμη γίνεται δύο φορές, μία από το διάδρομο (E-bus) και μία είτε από το LANai είτε από τη διεπαφή των πακέτων. Ακριβώς επειδή η πρόσβαση δεν γίνεται χρονικά αυθαίρετα, το LANai φαίνεται από το διάδρομο σαν σύγχρονη μνήμη και μπορεί να δρα σαν σκλάβος διαδρόμου (bus-slave) στην πλειοψηφία της 32-bit μνήμης ή των διαδρόμων E/E. Επιπρόσθετα, μπορεί να συμπεριφέρεται σαν αφέντης διαδρόμου (bus-master) για να μεταφέρει block δεδομένων από τον E-bus στην SRAM, όταν χρησιμοποιεί τη μηχανή DMA για διευθύνσεις. Η τελευταία, χρησιμοποιείται επίσης στον υπολογισμό των checksums των δεδομένων που μεταφέρει.

2.3.3 Δρομολόγηση

Το Myrinet χρησιμοποιεί δρομολόγηση πηγής, για να μεταφέρει πακέτα μεταξύ των κόμβων. Με αυτή την τεχνική ο κόμβος - πηγή, υπολογίζει το μονοπάτι που το πακέτο πρέπει να ακολουθήσει, για να φτάσει στον προορισμό του και το αποθηκεύει στην επικεφαλίδα του. Έτσι, κάθε πακέτο περιέχει μια ταξινομημένη λίστα από αναγνωριστικά των "συνδέσμων" εξόδου, που

χρησιμοποιούνται από κάθε ενδιάμεσο δρομολογητή, για να το οδηγήσουν σωστά. Το πρώτο αναγνωριστικό συνδέσμου αντιστοιχεί σ' αυτό που θα χρησιμοποιήσει ο πρώτος δρομολογητής, το δεύτερο σ' αυτό που θα χρησιμοποιήσει ο δεύτερος κοκ. Το κάθε αναγνωριστικό αφαιρείται από την επικεφαλίδα, αφού χρησιμοποιηθεί. Προκειμένου να υφίστανται μονοπάτια μεταξύ πηγής και πιθανού προορισμού, κάθε δίκτυο πρέπει να έχει την αναπαράσταση της τοπολογίας του. Τα μονοπάτια δημιουργούνται πριν από την αποστολή όλων των πακέτων κατά τη διαδικασία της αρχικοποίησης. Επιπρόσθετα, κάθε προσαρμογέας δικτύου ελέγχει για αλλαγές στην τοπολογία (αποσύνδεση κόμβων, σφάλματα, εκκίνηση νέων κόμβων κλπ) έτσι, ώστε να διατηρείται ο πίνακας δρομολόγησης ενήμερος.

2.3.4 Πρωτόκολλο διεπαφής δικτύου / Το λογισμικό GM

Για την αποφυγή του κόστους των κλήσεων πυρήνα για κάθε πρόσβαση στο δίκτυο, το βασικό πρωτόκολλο αντιστοιχίζει ολόκληρη τη μνήμη της διεπαφής δικτύου στο χώρο χρήστη. Οι διεργασίες χρήστη γράφουν τις αιτήσεις για αποστολή κατευθείαν στη μνήμη αυτή χωρίς να παρεμβάλλεται το λειτουργικό σύστημα. Το βασικό πρωτόκολλο δεν προσφέρει προστασία μνήμης, γι' αυτό και η διεπαφή δεν μπορεί να χρησιμοποιηθεί από πολλές διεργασίες.

Οι διεργασίες χρήστη προκαλούν (εκκινούν) μια απλή "πρωτόγονη" αποστολή, για να στείλουν ένα πακέτο δεδομένων.

Οι μεταφορές δεδομένων έχουν ένα σημαντικό αντίκτυπο στην καθυστέρηση και τη συνολική ρυθμαπόδοση που έχει ένα πρωτόκολλο, οπότε, για να επιτύχει κανείς υψηλή απόδοση, είναι βασικό να τις βελτιώσει.

Το GM[Myr03] είναι ένα σύστημα επικοινωνίας με μηνύματα για το Myrinet. Οι στόχοι σχεδίασης του GM περιλαμβάνουν χαμηλό CPU overhead, επεκτασιμότητα, χαμηλή καθυστέρηση και υψηλό εύρος ζώνης. Επιπλέον, Το GM έχει κάποια χαρακτηριστικά που το διαφοροποιούν από παρόμοια συστήματα:

- Το overhead που προκαλεί είναι της τάξης του ενός microsecond ανά πακέτο σε όλες τις αρχιτεκτονικές
- Μπορεί να διαθέτει σε διεργασίες ταυτόχρονη πρόσβαση στη διεπαφή δικτύου με προστασία μνήμης και χωρίς να παρεμβάλλεται το λειτουργικό σύστημα.
- Διαθέτει αξιόπιστη και με σειρά προτεραιότητας παράδοση πακέτων μεταξύ κόμβων παρουσία σφαλμάτων δικτύου. Το GM αντιλαμβάνεται και επαναποστέλει πακέτα που είτε έχουν χαθεί, είτε τα δεδομένα τους μεταβλήθηκαν κατά τη διάρκεια της μετάδοσής τους. Επίσης,

επαναδρομολογεί πακέτα στο δίκτυο προσπερνώντας σφάλματα δικτύου, όταν υπάρχουν εναλλακτικά μονοπάτια.

- Υποστηρίζει παραπάνω από 10 000 κόμβους.
- Διαθέτει δύο επίπεδα προτεραιότητας και έτσι επιτρέπει προώθηση προσδεδεμένης μνήμης (bounded), χωρίς κίνδυνο αδιεξόδου.
- Ανιχνεύει και "χαρτογραφεί" την τοπολογία των δικτύων Myrinet αυτόματα.

Παρόλα αυτά, ακριβώς επειδή είναι ένα "ελαφρύ" στρώμα επικοινωνίας, οι δυνατότητές του έχουν όρια που διακρίνονται, αν μέσω του GM υλοποιηθεί ένα βαρύτερο στρώμα επικοινωνίας:

- Δεν είναι δυνατόν να σταλούν ή να ληφθούν μηνύματα σε μνήμη που δεν μπορεί να προσπελαστεί με DMA.
- Δεν υποστηρίζονται λειτουργίες scatter-gather απευθείας.

Το σύστημα επικοινωνίας GM προσφέρει αξιόπιστη, ordered παράδοση μηνυμάτων μεταξύ άκρων που ονομάζονται ports με δύο επίπεδα προτεραιοτήτων. Από αυτό το μοντέλο λείπει η έννοια της σύνδεσης· δε χρειάζεται για το λογισμικό πελάτη να εκκινήσει μία σύνδεση με κάποιο απομακρυσμένο port έτσι ώστε να μπορέσει να επικοινωνήσει μαζί του. Το λογισμικό κατασκευάζει ένα μήνυμα και το στέλνει σε οποιοδήποτε port στο δίκτυο. Αυτό το παράδοξο χαρακτηριστικό, δηλαδή το γεγονός ότι η επικοινωνία στο GM είναι αξιόπιστη, παρόλο που δε δημιουργείται σύνδεση σε κάθε αποστολή, βασίζεται στη διατήρηση αξιόπιστων συνδέσεων μεταξύ ζευγών hosts στο δίκτυο και στην πολύπλεξη της κίνησης μεταξύ των ports πάνω από αυτές τις αξιόπιστες συνδέσεις.

2.3.4.1 GM ports / ακροσημεία

Σε λειτουργικά συστήματα που προσφέρουν προστασία μνήμης, το GM προσφέρει πρόσβαση στο δίκτυο με προστατευμένη μνήμη. Θα ήταν αδύνατο για οποιαδήποτε εφαρμογή πελάτη GM να το χρησιμοποιήσει για πρόσβαση σε μνήμη διαφορετική από τη μνήμη της εφαρμογής, εκτός αν ορίζεται σαφώς από το GM API. Η πιστοποιημένη πηγή του κάθε ληφθέντος μηνύματος είναι προσβάσιμη στον παραλήπτη, με αποτέλεσμα να μπορεί να απορρίπτει μηνύματα από μη πιστοποιημένες πηγές.

Το μεγαλύτερο μήνυμα που μπορεί να στείλει το GM είναι $2^{31} - 1$ bytes. Ωστόσο, δεδομένου ότι οι απομονωτές μηνυμάτων πρέπει να βρίσκονται σε

μνήμη που μπορεί να προσπελαστεί με DMA, το μέγιστο μέγεθος μηνύματος περιορίζεται από το ποσό της DMA μνήμης που μπορεί να αποδώσει το λειτουργικό σύστημα στον οδηγό του GM.

Η προτεραιότητα διατηρείται μόνο για μηνύματα της ίδιας πόρτας αποστολής, που απευθύνονται στην ίδια πόρτα λήψης και που έχουν την ίδια προτεραιότητα. Μηνύματα με διαφορετική προτεραιότητα δεν μπλοκάρουν το ένα το άλλο. Έτσι, μηνύματα χαμηλότερης προτεραιότητας μπορεί να "προσπεράσουν" αυτά της υψηλότερης. Συνήθως χρησιμοποιούνται μηνύματα της ίδιας προτεραιότητας ή το κανάλι υψηλής προτεραιότητας χρησιμοποιείται για μηνύματα ελέγχου ή για προώθηση μηνύματος σε έναν κόμβο μόνο.

Η αποστολή και η λήψη στο GM γίνεται με tokens, αναπαριστώντας το χώρο που δεσμεύεται στον πελάτη σε διάφορες εσωτερικές ουρές του GM. Στην αρχικοποίηση, ο πελάτης κατέχει `gm_num_send_tokens()` tokens αποστολής και `gm_num_receive_tokens()` tokens λήψης. Ο πελάτης μπορεί να καλέσει συγκεκριμένες συναρτήσεις μόνο όταν κατέχει ένα τέτοιο token, και καλώντας μια τέτοια συνάρτηση το απελευθερώνει. Η εφαρμογή πελάτη ευθύνεται για τον αριθμό των tokens που κατέχει και δεν πρέπει να καλεί μια συνάρτηση GM που χρειάζεται ενός συγκεκριμένου τύπου token, αν δεν το κατέχει ήδη. Μια τέτοια κλήση συνάρτησης μπορεί να προκαλέσει απροσδόκητα αποτελέσματα, αν και το GM συνήθως αναφέρει τέτοια σφάλματα τα οποία τελικά δεν προκαλούν προβλήματα στη σταθερότητα και την ασφάλεια του συστήματος.

Όπως αναφέρθηκε προηγουμένως, η αποστολή ρυθμίζεται με tokens. Ο πελάτης μιας πόρτας μπορεί να στείλει ένα μήνυμα μόνο όταν κρατά ένα token αποστολής για τη συγκεκριμένη πόρτα. Καλώντας μια συνάρτηση αποστολής του GM ο πελάτης το απελευθερώνει και περνά ένα callback και ένα δείκτη σε context σε αυτή τη συνάρτηση. Όταν η αποστολή ολοκληρωθεί το GM καλεί το callback περνώντας έναν δείκτη στην πόρτα του GM, τον δείκτη context από τον πελάτη και έναν κώδικα κατάστασης που δείχνει αν η αποστολή ολοκληρώθηκε επιτυχώς ή με κάποιο σφάλμα. Όταν το GM καλεί τη συνάρτηση callback, το token περνά διαφανώς πίσω στον πελάτη. Οι περισσότερες εφαρμογές GM που βασίζονται στον τρόπο που το GM χειρίζεται τα σφάλματα, θεωρούν οποιαδήποτε επιστροφή διαφορετική από `GM_SUCCESS` σαν κρίσιμο σφάλμα. Ωστόσο, οι πιο εξειδικευμένες εφαρμογές μπορούν να αντιμετωπίσουν σωστότερα τέτοια σφάλματα βασιζόμενες στις επεκτάσεις του GM API για χειρισμό σφαλμάτων. Αυτές οι επεκτάσεις ορίζονται σαφώς στο εγχειρίδιο του GM. Είναι σημαντικό να σημειωθεί ότι η συνάρτηση callback θα κληθεί μόνο μέσα στην κλήση της `gm_unknown()` από τον πελάτη, τη συνάρτηση χειρισμού άγνωστων γεγονότων του GM που ο πελάτης καλεί όταν αντιληφθεί ένα μη αναγνωρίσιμο μήνυμα (ή γενονός).

Το token αποστολής επιστρέφεται διαφανώς στον πελάτη όταν κληθεί η

συνάρτηση callback ή σε περίπτωση που η αποστολή γίνει χωρίς callback με τις συναρτήσεις `gm_send()` ή `gm_send_to_peer()`, με την επιστροφή ενός δείκτη `GM_SENT_EVENT`.

Η λήψη στο GM ρυθμίζεται επίσης με tokens. Αφού ανοιχτεί μια πόρτα, ο πελάτης κατέχει διαφανώς `gm_num_receive_tokens()` tokens λήψης, επιτρέποντας στην πόρτα να προσφέρει (provide) στο GM συγκεκριμένους απομονωτές για λήψη, χρησιμοποιώντας την `gm_provite_receive_buffer()`. Με κάθε κλήση σ'αυτή τη συνάρτηση, ο πελάτης ελευθερώνει ένα token λήψης. Επίσης, με κάθε απομονωτή να περνιέται σαν όρισμα στην `gm_provite_receive_buffer()`, ο πελάτης περνά τον αντίστοιχο ακέραιο SIZE που ορίζει πως το μήκος (length) του απομονωτή λήψης (receive buffer) είναι τουλάχιστο `gm_max_length_for_size()` bytes.

Προτού ο πελάτης μιας πόρτας μπορέσει να λάβει ένα μήνυμα συγκεκριμένου μεγέθους και προτεραιότητας η εφαρμογή πελάτη πρέπει να δώσει στο GM ένα receive token του αντίστοιχου μεγέθους και προτεραιότητας. Το token αυτό καθορίζει τον απομονωτή που θα αποθηκεύσει τη λήψη που ταιριάζει. Όταν ένα τέτοιο μήνυμα ληφθεί, θα μεταφερθεί στον απομονωτή λήψης που καθορίζεται από το token. Σημειωτέον ότι πολλά tokens ιδίου μεγέθους και προτεραιότητας μπορούν να προσφερθούν στην ίδια πόρτα.

Στη συνέχεια, αφού προσφερθούν απομονωτές λήψης με χαρακτηριστικά ίδια με αυτά των μηνυμάτων που μπορούν να ληφθούν, ο πελάτης πρέπει να κάνει σταθμοσκόπηση (κάνει poll) για γεγονότα λήψης χρησιμοποιώντας μία από τις συναρτήσεις `gm_receive()`, `gm_blocking_receive()`, `gm_blocking_receive_no_spin()`. Η επιστροφή αυτών των συναρτήσεων είναι ένα `gm_receive_event_t`. Η λήψη μηνυμάτων `GM_RECV_EVENT` και `GM_HIGH_RECV_EVENT` δηλώνει ληφθέντα πακέτα χαμηλής και υψηλής προτεραιότητας αντίστοιχα. Όλα τα υπόλοιπα πρέπει να περνούν στην `gm_unknown()`. Τέτοια γεγονότα χρησιμοποιούνται εσωτερικά στο GM και ο πελάτης δεν πρέπει να απασχολείται με το περιεχόμενο αυτών των πακέτων εκτός αν ορίζεται διαφορετικά.

Για την αποφυγή αδιεξόδου σε μια πόρτα, η εφαρμογή πελάτη πρέπει να διασφαλίζει ότι πάντα κατέχει tokens λήψης για οποιοδήποτε δεκτό συνδυασμό μεγέθους και προτεραιότητας για περισσότερο από ένα συγκεκριμένο χρονικό διάστημα. Πρέπει επίσης να διασφαλίζει ότι η πόρτα είναι ενήμερη για τους μη αποδεκτούς συνδυασμούς αυτών των χαρακτηριστικών και ότι η ίδια η εφαρμογή δεν πρέπει να στείλει σε κάποια απομακρυσμένη πόρτα ένα τέτοιο μη αποδεκτό μήνυμα.

Κατά συνθήκη, όταν σε μια πόρτα τελειώσουν τα tokens χαμηλής προτεραιότητας για οποιοδήποτε συνδυασμό μεγέθους, ο πελάτης μπορεί να παρακάμψει τη διαδικασία αναπλήρωσής τους ενώ εκκρεμεί η ολοκλήρωση συγκεκριμένου αριθμού από αποστολές υψηλής προτεραιότητας. Παρόλα αυτά, πρέπει πάντα

να αντικαθιστά όλους τους τύπους από tokens υψηλής προτεραιότητας χωρίς να περιμένει να ολοκληρωθούν αντίστοιχες αποστολές. Χρησιμοποιώντας αυτή την τεχνική, μπορεί να επιτευχθεί αξιόπιστη προώθηση ενός κόμβου χωρίς τον κίνδυνο αδιεξόδου.

2.3.4.2 Αρχικοποίηση του GM

Πριν από οποιαδήποτε κλήση συνάρτησης του GM πρέπει να κληθεί η `gm_init()` και αφού ολοκληρηθούν οι κλήσεις και πριν από την έξοδο της εφαρμογής πρέπει να κληθεί η `gm_finalize()`. Κάθε κλήση στην `gm_init()` πρέπει να αντιστοιχηθεί με μια κλήση στη `gm_finalize()` πριν το τέλος της εφαρμογής. Παρόλο που το GM χειρίζεται αυτόματα έξοδο από την εφαρμογή χωρίς να έχουν προηγηθεί απαραίτητα ζεύγη τέτοιων κλήσεων σε λειτουργικά συστήματα με προστασία μνήμης, προτείνεται αυτό να μη γίνεται για λόγους σωστής καθολικής συμπεριφοράς.

Μια πόρτα του GM αρχικοποιείται με κλήση της συνάρτησης

```
gm_open(struct gm_port ** PORT,
        unsigned int UNIT,
        unsigned int PORT_ID,
        char * PORT_NAME,
        enum gm_api_version VERSION)
```

όπου το `PORT_ID`είναι το αναγνωριστικό της πόρτας που ανοίγει στη διεπαφή Myrinet με το αναγνωριστικό `UNIT`. Ο δείκτης που επιστρέφεται στο `*PORT` πρέπει να περνιέται στις επόμενες κλήσεις συναρτήσεων του GM. Το `PORT_NAME` είναι μια συμβολοακολουθία μέχρι `gm_max_port_name_length()` bytes που περιγράφουν τον πελάτη. Το όνομα χρησιμοποιείται προς το παρόν για λόγους εκσφαλμάτωσης αλλά θα είναι διαθέσιμη σε όλους τους πελάτες GM στο δίκτυο μέσω ενός μηχανισμού που θα υλοποιηθεί στο μέλλον. Το VERSION αναφέρεται στην έκδοση του GM API που χρησιμοποείται από την εφαρμογή πελάτη.

Οι δείκτες `struct gm_port *` είναι μια δομή που δεν πρέπει να γίνει dereference σε καμία περίπτωση από τον πελάτη.

Μετά την κλήση της συνάρτησης `gm_open()` και ουσιαστικά την αρχικοποίηση της πόρτας, ο πελάτης κατέχει διαφανώς `gm_num_send_tokens()` tokens αποστολής και `gm_num_receive_tokens()` tokens λήψης. Οι περισσότερες εφαρμογές GM χρησιμοποιούν όλα τα tokens λήψης αμέσως μετά την αρχικοποίηση της πόρτας για να προσφέρουν τους απομονωτές στο GM (receive buffers) με την κλήση της `gm_provide_receive_buffer()`.

Στη συνέχεια, μετά από τον ορισμό των απομονωτών λήψης, ο πελάτης πρέπει να καλέσει την `gm_set_acceptable_sizes()` για κάθε προτε-

ραιότητα (GM_LOW_PRIORITY και GM_HIGH_PRIORITY) για να καταστεί σαφές ποια μηνύματα περιμένει να φτάσουν στην πόρτα. Παρόλο που η κλήση της τελευταίας συνάρτησης δεν είναι απαραίτητη, αυτή η διαδικασία επιταχύνει την έκδοση ενός κωδικού σφάλματος αν ένα μη αποδεκτό μήνυμα φτάσει στην πόρτα. Αν δε γίνει η συγκεκριμένη κλήση, τότε ένα τέτοιο σφάλμα δε θα αναφερθεί παρά μόνο αφού περάσει ο χρόνος timeout που είναι περίπου ένα λεπτό. Έτσι μια κλήση αυτής της συνάρτησης μπορεί να γλιτώσει χρόνο σε διαδικασία ανάπτυξης εφαρμογών.

2.3.4.3 Μνήμη

Το GM μπορεί να στείλει μηνύματα από μνήμη που είναι δεσμευμένη με τη gm_dma_calloc() ή τη gm_dma_malloc() ή μνήμη που έχει γίνει register για μεταφορές DMA με τη συνάρτηση gm_register_memory(). Αν ο πελάτης προσπαθήσει να στείλει δεδομένα από μνήμη που δεν μπορεί να προσπελαστεί με DMA, το GM θα στείλει bytes που περιέχουν 0xaa. Αν προσπαθήσει να λάβει δεδομένα σε μνήμη μη DMA, θα απορριφθούν σιωπηλά και ένα σφάλμα θα αναφερθεί στο log του πυρήνα.

Η συνάρτηση gm_allow_remote_memory_access(PORT) επιτρέπει σε απομακρυσμένες διεργασίες να αλλάξουν τη μνήμη που κατέχει η διεργασία. Με την κλήση της, οποιοδήποτε απομακρυσμένη πόρτα GM μπορεί να μεταβάλλει τα δεδομένα οποιασδήποτε μνήμης DMA που είναι δεσμευμένη στη συγκεκριμένη πόρτα. Η προσπέλαση μπορεί να γίνει π.χ. με μια κλήση της gm_directed_send_with_callback(gm_put()).

2.3.4.4 Αποστολή

Όπως αναφέρθηκε προηγουμένως, η αποστολή μηνυμάτων στο GM ρυθμίζεται από έναν απλό μηχανισμό με tokens για να μην υπερχειλίσουν οι εσωτερικές ουρές του που έχουν ανω φραγμένο όριο. Η εφαρμογή του πελάτη πρέπει να κατέχει ένα token αποστολής πριν καλέσει την gm_send_with_callback(). Μετά την αρχικοποίηση, η εφαρμογή κατέχει όλα τα tokens αποστολής, αριθμό που καθορίζεται από την gm_num_send_tokens() και τα περνά ένα ένα διαφανώς στη βιβλιοθήκη του GM με κάθε κλήση της gm_send_with_callback() ή της gm_send_to_peer_with_callback(). Το token ανακτάται από το GM όταν η αποστολή ολοκληρωθεί όπου καλείται η συνάρτηση του callback του πελάτη και έτσι διαφανώς επιστρέφεται το token σε αυτόν. Το περιεχόμενο του μηνύματος προς αποστολή δεν πρέπει να μεταβληθεί στο διάστημα μεταξύ της κλήσης της gm_send_with_callback() και της ολοκλήρωσης της αποστολής. Σε αντίθετη περίπτωση θα φτάσουν στον πελάτη μη ορισμένα δεδομένα (undefined data).

```
void gm_send_with_callback (struct gm_port *p,
                            void *message,
                            unsigned int size,
                            gm_size_t len,
                            unsigned int priority,
                            unsigned int target_node_id,
                            unsigned int target_port_id,
                            gm_send_completion_callback_t callback,
                            void *context)
```

Στην ειδική περίπτωση που η παράμετρος TARGET_PORT_ID είναι η ίδια με την TARGET_NODE_ID τότε μπορεί να χρησιμοποιηθεί η συνάρτηση gm_send_to_peer_with_callback() που παρακάμπτει τη συγκεκριμένη παράμετρο. Αυτή η διαδικασία μπορεί να βελτιώσει την απόδοση στις διεπαφές Myrinet των 32-bit.

2.3.4.5 Λήψη

Παρόμοια με την αποστολή, ο μηχανισμός λήψης μηνυμάτων στο GM ρυθμίζεται επίσης από έναν μηχανισμό με tokens. Μετά την αρχικοποίηση, ο πελάτης κατέχει όλα τα gm_num_receive_tokens() tokens λήψης. Η εφαρμογή προσφέρει τα tokens στο GM καλώντας τη gm_provide_receive_buffer (PORT, BUFFER, SIZE, PRIORITY) που δείχνει ότι το GM μπορεί να λάβει οποιοδήποτε μήνυμα μέσα στον απομονωτή BUFFER με την προϋπόθεση ότι το SIZE και το PRIORITY ταιριάζουν ακριβώς αυτών του μηνύματος που λαμβάνεται. Τελικά, το GM θα χρησιμοποιήσει τον απομονωτή για να αποθηκεύσει το μήνυμα. Αντίθετα με άλλα συστήματα επικοινωνίας με μηνύματα, το GM απαιτεί το μέγεθος του μηνύματος να είναι ακριβώς το ίδιο με το μέγεθος που αναφέρεται στο token. Δε θα χρησιμοποιηθεί σε καμία περίπτωση ο επόμενος μεγαλύτερος απομονωτής όταν το μέγεθος ενός εισερχομενου μηνύματος δεν είναι σωστό. Όλοι οι απομονωτές που περνούν στην gm_provide_receive_buffer πρέπει να μπορούν να προσπελαστούν με DMA. Επίσης πρέπει να είναι aligned ή να βρίσκονται σε μνήμη που έχει δεσμευτεί με τη gm_dma_calloc() ή τη gm_dma_malloc() και πρέπει να έχουν μήκος τουλάχιστο gm_max_length_for_size(SIZE)

```
void gm_provide_receive_buffer(gm_port_t *p,
                               void *ptr,
                               unsigned size,
                               unsigned priority)
```

Συνήθως οι εφαρμογές GM προσφέρουν τουλάχιστον δύο receive buffers για κάθε μέγεθος και προτεραιότητα ενός μηνύματος που πιθανώς θα ληφθεί για μεγιστοποίηση της απόδοσης επιτρέποντας την επεξεργασία του ενός όσο ο άλλος γεμίζει με δεδομένα από το δίκτυο. Ωστόσο, ένας receive buffer για κάθε συνδυασμό μεγέθους - προτεραιότητας αρκεί για σωστή λειτουργία. Επιπρόσθετα, μια καλή ιδέα είναι να υπάρχουν περισσότεροι buffers για μικρά μεγέθη ώστε να μπορούν να ληφθούν πολλά μικρά μηνύματα όσο ο host είναι απασχολημένος σε υπολογισμούς. Δεν υπάρχει λόγος να υπάρχουν tokens για λήψεις μεγέθους μικρότερου του `gm_min_message_size()`.

Μετά τη δέσμευση των tokens λήψης, χρειάζεται να γίνει σταθμοσκόπηση (poll) για εισερχόμενα γεγονότα. Αυτό γίνεται με τη συνάρτηση `gm_receive_pending(port)` που επιστρέφει μια τιμή διάφορη του μηδενός αν μια λήψη γίνεται εκείνη τη στιγμή, ή μηδέν αν δεν έχει έρθει κανένα μήνυμα. Η συνάρτηση

```
gm_next_event_peek(struct gm_port *P,
                   gm_u16_t *SENDER)
```

μπορεί επίσης να χρησιμοποιηθεί για να δείχνει στο γεγονός που είναι στην κεφαλή της ουράς. Η επιστροφή είναι ένας τύπος γεγονότος (ή μηδέν αν δεν υπάρχει γεγονός προς λήψη). Η παράμετρος SENDER γεμίζει με τον αποστολέα του μηνύματος αν το γεγονός είναι λήψης. Ο πελάτης μπορεί επίσης να κάνει poll για γεγονότα με την `gm_receive(PORT)` που επιστρέφει έναν δείκτη σε μια δομή γεγονότος, τύπου `gm_event_t`. Αν δεν υπάρχει event στην ουρά εισερχομένων μηνυμάτων, τότε επιστρέφεται ένας δείκτης σε ένα "ψεύτικο" γεγονός, στο `GM_NO_RECV_EVENT`. Το γεγονός που επιστρέφεται από αυτή τη συνάρτηση θεωρείται σωστό μόνο όταν κληθεί εκ νέου η `gm_receive()`.

Υπάρχουν πολλοι τύποι αυτής της συνάρτησης, που όλοι μπορούν να χρησιμοποιηθούν με ασφάλεια στην ίδια εφαρμογή.

- Η `gm_receive()` επιστρέφει το πρώτο τρέχον γεγονός λήψης ή `GM_NO_RECV_EVENT` αν δεν υπάρχει
- Η `gm_blocking_receive()` είναι ίδια με την `gm_receive()` με τη διαφορά ότι κάνει poll για ένα millisecond προτού κοιμηθεί.
- Η `gm_blocking_receive_no_spin()` είναι ίδια με την `gm_blocking_receive()` με τη διαφορά ότι κοιμάται αμέσως αν δεν υπάρχει γεγονός. Χρησιμοποιείται γενικά σε περιβάλλοντα που έχουν παραπάνω από μια διεργασία ανά επεξεργαστή.

Όταν ληφθεί ένα γεγονός από μία από τις τρεις αυτές συναρτήσεις, ο πελάτης πρέπει είτε να επεξεργαστεί αυτό το γεγονός, αν το αναγνωρίζει, είτε

να το περάσει στην gm_unknown() αν δεν μπορεί να το αναγνωρίσει. Όλα τα πεδία στο γεγονός λήψης είναι σε σειρά δικτύου (network byte order) και πρέπει να προσαρμοστούν σε σειρά του host (host byte order). Ο πελάτης δεν είναι υποχρεωμένος να χειρίζεται όλα τα εισερχόμενα γεγονότα και μπορεί να τα περνά όλα στην gm_unknown() αλλά οποιαδήποτε χρήσιμη εφαρμογή θα χειρίζεται τα GM_RECV_EVENT ή GM_HIGH_RECV_EVENT για να μπορεί να έχει πρόσβαση στα δεδομένα.

```
gm_recv_event_t* gm_receive(struct gm_port *P)
```

```
gm_recv_event_t* gm_blocking_receive(struct gm_port *P)
```

```
gm_recv_event_t* gm_blocking_receive_no_spin(struct gm_port *P)
```

- GM_NO_RECV_EVENT. Δεν υπάρχει γεγονός στην ουρά εισερχομένων.
- _GM_SLEEP_EVENT. Αναφέρεται σε μια από τις δύο συναρτήσεις gm_blocking_receive() ή gm_blocking_receive_no_spin() και αντιστοιχεί σε sleep event λόγω μη εισερχόμενου μηνύματος
- GM_RECV_EVENT ή GM_HIGH_RECV_EVENT. Αυτό το γεγονός δείχνει ότι ένα κανονικό εισερχόμενο γεγονός έχει λάβει χώρα. Η παρακάτω πληροφορία είναι διαθέσιμη στον πελάτη μέσω της δομής event->recv:
  - ⋆ length.
    Ο αριθμός των bytes των εισερχόμενων δεδομένων
  - ⋆ size.
    Το μέγεθος του buffer στον οποίο το μήνυμα έχει ληφθεί.
  - ⋆ buffer.
    Ένας δείκτης στον απομονωτή που έχει ήδη περαστεί στην gm_provide_receive_buffer() έτσι ώστε να επιτραπεί η λήψη του μηνύματος.
  - ⋆ sender_node_id
  - ⋆ sender_port_id
  - ⋆ tag.
    Η ετικέτα που έχει περαστεί στη συνάρτηση gm_provide_receive_buffer_with_tag() ή μηδέν αν έχει χρησιμοποιηθεί η gm_provide_receive_buffer().
    Ο τύπος GM_HIGH_RECV_EVENT δείχνει τη λήψη ενός πακέτου υψηλής προτεραιότητας ενώ ο GM_HIGH_RECV_EVENT χαμηλής προτεραιότητας, αντίστοιχα.

- GM_PEER_RECV_EVENT ή GM_HIGH_PEER_RECV_EVENT. Αυτά τα γεγονότα μπορούν να αγνοηθούν με ασφάλεια αφού το πέρασμά τους στην gm_unknown() θα τα μετατρέψει σε κανονικά GM_RECV_EVENT γεγονότα που θα περαστούν στον πελάτη στην επόμενη κλήση στην gm_receive().

  Αυτοί οι δύο τύποι είναι ακριβώς όπως και οι GM_RECV_EVENT ή GM_HIGH_RECV_EVENT με τη διαφορά ότι τα πεδία sender_node_id και sender_port_id είναι ταυτόσημα και το ένα από αυτά έχει παραληφθεί. Μπορούν να αντιμετωπίζονται λοιπόν με τον ίδιο τρόπο με τους παραπάνω τύπους μηνυμάτων.

- GM_FAST_RECV_EVENT

- GM_FAST_HIGH_RECV_EVENT

- GM_FAST_PEER_RECV_EVENT

- GM_FAST_HIGH_PEER_RECV_EVENT

  Αυτά τα γεγονότα μπορούν να αγνοηθούν με απόλυτη ασφάλεια, να περαστούν δηλαδή στην gm_unknown() και σε αυτή την περίπτωση θα μετατραπούν σε ένα κανονικό μήνυμα τύπου GM_RECV_EVENT και θα περαστούν στον πελάτη στην επόμενη κλήση της gm_receive(). Η διαδικασία μετατροπής, θα αντιγράψει το εισερχόμενο μήνυμα από την ουρά εισερχομένων στον απομονωτή εισερχομένων.

  Αυτοί οι τύποι δηλώνουν ότι ένα μήνυμα μικρού μεγέθους έχει ληφθεί και τα δεδομένα του είναι αποθηκευμένα στην ουρά εισερχομένων, για βελτίωση της απόδοσης μικρού μεγέθους μηνυμάτων. Ο τύπος PEER δηλώνει αυτό που έχει οριστεί παραπάνω. Ο τύπος HIGH δηλώνει ότι το μήνυμα έχει σταλεί με υψηλή προτεραιότητα.

  Σε εφαρμογές που χρησιμοποιούν μικρού μεγέθους μηνύματα που τα επεξεργάζονται αμέσως και διαγράφονται με τη λήψη τους, μπορούν να χρησιμοποιηθούν οι παραπάνω τύποι για αύξηση της απόδοσης. Αν μετά την επεξεργασία του μηνύματος χρειάζεται και η αποθήκευσή του στον buffer εισερχομένων, μπορεί είτε να κληθεί η gm_memmorize_message() είτε να περαστεί στην gm_unknown().

  Η διαφορά με τα παραπάνω όσον αφορά στην πληροφορία που περιέχουν είναι ένα πεδίο message που περιέχει τον δείκτη στο εισερχόμενο μήνυμα που είναι αποθηκευμένο στην ουρά εισερχομένων και είναι εγγυημένο ότι θα είναι έγκυρο μέχρι την επόμενη κλήση στην gm_receive().

Οποιοδήποτε εισερχόμενο γεγονός που δεν αναγνωρίζεται από μια εφαρμογή πρέπει να περνιέται απευθείας στην `gm_unknown()`. Η συνάρτηση `gm_unknown()` θα ελευθερώσει ο,τιδήποτε θα χρειαζόταν να ελευθερωθεί αν ήταν ένα κανονικό μήνυμα. Επίσης, επιπρόσθετοι τύποι μηνυμάτων που λαμβάνονται από μια εφαρμογή περνούν στην `gm_unknown()` η οποία τα χειρίζεται ανάλογα. Αυτά τα μηνύματα μπορούν να χρησιμοποιηθούν από το GM σαν alarms ή blocking λήψεις.

Το κίνητρο για την τοποθέτηση των μικρού μεγέθους μηνυμάτων στην ουρά εισερχομένων παρά το γεγονός ότι αυτή η διαδικασία μπορεί να χρειάζεται μία παραπάνω αντιγραφή από την πλευρά του πελάτη, είναι το ακόλουθο σύνολο παρατηρήσεων:

- ⋆ Ένα μεγάλο μέρος των εισερχόμενων μηνυμάτων μικρού μεγέθους είναι μηνύματα ελέγχου που γίνουν αντικείμενο επεξεργασίας αμέσως μετά τη λήψη τους και στη συνέχεια να μη χρειαστεί να αντιγραφούν στον πιο μόνιμο απομονωτή και με αποτέλεσμα να είναι περιττή μία ακόμα κλήση στην `gm_receive()`.
- ⋆ Το κόστος μίας ακόμα κλήσης DMA για να τοποθετηθεί το μήνυμα στον απομονωτή αντί για την ουρά εισερχομένων, είναι πιο ακριβό σε όρους χρόνου για μικρά μηνύματα από ότι να εκτελέσει την αντιγραφή ο host.

Επομένως, τοποθετώντας μικρού μεγέθους μηνύματα στην ουρά εισερχομένων, και όχι στον πιο μόνιμο απομονωτή ληφθέντων μηνυμάτων, αυξάνει την απόδοση και αξίζει την επιπρόσθετη πολυπλοκότητα.

Για την αποφυγή αδιεξόδου, η εφαρμογή πελάτη πρέπει να βεβαιωθεί ότι το GM δεν υπολείπεται σε tokens λήψης για ένα οποιοδήποτε πιθανό μήνυμα για περισσότερο από ένα άνω φραγμένο χρονικό διάστημα. Γενικά, αν εξαιρέσουμε την περίπτωση προώθησης μηνυμάτων, αυτό σημαίνει ότι μετά από κάθε επιτυχή κλήση στην `gm_receive()` ο πελάτης πρέπει να καλέσει την `gm_provide_receive_buffer()` για αντικατάσταση του token λήψης με ένα του ίδιου μεγέθους και της ίδιας προτεραιότητας πριν μια νέα κλήση της `gm_receive()` ή της `gm_send()`. Αν συμβεί μια τέτοια κατάσταση αδιεξόδου για μεγάλο διάστημα (της τάξης του ενός λεπτού), ή αρκετά συχνά (κάθε μια χρονική στιγμή ανάμεση σε περιόδους του ενός λεπτού), τότε οι απομακρυσμένες αποστολές στη συγκεκριμένη πόρτα λήψης θα ″λήξουν″.

2.3.5 Ένα σενάριο αποστολής

Με τη βοήθεια του σχήματος 2.5 και με αναφορά το σχήμα 2.4 παρουσιάζεται ένα σενάριο αποστολής με χρήση του GM σε δίκτυο Myrinet[BRB98].

1. Όπως αναφέρθηκε προηγουμένως, η αποστολή με το GM πάνω από Myrinet προϋποθέτει τα δεδομένα να βρίσκονται σε χώρο που μπορεί να αποτελέσει πηγή για μεταφορά DMA. Έτσι, στο πρώτο βήμα τα δεδομένα από τη μνήμη του χώρου χρήστη αντιγράφονται σε μια περιοχή DMA. Βέβαια, θεωρούμε δεδομένο ότι τα δεδομένα πρώτα βρίσκονται στο χώρο χρήστη. Θα μπορούσαμε να θεωρήσουμε ότι γίνεται ανάγνωση από κάποιο μέσο αποθήκευσης κατευθείαν στην περιοχή DMA. Αλλά ας αφήσουμε τη συγκεκριμένη προσέγγιση για αργότερα.

2. Ο μικροεπεξεργαστής LANai πρέπει να γνωρίζει ποιοι απομονωτές στην περιοχή DMA είναι κενοί και ποιοι περιέχουν δεδομένα. Για το σκοπό αυτό, τη στιγμή που δεσμεύονται οι buffers στην περιοχή DMA για μελλοντική χρήση τους από την εφαρμογή χρήστη, το LANai με χρήση των tokens μπορεί να γνωρίζει ανά πάσα στιγμή σε ποιον απομονωτή αναφέρεται ο χρήστης. Έτσι, ταυτόχρονα με τη δέσμευση των απομονωτών στην περιοχή DMA το LANai αποθηκεύει ένα token με πληροφορίες για τον αντίστοιχο buffer που δείχνει.

3. Σαν άμεσο αποτέλεσμα του βήματος 2, το LANai προγραμματίζει τη μηχανή DMA που επικοινωνεί με το δίαυλο PCI και άρα μπορεί να αντιγράψει δεδομένα από και προς την κεντρική μνήμη του συστήματος, δίνοντας τη φυσική διεύθυνση του απομονωτή που θα αντιγραφεί στη στατική μνήμη της κάρτας δικτύου Myrinet.

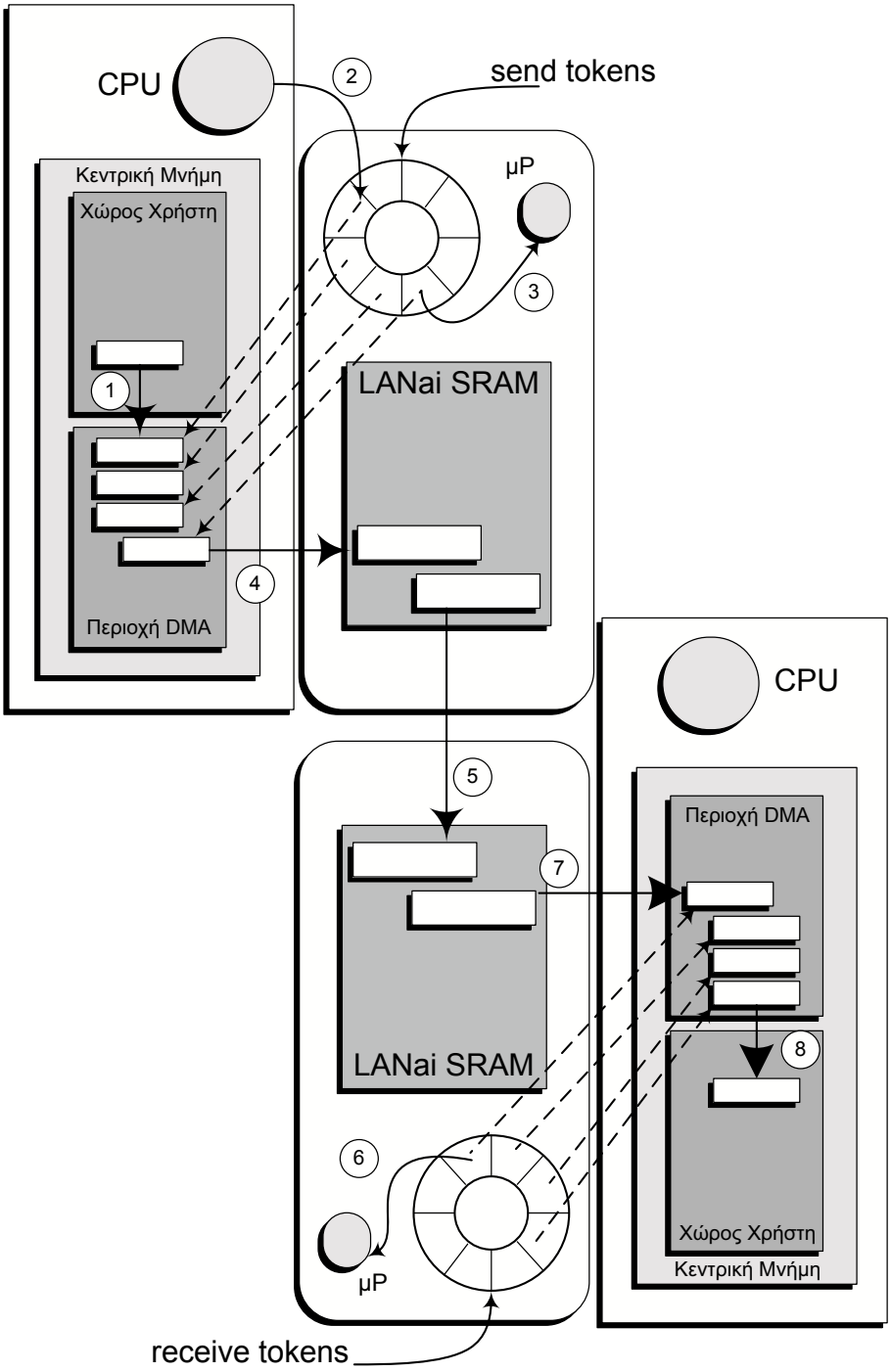
4. Η μηχανή DMA (host-to-LANai) αντιγράφει τα δεδομένα στη στατική μνήμη του LANai.

5. Η μηχανή DMA (LANai-to-network, dma send) αντιγράφει τα δεδομένα στο καλώδιο. Η μηχανή DMA (network-to-LANai, dma receive) αντιγράφει τα δεδομένα στη στατική μνήμη της κάρτας δικτύου του παραλήπτη.

6. Ο LANai του παραλήπτη ελέγχει τα tokens λήψης για δείκτης προς απομονωτές που μπορούν να φιλοξενήσουν δεδομένα σε περιοχή DMA της κεντρικής μνήμης του συστήματος του παραλήπτη. Προγραμματίζει τη μηχανή DMA (lanai-to-host) για την αντιγραφή.

7. Η μηχανή DMA (LANai-to-host) πραγματοποιεί την αντιγραφή και τα δεδομένα βρίσκονται σε περιοχή DMA της κεντρικής μνήμης του συστήματος του παραλήπτη.

8. Σε περίπτωση που τα δεδομένα πρέπει να μεταφερθούν από την DMA περιοχή τότε εκτελείται και το παρόν βήμα.



Σχήμα 2.5: Μια αποστολή σε δίκτυο Myrinet με χρήση του GM

Κεφάλαιο 3

Συστήματα αποθήκευσης στο Linux

Στόχος του παρόντος κεφαλαίου είναι η ανάλυση του επιπέδου του λειτουργικού συστήματος LINUX που είναι υπεύθυνο για την υποστήριξη συσκευών αποθήκευσης.

3.1 Το Linux

Το Linux είναι ένα ελεύθερο τύπου-Unix λειτουργικό σύστημα που αρχικά δημιουργήθηκε από τον Linus Torvalds και στη συνέχεια αναπτύχθηκε με τη βοήθεια προγραμματιστών από όλον τον κόσμο. Το Linux είναι μια ανεξάρτητη POSIX υλοποίηση και στα χαρακτηριστικά του συμπεριλαμβάνει πραγματικό multitasking, πραγματικό πολυχρηστικό περιβάλλον, virtual memory, shared libraries, demand loading, TCP/IP networking και πολλά άλλα χαρακτηριστικά που δικαιολογούν τον τίτλο ″τύπου-Unix″. Είναι κατασκευασμένο υπό την GPL άδεια, δηλαδή ο πηγαίος κώδικάς του είναι διαθέσιμος στον καθένα (open-source). Μπορεί να χρησιμοποιηθεί για πολλούς σκοπούς, όπως το networking, η ανάπτυξη προγραμμάτων, ακόμα και για πλατφόρμα για απλούς χρήστες. Λόγω της φύσης και της ευελιξίας του, το Linux έχει γίνει πολύ διάσημο παγκοσμίως και ένας μεγάλος αριθμός προγραμματιστών έχει επικεντρώσει το ενδιαφέρον του πάνω σε αυτό.

Μερικά από τα χαρακτηριστικά του συνοψίζονται παρακάτω:

- 32 ή 64 bit, ανάλογα με την αρχιτεκτονική
- Multitasking: πολλά προγράμματα μπορούν να τρέχουν ταυτόχρονα
- Multiuser: πολλοί χρήστες μπορούν να το χρησιμοποιούν ταυτόχρονα
- Multiplatform: μπορεί να τρέξει σε μια πληθώρα επεξεργαστών

- Multiprocessing: Παράλληλη επεξεργασία μέχρι και 16 επεξεργαστές
- Clustering: μπορεί να στηθεί σε συστοιχία υπολογιστών
- Προστασία μνήμης, ώστε κάποιο πρόγραμμα να μην μπορεί να κολλήσει τον υπολογιστή
- Εκτελέσιμα με την τεχνολογία Load on Demand: φορτώνονται μόνο τα τμήματα των προγραμμάτων που χρησιμοποιούνται
- Σελιδοποίηση της Virtual Μνήμης
- Δυναμικά διασυνδεδεμένες βιβλιοθήκες (DLLs) καθώς και στατικές επίσης
- Συμβατό με το Unix (POSIX, System V και BSD) σε επίπεδο κώδικα.
- POSIX έλεγχος διεργασιών. Επίσης, είναι δυνατή η χρήση του QNX στύλ scheduling
- Υποστήριξη μιας πληθώρας από filesystems, μεταξύ των οποίων τα FAT32, NTFS, BSD ufs, HFS, QNX fs κ.α.
- Μέχρι και 64 εικονικές κονσόλες
- Ο πηγαίος κώδικας είναι διαθέσιμος, συμπεριλαμβανομένου του πυρήνα και των drivers
- Πιθανότατα ο πιο γρήγορος 100Mbit-Ethernet TCP/IP κώδικας
- Σε σχέση με τα άλλα Unix, έχει τους περισσότερους drivers για περιφερειακά, μεταξύ των οποίων είναι κάρτες ήχου, κάρτες δικτύου ethernet, κάρτες ATM, κάρτες ISDN κ.α.
- Πρωτόκολλα δικτύου, όπως TCP/IP v4 και v6, IPX/SPX, TokenRing, Ethertalk, Appletalk κ.α.
- Πληθώρα δικτυακών χαρακτηριστικών, όπως masquerading, tunneling, forwarding, routing, firewalls κ.α.

Το Linux είναι στενά συνδεδεμένο με το GNU project, του οποίου στόχος είναι η δημιουργία ελεύθερου λογισμικού. Το βασικότερο κομμάτι των εφαρμογών χώρου χρήστη σε ένα Linux σύστημα προέρχονται από το GNU project.

Τα λειτουργικά συστήματα τύπου-Unix, όπως και το Linux, βασίζονται στην έννοια αρχείο που, όπως αναφέρθηκε στο Κεφάλαιο 1, είναι μια δομημένη ακολουθία από bytes. Σύμφωνα με αυτή την προσέγγιση οι συσκευές Ε/Ε αντιμετωπίζονται ως ″ειδικά″ (special) αρχεία που λέγονται device files.

3.1.1 Ο πυρήνας του Linux

Στο Linux διάφορες συντρέχουσες διεργασίες επιτελούν διαφορετικές λειτουργίες και είναι αναγκαίο να διασφαλίζεται η εύρυθμη εκτέλεσή τους. Η κάθε διεργασία ζητά πόρους του συστήματος, όπως υπολογιστική δύναμη, μνήμη, συνδεσιμότητα δικτύου ή κάτι άλλο. Ο πυρήνας είναι ένα μεγάλο κομμάτι εκτελέσιμου κώδικα που διαχειρίζεται τους πόρους[DPB05]. Παρόλο που ο διαχωρισμός ανάμεσα στις λειτουργίες του πυρήνα δεν είναι σαφώς ορισμένος, ο ρόλος του είναι:

- Process Management (διαχείριση διεργασιών)
  Ο πυρήνας δημιουργεί και καταστρέφει διεργασίες και χειρίζεται τη σύνδεσή τους με τον "έξω κόσμο" (είσοδος / έξοδος). Η επικοινωνία ανάμεσα στις διεργασίες (μέσω σημάτων, σωληνώσεων ή αρχές διαδιεργασιακής επικοινωνίας) είναι βασική στη λειτουργικότητα του όλου συστήματος και είναι επίσης στον έλεγχο του πυρήνα. Το γεγονός ότι ο πυρήνας επιτελεί τη διαχείριση των διεργασιών υλοποιεί ένα επίπεδο αφαίρεσης για διεργασίες που εκτελούνται σε έναν ή περισσότερους επεξεργαστές.

- Memory Management (διαχείριση Μνήμης)
  Η μνήμη του υπολογιστικού συστήματος είναι ένας σημαντικός πόρος και η πολιτική που χρησιμοποιείται για τη διαχείρισή της είναι βασική για την απόδοση του συστήματος. Ο πυρήνας χτίζει ένα χώρο εικονικής διευθυνσιοδότησης για οποιαδήποτε διεργασία πάνω από τους πεπερασμένους διαθέσιμους πόρους. Τα διαφορετικά κομμάτια του πυρήνα αλληλεπιδρούν με το υποσύστημα διαχείρισης μνήμης μέσω ενός συνόλου συναρτήσεων που ποικίλλουν από το απλό ζεύγος `malloc()` / `free()` μέχρι και πολύ πιο πολύπλοκες συναρτήσεις

- Filesystems (συστήματα αρχείων)
  Το Linux βασίζεται στη λογική του συστήματος αρχείων˙ σχεδόν όλα μπορούν να είναι ένα αρχείο. Ο πυρήνας χτίζει ένα δομημένο σύστημα αρχείων πάνω από αδόμητο υλικό και το αποτέλεσμα, ένα μοντέλο αφαίρεσης με αρχεία (file abstraction), χρησιμοποείται διαμέσου όλου του συστήματος. Επιπρόσθετα, το Linux υποστηρίζει διάφορους τύπους συστημάτων αρχείων, δηλαδή διαφορετικούς τρόπους οργάνωσης δεδομένων στο φυσικό μέσο. Για παράδειγμα, οι σκληροί δίσκοι μπορούν να διαμορφωθούν με το πρότυπο σύστημα αρχείων ext3, το κοινώς χρησιμοποιούμενο FAT ή και αρκετά άλλα.

- Device Control (έλεγχος συσκευών)
  Όλες σχεδόν οι λειτουργίες συστήματος τελικά καταλήγουν σε μια

φυσική συσκευή. Με εξαίρεση τον επεξεργαστή, τη μνήμη και λίγες άλλες οντότητες, όλος ο έλεγχος λειτουργιών μιας συσκευής επιτυγχάνεται με κώδικα συγκεκριμένο στην εκάστοτε συσκευή. Ο κώδικας αυτός ονομάζεται οδηγός συσκευής. Ο πυρήνας πρέπει να έχει ενσωματωμένο έναν οδηγό συσκευής για κάθε περιφερειακή συσκευή παρούσα στο σύστημα, από το σκληρό δίσκο μέχρι το πληκτρολόγιο.

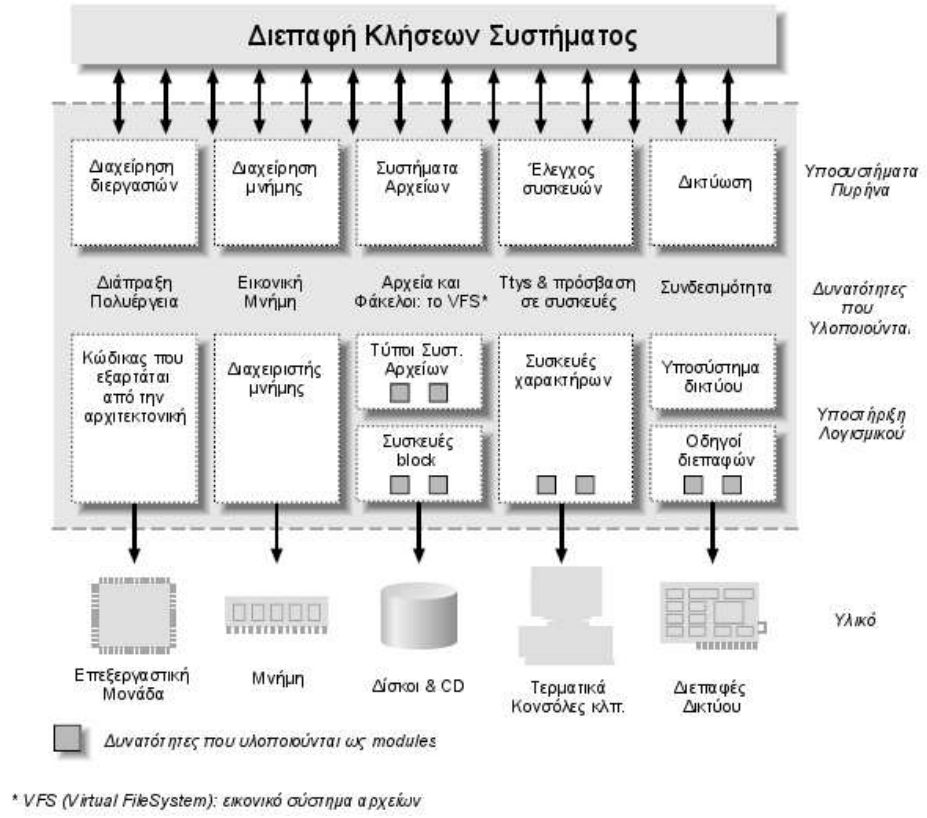
- Networking (δικτύωση)
  Το λειτουργικό σύστημα πρέπει να διαχειρίζεται τη δικτύωση, γιατί οι περισσότερες λειτουργίες δικτύου δεν είναι συγκεκριμένες σε μια διεργασία· τα εισερχόμενα πακέτα είναι ασύγχρονα γεγονότα. Τα πακέτα πρέπει να συγκεντρωθούν, να αναγνωριστούν και να διεκπεραιωθούν, πριν τα αναλάβει μια διεργασία. Το σύστημα είναι υπεύθυνο να παραδίδει πακέτα δεδομένων μεταξύ εφαρμογών και διεπαφών δικτύου και να ελέγχει την εκτέλεση των εφαρμογών ανάλογα με τη δικτυακή τους λειτουργία. Επιπρόσθετα, η δρομολόγηση και η ανάλυση διευθύνσεων υλοποιούνται μέσα στον πυρήνα.

3.1.2 Modules

΄Ενα πλεονέκτημα του Linux είναι το γεγονός ότι ο πυρήνας μπορει να επεκτείνει τις δυνατότητες του, ενώ τρέχει. Κάθε κομμάτι κώδικα που μπορεί να ενσωματωθεί στον πυρήνα, ενώ τρέχει, λέγεται module. Το Linux προσφέρει υποστήριξη για αρκετά διαφορετικούς τύπους (ή κλάσεις) από modules, που περιλαμβάνουν και οδηγούς συσκευών, χωρίς να περιορίζονται μόνο σε αυτούς. Κάθε module φτιάχνεται από εκτελέσιμο κώδικα (χωρίς να είναι linked σε ολοκληρωμένο εκτελέσιμο αρχείο) που μπορεί να γίνει link δυναμικά στον πυρήνα που τρέχει μέσω της εφαρμογής insmod και να γίνει unlink με την εφαρμογή rmmod.

3.1.3 Κλάσεις συσκευών και modules

Από την οπτική γωνία του πυρήνα, οι συσκευές διακρίνονται σε τρείς τύπους. Κάθε module, συνήθως, υλοποιεί έναν από αυτούς τους τύπους και έτσι μπορούν να χωριστούν σε char modules (για συσκευές χαρακτήρων), block modules (για συσκευές block) και σε network modules. Ο διαχωρισμός αυτός δεν είναι σαφής, αφού εξαρτάται από τον προγραμματιστή αν θα ενσωματώσει διαφορετικούς οδηγούς για διαφορετικές συσκευές σε ένα κοινό module. Προτιμότερο είναι, για λόγους επεκτασιμότητας και λειτουργικότητας, να χτίζεται ένα καινούριο module για κάθε λειτουργία που προστίθεται στον πυρήνα.



Σχήμα 3.1: Μία όψη του πυρήνα σε υποσυστήματα

- Character devices
  Μια συσκευή χαρακτήρων μπορεί να προσπελαστεί σαν μια ροή από bytes (όπως ένα αρχείο)˙ ο οδηγός συσκευής χαρακτήρων υλοποιεί αυτή τη συμπεριφορά. ΄Ενας τέτοιος οδηγός υλοποιεί τουλάχιστο τις κλήσεις συστήματος open, close, read και write. Η διαφορά τους με ένα κοινό αρχείο είναι ότι, ενώ στο αρχείο μπορούν να προσπελαστούν δεδομένα από οπουδήποτε, στη συσκευή χαρακτήρων η πρόσβαση είναι ακολουθιακή. Σαν εξαίρεση, υπάρχουν συσκευές χαρακτήρων (όπως τα frame grabbers) που μπορούν να προσπελαστούν συνολικά με χρήση της mmap ή της lseek,

- Block devices
  Όπως οι συσκευές χαρακτήρων, έτσι και οι συσκευές block μπορούν να προσπελαστούν από "κόμβους" του συστήματος στον φάκελο /dev. Μια τέτοια συσκευή μπορεί να ενθυλακώσει ένα σύστημα αρχείων. Στα

περισσότερα συστήματα Unix, μια συσκευή block μπορεί να διαχειριστεί λειτουργίες που μεταφέρουν ένα ή περισσότερα blocks που συνήθως έχουν μέγεθος 512 (ή μια μεγαλύτερη δύναμη του 2) bytes. Στο Linux, οι εφαρμογές μπορούν να διαβάζουν οποιοδήποτε αριθμό από bytes με αποτέλεσμα οι συσκευές char και οι συσκευές block να διαφέρουν μόνο στον τρόπο με τον οποίο τις χειρίζεται ο πυρήνας.

- Networking Interfaces
  Οποιαδήποτε κίνηση δικτύου γίνεται μέσω μιας διεπαφής, που είναι μια συσκευή που μπορεί να ανταλλάζει δεδομένα με άλλα συστήματα. Συνήθως μια τέτοια διεπαφή είναι υλικό, αλλά θα μπορούσε να είναι εικονική, όπως το loopback interface. Μια διεπαφή δικτύου είναι υπεύθυνη για αποστολή και λήψη πακέτων δεδομένων, χωρίς να γνωρίζει πώς οι κινήσεις αντιστοιχούνται στα μεταφερόμενα πακέτα, γιατί οδηγείται από το υποσύστημα δικτύου του πυρήνα.

  Πολλές συνδέσεις δικτύου είναι προσανατολισμένες σε ροή δεδομένων αλλά οι συσκευές δικτύου, συνήθως, είναι σχεδιασμένες με βάση τη μετάδοση πακέτων. Ένας οδηγός συσκευής δικτύου δε γνωρίζει τίποτα από ατομικές συνδέσεις, γνωρίζει μόνο από χειρισμό πακέτων.

  Αφού μια συσκευή δικτύου δεν είναι προσανατολισμένη σε ροή δεδομένων, η διεπαφή δικτύου δεν αντιστοιχίζεται σε ένα κόμβο στο σύστημα αρχείων, όπως μια συσκευή χαρακτήρων (π.χ. η /dev/ttyS1). Τα Unix συστήματα χειρίζονται μια συσκευή δικτύου δίνοντάς της ένα μοναδικό αναγνωριστικό (όπως το eth0), που δεν έχει μια αντιστοίχιση στο σύστημα αρχείων. Ο τρόπος προσπέλασης αυτής της διεπαφής δεν είναι με read και write αλλά με κλήσεις σε συναρτήσεις που σχετίζονται με το χειρισμό και τη μετάδοση πακέτων.

Υπάρχουν και άλλοι τρόποι διαχωρισμού modules οδηγών συσκευών που είναι κάθετα αντίθετοι (orthogonal), συγκριτικά με τους παραπάνω τύπους. Γενικά, κάποιοι τύποι συσκευών λειτουργούν με επιπρόσθετα επίπεδα συναρτήσεων του πυρήνα για ένα δεδομένο τύπο συσκευών. Για παράδειγμα, υπάρχουν USB modules, serial modules, SCSI modules κλπ. Κάθε συσκευή USB οδηγείται από ένα USB module που λειτουργεί με το υποσύστημα USB, αλλά η συσκευή εμφανίζεται στο σύστημα σαν μια συσκευή χαρακτήρων (όπως μια σειριακή πόρτα USB), μια συσκευή block (μια USB συσκευή που διαβάζει κάρτες μνήμης) ή μια συσκευή δικτύου (μια USB διεπαφή δικτύου Ethernet).

3.2 Χειρισμός Μνήμης

Το Linux είναι ένα σύστημα εικονικής μνήμης, δηλαδή οι διευθύνσεις που χρησιμοποιούν οι εφαρμογές χρήστη δεν αντιστοιχούν απευθείας σε φυσικές διευθύνσεις που χρησιμοποιεί το υλικο. Η εικονική μνήμη δίνει ένα στρώμα αφαίρεσης στον τρόπο με τον οποίο το λειτουργικό σύστημα χειρίζεται τη διαθέσιμη μνήμη στο φυσικό επίπεδο. Οι εφαρμογές που εκτελούνται στο σύστημα, μπορούν να δεσμεύουν περισσότερη μνήμη από τη διαθέσιμη καθώς και μνήμη που αντιστοιχεί σε μια συσκευή.

Το Linux χρησιμοποιεί συγκεκριμένους όρους για τη διευθυνσιοδότηση της μνήμης.

- User virtual addresses
  Είναι οι κανονικές διευθύνσεις που φαίνονται από τις εφαρμογές χώρου χρήστη. Είναι 32 ή 64 bits σε μήκος ανάλογα με την αρχιτεκτονική του υλικού. Κάθε διεργασία έχει διαφορετικό virtual address space.

- Physical addresses
  Είναι διευθύνσεις που χρησιμοποιούνται μεταξύ του επεξεργαστή και της μνήμης του συστήματος. Είναι είτε 32 είτε 64 bits σε μήκος.

- Bus addresses
  Είναι οι διευθύνσεις που χρησιμοποιούνται από τους διαδρόμους περιφερειακών και τη μνήμη. Συχνά είναι ίδιες με τις διευθύνσεις που χρησιμοποιεί ο επεξεργαστής αλλά όχι πάντα. Κάποιες αρχιτεκτονικές μπορούν να προσφέρουν μια μονάδα διαχείρισης μνήμης εισόδου / εξόδου που κάνει remap διευθύνσεις μεταξύ ενός διαδρόμου και της κύριας μνήμης.

- Kernel logical addresses
  Αυτές οι διευθύνσεις κατασκευάζουν το κανονικό address space του πυρήνα. Κάνουν map κάποιο κομμάτι της μνήμης (ή ακόμα και ολόκληρη) και χρησιμοποιούνται σαν να είναι φυσικές διευθύνσεις. Σε πολλές αρχιτεκτονικές διαφέρουν από τις φυσικές διευθύνσεις μόνο κατά ένα offset.

- Kernel virtual addresses
  Είναι παρόμοιες με τις λογικές διευθύνσεις στο βαθμό που αποτελούν ένα mapping από μια kernel-space διεύθυνση σε μια physical διεύθυνση.

3.2.1 Φυσικές διευθύνσεις και σελίδες

Η πραγματική (φυσική) μνήμη χωρίζεται σε διακριτές μονάδες που λέγονται σελίδες. Ο χειρισμός της μνήμης από το λειτουργικό σύστημα γίνεται στο μεγαλύτερο βαθμό του με βάση τις σελίδες. Το μέγεθος μιας σελίδας κυμαίνεται ανάλογα με την αρχιτεκτονική, παρόλο που τα περισσότερα συστήματα σήμερα χρησιμοποιούν ένα σταθερό μέγεθος 4096 bytes.

Μια διεύθυνση μνήμης (εικονική ή και φυσική) διαιρείται σε έναν αριθμό σελίδας και σε ένα offset μέσα στη σελίδα. Αν χρησιμοποιούνται, για παράδειγμα, 4096 bytes, τα 12 λιγότερο σηματικά bits είναι το offset και τα υπόλοιπα περισσότερο σημαντικά bits είναι ο αριθμός της σελίδας. Αν αποκοπεί το offset και γίνει ολίσθηση του αριθμού προς τα δεξιά το αποτέλεσμα είναι το page frame number (pfn). Αυτή η ολίσθηση είναι κοινή πρακτική.

3.2.2 Μνήμη High και Low

Η διαφορά μεταξύ λογικών και εικονικών διευθύνσεων του πυρήνα φαίνεται καθαρά σε 32 bit συτήματα εξοπλισμένα με μεγάλου μεγέθους μνήμη. Με 32 bits είναι δυνατόν να διευθυνσιοδοτηθούν 4GB μνήμης. Το Linux σε τέτοια συστήματα περιοριζόταν σε ελάχιστα λιγότερη μνήμη από αυτή, λόγω του μοντέλου που εγκαθιστά και διαχειρίζεται τον εικονικό χώρο διευθύνσεων (virtual address space).

Ο πυρήνας (στην x86 αρχιτεκτονική) χωρίζει τον 4GB εικονικό χώρο διευθύνσεων στο χώρο χρήστη και στο χώρο πυρήνα. Το ίδιο σύνολο mappings χρησιμοποιείται και στις δύο περιπτώσεις. ΄Ενα τυπικό χώρισμα αφιερώνει 3GB στο χώρο χρήστη και 1 GB στο χώρο πυρήνα. Ο κώδικας του πυρήνα και οι δομές του πρέπει να χωρέσουν εκεί, όμως το μεγαλύτερο χώρο καταλαμβάνουν τα εικονικά mappings για την πραγματική μνήμη. Ο πυρήνας δεν μπορεί να διαχειρίζεται μνήμη που δεν απεικονίζεται στο χώρο διευθύνσεών του. Με άλλα λόγια ο πυρήνας χρειάζεται μια δική του εικονική διεύθυνση για οποιοδήποτε κομμάτι μνήμης που προσπελάζω. ΄Ετσι, για πολλά χρόνια, το μεγαλύτερο μέγεθος φυσικής (πραγματικής) μνήμης που θα μπορούσε να χειριστεί ο πυρήνας ήταν το μέγεθος που θα μπορούσε να απεικονιστεί μέσα στο κομμάτι του πυρήνα στο χώρο εικονικών διευθύνσεων, μειωμένο κατά το χώρο που καταλαμβάνει ο ίδιος ο πυρήνας. Σαν αποτέλεσμα, τα συστήματα Linux βασισμένα σε x86 αρχιτεκτονική μπορούσαν να λειτουργούν με ένα μέγιστο από κάτι λιγότερο του 1GB φυσικής (πραγματικής) μνήμης.

Σε απάντηση της εμπορικής πίεσης, οι κατασκευαστές επεξεργαστών προσέθεσαν δυνατότητες «επέκτασης διευθύνσεων» στα προϊόντα τους, χωρίς αυτό να επηρεάζει τις 32 bit εφαρμογές. Το αποτέλεσμα είναι ότι σε πολλές περιπτώσεις ακόμα και 32 bit επεξεργαστές μπορούν να διευθυνσιοδοτήσουν

περισσότερα από 4GB φυσικής μνήμης. Βέβαια, παραμένει ο περιορισμός της ένα-προς-ένα απεικόνισης φυσικής μνήμης με λογικές διευθύνσεις. Μόνο το χαμηλότερο κομμάτι της μνήμης (μέχρι 1 ή 2 GB, ανάλογα το υλικό) έχει λογικές διευθύνσεις· το υπόλοιπο (high memory) δεν έχει. Πριν την προσπέλαση μιας συγκεκριμένης σελίδας high-memory, ο πυρήνας πρέπει να κατασκευάσει ένα εικονικό mapping, για να μπορεί να έχει τη σελίδα αυτή διαθέσιμη στο χώρο διευθύνσεών του. Έτσι αρκετές δομές δεδομένων του πυρήνα πρέπει να τοποθετηθούν στη low-memory· η high-memory δεσμεύεται κυρίως για εφαρμογές χώρου χρήστη.

- Low memory
  Μνήμη για την οποία υπάρχουν λογικές διευθύνσεις στο χώρο διευθύνσεων του πυρήνα. Στα κοινά συστήματα, συνήθως, είναι όλη η διαθέσιμη πραγματική μνήμη.

- High memory
  Μνήμη για την οποία δεν υπάρχουν λογικές διευθύνσεις, γιατί ξεπερνά το όριο του χώρου διευθύνσεων που έχει δεσμευτεί για τις εικονικές διευθύνσεις του πυρήνα.

3.2.3 Page Tables

Σε συστήματα i386 το όριο μεταξύ low και high memory είναι συνήθως κάτω του 1GB, παρόλο που τα όρια μπορούν να αλλάζουν στις ρυθμίσεις του πυρήνα. Αυτό το όριο δεν έχει καμία σχέση με το παλιό όριο των 640KB, που μπορούσε κανείς να βρει στα PC, και δεν εξαρτάται από το υλικό. Είναι αντίθετα ένα όριο που το θέτει ο πυρήνας, αφού διαιρεί το χώρο διευθύσεων σε χώρο χρήστη και χώρο πυρήνα.

Σε οποιοδήποτε σύγχρονο σύστημα, ο επεξεργαστής πρέπει να έχει ένα μηχανισμό μετατροπής των εικονικών διευθύνσεων σε φυσικές. Αυτός ο μηχανισμός ονομάζεται πίνακας σελίδων (page table). Είναι ένα βασικός πολυεπίπεδος πίνακας με δενδρική δομή που περιέχει virtual-to-physical mappings και κάποιες σημαίες. Ένας μεγάλος αριθμός κοινότυπων λειτουργιών που γίνονται από οδηγούς συσκευών συμπεριλαμβάνουν τα page tables.

3.2.4 Virtual Memory Areas (Περιοχές εικονικής μνήμης)

Η περιοχή εικονικών διευθύνσεων (VMA) είναι μια δομή του πυρήνα που τη χρησιμοποιεί, για να διαχειρίζεται περιοχές του χώρου διευθύνσεων μιας διεργασίας. Μια VMA παριστάνει μια ομογενή περιοχή στην εικονική μνήμη μιας διεργασίας: ένα διαδοχικό εύρος από εικονικές διευθύνσεις που έχουν τις

ίδιες σημαίες δικαιωμάτων και αντιγράφονται από το ίδιο αντικείμενο (backed up by the same object). Αποτελείται (τουλάχιστον) από τα εξής:

- Μια περιοχή για τον εκτελέσιμο κώδικα του αρχείου (που συχνά αποκαλείται text)
- Αρκετές περιοχές για να δεδομένα είτε αρχικοποιημένα είτε όχι και τη στοίβα του προγράμματος.
- Μια περιοχή για κάθε ενεργό mapping μνήμης.

Με τον όρο mapping μιας συσκευής εννοούμε την απεικόνιση (αντιστοιχία) ενός εύρους διευθύνσεων χώρου χρήστη στη μνήμη της συσκευής. Όταν η εφαρμογή διαβάζει ή γράφει στο καταχωρημένο όριο διευθύνσεων, στην πραγματικότητα ελέγχει τη συσκευή. Η περιοχή που γίνεται map πρέπει να είναι πολλαπλάσιο του `PAGE_SIZE`, αφού ο πυρήνας χειρίζεται μνήμη με βάση τις σελίδες, και πρέπει να βρίσκεται μέσα στην υπάρχουσα φυσική μνήμη ξεκινώντας από μια διεύθυνση πολλάσια του `PAGE_SIZE`.

3.2.5 Ζώνες Μνήμης (Memory Zones)

Σε ένα ιδανικό υπολογιστικό σύστημα, ένα page frame είναι μια μονάδα μνήμης που μπορεί να χρησιμοποιηθεί για οτιδήποτε, όπως να αποθηκεύσει δεδομένα του χρήστη ή του πυρήνα, να κρατήσει προσωρινά δεδομένα από κάποια συσκευή αποθήκευσης κλπ. Οποιαδήποτε σελίδα δεδομένων (page of data) μπορεί να αποθηκευτεί σε ένα page frame χωρίς όρια.

Παρόλα αυτά, τα σύγχρονα συστήματα έχουν περιορισμούς υλικού που μπορεί να θέτουν όρια στον τρόπο με τον οποίο μπορούν να χρησιμοποιηθούν τα page frames (`struct pages`). Συγκεκριμένα ο πυρήνας του Linux πρέπει να αντιμετωπίσει το γεγονός ότι οι σύγχρονοι 32-bit επεξεργαστές με μεγάλου μεγέθους μνήμη δεν μπορούν να προσπελάσουν απευθείας όλη τη φυσική (πραγματική) μνήμη, γιατί ο χώρος διευθύνσεών τους είναι μικρός.

Για την αντιμετώπιση αυτού του βασικού περιορισμού ο πυρήνας του Linux διαιρεί τη φυσική μνήμη σε ζώνες (memory zones).

- `ZONE_DMA`
- `ZONE_NORMAL`
- `ZONE_HIGHMEM`

Οι δύο πρώτες περιοχές (ZONE_DMA και ZONE_NORMAL) περιέχουν διευθύνσεις μνήμης που μπορούν να προσπελαστούν απευθείας από τον πυρήνα, ενώ αντίθετα η ZONE_HIGHMEM περιέχει page frames που δεν μπορούν να προσπελαστούν απευθείας, λόγω των ορίων που τίθενται από το περιορισμένο μέγεθος του χώρου εικονικών διευθύνσεων.

3.3 Αρχιτεκτονική Εισόδου / Εξόδου

Για την εύρυθμη λειτουργία ενός υπολογιστικού συστήματος, πρέπει να υπάρχουν διάδρομοι δεδομένων που επιτρέπουν στον επεξεργαστή, στην κεντρική μνήμη του συστήματος και στις περιφερειακές συσκευές Ε/Ε να επικοινωνούν. Αυτοί οι διάδρομοι ονομάζονται *buses* (δίαυλοι) και λειτουργούν σαν το πρωτεύον κανάλι επικοινωνίας μέσα στο υπολογιστικό σύστημα.

Σε οποιοδήποτε σύστημα μπορεί κανείς να βρει ένα διάδρομο συστήματος που συνδέει όλες τις εσωτερικές συσκευές. Ένας τυπικός διάδρομος είναι ο PCI (Peripheral Component Interconnect). Συνήθως στο υπολογιστικό σύστημα υπάρχουν διάφοροι τέτοιοι διάδρομοι, διαφορετικών τύπων που συνδέονται με συσκευές που ονομάζονται *bridges*. Δύο πολύ γρήγοροι διάδρομοι είναι αφιερωμένοι στις μεταφορές δεδομένων από και προς την κεντρική μνήμη· ο frontside bus συνδέει τον επεξεργαστή με τον ελεγκτή της μνήμης, ενώ ο backside bus συνδέει τον επεξεργαστή απευθείας στην εξωτερική «γρήγορη μνήμη». Η host bridge συνδέει το διάδρομο συστήματος με τον frontside bus.

Οποιαδήποτε συσκευή Ε/Ε συνδέεται σε ένα και μόνο διάδρομο. Ο τύπος του διαδρόμου επηρεάζει τον εσωτερικό σχεδιασμό της συσκευής, καθώς και το πώς αντιμετωπίζεται από τον πυρήνα. Το μονοπάτι που συνδέει τον επεξεργαστή με μια συσκευή εισόδου / εξόδου λέγεται γενικά I/O bus (διάδρομος εισόδου εξόδου).

3.3.1 I/O ports

Κάθε συσκευή που συνδέεται στο διάδρομο Ε/Ε έχει ένα δικό της σύνολο με διευθύνσεις Ε/Ε που συνήθως λέγονται ακροσημεία (ports) Ε/Ε. Τα I/O ports μπορούν να απεικονιστούν σε διευθύνσεις του physical address space (χώρου φυσικών διευθύνσεων). Μ' αυτόν τον τρόπο ο επεξεργαστής μπορεί να επικοινωνήσει με μια συσκευή Ε/Ε δίνοντας εντολές γλώσσας μηχανής (assembly) που λειτουργούν απευθείας στη μνήμη (όπως MOV, AND, OR κλπ.). Σήμερα οι συσκευές χρησιμοποιούν αυτό το μοντέλο, γιατί είναι γρήγορο και μπορεί να συνδυαστεί με DMA[JC05].

Ένας σημαντικός στόχος για τους σχεδιαστές συστημάτων είναι να προσφέρουν μια ενοποιημένη προσέγγιση στον προγραμματισμό για Ε/Ε, χωρίς να

θυσιάζουν την απόδοση. Έτσι, τα I/O ports της κάθε συσκευής δομούνται σε ένα σύνολο από ειδικούς καταχωρητές (specialized registers). Ο επεξεργαστής γράφει τις εντολές που θέλει να στείλει στη συσκευή στον device control register και διαβάζει μια τιμή που αναπαριστά την εσωτερική κατάσταση της συσκευής από τον device status register. Επίσης, ο επεξεργαστής ανακτά δεδομένα από τη συσκευή διαβάζοντας bytes από τον device input register και γράφει δεδομένα, γράφοντας bytes στον device output register. Για τη μείωση του κόστους, μπορεί η ίδια I/O port να χρησιμοποιηθεί για διαφορετικούς σκοπούς. Για παράδειγμα κάποια bits περιγράφουν την κατάσταση της συσκευής, άλλα αναφέρουν την εντολή που θα εκτελεστεί. Αντίστοιχα η ίδια I/O port μπορεί να χρησιμοποιηθεί σαν ένας καταχωρητής εισόδου ή εξόδου.

Ένα I/O interface (διεπαφή Ε/Ε) είναι ένα κύκλωμα υλικού μεταξύ ενός συνόλου I/O ports και του αντίστοιχου ελεγκτή συσκευής. Λειτουργεί σαν ένας μεταφραστής των δεδομένων μέσα στο I/O port σε εντολές και δεδομένα για τη συσκευή. Σε αντίθετη κατεύθυνση ανιχνεύει αλλαγές στην κατάσταση της συσκευής και ανάλογα ανανεώνει την I/O port που παίζει το ρόλο του status register. Υπάρχουν δύο είδη διεπαφών˙ το Custom I/O interface και το General Purpose I/O interface.

Μια σύνθετη συσκευή μπορεί να χρειάζεται έναν ελεγκτή συσκευής, για να τη διαχειριστεί. Οι δύο βασικοί ρόλοι που παίζει ένας τέτοιος ελεγκτής είναι να:

- Μεταφράζει τις εντολές υψηλού επιπέδου που λαμβάνει από το I/O interface και αναγκάζει τη συσκευή να εκτελέσει συγκεκριμένες εντολές δίνοντας συγκεκριμένα ηλεκτρικά σήματα σ'αυτή.

- Αλλάζει και μεταφράζει σωστά τα σήματα που λαμβάνει από τη συσκευή και μεταβάλλει τις τιμές του status register.

Ένας συνήθισμένος ελεγκτής είναι ο disk controller (ελεγκτής δίσκου), που λαμβάνει εντολές υψηλού επιπέδου, όπως

> "γράψε το συγκεκριμένο block δεδομένων"

από τον μικροεπεξεργαστή (μέσω του I/Ointerface) και τις μετατρέπει σε χαμηλού επιπέδου λειτουργίες όπως

> "τοποθέτησε την κεφαλή στο δεξιό track" και
> "γράψε τα δεδομένα πάνω στο track".

Σήμερα, οι ελεγκτές δίσκων είναι αρκετά περίπλοκοι, αφού μπορούν να κρατήσουν τα δεδομένα σε πολύ γρήγορες «κρυφές» μνήμες και να αναδιατάσσουν τις

εντολές του επεξεργαστή, βελτιστοποιώντας έτσι την πρόσβαση στη συσκευή ανάλογα με τη γεωμετρία της.

Αρκετές συσκευές περιέχουν τη δική τους μνήμη που λέγεται συνήθως I/O shared memory (μοιραζόμενη μνήμη Ε/Ε). Για παράδειγμα όλες οι πρόσφατες κάρτες γραφικών περιέχουν δεκάδες MB μνήμης που χρησιμοποιούνται, για να αποθηκεύσουν την εικόνα, πριν την εμφανίσουν στην οθόνη.

Η πολυπλοκότητα των διαδρόμων Ε/Ε καθιστά περίπλοκη τη διαχείριση των συσκευών με τον κλασικό τρόπο του πυρήνα του Linux. Έτσι νεότερες εκδόσεις του πυρήνα περιέχουν ένα σύνολο δομών δεδομένων που προσφέρουν έναν ενιαίο τρόπο πρόσβασης στους διαδρόμους, στις συσκευές και στο σύστημα. Αυτό το μοντέλο λέγεται device driver model. Περιέχει ειδικά συστήματα αρχείων (sysfs) και δομές δεδομένων στενά συνδεδεμένες με αυτό (kobjects, ksets, subsystems, devices, device drivers, buses, classes κλπ.).

3.3.2 Συσκευές στο Linux

3.3.2.1 Αρχεία συσκευών

Τα αρχεία συσκευών χρησιμοποιούνται από την αρχή της ανάπτυξης του λειτουργικού συστήματος UNIX. Ένα αρχείο συσκευής είναι ουσιαστικά ένα πραγματικό αρχείο με τη διαφορά ότι το inode[1] του δε δείχνει σε blocks δεδομένων αλλά στη διεύθυνση ενός αναγνωριστικού της συσκευής που αναφέρεται.

Παραδοσιακά, τα ειδικά αρχεία συσκευών, χαρακτηρίζονται από δύο αριθμούς· τον major number και τον minor number. Όλες οι συσκευές που έχουν τον ίδιο αριθμό major (άρα και τον ίδιο τύπο), μοιράζονται το σύνολο των file operations, γιατί έχουν τον ίδιο οδηγό συσκευής. Ο δεύτερος αριθμός, ο minor number, αναγνωρίζει μια συγκεκριμένη συσκευή ανάμεσα στο σύνολο αυτών που έχουν τον ίδιο major number. Για παράδειγμα ένα σύνολο σκληρών δίσκων, που τους διαχειρίζεται ο ίδιος ελεγκτής, έχουν τον ίδιο major και διαφορετικούς minor αριθμούς.

Συνήθως, ένα αρχείο συσκευής αντιστοιχίζεται με μια συγκεκριμένη συσκευή (όπως με ένα σκληρό δίσκο, /dev/hda) ή με ένα φυσικό ή εικονικό κομμάτι μιας συσκευής (όπως ένα partition, /dev/hda2). Ωστόσο, σε κάποιες περιπτώσεις, ένα αρχείο συσκευής δεν αντιστοιχεί σε πραγματική συσκευή, αλλά αναπαριστά μια εικονική συσκευή, όπως για παράδειγμα το /dev/null (αρχείο με μηδενικό μέγεθος) που αντιστοιχεί σε μια "μαύρη τρύπα"· όλα τα δεδομένα που γράφονται σε αυτό το αρχείο αγνοούνται (καταστρέφονται).

[1]Οι πληροφορίες που χρειάζονται από το σύστημα αρχείων για να μπορεί να χειριστεί ένα αρχείο περιέχονται σε μία δομή δεδομένων που ονομάζεται *inode*. Κάθε αρχείο διαθέτει το δικό του inode το οποίο χρησιμοποιεί το σύστημα αρχείων για να αναγνωρίσει το αρχείο.

Όνομα	Τύπος	Major	Minor	Περιγραφή
/dev/fd0	char	2	0	Μαλακή δισκέτα
/dev/hda	block	3	0	Πρώτος σκληρός δίσκος IDE
/dev/hda2	block	3	2	2^o partition του $1^{ου}$ σκληρού δίσκου IDE
/dev/hdb	block	3	64	Δεύτερος IDE σκληρός δίσκος
/dev/hdb3	block	3	67	3^o partition του $2^{ου}$ σκληρού δίσκου IDE
/dev/ttyp0	char	3	0	Τερματικό
/dev/console	char	5	1	Κονσόλα
/dev/lp1	char	6	1	Παράλληλος εκτυπωτής
/dev/ttyS0	char	4	64	Πρώτη σειριακή θύρα
/dev/rtc	char	10	135	Ρολόι πραγματικού χρόνου
/dev/null	char	1	3	Κενή συσκευή

Πίνακας 3.1: Παράδειγμα αρχείων συσκευών

Τα αρχεία συσκευών δημιουργούνται μία φορά σε ένα σύστημα. Παρόλα αυτά, σήμερα, ο αριθμός των συσκευών που υπάρχουν ξεπερνά τα όρια του 8-bit major number και έτσι αυξήθηκε το μέγεθος του major number σε 12bits και του minor σε 20 bits. Ο τρόπος χειρισμού των αρχείων συσκευών από το λειτουργικό σύστημα είναι σήμερα δυναμικός. Θα ήταν αδιανόητο να υπάρχουν στον υποκατάλογο /dev (εκεί όπου συνήθως αποθηκεύονται τα αρχεία συσκευών) τα αρχεία που αντιστοιχούν σε όλες τις συσκευές που πιθανώς να συνδεθούν στο υπολογιστικό σύστημα.

3.3.3 Οδηγοί Συσκευών (Device Drivers)

΄Ενας device driver είναι ένα σύνολο λειτουργιών του πυρήνα που επιτρέπει σε μια συσκευή να ανταποκρίνεται στο προγραμματιστικό περιβάλλον το οποίο ορίζεται από το σύνολο των λειτουργιών του Virtual File System (VFS)[2] που ελέγχουν τη συσκευή.

΄Ενας οδηγός συσκευής είναι το σύνολο των λειτουργιών του πυρήνα που ελέγχει τη συσκευή. Οι λειτουργίες αυτές ανήκουν στο VFS και συνθέτουν το προγραμματιστικό περιβάλλον με το οποίο η συσκευή αλληλεπιδρά με τον πυρήνα. Η πραγματική υλοποίηση όλων αυτών των λειτουργιών αφήνεται στον οδηγό συσκευής. Επειδή κάθε συσκευή έχει διαφορετικό ελεγκτή Ε/Ε και επομένως διαφορετικές εντολές και καταστάσεις, οι περισσότερες συσκευές Ε/Ε έχουν τους δικούς τους drivers.

[2]Το VFS είναι ένα στρώμα λογισμικού του πυρήνα που χειρίζεται όλες τις κλήσεις συστήματος συγκριτικά με το σύστημα αρχείων του UNIX

Υπάρχουν πολλοί τύποι οδηγών συσκευών. Διαφέρουν κυρίως ως προς το επίπεδο υποστήριξης που προσφέρουν σε εφαρμογές χώρου χρήστη, καθώς και στις στρατηγικές προσωρινής αποθήκευσης των δεδομένων που συλλέγουν από τις συσκευές. Ένας οδηγός συσκευής δεν περιέχει μόνο τις συναρτήσεις που λειτουργούν τη συσκευή. Πριν να χρησιμοποιηθεί μια συσκευή πρέπει, να γίνουν ορισμένες ενέργειες που τις αναλαμβάνει εξολοκλήρου ο οδηγός.

- Δήλωση του οδηγού και της συσκευής στον πυρήνα.
- Αρχικοποίηση του οδηγού και των δομών που χρειάζεται για να χειριστεί τη συσκευή.
- Παρακολούθηση των λειτουργιών εισόδου εξόδου, ενδιάμεση αποθήκευση δεδομένων.
- Πρόσβαση στη μοιραζόμενη μνήμη.
  Ανάλογα με τον τύπο της συσκευής και του διαδρόμου, η μοιραζόμενη μνήμη της συσκευής στην αρχιτεκτονική του PC μπορεί να απεικονιστεί σε διαφορετικό εύρος φυσικών διευθύνσεων. Για παράδειγμα, όσον αφορά στις συσκευές που συνδέονται στο διάδρομο PCI, η μοιραζόμενη μνήμη απεικονίζεται σε 32-bit φυσικές διευθύνσεις κοντά στο όριο των 4GB.

Ένας οδηγός συσκευής μπορεί να χρησιμοποιήσει το DMA με δύο διαφορετικούς τρόπους· σύγχρονα και ασύγχρονα. Η πρώτη περίπτωση αναφέρεται για παράδειγμα σε μια συσκευή αναπαραγωγής ήχου. Το ασύγχρονο DMA χρησιμοποιείται για παράδειγμα σε μια περίπτωση που μια κάρτα δικτύου λαμβάνει ένα πλαίσιο από ένα τοπικό δίκτυο.

Κάθε μεταφορά δεδομένων με DMA περιλαμβάνει τουλάχιστο έναν απομονωτή μνήμης, που περιέχει τα δεδομένα που θα διαβαστούν ή θα εγγραφούν από τη συσκευή. Γενικά, πριν εκκινηθεί η μεταφορά, ο οδηγός της συσκευής πρέπει να βεβαιωθεί ότι το κύκλωμα του DMA μπορεί να προσπελάσει απευθείας κομμάτια της κεντρικής μνήμης του συστήματος.

Μέχρι τώρα, έχουμε αναλύσει τρεις τύπους διευθύνσεων μνήμης· λογικές, εικονικές και φυσικές. Παρόλα αυτά, υπάρχει και ένας τέταρτος τύπος, που είναι οι διευθύνσεις διαδρόμου (bus addresses). Αντιστοιχούν στις διευθύνσεις μνήμης που χρησιμοποιούνται από όλες τις συσκευές του διαδρόμου εκτός του επεξεργαστή.

Σε μια λειτουργία DMA, η μεταφορά των δεδομένων γίνεται χωρίς τη μεσολάβηση του επεξεργαστή με αποτέλεσμα τα δεδομένα να οδηγούνται απευθείας από το διάδρομο δεδομένων και το κύκλωμα του DMA. Γι' αυτό το λόγο, όταν ο πυρήνας εγκαθιστά μια DMA λειτουργία, πρέπει να γράψει τη διεύθυνση διαδρόμου του απομονωτή μνήμης που μετέχει στη συγκεκριμένη I/O port του DMA ή της συσκευής Ε/Ε.

3.3.4 Direct I/O

Στον πυρήνα του Linux, όπως έχουμε δει μέχρι τώρα, η πρόσβαση ενός αρχείου είτε μέσω του συστήματος αρχείων είτε μέσω αναφοράς στα blocks που είναι αποθηκευμένο είτε μέσω απεικόνισης του αρχείου στη μνήμη φαινομενικά γίνεται με τον ίδιο τρόπο. Υπάρχουν βέβαια εφαρμογές, όπως συστήματα βάσεων δεδομένων υψηλής απόδοσης, που χρειάζεται να έχουν πλήρη έλεγχο στον τρόπο που γίνονται οι μεταφορές των δεδομένων από και προς το μέσο αποθήκευσης.

Αν αυτές οι εφαρμογές «εμπιστευτούν» τον πυρήνα για τη μεταφορά των δεδομένων από το μέσο αποθήκευσης στην κεντρική μνήμη για παράδειγμα, τότε πρώτον αρκετές σελίδες δεδομένων σπαταλώνται, γιατί ήδη υπάρχουν στην κεντρική μνήμη, δεύτερον οι κλήσεις συστήματος read() και write() καθυστερούν, λόγω του readahead και των πλεοναζουσών λειτουργιών που χειρίζονται τη γρήγορη «κρυφή» μνήμη και τρίτων, γίνονται διπλές μεταφορές από το μέσο αποθήκευσης σε έναν απομονωτή στο χώρο του πυρήνα, και από τον απομονωτή αυτόν στο χώρο χρήστη.

Επειδή οι συσκευές block πρέπει να λειτουργούν με DMA και με interrupts, λειτουργίες που μπορούν να εκτελεστούν μόνο σε χώρο πυρήνα, υπάρχει υποστήριξη να παρακαμφθούν οι κλασικές μέθοδοι ανάκτησης δεδομένων με χρήση του μοντέλου Direct I/O (απευθείας Ε/Ε) που είναι ένας τρόπος να παρακαμφθεί στη μεταφορά η page cache.

Σε όλα τα σύγχρονα λειτουργικά συστήματα, όλη η Είσοδος / Έξοδος αποθηκεύεται προσωρινά σε απομονωτές από τον κώδικα του πυρήνα (συχνά από αρκετά επίπεδα από κρυφές μνήμες). Με αυτή την προσωρινή αποθήκευση όλων των δεδομένων εισόδου / εξόδου σαν τυπική διαδικασία, οι εφαρμογές βελτιώνουν την απόδοσή τους, καθώς επιτρέπει σημαντική μείωση των λειτουργιών εισόδου / εξόδου κατά τη διάρκεια λειτουργίας του συστήματος.

Υπάρχουν πολλές ευριστικές συναρτήσεις για τη βέλτιστη αντικατάσταση των δεδομένων στις κρυφές μνήμες, όταν το σύστημα έχει λίγη μόνο ελεύθερη μνήμη (αντικατάσταση σελίδων).

Μειονεκτήματα:

Όταν η διαδικασία αυτή ακολουθηθεί, στην περίπτωση ανάκτησης δεδομένων από κάποιο μέσο αποθήκευσης (σκληρό δίσκο για παράδειγμα), γίνεται DMA από / προς την cache (κρυφή μνήμη) και όχι από / προς τον απομονωτή της πηγής / προορισμού που έχει δεσμευτεί από την εφαρμογή χρήστη και βρίσκεται στη μνήμη του χώρου χρήστη. Αυτές οι αντιγραφές με τη σειρά τους, επιβαρύνουν τον επεξεργαστή και το κόστος της μνήμης, μεταφέροντας τα δεδομένα από μια κρυφή μνήμη του πυρήνα σε ένα κομμάτι μνήμης του χώρου χρήστη για τη λειτουργία της ανάγνωσης για παράδειγμα.

Σε περιπτώσεις εφαρμογών που τηρούν διαδικασίες για αυτοματοποιημένη

προσωρινή αποθήκευση δεδομένων (self caching applications), όπως κάποια συστήματα διαχείρισης βάσεων δεδομένων, η διαδικασία αντιγραφών και προσωρινής αποθήκευσης του πυρήνα είναι άχρηστη, αφού το μόνο που προσφέρει είναι σπατάλη μνήμης, εύρους ζώνης μεταφοράς δεδομένων στους διαδρόμους του συστήματος και άχρηστους κύκλους του επεξεργαστή.

Με ένα αρκετά γρήγορο μέσο αποθήκευσης, ο διάδρομος μνήμης και το εύρος ζώνης μεταφοράς δεδομένων του επεξεργαστή θα είναι ένα σοβαρό bottleneck για τις εφαρμογές που δεν εκμεταλλεύονται την προσωρινή αποθήκευση του πυρήνα.

Οι περιορισμοί που τίθενται στη χρήση του Direct I/O είναι δύο:

- ο απομονωτής από τον οποίο γίνεται η ανάγνωση πρέπει να είναι ευθυγραμμισμένος (aligned) στο blocksize του συστήματος αρχείων (softblocksize).
- το μέγεθος του απομονωτή πρέπει να είναι πολλαπλάσιο του blocksize.

3.4 Block Device Drivers

΄Ενας οδηγός συσκευής block (block device driver) δίνει πρόσβαση σε συσκευές που μεταφέρουν δεδομένα προσβάσιμα με τυχαίο τρόπο σε δεδομένου μεγέθους blocks, όπως ένας σκληρός δίσκος. Ο πυρήνας του Linux βλέπει τις συσκευές block σαν διαφορετικές «εκ βάσεως» από τις συσκευές χαρακτήρων˙ σαν αποτέλεσμα οι συσκευές block έχουν ένα δικό τους χαρακτήρα και μια διακριτή διεπαφή με το υπόλοιπο σύστημα [Kou].

Οι «efficient» συσκευές block αποτελούν ένα σημαντικό στοιχείο απόδοσης ολόκληρου του συστήματος και δεν αρκούνται μόνο σε αναγνώσεις / εγγραφές. Τα σύγχρονα συστήματα μεταφέρουν δεδομένα, που δε χρησιμοποιούνται άμεσα, σε δευτερεύουσα μνήμη αποθήκευσης. Αυτή η μνήμη ονομάζεται εικονική και συνήθως είναι συσκευές δίσκων. Οι οδηγοί συσκευών block είναι ο σύνδεσμος μεταξύ του πυρήνα, της κύριας μνήμης και της δευτερεύουσας αποθήκευσης (secondary storage). ΄Ετσι, μπορούμε να θεωρήσουμε ότι επιτελούν σημαντικό ρόλο στο υποσύστημα της εικονικής μνήμης.

Το μεγαλύτερο μέρος της σχεδίασης του στρώματος για συσκευές block επικεντρώνεται στην απόδοση. Το σύστημα δεν μπορεί να διασφαλίσει την εύρυθμη λειτουργία του, αν το υποσύστημα εισόδου εξόδου για συσκευές block δεν είναι ρυθμισμένο κατάλληλα. Η διεπαφή οδηγού συσκευών block του Linux επιτρέπει τη μέγιστη αξιοποίηση των δυνατοτήτων της συσκευής, αλλά επιβάλλει μεγάλη πολυπλοκότητα.

Για την βελτίωση της απόδοσης των συσκευών αποθήκευσης χρησιμοποιούνται δύο βασικές τεχνικές:

- Χρήση ιεραρχίας μνήμης (caching):

  Η πρόσβαση στις συσκευές αποθήκευσης είναι αργή σε σχέση με την πρόσβαση στη μνήμη. Για καλύτερη απόδοση και ταχύτητα απόκρισης χρησιμοποιούνται κρυφές μνήμες (caches) στην κυρία μνήμη, για να κρατάνε τα δεδομένα των συσκευών αυτών. Οι κρυφές μνήμες χρησιμοποιούνται και στις εγγραφές και στις αναγνώσεις. Η βελτίωση της απόδοσης εξασφαλίζεται από την τοπικότητα της αναφοράς (locality of reference), η οποία ισχύει και στα δεδομένα των αποθηκευτικών συσκευών.

- Χρονοδρομολόγηση αιτήσεων E/E (I/O scheduling):

  Όσο καλή και αν είναι η χρήση των κρυφών μνημών, κάποια στιγμή θα πρέπει να εγγραφούν ή να αναγνωσθούν δεδομένα από την αποθηκευτική συσκευή. Οι λειτουργίες αυτές πραγματοποιούνται με τη δημιουργία αιτήσεων E/E (I/O requests) στην συσκευή αυτή. Στις περισσότερες αποθηκευτικές συσκευές είναι αποδοτικότερο να μην ικανοποιούνται άμεσα οι αιτήσεις αυτές, αλλά να συγκεντρώνεται ένα σύνολο αιτήσεων μέσω κάποιου αλγόριθμου και να ικανοποιούνται μαζικά. Για παράδειγμα στους σκληρούς δίσκους είναι πιο αποδοτικό να συγκεντρώνονται και να ικανοποιούνται μαζί οι αιτήσεις που αφορούν γειτονικά block, ώστε να ελαχιστοποιηθεί η μετακίνηση του βραχίονα που χρησιμοποιείται. Απαραίτητη προϋπόθεση είναι ο παραπάνω αλγόριθμος να μην αυξάνει σε αισθητό βαθμό την καθυστέρηση που βιώνει ο χρήστης, όταν αλληλεπιδρά με τις συσκευές αποθήκευσης. Οι αλγόριθμοι αυτοί ονομάζονται αλγόριθμοι χρονοδρομολόγησης E/E (I/O scheduling alogorithms).

3.4.1 Βασικές δομές δεδομένων του στρώματος block

Σελίδες

Το Linux, όπως και τα περισσότερα λειτουργικά συστήματα, αναλαμβάνει να δημιουργήσει έναν ενιαίο εικονικό χώρο μνήμης στις διεργασίες που εκτελούνται σε αυτό. Για να το πετύχει αυτό, ένας από τους μηχανισμούς που χρησιμοποιεί είναι αυτός της σελιδοποίησης, όπου η φυσική μνήμη του συστήματος χωρίζεται σε σελίδες ίδιου μεγέθους. Η τιμή του μεγέθους της σελίδας για την αρχιτεκτονική i386 είναι 4096 bytes. Η δομή δεδομένων που ορίζεται για την περιγραφή της κάθε φυσικής σελίδας είναι η `struct page`.

Block μονάδες Ε/Ε

Η βασική μονάδα Ε/Ε για τις συσκευές block είναι η δομή `struct bio`, η ονομασία της οποίας προέρχεται από τη φράση Block I/O. Οποιοδήποτε κομμάτι του Linux θέλει να επικοινωνήσει με την συσκευή block θα χρησιμοποιήσει, έμμεσα ή άμεσα, την δομή αυτή. Ένα αντικείμενο bio περιέχει ένα σύνολο σελίδων, οι οποίες σχετίζονται με την εν λόγω μεταφορά δεδομένων. Συγκεκριμένα, σε περίπτωση αποθήκευσης δεδομένων στη συσκευή οι σελίδες αυτές περιέχουν τα δεδομένα προς εγγραφή. Αντίθετα, σε περίπτωση ανάκτησης δεδομένων, οι σελίδες αυτές αποτελούν τον χώρο μνήμης, στον οποίο θα μεταφερθούν τα δεδομένα. Οι σελίδες που σχετίζονται με μια μονάδα bio περιέχονται σε έναν πίνακα με στοιχεία τύπου `struct bio_vec`. Η δομή αυτή ορίζεται ως εξής:

```
struct bio_vec {
   struct page *bv_page;
   unsigned int bv_len;
   unsigned int bv_offset;
}
```

Εκτός από τον ίδιο τον πίνακα η δομή `struct bio` περιέχει ορισμένα πεδία για την διαχείριση του πίνακα αυτού.

3.4.2 Block αιτήσεις Ε/Ε

Τις περισσότερες φορές οι οδηγοί συσκευών block δεν επεξεργάζονται αυτόνομες μονάδες bio, αλλά ένα σύνολο από αυτές, αφού, όπως προαναφέρθηκε, το Linux χρησιμοποιεί χρονοδρομολόγηση Ε/Ε, ώστε να ομαδοποιεί τις αιτήσεις κατάλληλα. Οι μονάδες bio συλλέγονται σε δομές `struct request`, οι οποίες και μεταφέρονται στον οδηγό της συσκευής. Κάθε αντικείμενο *request* περιέχει ένα σύνολο από αντικείμενα τύπου *bio*, τα οποία συνδέονται μέσω μια λίστας. Η λίστα αυτή υλοποιείται με τη βοήθεια των πεδίων `struct bio *bio` και `struct bio *biotail` της δομής *request* και των πεδίων `bi_next` των δομών *bio*. Επιπρόσθετα, ορίζεται και ένα πεδίο *cbio*, το οποίο αντιστοιχεί στην τρέχουσα μονάδα *bio*.

Η δομή request περιέχει και κάποια ακόμα πεδία, που έχουν βοηθητικό ρόλο, και μπορούν να χρησιμοποιηθούν είτε από το στρώμα συσκευών block είτε από το χρονοδρομολογητή Ε/Ε είτε ακόμα και από τον οδηγό της συσκευής.

3.4.3 Ουρές αιτήσεων

Η δομή δεδομένων που είναι υπεύθυνη για την συλλογή και αποθήκευση των αιτήσεων είναι η `struct_request_queue` ή ισοδύναμα η `request_queue_t`. Η δομή αυτή περιέχει όλες τις πληροφορίες και συναρτήσεις που είναι απαραίτητες για κάθε πτυχή της υλοποίησης του στρώματος block στο Linux. Ο σχεδιασμός είναι τέτοιος, ώστε να μπορεί να προσαρμοστεί σε κάθε πιθανή ανάγκη μιας συσκευής block, αλλά και για την υποστήριξη πολλαπλών αλγορίθμων χρονοδρομολόγησης. Ο αλγόριθμος που χρησιμοποιείται για την χρονοδρομολόγηση Ε/Ε, ονομάζεται και αλγόριθμος ανελκυστήρα (elevator algorithm). Για την υποστήριξη πολλαπλών αλγόριθμων ορίστηκε μια διεπαφή για τη συνεργασία με το στρώμα block του Linux. Κάθε τέτοιος αλγόριθμος θα πρέπει να υλοποιήσει ένα σύνολο από συναρτήσεις με καθορισμένο σκοπό. Οι συναρτήσεις αυτές μαζί με κάποια άλλα βοηθητικά στοιχεία για την υλοποίηση του αλγορίθμου περιέχονται στη δομή `elevator_t`. Η δομή αυτή χρησιμοποιείται από την ουρά, ώστε να εφαρμόστει με διάφανο τρόπο η πολιτική του κάθε αλγορίθμου. Σε κάθε ουρά υπάρχει ένα πεδίο elevator τύπου `elevator_t` για αυτόν τον σκοπό.

3.4.4 Διεπαφή οδηγού συσκευής block

Κατά τη δημιουργία ενός οδηγού συσκευής block, χρησιμοποιείται μια συγκεκριμένη διεπαφή για την επικοινωνία με το στρώμα block του Linux. Σε αυτή τη παράγραφο, θα σκιαγραφηθούν μερικές βασικές συναρτήσεις της διεπαφής αυτής.

Διεπαφή gendisk

Οι περισσότερες λειτουργίες για μια συσκευή block υλοποιούνται από το στρώμα gendisk, το οποίο προσφέρει ορισμένες συναρτήσεις για την αλληλεπίδραση με τους οδηγούς των συσκευών block. Για κάθε συσκευή ο οδηγός θα πρέπει να χρησιμοποιήσει μια μεταβλητή τύπου struct gendisk. Για τη δέσμευση των πόρων που χρειάζονται για μια τέτοια μεταβλητή χρησιμοποιείται η συνάρτηση `alloc_disk()`. Για την απελευθέρωση των πόρων αυτών θα πρέπει να κληθεί η συνάρτηση `put_disk()`. Μετά την αρχικοποίηση ενός gendisk η συσκευή που αναπαριστά μπορεί να εισαχθεί στο σύστημα με τη συνάρτηση `add_disk()`. Η αντίθετη διαδικασία πραγματοποιείται με τη συνάρτηση `del_gendisk()`.

Ουρές αιτήσεων

Κάθε οδηγός συσκευής block μπορεί να χρησιμοποιείται για περισσότερες από μια συσκευές. Για κάθε συσκευή ο οδηγός, στη συντριπτική πλειοψηφία των περιπτώσεων, δημιουργεί μια ουρά, η οποία απεικονίζεται μέσω του τύπου `struct request_queue`. Η δέσμευση μνήμης για τη δομή αυτή αλλά και η αρχικοποίησή της πραγματοποιείται μέσω της συνάρτησης `blk_init_queue()`.

Η συνάρτηση `request_fn` είναι η βασική συνάρτηση που χρειάζεται να υλοποιήσει ο οδηγός και ως σκοπό έχει την επεξεργασία των αιτήσεων. Η μεταβλητή lock είναι ουσιαστικά ένας σηματοφορέας, μια μεταβλητή συγχρονισμού, η οποία προστατεύει την ουρά από ταυτόχρονη πρόσβαση. Όταν ο οδηγός δε χρειάζεται πλέον την ουρά μπορεί να την καταστρέψει χρησιμοποιώντας τη συνάρτηση `blk_cleanup_queue()`.

Η συνάρτηση `request_fn` θα κληθεί, όταν το στρώμα block του Linux αποφανθεί ότι θα πρέπει να προωθηθούν ορισμένες αιτήσεις από την ουρά στη συσκευή. Η κλήση της συνάρτησης αυτής θα γίνει, αφού πρώτα η ουρά έχει κλειδωθεί μέσω του σηματοφορέα lock.

Επεξεργασία αιτήσεων

Όταν κληθεί η συνάρτηση `request_fn()`, για να εξαγάγει από την ουρά την επόμενη αίτηση, θα πρέπει να καλέσει τη συνάρτηση:

```
struct request *elv_next_request(request_queue_t *q);
```

Η συνάρτηση αυτή θα επιστρέψει την επόμενη αίτηση προς επεξεργασία ή NULL, αν δεν υπάρχει καμία τέτοια αίτηση. Για την επεξεργασία της αίτησης ο οδηγός θα πρέπει να εξαγάγει διαδοχικά τις μονάδες *bio* που απαρτίζουν την αίτηση. Για την διαδικασία αυτή έχει οριστεί μια μακροεντολή, ώστε να είναι δυνατή η αλλαγή του τρόπου αποθήκευσης των μονάδων σε μια αίτηση, χωρίς να χρειαστεί να αλλάξουν όλοι οι χαμηλού επιπέδου οδηγοί. Η μακροεντολή αυτή ορίζεται ως εξής:

```
rq_for_each_bio(struct bio *bio, struct request *rq)
```

Καθώς προχωράει η επεξεργασία της αίτησης είναι θεμιτό να ανανεώνεται η δομή της αίτησης, σχετικά με την κατάστασή της. Για αυτό το λόγο χρησιμοποιούνται οι συναρτήσεις:

```
end_that_request_first()
end_that_request_chunk()
```

Οι συναρτήσεις αυτές επιστρέφουν 0 σε περίπτωση που η επεξεργασία της αίτησης αυτής έχει ολοκληρωθεί και 1 σε περίπτωση που δεν έχει ολοκληρωθεί. Όταν ολοκληρωθεί η επεξεργασία της αίτησης, ο οδηγός θα πρέπει να εξαγάγει την αίτηση από την ουρά και να την τερματίσει. Οι συναρτήσεις που

χρησιμοποιούνται είναι οι:

```
blkdev_dequeue_request();
end_that_request_last();
```

Η πρώτη χρησιμοποιείται για την εξαγωγή της αίτησης από την ουρά, ενώ η δεύτερη για να ενημερώσει το στρώμα block ότι περατώθηκε η επεξεργασία της αίτησης από τον οδηγό της συσκευής.

Επεξεργασία μονάδων bio

Οι μονάδες E/E bio συνοδεύονται και αυτές από ορισμένες συναρτήσεις, οι οποίες είτε χρησιμοποιούνται από το στρώμα block του Linux είτε από τον ίδιο τον οδηγό της συσκευής. Οι οδηγοί πολλές φορές είναι απαραίτητο να επεξεργαστούν όλες τις σελίδες που περιέχονται σε μία μονάδα E/E bio. Για τη διευκόλυνσή τους έχει οριστεί η μακροεντολή `bio_for_each_segment()`, η οποία χρησιμοποιεί το πεδίο `bi_cnt` της μονάδας bio.

Μια άλλη βασική συνάρτηση για τις μονάδες bio είναι η

```
bio_endio();
```

Η συνάρτηση αυτή αναβαθμίζει τα κατάλληλα πεδία της μονάδας bio μετά από τη μερική ή ολική επεξεργασία της. Σε περίπτωση που η επεξεργασία ολοκληρωθεί και δεν υπάρχουν άλλα δεδομένα, η συνάρτηση αυτή φροντίζει για τον τερματισμό της μονάδας αυτής. Η `bio endio()`, στις περισσότερες περιπτώσεις δε χρησιμοποιείται απευθείας από τον οδηγό, αλλά καλείται από τις συναρτήσεις `end_that_request_first()` ή `end_that_request_chunk()`.

Παραμετροποίηση της ουράς αιτήσεων

Η διεπαφή του Linux για τους οδηγούς συσκευών προσφέρει πολλές δυνατότητες για ρύθμιση της λειτουργίας του στρώματος block. Οι ρυθμίσεις αυτές είναι ξεχωριστές για κάθε συσκευή και γίνονται στο επίπεδο της ουράς αιτήσεων. Παρακάτω παρουσιάζονται ορισμένα παραδείγματα τέτοιων ρυθμίσεων. Η παρουσίαση αυτή σε καμία περίπτωση δεν είναι πλήρης. Σκοπός της είναι η κατανόηση του τρόπου με τον οποίο είναι δυνατόν να ρυθμιστεί η ουρά αιτήσεων, ώστε να ανταποκρίνεται στις ανάγκες κάθε συσκευής. Η κάθε συσκευή τις περισσότερες φορές μπορεί να επεξεργαστεί έναν περιορισμένο αριθμό από τομείς κάθε φορά. Συνεπώς, για να μη χρειαστεί να διασπαστούν οι αιτήσεις που παράγει το στρώμα block του Linux, ώστε να τις επεξεργαστεί η συσκευή, ο κάθε οδηγός μπορεί να ορίσει το μέγιστο αριθμό τομέων που μπορεί να περιέχει μια αίτηση. Η λειτουργία αυτή γίνεται μέσω της συνάρτησης:

```
blk_queue_max_sectors();
```

Στις περισσότερες περιπτώσεις οι συσκευές που χρησιμοποιούνται έχουν μέγεθος τομέα 512 bytes. Ωστόσο υπάρχουν ορισμένες συσκευές στις οποίες το

μέγεθος του τομέα έχει άλλη τιμή. Η ρύθμιση της τιμής αυτής μπορεί να γίνει μέσω της συνάρτησης:

`blk_queue_hardsect_size()`

Πολλές φορές είναι απαραίτητο ο οδηγός να κάνει ορισμένες λειτουργίες, ώστε να προετοιμάσει την κάθε αίτηση, πριν την επεξεργαστεί μέσω της συνάρτησης `request_fn`. Η παρακάτω συνάρτηση, επιτρέπει στον οδηγό να ορίσει μια συνάρτηση για την προεπεξεργασία μιας αίτησης:

`blk_queue_prep_rq()`

Κεφάλαιο **4**

Σχεδιασμός

4.1 Μια τυπική δικτυακή συσκευή αποθήκευσης

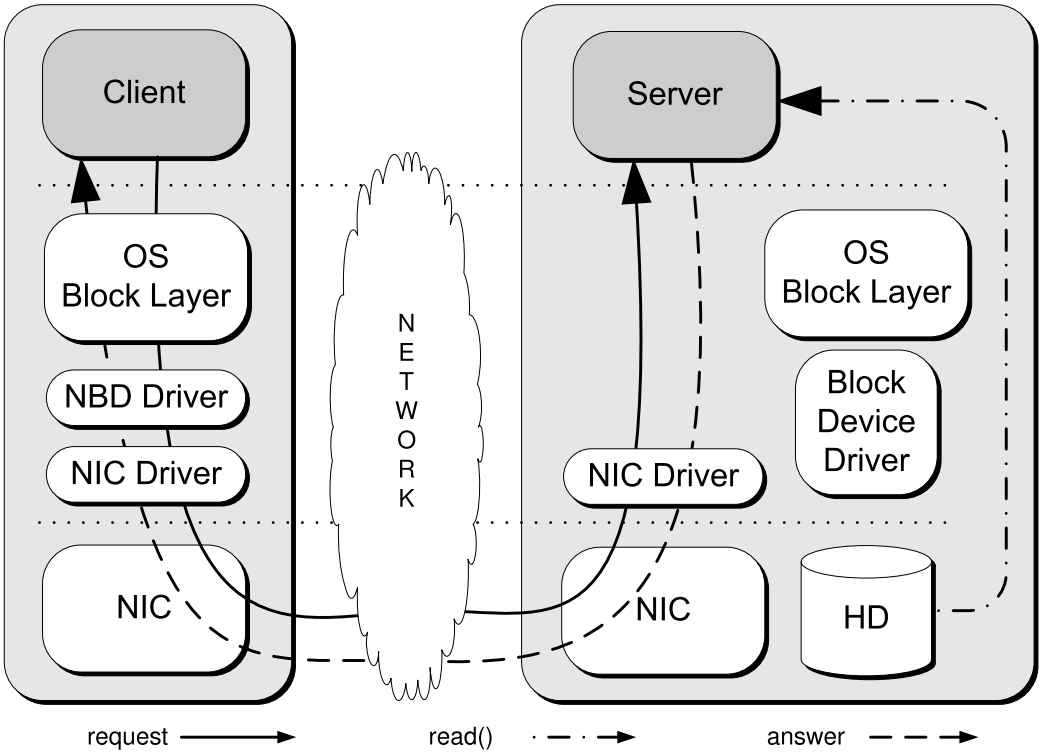
Μια δικτυακή συσκευή block είναι μια αφαίρεση σε επίπεδο λογισμικού στην τοπική αποθήκευση στο στρώμα των συσκευών block. Η ιδέα στην οποία βασίζεται είναι η προσομοίωση μιας τοπικής συσκευής στο ανώτερο στρώμα, ενώ δεσμεύει και αποδεσμεύει πόρους μέσω δικτύου για απομεμακρυσμένη χρησιμοποίησή τους. Χρησιμοποιείται ευρέως για αποθήκευση πλεονάζουσας πληροφορίας.

Στον πυρήνα του Linux συμπεριλαμβάνεται μια τέτοια δικτυακή συσκευή αποθήκευσης. Χρησιμοποιώντας ένα μοντέλο εξυπηρετητή – πελάτη, επιτρέπει σε ένα σύστημα να χρησιμοποιήσει μια αποθηκευτική συσκευή ενός άλλου απομακρυσμένου συστήματος. Το πρώτο σύστημα αποτελεί τον πελάτη και το δεύτερο τον εξυπηρετητή. Μια τέτοια συσκευή απαντάται στη βιβλιογραφία σαν Network Block Device (NBD)[Pav].

Ο εξυπηρετητής αποτελείται αποκλειστικά από μια εφαρμογή χώρου χρήστη που αναλαμβάνει να ικανοποιεί αιτήσεις block που του αποστέλλονται από τον πελάτη. Ο πελάτης αποτελείται από μια εφαρμογή χώρου χρήστη και έναν οδηγό συσκευής αποθήκευσης στο χώρο πυρήνα. Η δικτυακή συσκευή αποθήκευσης NBD προσομοιώνει μια συσκευή block, όπως ένας σκληρός δίσκος ή ένα κομμάτι ενός σκληρού δίσκου στο τοπικό σύστημα αλλά στην πραγματικότητα, ανακτά τα δεδομένα μέσω δικτύου.

4.1.1 Πελάτης

Ο πελάτης πρέπει να προσομοιώσει μια συσκευή αποθήκευσης στο τοπικό σύστημα, να μετατρέψει τις αιτήσεις για ανάκτηση δεδομένων από την εικονική συσκευή σε αιτήσεις για δικτυακή ανάκτηση δεδομένων και να λάβει την εξυπηρετημένη αίτηση επιτυχώς.



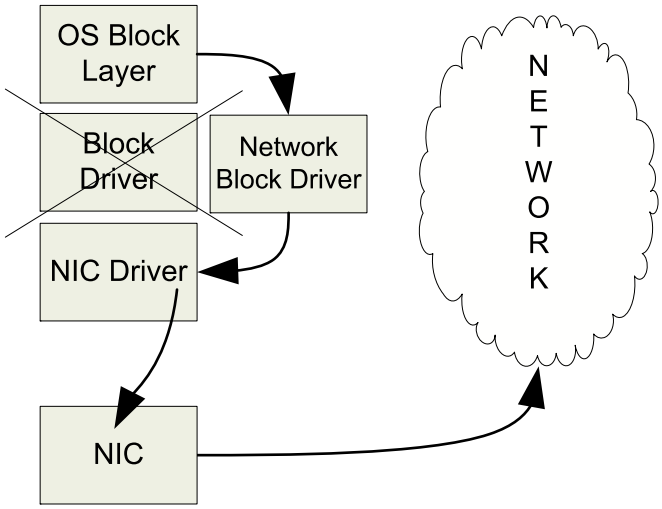
Σχήμα 4.1: Μια τυπική δικτυακή συσκευή block

Όπως βλέπουμε στο σχήμα 4.2 έχουμε μία ένα προς ένα αντιστοίχηση αιτήσεων για δεδομένα από το μέσο αποθήκευσης σε αιτήσεις που κατευθύνονται προς το δίκτυο. Στο σχήμα 4.3 φαίνονται αναλυτικά οι αντιγραφές δεδομένων και το μονοπάτι που ακολουθούν στον πελάτη, από τη στιγμή που θα γίνει μια αίτηση για ανάγνωση μέχρι τη στιγμή που αυτή θα εξυπηρετηθεί.

Ο πυρήνας της εφαρμογής του πελάτη είναι: Πιο συγκεκριμένα για το σχήμα 4.3, έχουμε:

1. Στο βήμα 1 μια εφαρμογή χρήστη εκτελεί μια κλήση συστήματος για ανάγνωση δεδομένων από μια συσκευή αποθήκευσης (που στην προκειμένη περίπτωση είναι δικτυακή).

2. Ο πυρήνας του λειτουργικού συστήματος ελέγχει αν τα δεδομένα που ζητούνται βρίσκονται σε κάποια προσωρινή ή "κρυφή" μνήμη (page_cache) και ανάλογα προετοιμάζει το κομμάτι που αναλαμβάνει τέτοιες αιτήσεις (OS block layer). Σε περίπτωση που τα δεδομένα υπάρχουν σε προσωρινή μνήμη συνεχίζει στο βήμα 11.



Σχήμα 4.2: Η αρχιτεκτονική του Πελάτη

3. Εδώ οι αιτήσεις για block (blk_requests) μεταφράζονται στις αιτήσεις του μοντέλου επικοινωνίας με τον εξυπηρετητή και προωθούνται στον οδηγό συσκευής της κάρτας δικτύου με όποιες αλλαγές χρειάζονται, για να γίνει η αντιγραφή στην κάρτα δικτύου.

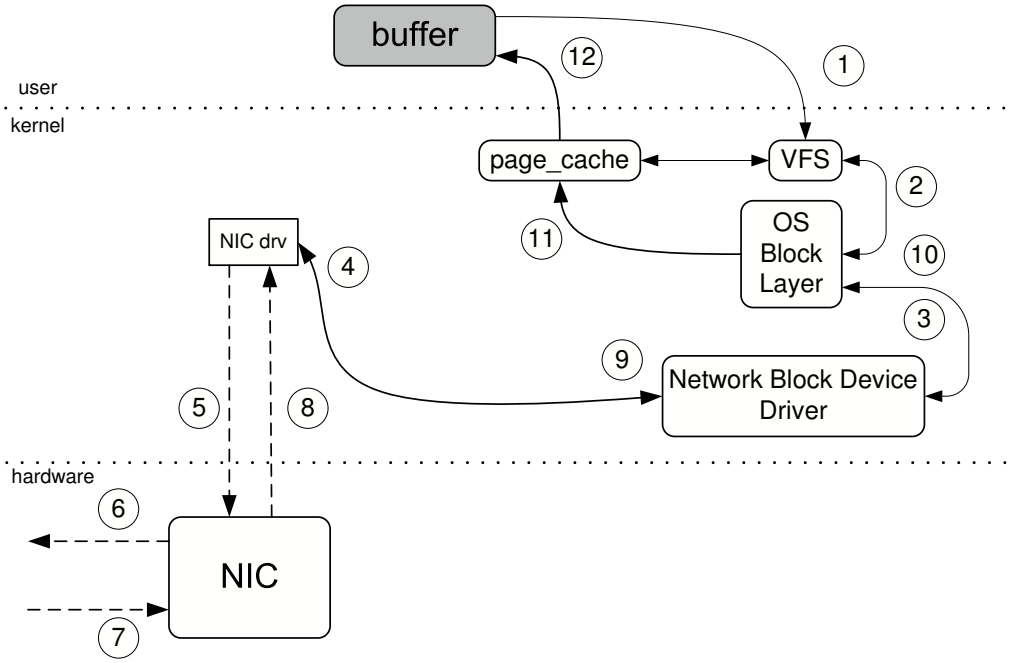
4. Εδώ γίνεται DMA σε μια προσωρινή μνήμη του οδηγού της κάρτας δικτύου, για να προετοιμαστεί η αποστολή στο καλώδιο.

5. Αντιγράφεται η αίτηση στη στατική μνήμη που, ενδεχομένως, διαθέτει η κάρτα δικτύου.

6. Αποστέλλεται η αίτηση στο καλώδιο.

7. Με τη λήψη των δεδομένων από τον εξυπηρετητή, τα δεδομένα βρίσκονται σε μια προσωρινή μνήμη της κάρτας δικτύου.

8. Από τη μνήμη αυτή, αντιγράφονται με dma σε χώρο μνήμης του πυρήνα, όπου αποκόπτονται οι επικεφαλίδες των πακέτων δικτύου ή γίνονται οι απαραίτητες διαδικασίες, ώστε τα πακέτα να μην περιέχουν δεδομένα πληροφοριών για το δίκτυο.



Σχήμα 4.3: Ο Πελάτης

9. Τα δεδομένα περνούν στον έλεγχο του οδηγού δικτυακής συσκευής block. Μεταφράζεται το μήνυμα σε αίτηση block.

10. Το στρώμα χειρισμού συσκευών block του λειτουργικού αναλαμβάνει την αίτηση και τα δεδομένα που αφορούν στα block που ζητήθηκαν.

11. Τα δεδομένα αντιγράφονται στην προσωρινή (κρυφή) μνήμη του συστήματος.

12. Τέλος, τα δεδομένα γράφονται στο κομμάτι μνήμης της διεργασίας που τα ζήτησε και η αίτηση είναι πλέον ικανοποιημένη μαζί με την κλήση συστήματος για ανάγνωση.

4.1.2 Εξυπηρετητής

Η εφαρμογή του χώρου χρήστη που εξυπηρετεί αιτήσεις πρέπει να μπορεί να αναγνωρίσει τι ζητά ο πελάτης, να το ανακτήσει από το φυσικό μέσο αποθήκευσης και να το στείλει στο δίκτυο. Αυτό γίνεται σε 3 βήματα.

- Ανάγνωση της αίτησης.

- Ανάκτηση των δεδομένων.
- Αποστολή στο δίκτυο.

Αντίστοιχα σε ψευδοκώδικα ο πυρήνας της εφαρμογής είναι:

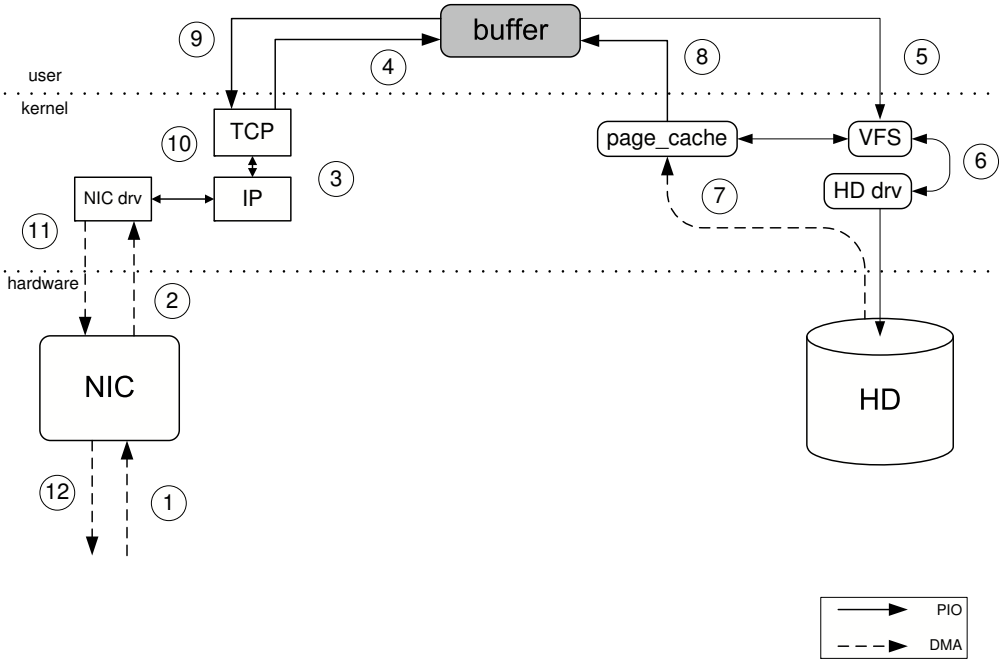
```
bd = open(/dev/local_block_device, O_READONLY);
for (;;) {
   read(remote_node,&request);
   read(bd, request, &buffer);
   send(buffer, remote_node);
}
```

όπου `remote_node` ο πελάτης, `request` τα δεδομένα που ζητά, bd ο block descriptor της φυσικής συσκευής και buffer το κομμάτι της μνήμης που έχει δεσμευτεί στο χώρο χρήστη από την εφαρμογή του εξυπηρετητή, όπου θα αποθηκευτούν προσωρινά τα δεδομένα και από το οποίο θα σταλούν στον πελάτη.

Στο σχήμα 4.4 φαίνεται η διαδρομή που ακολουθούν τα δεδομένα σε λογικό επίπεδο, από τη στιγμή που λαμβάνεται η αίτηση του πελάτη μέχρι να σταλεί η απάντηση στο δίκτυο.

Πιο συγκεκριμένα, λαμβάνεται η αίτηση από το καλώδιο στην προσωρινή μνήμη αποθήκευσης της κάρτας δικτύου (βήμα 1). Από εκεί αντιγράφεται με DMA στην κεντρική μνήμη στο χώρο πυρήνα (βήμα 2), όπου αποκόπτονται οι επικεφαλίδες του πακέτου και παραμένει το μήνυμα της αίτησης για ανάγνωση δεδομένων από τη δικτυακή συσκευή αποθήκευσης (βήμα 3). Τα δεδομένα αντιγράφονται στο χώρο χρήστη, στο χώρο δηλαδή της κεντρικής μνήμης που έχει δεσμευτεί από την εφαρμογή του εξυπηρετητή έτσι, ώστε να γίνει η αναγνώριση της αίτησης δηλαδή να καθοριστεί ποια blocks πρέπει να διαβαστούν από το φυσικό μέσο αποθήκευσης (βήμα 4).

Στο βήμα 5 η εφαρμογή του εξηπηρετητή ζητά διαφανώς από το λειτουργικό σύστημα τα blocks που αναφέρει η αίτηση του πελάτη. Το στρώμα του λειτουργικού συστήματος που χειρίζεται τις συσκευές Ε/Ε, ανατρέχει πρώτα στην προσωρινή μνήμη (page_cache) για να διαγνώσει αν τα δεδομένα που ζητούνται βρίσκονται ήδη εκεί. Αν ισχύει αυτό, τότε η εφαρμογή συνεχίζει τη λειτουργία της από το βήμα 8. Αλλιώς, το στρώμα του λειτουργικού συστήματος που χειρίζεται συσκευές αποθήκευσης προγραμματίζει τη μηχανή DMA του μέσου αποθήκευσης να αντιγράψει τα block που ζητούνται στην προσωρινή μνήμη στο χώρο πυρήνα, μέσω του οδηγού συσκευής αποθήκευσης (βήμα 6). Η αντιγραφή DMA γίνεται (βήμα 7), και από την προσωρινή μνήμη τα



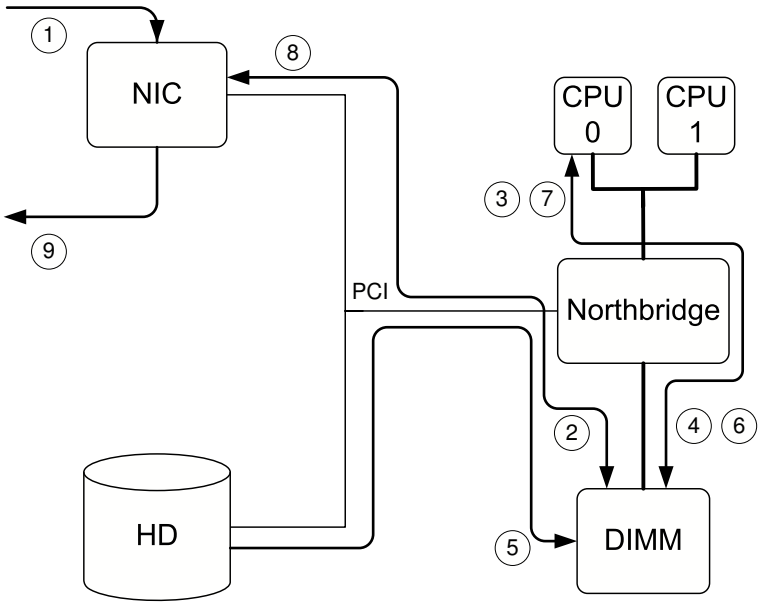
Σχήμα 4.4: Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής nbd (λογικό επίπεδο)

δεδομένα αντιγράφονται στη μνήμη χώρου χρήστη που είναι δεσμευμένη από την εφαρμογή του εξυπηρετητή (βήμα 8).

Στη συνέχεια αντιγράφονται εκ νέου στο χώρου του πυρήνα, όπου προστίθενται επικεφαλίδες και ό,τι άλλο χρειάζεται για να μεταφερθούν τα δεδομένα ως πακέτο μέσω δικτύου (βήματα 9,10). Από εκεί με μια κλήση DMA αντιγράφονται τα δεδομένα στην κάρτα δικτύου και πάλι με DMA στο καλώδιο (βήματα 11,12).

Στο φυσικό επίπεδο το μονοπάτι των δεδομένων φαίνεται στο σχήμα 4.5. Αυτό που παρατηρείται είναι ότι υπάρχουν διαδρομές από και προς τον επεξεργαστή, πράγμα που προσθέτει σημαντική επιβάρυνση στην υπολογιστική δύναμη του συστήματος του εξυπηρετητή.

Πιο αναλυτικά, στο βήμα 1 η αίτηση λαμβάνεται από την κάρτα δικτύου και αντιγράφεται (βήμα 2) στην κεντρική μνήμη. Για να "αποκωδικοποιηθεί" (μεταφραστεί) η αίτηση απαιτείται χρήση του επεξεργαστή και ένα αντίγραφο σε άλλο χώρο της μνήμης (βήματα 3, 4). Με μια κλήση DMA αντιγράφονται



Σχήμα 4.5: Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής nbd (φυσικό επίπεδο)

τα δεδομένα από το μέσο αποθήκευσης στην κεντρική μνήμη του συστήματος (βήμα 5) με ένα επιπλέον αντίγραφο (βήματα 6, 7).

Από την κεντρική μνήμη αντιγράφονται στην κάρτα δικτύου και από εκεί στο δίκτυο (βήματα 8, 9).

Τα βήματα 6 και 7 αναφέρονται στην αντιγραφή των δεδομένων από την `page_cache` στο χώρο χρήστη (buffer) ενώ θα μπορούσαν να παραλειφθούν με χρήση απευθείας E/E ή αλλιώς Direct I/O. Έτσι, η αντιγραφή από το μέσο αποθήκευσης στο χώρο της μνήμης, από όπου θα αποσταλούν τα δεδομένα, θα μπορούσε να γίνει σε ένα βήμα (στο 5), παραλείποντας τα 6, 7. Με αυτόν τον τρόπο η επιβάρυνση του επεξεργαστή θα περιοριζόταν στην ανάγνωση και μετάφραση της αίτησης από το δίκτυο (βήματα 3, 4).

Η χρήση Direct I/O είναι μια σημαντική βελτίωση στη συσκευή δικτυακής αποθήκευσης που ακολουθείται στα δύο επόμενα μοντέλα συσκευών.

4.1.3 Δίκτυο

΄Ενα σημαντικό, ίσως και το σημαντικότερο, κομμάτι του σχεδιασμού μιας δικτυακής συσκευής αποθήκευσης είναι το κομμάτι που αναφέρεται στη δικτυακή μεταφορά των δεδομένων. Η κλασική NBD του πυρήνα του Linux ακολουθεί την πεπατημένη οδό της μεταφοράς με το πρωτόκολλο TCP/IP με χρήση sockets για επικοινωνία ανάμεσα στον πελάτη και τον εξυπηρετητή.

Αυτό προσδίδει σημαντική επιβάρυνση στον επεξεργαστή, αφού ο πυρήνας καλείται να επεξεργαστεί τα πακέτα TCP/IP (επικεφαλίδες, επιπλέον αντιγραφές για λήψη / αποστολή), πράγμα ασύμφορο για μια δικτυακή συσκευή αποθήκευσης, όπου η ταχύτητα πρέπει να συμβαδίζει με την αξιοπιστία.

Μια σημαντική βελτίωση του συγκεκριμένου μοντέλου δικτυακής συσκευής αποθήκευσης είναι η χρησιμοποίηση διαφορετικού δικτυακού μέσου.

4.2 Μια δικτυακή συσκευή αποθήκευσης για Myrinet

Εδώ, το μοντέλο διαφοροποιείται, όσον αφορά στο δικτυακό κομμάτι του σχεδιασμού. Θα μπορούσε να χρησιμοποιηθεί το Ethernet Emulation του Myrinet, αλλά, παρόλα αυτά, το μόνο που θα μπορούσε να βελτιωθεί θα ήταν το latency μετάδοσης των δεδομένων στο φυσικό μέσο (στο καλώδιο). Γι' αυτό το λόγο θεωρήθηκε απαραίτητο να χρησιμοποιηθεί το μοντέλο δικτυακής επικοινωνίας με μηνύματα του Myrinet[Kou].

Συγκριτικά με την προηγούμενη συσκευή (την κλασική NBD), όπου η δικτυακή μεταφορά και μετάδοση γίνεται με το TCP/IP και το Ethernet, η συσκευή αποθήκευσης πάνω από Myrinet, με χρήση του GM, βελτιώνει κατά πολύ την ταχύτητα ανάκτησης δεδομένων, αφού πρώτον παραλείπονται τα βήματα μεσολάβησης του πυρήνα στην αντιγραφή από το μέσο αποθήκευσης στην κεντρική μνήμη του συστήματος και δεύτερον αποδεσμεύεται ο επεξεργαστής από το DMA που πρέπει να γίνει για την αποστολή / λήψη μηνυμάτων / πακέτων μέσω δικτύου. Παρόλα αυτά, τα δεδομένα περνούν και πάλι πάνω από το δίαυλο PCI με αποτέλεσμα να υπάρχει καθυστέρηση στο χρόνο που χρειάζεται να εξυπηρετηθεί η αίτηση.

4.2.1 Εξυπηρετητής

Ο εξυπηρετητής για μια συσκευή αποθήκευσης σε δίκτυο Myrinet δε διαφέρει σε τίποτα από τον προηγούμενο στο επίπεδο των απαιτήσεων. Ο ψευδοκώδικας είναι:

```
bd = open(/dev/local_block_device, O_READONLY, O_DIRECT);
for (;;) {
   gm_recv(&request);
   read(bd, request, &buffer);
   gm_send(buffer, remote_node);
}
```

Εδώ παρατηρούμε δύο σημαντικές διαφορές. Πρώτον, η ανάγνωση των δεδομένων γίνεται σε ένα βήμα από το φυσικό μέσο αποθήκευσης και όχι σε 3 όπως στο προηγούμενο μοντέλο (σχήμα 4.7). Ο πυρήνας δε λαμβάνει μέρος στο δικτυακό κομμάτι της εφαρμογής, αφού το Myrinet χρησιμοποιεί το μοντέλο επικοινωνίας χώρου χρήστη. Έτσι, η αντιγραφή των δεδομένων από το μέσο αποθήκευσης γίνεται απευθείας σε ένα κομμάτι του χώρου χρήστη (buffer, βήμα 4). Η αντιγραφή στη στατική μνήμη του LANai γίνεται απευθείας με DMA που καλείται από το μικροεπεξεργαστή της διεπαφής δικτύου Myrinet (LANai). Επομένως τα δεδομένα παρακάμπτουν τον πυρήνα κατά τη μεταφορά τους από το χώρο χρήστη στη στατική (μοιραζόμενη) μνήμη της κάρτας δικτύου και αντίστροφα.

Αλλά ας δούμε τη διαδρομή πιο αναλυτικά. Στο λογικό επίπεδο (σχήμα 4.6):

1. Η αίτηση λαμβάνεται από το δίκτυο (βήμα 1) με DMA στην ουρά εισερχομένων της διεπαφής Myrinet.

2. Από εκεί, αντιγράφεται με DMA απευθείας στη μνήμη του χώρου χρήστη που έχει δεσμευτεί από την εφαρμογή του εξυπηρετητή (βήμα 2).

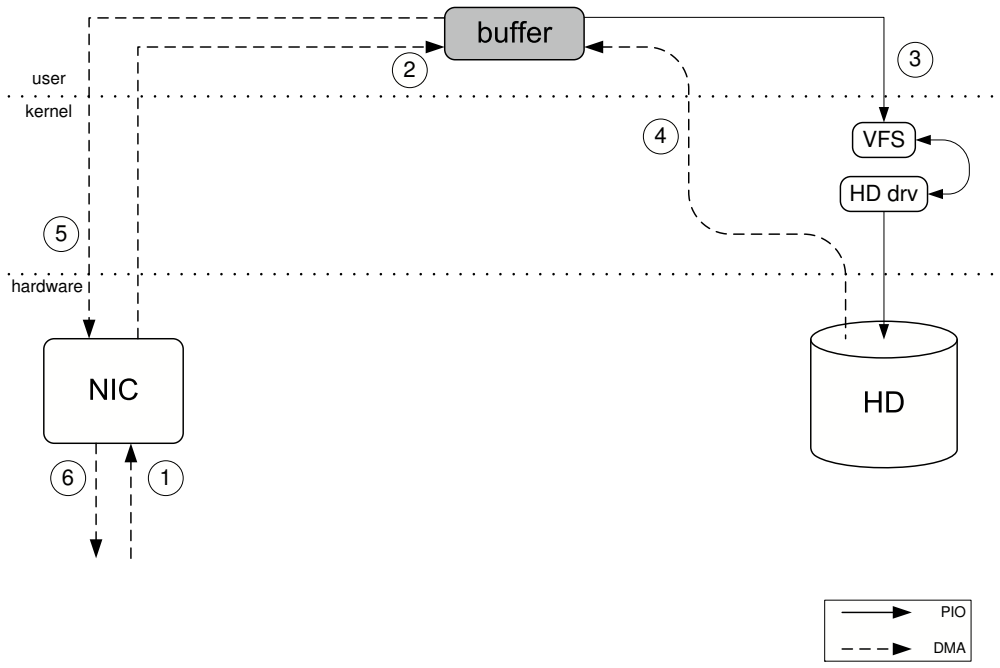
3. Μέσω μιας κλήσης συστήματος (βήμα 3) προγραμματίζεται η μηχανή DMA του μέσου αποθήκευσης για την αντιγραφή των δεδομένων που ζητούνται, από τη συσκευή στην κεντρική μνήμη, χωρίς τη μεσολάβηση προσωρινών απομονωτών.

4. Στο βήμα 4, πραγματοποιείται η αντιγραφή στη μνήμη του χώρου χρήστη και

5. Στο βήμα 5 τα δεδομένα αντιγράφονται με DMA και πάλι από την κεντρική μνήμη στη στατική μνήμη του Lanai.

6. Στο βήμα 6, τα δεδομένα έχουν μπει στην ουρά εξερχομένων της διεπαφής του Myrinet και με DMA αποστέλλονται στο καλώδιο.

Στο φυσικό επίπεδο (σχήμα 4.6):



Σχήμα 4.6: Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής nbd πάνω από Myrinet (λογικό επίπεδο)

1. η αίτηση λαμβάνεται από το καλώδιο στο βήμα 1.

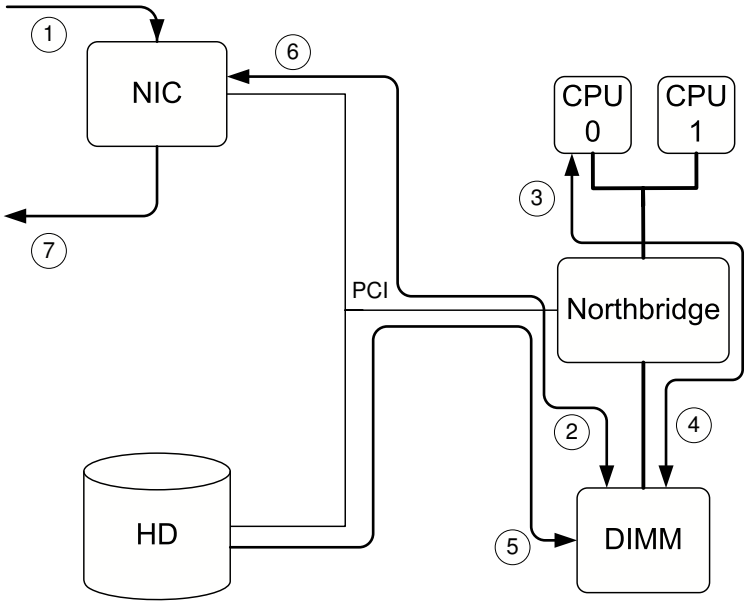
2. Από εκεί, τα δεδομένα αντιγράφονται στην κεντρική μνήμη (βήμα 2).

3,4 Εκεί μεταφράζεται η αίτηση από τον επεξεργαστή (βήματα 3, 4).

5 Γίνεται η κλήση για ανάγνωση από το μέσο αποθήκευσης και με DMA μεταφέρονται τα δεδομένα από το σκληρό δίσκο στην κεντρική μνήμη του συστήματος.

6 Από εδώ, και πάλι χωρίς μεσολάβηση του επεξεργαστή. αντιγράφονται τα δεδομένα στη στατική μνήμη της διεπαφής Myrinet (βήμα 6)

7 και στη συνέχεια αντιγράφονται τα δεδομένα με DMA από την εξερχόμενη ουρά, στο καλώδιο (βήμα 7).



Σχήμα 4.7: Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής nbd πάνω από Myrinet (φυσικό επίπεδο)

4.3 Η συσκευή δικτυακής αποθήκευσης gmblock

Το μοντέλο της συσκευής που υλοποιήσαμε, ακολουθεί τη βασική δομή της συσκευής δικτυακής αποθήκευσης, παραλείποντας τα βήματα που προσθέτουν επιβάρυνση στον επεξεργαστή και που καθυστερούν τη μεταφορά των δεδομένων μέσα στο υπολογιστικό σύστημα. Τα δεδομένα προσπερνούν τα βήματα στο δίαυλο PCI, αφού το μέσο αποθήκευσης καθώς και η διεπαφή του Myrinet βρίσκονται πάνω σε αυτόν.

Το δικτυακό κομμάτι του μοντέλου αναφέρεται σε μηνύματα GM πάνω από δίκτυο Myrinet, γιατί με αυτόν τον τρόπο επιτυγχάνεται αποδέσμευση του πυρήνα και του επεξεργαστή από τις αντιγραφές των δεδομένων για αποστολή / λήψη μηνυμάτων. Αυτό είναι και το κυριότερο πλεονέκτημα της επικοινωνίας σε επίπεδο χώρου χρήστη.

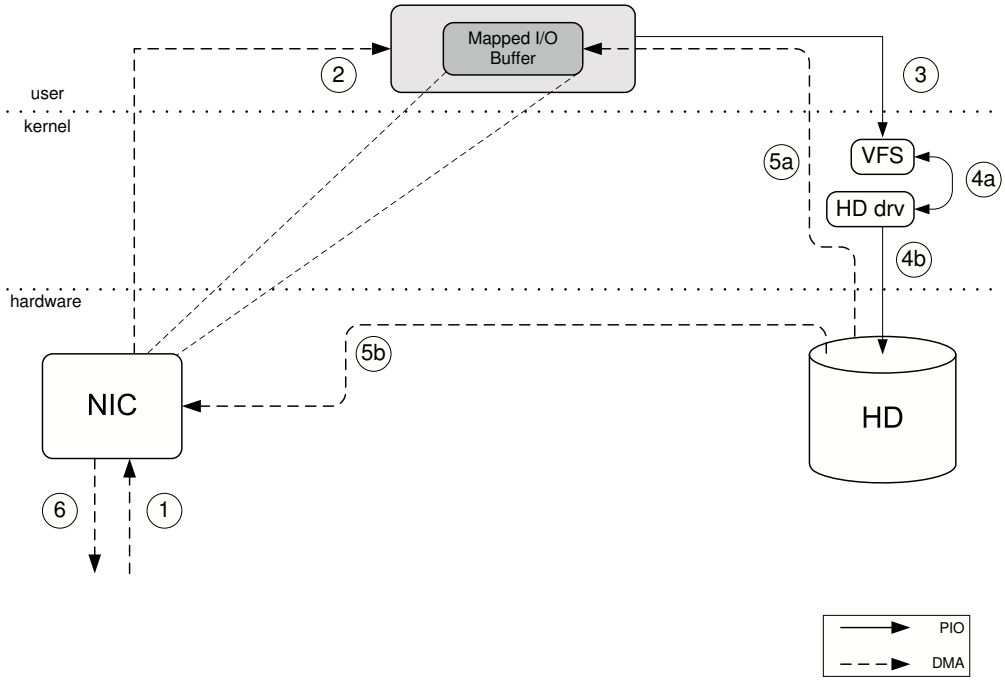
Ο πελάτης όπως αναφέρθηκε προηγουμένως, αναλαμβάνει να προσομοιώσει μια συσκευή αποθήκευσης, μεταφράζοντας τις αιτήσεις block σε δικτυακές αιτήσεις.

4.3.1 Εξυπηρετητής

Πιο συγκεκριμένα, η διαδρομή των δεδομένων στη συσκευή φαίνεται στο σχήμα 4.8 σε λογικό επίπεδο και σε φυσικό επίπεδο στο σχήμα 4.9. Ο ψευδοκώδικας του εξυπηρετητή, όσον αφορά στις απαιτήσεις, δε διαφέρει από τις άλλες δύο συσκευές. Το μοντέλο, βέβαια, διαφέρει σημαντικά στο επίπεδο του λειτουργικού συστήματος και στο στον τρόπο με τον οποίο ο πυρήνας και ο χώρος χρήστη χειρίζονται τις αιτήσεις ανάγνωσης από το φυσικό μέσο αποθήκευσης.

```
bd = open(/dev/local_block_device, O_READONLY, O_DIRECT);
for (;;) {
   gm_recv(&request);
   read(bd, request, &buffer);
   gm_send(buffer, remote_node);
}
```

Ας δούμε όμως αναλυτικά το μονοπάτι των δεδομένων στο σύστημα. Αρχικά, η αίτηση φτάνει στη διεπαφή Myrinet και αντιγράφεται με DMA στην κεντρική μνήμη του συστήματος. Ο επεξεργαστής αναλαμβάνει να μεταφράσει την αίτηση σε sectors και να προγραμματίσει μια μηχανή DMA προκειμένου να ανακτήσει από το μέσο αποθήκευσης τα συγκεκριμένα κομμάτια block, στα οποία αναφέρεται η αίτηση και να τα εγγράψει στη στατική μνήμη του Lanai από την οποία με μια ακόμα DMA μεταφορά θα αποσταλούν στο δίκτυο.



Σχήμα 4.8: Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής gmblock (λογικό επίπεδο)

Στο λογικό επίπεδο (σχήμα 4.8):

1. Η αίτηση φτάνει στη διεπαφή Myrinet στην ουρά εισερχομένων μηνυμάτων.

2. Αντιγράφεται στο χώρο μνήμης που ελέγχει ο οδηγός συσκευής της κάρτας Myrinet.

3. Με μια αίτηση ανάγνωσης δεδομένων από το στρώμα διαχείρισης συσκευών block του πυρήνα,

4a. το VFS (λόγω της κλήσης συστήματος για ανάγνωση από το μέσο αποθήκευσης) βάζει την αίτηση για τα δεδομένα που ζητούνται από τον πελάτη στην ουρά αιτήσεων του οδηγού συσκευής του μέσου.

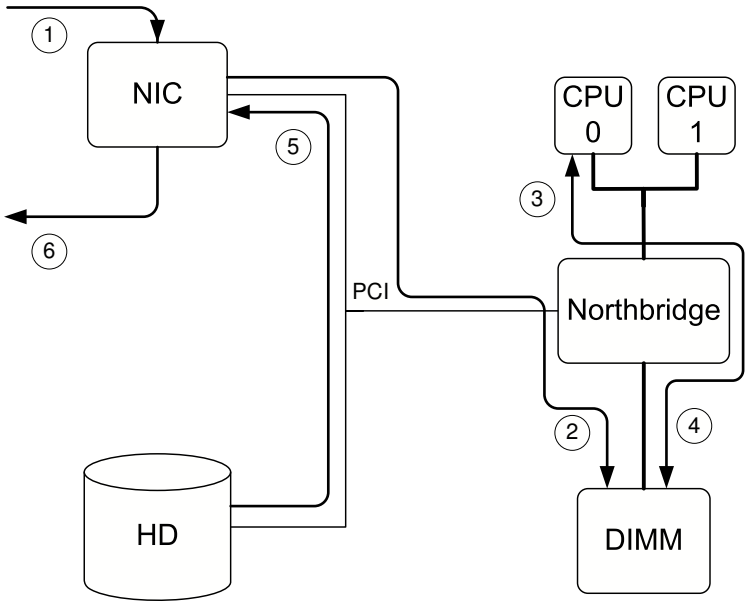
4b. Ο οδηγός του μέσου αποθήκευσης προγραμματίζει τη μηχανή DMA που βρίσκεται πάνω στο μέσο, για να εκκινήσει την αντιγραφή των δεδομένων που ζητούνται από τον πελάτη.

5a. Στο βήμα αυτό στο λογικό επίπεδο, αντιγράφονται τα δεδομένα στη στατική μνήμη του Lanai που απεικονίζεται στο χώρο φυσικών διευθύνσεων διαύλου PCI.

5b. Το βήμα αυτό αποτελεί ουσιαστικά τη φυσική μεταφορά των δεδομένων.

6. Τα δεδομένα αποστέλλονται με μια μεταφορά DMA στο δίκτυο, αφού τροποποιηθούν σε μηνύματα Myrinet.

Στο φυσικό επίπεδο (σχήμα 4.9) η μεταφορά των δεδομένων ακολουθεί την εξής διαδρομή:



Σχήμα 4.9: Το μονοπάτι των δεδομένων στον Εξυπηρετητή της συσκευής gmblock (φυσικό επίπεδο)

1. Η αίτηση φτάνει στη διεπαφή δικτύου Myrinet και αποθηκεύεται στον απομονωτή εισερχομένων μηνυμάτων.

2. Στη συνέχεια αντιγράφεται στη κεντρική μνήμη του συστήματος.

3.,4. Εκδίδεται η αίτηση ανάγνωσης δεδομένων από το μέσο αποθήκευσης, μέσω του ελεγκτή του συσκευής αποθήκευσης

5. Τα δεδομένα με μια μεταφορά DMA γράφονται στη στατική μνήμη του LANai.

6. Τα δεδομένα αποστέλλονται στο δίκτυο με μία μεταφορά DMA μέσω της μηχανής DMA εξερχομένων μηνυμάτων.

Στις δύο προηγούμενες περιπτώσεις, η ανάκτηση των δεδομένων από το μέσο αποθήκευσης και η αποθήκευσή τους στον απομονωτή από τον οποίο αποστέλλονταν, γινόταν με ενδιάμεση αποθήκευση είτε στην page cache (χωρίς O_DIRECT) είτε στην κεντρική μνήμη του συστήματος, στο χώρο χρήστη. Εδώ, ένα κομμάτι της μνήμης του Lanai απεικονίζεται σε φυσικές διευθύνσεις μνήμης, με αποτέλεσμα εγγραφή σε αυτό το κομμάτι σε λογικό επίπεδο να ισοδυναμεί με εγγραφή στη στατική μνήμη του Lanai στο φυσικό επίπεδο. Έτσι με μία αντιγραφή DMA γίνεται η μεταφορά των δεδομένων από το μέσο αποθήκευσης στο χώρο, από όπου θα αποσταλούν τα δεδομένα. Το αποτέλεσμα είναι η παράκαμψη όλων των βημάτων που αποτελούν επιβάρυνση είτε στον επεξεργαστή είτε στο bandwidth του διαύλου PCI.

Όπως φαίνεται στο σχήμα 4.9 η ανάγνωση των δεδομένων από το δίσκο γίνεται σε ένα βήμα, απευθείας στη μοιραζόμενη μνήμη του Lanai. Έτσι η κεντρική μνήμη του συστήματος δεν επηρεάζεται από την αποστολή εξυπηρετημένων αιτήσεων ούτε ο επεξεργαστής επιβαρύνεται με μεταφράσεις σελίδων και περιττές αντιγραφές.

Κεφάλαιο **5**

Υλοποίηση

Η υλοποίηση του συγκεκριμένου μοντέλου δικτυακής ανάκτησης δεδομένων από συσκευή αποθήκευσης μέσω δικτύου Myrinet χωρίς τη μεσολάβηση του πυρήνα / επεξεργαστή ή της κεντρικής μνήμης του συστήματος μπορεί να αναλυθεί σε δύο βασικά τμήματα.

- Υλοποίηση εφαρμογών ακολουθώντας το μοντέλο εξυπηρετητή - πελάτη με αιτήσεις που μεταφράζονται από αιτήσεις block σε αιτήσεις gmblock (GM messages + block) και αντίστροφα.

- Αλλαγές σε βασικά σημεία του οδηγού της κάρτας δικτύου Myrinet (GM) και του πυρήνα του Linux για την υποστήριξη των λειτουργιών που απαιτούνται για τη λειτουργία αυτού του μοντέλου.

Η ανάλυση των σημαντικότερων σημείων των εφαρμογών που υλοποιήθηκαν ακολουθείται από τις αλλαγές που έγιναν στον οδηγό της κάρτας Myrinet και στον πυρήνα του Linux.

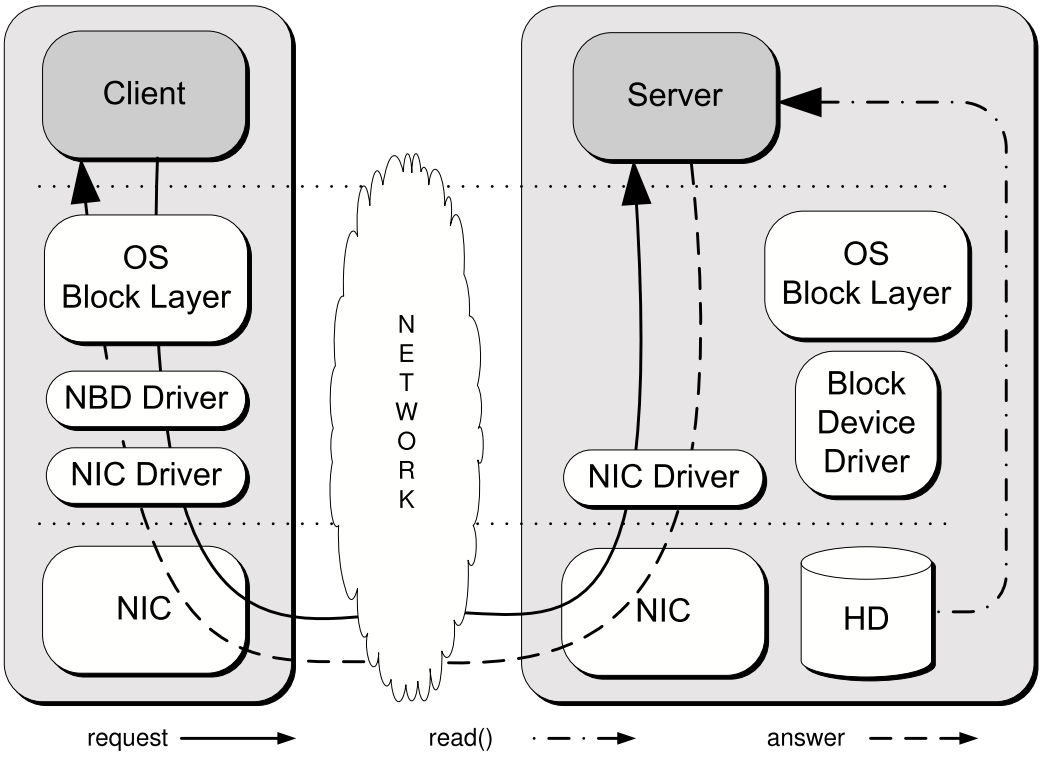
5.1 Εισαγωγή

Ο εξυπηρετητής της δικτυακής συσκευής αποθήκευσης που υλοποιήθηκε είναι μια εφαρμογή χώρου χρήστη που αναλαμβάνει να εξυπηρετήσει αιτήσεις για ανάγνωση από μια συσκευή αποθήκευσης. Ουσιαστικά η λειτουργία του συνίσταται στη μετάφραση των αιτήσεων που δέχεται μέσω του δικτύου Myrinet σε αιτήσεις για ανάγνωση block δεδομένων από ένα μέσο αποθήκευσης, όπως ο σκληρός δίσκος, και στην αποστολή των δεδομένων στο δίκτυο, χωρίς τη μεσολάβηση της κεντρικής μνήμης του συστήματος ή του επεξεργαστή.

Ο πελάτης της δικτυακής συσκευής gmblock είναι μια εφαρμογή που εκτελείται σε χώρο πυρήνα, φορτώνεται ως module στον πυρήνα και αναλαμβάνει να μεταφράσει τις αιτήσεις για ανάγνωση block δεδομένων από τη συσκευή

αποθήκευσης σε πακέτα μηνυμάτων δικτύου Myrinet, τα οποία και αποστέλλει στον εξυπηρετητή. Η εφαρμογή - πελάτης επίσης αναλαμβάνει να περιμένει τις εξυπηρετημένες αιτήσεις και να ολοκληρώσει τις αιτήσεις για ανάγνωση block δεδομένων, όταν αυτές φτάσουν στο σύστημα.

Το μοντέλο επικοινωνίας μεταξύ του εξυπηρετητή και του πελάτη βασίζεται σε δύο δομές δεδομένων που αναλαμβάνουν την κοινοποίηση βασικών μεταβλητών και σταθερών σε όλο τον κώδικα και ουσιαστικά συντελούν στη δικτυακή σύνδεση πελάτη εξυπηρετητή, στην εκκίνηση της επεξεργασίας μιας αίτησης στον εξυπηρετητή, στον αμοιβαίο αποκλεισμό ταυτόχρονων νημάτων (με σηματοφορείς) και στον επιτυχή τερματισμό της αίτησης στο άκρο του πελάτη.



Σχήμα 5.1: Το μοντέλο που υλοποιήθηκε

Η υλοποίηση έγινε σε μια πειραματική πλατφόρμα δύο μηχανημάτων συνδεδεμένων σημείο-προς-σημείο με κάρτες Myrinet M3F-PCI64B-2, επεξεργαστές AMD Athlon@900Mhz και Intel Pentium II@300Mhz, μνήμη 768MB και 384MB αντίστοιχα και σκληρούς δίσκους 10GB και 6GB αντίστοιχα. Το λειτουργικό σύστημα που έγινε η υλοποίηση είναι GNU/Linux διανομής Debian έκδοσης

unstable στη βασική του εγκατάσταση με έκδοση πυρήνα 2.6.17. Το λογισμικό GM (firmware, οδηγός συσκευής / kernel module, βιβλιοθήκη χώρου χρήστη) είναι έκδοσης gm-2.0.26.

5.1.1 Το module - πελάτης

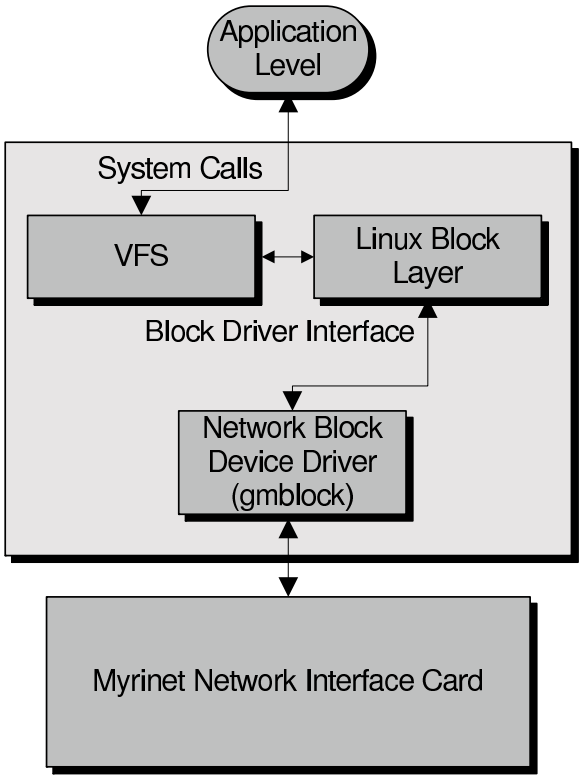
Όπως αναφέρθηκε προηγουμένως, ο πελάτης είναι ένα kernel module για έκδοση πυρήνα 2.6.17, που προσομοιώνει μια συσκευή αποθήκευσης. Ουσιαστικά, επειδή η υλοποίηση έγινε για ανάγνωση μόνο από τη συσκευή, πρόκειται για read-only συσκευή αποθήκευσης, δηλαδή πρόκειται για μια συσκευή ανάκτησης αποθηκευμένων δεδομένων από δίκτυο. Η διαφορά αυτής της συσκευής από μια κοινή (read only) συσκευή αποθήκευσης είναι ότι αντικαθιστά την ανάκτηση των δεδομένων από το φυσικό μέσο σε ανάκτηση δεδομένων από το δίκτυο μέσω μιας σύνδεσης δικτύου Myrinet, χωρίς όμως να υπάρχει διαφορετικός χειρισμός από την πλευρά του χρήστη. Πιο απλά, στο ανώτερο επίπεδο, το στρώμα χειρισμού block συσκευών του πυρήνα μιλά με το χρήστη και με τον οδηγό της συσκευής (το module που έχουμε υλοποιήσει), με τις ίδιες κλήσεις συστήματος και με τον ίδιο τρόπο που θα χρησιμοποιούσε για κλασικές συσκευές αποθήκευσης σε φυσικό μέσο (διαπροσωπία συσκευών block, σχήμα 5.2).

5.1.2 Ο εξυπηρετητής

Ο εξυπηρετητής είναι μια εφαρμογή χώρου χρήστη που χειρίζεται αιτήσεις που δέχεται από δίκτυο Myrinet σε συγκεκριμένη δομή, τις μεταφράζει διαφανώς σε αιτήσεις block, διαβάζοντας από το φυσικό μέσο αποθήκευσης τα δεδομένα που περιέχει η αίτηση και στη συνέχεια τα αποστέλλει στο δίκτυο, παρακάμπτοντας την κεντρική μνήμη και τον επεξεργαστή. Ένα σημαντικό πλεονέκτημα του εξυπηρετητή είναι ότι βρίσκεται στο χώρο χρήστη (user space), και καρπώνεται όλα τα πλεονεκτήματα των συστημάτων επικοινωνίας χώρου χρήστη (user level communication systems).

Το κομμάτι της υλοποίησης του εξυπηρετητή έχει στόχο την ταχύτερη πρόσβαση στα δεδομένα με όσο το δυνατόν λιγότερη παρέμβαση του επεξεργαστή και της κεντρικής μνήμης. Για να επιτευχθεί αυτό (σύμφωνα με τις προδιαγραφές του σχεδιασμού), πρέπει:

- το μέσο αποθήκευσης να εκτελέσει την αντιγραφή των δεδομένων απευθείας στη στατική μνήμη του LANai.
- ο οδηγός συσκευής της κάρτας δικτύου του Myrinet να μπορεί να στείλει δεδομένα μέσω μιας εφαρμογής χρήστη, απευθείας από τη στατική μνήμη



Σχήμα 5.2: Στρώματα χειρισμού της συσκευής gmblock

της κάρτας και όχι από την κεντρική μνήμη του συστήματος. Πρέπει δηλαδή να μπορεί να χειριστεί buffers στη στατική μνήμη του LANai, σαν να ήταν GM buffers στην κεντρική μνήμη του συστήματος (host RAM).

Το μοντέλο αυτό (που παρουσιάστηκε στο Κεφάλαιο 4) απαιτεί συνδυασμό αλλαγών τόσο στον πυρήνα του Linux, για να μπορεί να γίνει η DMA μεταφορά από το σκληρό δίσκο στη στατική μνήμη του LANai όσο και στον οδηγό της κάρτας Myrinet για να μπορεί αφενός να υποστηρίξει αυτή τη δυνατότητα (της αντιγραφής σε φυσικές διευθύνσεις διαύλου PCI), και αφετέρου να παρακάμψει το βήμα της αντιγραφής των δεδομένων από την κεντρική μνήμη, δεδομένου ότι το προς αποστολή μήνυμα βρίσκεται ήδη στη στατική μνήμη του LANai.

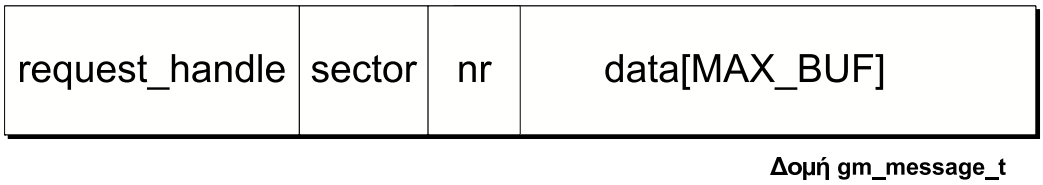
Παρακάτω εξηγείται αρχικά ο κώδικας των εφαρμογών και στη συνέχεια οι αλλαγές που έγιναν στον πυρήνα και στον οδηγό της συσκευής προκειμένου να υλοποιηθεί το μοντέλου που περιγράφεται.

5.1.3 Βασικές Δομές Επικοινωνίας

Οι βασικές δομές του μοντέλου επικοινωνίας των δύο εφαρμογών είναι τρεις:

- `gm_message_t`
  Η δομή του πακέτου που μεταδίδεται με το συγκεκριμένο μοντέλο είναι η `gm_message_t` και παρουσιάζεται στο σχήμα 5.3. Ουσιαστικά πρόκειται για μια αίτηση gmblock, δηλαδή μια αίτηση για ανάγνωση δεδομένων από μια συσκευή αποθήκευσης (block request) η οποία βρίσκεται σε δίκτυο Myrinet και μπορεί να προσπελαστεί με GM (gmblock).

  Αποτελείται από 4 πεδία:



Σχήμα 5.3: Η αίτηση gmblock

  - ⋆ `request_handle`:
    Είναι ο αριθμός που διαχωρίζει τις αιτήσεις μεταξύ τους. Αποτελεί στιγμιαία μοναδικό αριθμό και ουσιαστικά πρόκειται για το μοναδικό τρόπο με τον οποίο μπορεί ο πελάτης να γνωρίζει ποια αίτηση έλαβε. Το μοναδικό αναγνωριστικό είναι η διεύθυνση της μνήμης της αίτησης *req*. Είναι μοναδικό μόνο, όταν η αίτηση δεν έχει εξυπηρετηθεί και ενδεχομένως αυτό και μόνο είναι αρκετό.
  - ⋆ `sector`:
    Πρόκειται για τον αρχικό sector από τον οποίο ζητούνται δεδομένα.
  - ⋆ `nr` (Number of Sectors):
    Πρόκειται για τον αριθμό των sectors που έχουν ζητηθεί.
  - ⋆ `data`:
    Είναι τα δεδομένα που ζητήθηκαν.

- `gm_con_t`
  Είναι ο τύπος της δομής που κρατά τα στοιχεία της σύνδεσης (gm connection type) καθώς και οτιδήποτε άλλο χρειάζεται, για να διατηρηθεί η εύρυθμη λειτουργία των εφαρμογών (σηματοφορείς, μεταβλητές

```
typedef struct gm_con {
    struct gm_port *port;
    unsigned int board_num;
    unsigned int port_id;
    char *name;
    enum gm_api_version ver;
    gm_u32_t target_node_id;
    gm_u32_t local_node_id;
    gm_u32_t global_node_id;
    atomic_t gm_lock;
    atomic_t callback_lock;
    int callback;
    gm_status_t status;
} gm_con_t;
```

Σχήμα 5.4: Η δομή gm_con_t

κατάστασης, μετρητές κλπ.). Τα πεδία της αναλύονται παρακάτω και φαίνονται στο σχήμα 5.4.

- `port`
  Ουσιαστικά είναι δείκτης στο ακροσημείο που δεσμεύει συγκεκριμένο κομμάτι της μνήμης της κάρτας δικτύου Myrinet για επικοινωνία (αποστολή / λήψη μηνυμάτων). Χρησιμεύει, για να περνιέται σαν παράμετρος στις συναρτήσεις επικοινωνίας `gm_*` και ανατίθεται από την `gm_open()`. Για περισσότερες λεπτομέρειες βλέπε Κεφάλαιο 2 / GM.
- `board_num`
  Είναι ο αριθμός της κάρτας δικτύου Myrinet.
- `port_id`
  Είναι το αναγνωριστικό του ακροσημείου (port) του συστήματος επικοινωνίας GM.
- `pname`
  Είναι το όνομα του ακροσημείου.
- `ver`
  Είναι η έκδοση του gm API που χρησιμοποείται.

  - ⋆ `target_node_id`
    Πρόκειται για το αναγνωριστικό του κόμβου - προορισμού για μια αποστολή.
  - ⋆ `local_node_id`
    Είναι το αναγνωριστικό του τοπικού κόμβου.
  - ⋆ `global_node_id`
    Είναι το αναγωνριστικό του τοπικού κόμβου, όπως φαίνεται στους άλλους (απομακρυσμένους) κόμβους.
  - ⋆ `gm_lock`
    Είναι ένας σηματοφορέας δηλωμένος ως `atomic_t`, για να κλειδώνει το gm.
  - ⋆ `callback_lock`
    Άλλος ένας σηματοφορέας για κλείδωμα των διεργασιών, μέχρι να ληφθεί το callback.
  - ⋆ `callback`
    ΄Ενας μετρητής για τον αριθμό των callbacks που εκκρεμούν.
  - ⋆ `status`
    Μια μεταβλητή κατάστασης ανάλογα με την οποία η εφαρμογή μπορεί να διαχωρίσει μηνύματα σφαλμάτων ή να διακόψει τη λειτουργία της σε περίπτωση σφάλματος.

- `gm_globals_t`
  Η δομή αυτή χρησιμεύει ουσιαστικά στην ομαδοποίηση των δύο προηγούμενων και στην απλούστευση του χειρισμού της σύνδεσης. ΄Ετσι μια συνάρτηση μπορεί να παίρνει αυτή τη δομή σαν όρισμα αντί για τις επιμέρους δομές / μεταβλητές μία προς μία.
  - ⋆ `connection`
    Είναι ένας δείκτης σε δομή `gm_con_t`.
  - ⋆ `my_message`
    Είναι ένας δείκτης σε δομή `gm_message_t`.

Για το χειρισμό των δομών αυτών επιλέχθηκε η δημιουργία μακροεντολών που διευκολύνουν την πρόσβαση στις τιμές των μεταβλητών.

5.2 Ο πελάτης

Ο πελάτης του μοντέλου που υλοποιήθηκε, όπως προαναφέρθηκε, είναι ένα module για τον πυρήνα του Linux (έκδοσης 2.6.17). Προσομοιώνει μια συσκευή

block, στην οποία υλοποιούνται μόνο οι διαδικασίες ανάγνωσης. Ας δούμε όμως αναλυτικά τα βασικά του σημεία.

Εκτός από τις βασικές δομές του μοντέλου, `gm_message_t` και `gm_con_t`, ο πελάτης χρησιμοποιεί μια δομή `Device` για τον χειρισμό αυτής της εικονικής συσκευής αποθήκευσης (σχήμα 5.5).

- `Device`
  Η δομή Device ομαδοποιεί όλα τα σημεία που αφορούν στην κατάσταση ενός στιγμιότυπου (instance) της εικονικής συσκευής αποθήκευσης. Τα πεδία της είναι:

  ⋆ `size`
  Το μέγεθος της συσκευής.

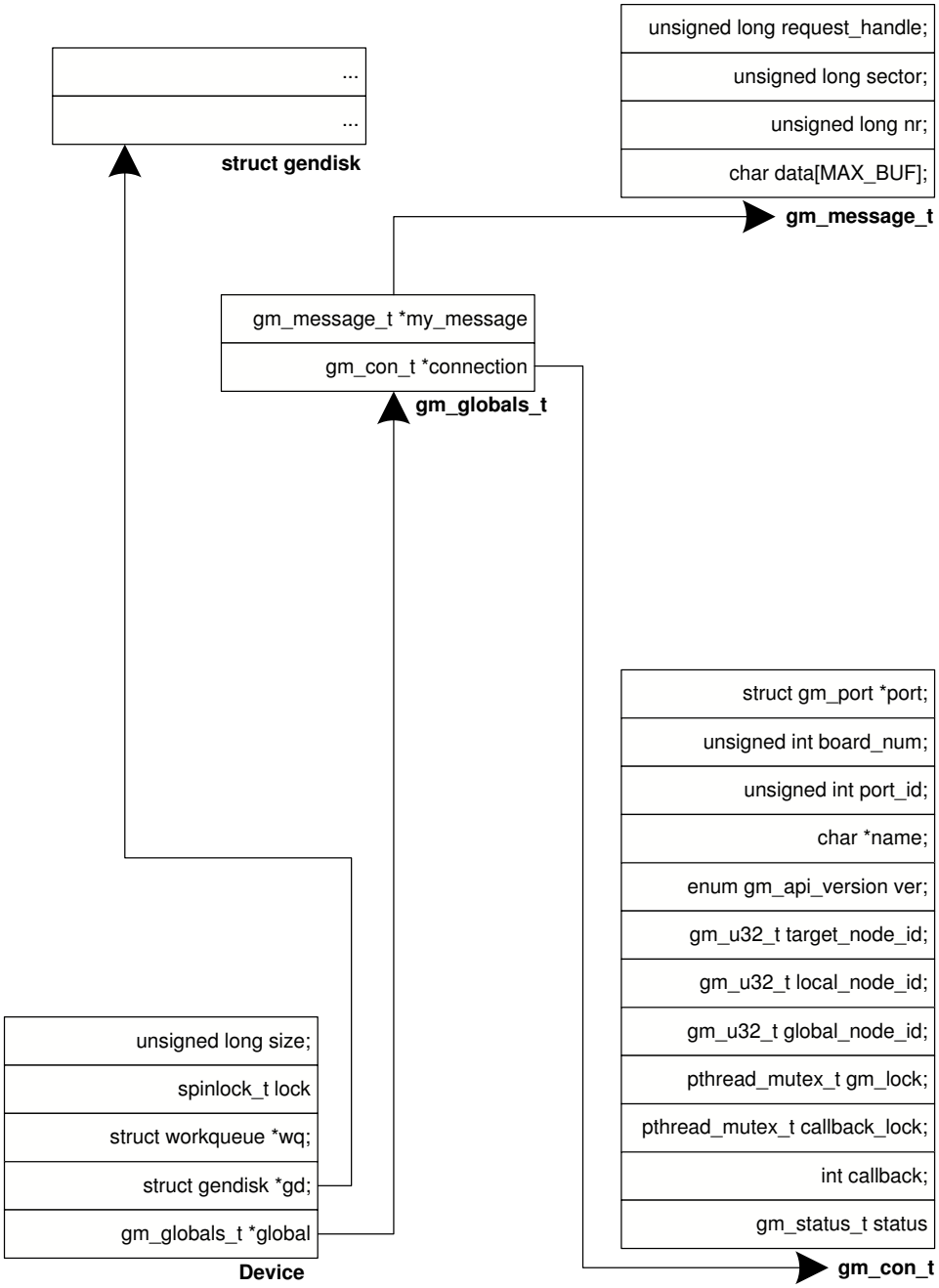
  ⋆ `lock`
  Ένας σηματοφορέας κλειδώματος της ουράς αιτήσεων της συσκευής.

  ⋆ `wq`
  Μια workqueue για την αναμονή των απαντήσεων από τον εξυπηρετητή, αφού το μοντέλο επικοινωνίας είναι ασύγχρονο.

  ⋆ `gd` (gendisk)
  Ένας δείκτης τύπου struct gendisk (generic disk). Η δομή gendisk είναι ο πυρήνας του στρώματος χειρισμού συσκευών block του Linux. Κάποια από τα πεδία που περιέχει εξηγούνται παρακάτω.

    ∗ `block_device_operations`
    Είναι μια δομή που ορίζει τις λειτουργίες που αντιστοιχίζονται σε αφαιρετικό επίπεδο στο στρώμα χειρισμού συσκευών αποθήκευσης του Linux. Δηλαδή μια συγκεκριμένη συσκευή χειρίζεται με διαφορετικό τρόπο τη `read()` από μια άλλη. Παρόλα αυτά, μια εφαρμογή (σε επίπεδο χρήστη) θα εκτελέσει την κλήση συστήματος `read()`, χωρίς να γνωρίζει πώς υλοποιείται η συγκεκριμένη κλήση σε κάθε περίπτωση.

    ∗ `queue`
    Είναι η ουρά που αποθηκεύει τις αιτήσεις που γίνονται στη συγκεκριμένη συσκευή. Η ουρά αιτήσεων αποτελεί το σημαντικότερο ίσως κομμάτι του οδηγού μιας συσκευής block, αφού είναι η δομή που κρατά τις αιτήσεις για ανάγνωση / εγγραφή.



Σχήμα 5.5: Οι δομές δεδομένων σύνδεσης και μηνύματος

Η λειτουργία του module του πελάτη ακολουθεί το εξής μοντέλο:

1. Το module λαμβάνει αιτήσεις για ανάγνωση δεδομένων από το VFS.

2. Μεταφράζει τις αιτήσεις block σε αιτήσεις gmblock. Για κάθε αίτηση για ανάγνωση δεδομένων δίνεται ο αρχικός sector και ο αριθμός των sectors που πρέπει να διαβαστούν. Το module κατασκευάζει το μήνυμα Myrinet από τους δύο αυτούς αριθμούς καθώς και από το δείκτη που δείχνει στις αιτήσεις block.

3. Οι αιτήσεις gmblock αποστέλλονται στο δίκτυο.

4. Ταυτόχρονα, ένα kernel thread (με τη μορφή ενός `workqueue`) εκκινείται από την `gmblock_init()` και εκτελείται ατέρμονα περιμένοντας να λάβει μηνύματα Myrinet, δηλαδή τις απαντήσεις / εξυπηρετημένες αιτήσεις gmblock, από τον εξυπηρετητή, και να τις προωθήσει κατάλληλα στους buffers που γεμίζονται με τα δεδομένα που ζητούνται.

5. Όταν οι αιτήσεις gmblock ληφθούν, αναγνωρίζονται από το kernel thread, μεταφράζονται ξανά σε αιτήσεις block και με κατάλληλο κλείδωμα της ουράς αιτήσεων, ολοκληρώνονται.

5.2.1 Αρχικοποίηση και Καταχώρηση

Ο πελάτης, όπως και κάθε kernel module (έκδοσης 2.6) αποτελείται από δύο βασικές συναρτήσεις `init()` και `cleanup()` που, όπως αναφέρουν τα ονόματά τους, αναλαμβάνουν να εκκινήσουν τη διεργασία του module και να την τερματίσουν ασφαλώς. Η συνάρτηση `gmblock_init()` αναλαμβάνει την εκκίνηση όλων των διεργασιών που αφορούν στη σύνδεση του module με το δίκτυο και στην επικοινωνία του με τα υπόλοιπα μέρη του πυρήνα. Πιο συγκεκριμένα, αναλαμβάνει να αρχικοποιήσει τη μεταβλητή δομής `gm_con_t` και τη δομή `Device`.

1. *Αρχικοποίηση του GM.*

2. *Αρχικοποίηση της συσκευής Block.*

3. *Αρχικοποίηση της ουράς αιτήσεων και της συνάρτησης χειρισμού της.*

4. *Καταχώρηση της συσκευής.*

Σχήμα 5.6: Εκκίνηση της διεργασίας του module

Στα σχήματα 5.6, 5.7 και 5.8 βλέπουμε τις λειτουργίες που γίνονται, για να καταχωρηθεί η συσκευή.

---

1. *Αρχικοποίηση της δομής* `gm_con_t`.

2. *Ανάθεση ονόματος κόμβου σύμφωνα με το χάρτη κόμβων του δικτύου Myrinet.*

3. *Ορισμός κόμβου - προορισμού (εξυπηρετητή) σύμφωνα με το χάρτη κόμβων του δικτύου Myrinet.*

4. *Δέσμευση απομονωτών μνήμης για εισερχόμενα και εξερχόμενα μηνύματα GM.*

Σχήμα 5.7: Αρχικοποίηση της σύνδεσης GM

΄Ενα βασικό σημείο στο κομμάτι της καταχώρησης της συσκευής είναι ο ορισμός της συνάρτησης χειρισμού της ουράς αιτήσεων. Όπως έχουμε δει στο Κεφάλαιο 3, η request function είναι ο πυρήνας ενός οδηγού συσκευής block, αφού είναι η συνάρτηση που καθορίζει πώς θα χειρίζεται το λειτουργικό σύστημα την εκάστοτε αίτηση για ανάγνωση / εγγραφή από / στο μέσο αποθήκευσης.

---

1. *Ορισμός μεγέθους συσκευής.*

2. *Ορισμός συνάρτησης χειρισμού της ουράς αιτήσεων της συσκευής.*

3. *Ανάθεση παραμέτρων (*`major / minor` *κλπ)*

4. *Καταχώρηση της συσκευής*

Σχήμα 5.8: Αρχικοποίηση και καταχώρηση της συσκευής gmblock

5.2.2 Χειρισμός Αιτήσεων

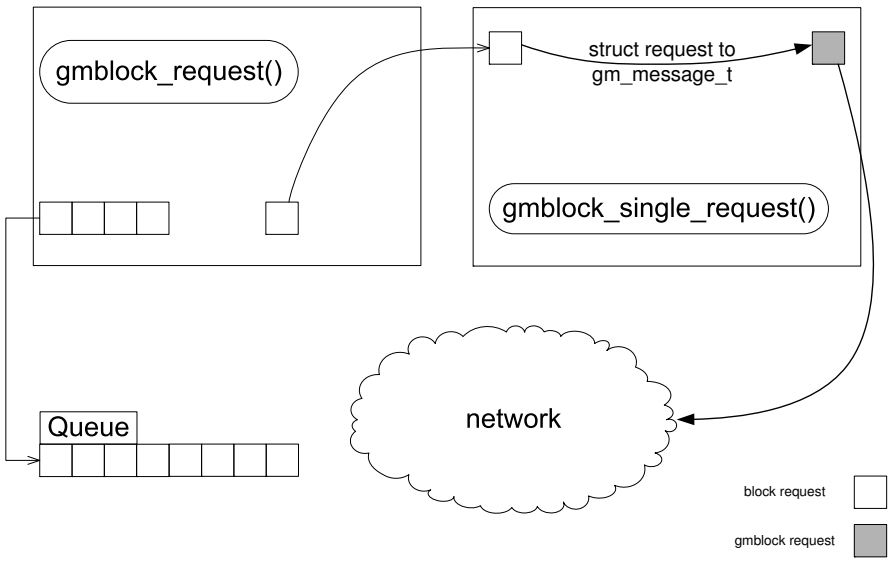
Εδώ η συνάρτηση `gmblock_request` ορίζεται με τη συνάρτηση `blk_init_queue`, και χρησιμοποιεί την `request_queue *Queue`, για να ενημερώθεί ο πυρήνας.

Όπως φαίνεται στο σχήμα 5.9 η συνάρτηση `gmblock_request` παίρνει σαν όρισμα την ουρά αιτήσεων της συσκευής (Queue) που έχει αρχικοποιηθεί στο σχήμα 5.8. Λαμβάνει την πρώτη μη εξυπηρετημένη αίτηση block και εκτελεί έναν ατέρμονα βρόχο, όσο υπάρχουν ακόμα ανεξυπηρετημένες αιτήσεις block στην ουρά. Τις βγάζει από την ουρά και καλεί τη συνάρτηση χειρισμού μίας μοναδικής αίτησης (την `gmblock_single_request()`). Οι αιτήσεις βγαίνουν από την ουρά γιατί η υλοποίηση της συγκεκριμένης συσκευής απαιτεί να έξυπηρετούνται ασύγχρονα. Η `gmblock_request()` εκτελείται σε context που δεν μπορεί να μπλοκάρει, να κοιμηθεί. Η εξυπηρέτηση όμως των αιτήσεων απαιτεί την αποστολή δεδομένων στο δίκτυο και την αναμονή λήψης μηνυμάτων από το δίκτυο με αποτέλεσμα να χρειάζεται να κοιμηθεί το νήμα που εκτελεί τις λήψεις μηνυμάτων και έτσι να μην μπορούν να εκτελεστούν αυτές οι λειτουργίες μέσα στην `gmblock_request()`. Έτσι, η λογική που ακολουθήθηκε είναι η έξοδός της από την ουρά, το μπλοκάρισμα της ουράς με μια συνάρτηση προετοιμασίας (έτσι, ώστε να είναι μόνο μία αίτηση σε εξέλιξη κάθε φορά) και η αποστολή της αίτησης στο δίκτυο Myrinet με την `gmblock_single_request()`.

Η συνάρτηση προετοιμασίας χειρισμού της ουράς αιτήσεων (preparation function) χρησιμεύει στον ορισμό παραμέτρων για την επεξεργασία αιτήσεων μέσα στην ουρά. Έτσι, στην παρούσα υλοποίηση, αφού ο στόχος ήταν η απόδειξη ότι το βελτιστοποιημένο μονοπάτι των δεδομένων είναι προσπελάσιμο, μέσω της συνάρτησης προετοιμασίας της ουράς αιτήσεων, ο πελάτης μπορεί να χειριστεί μόνο μία αίτηση ανά φορά (βλέπε Κεφάλαιο 6 / Επεκτάσεις).

Η `gmblock_single_request()` λαμβάνει σαν όρισμα ένα δείκτη στην τρέχουσα αίτηση για επεξεργασία, τη μεταφράζει σε μήνυμα Myrinet για αίτηση προς τον εξυπηρετητή, και αποστέλλει το μήνυμα απευθείας από το χώρο που έχει δεσμευτεί σε μνήμη DMA στην αρχικοποίηση του GM. Εδώ είναι σημαντικό να παρατηρήσουμε τον τρόπο αναγνώρισης της αίτησης. Το μοναδικό αναγνωριστικό (request_handle) που ορίζεται στο μοντέλο είναι η διεύθυνση του δείκτη της αίτησης block που μεταφράζεται σε αίτηση προς τον εξυπηρετητή της συσκευής. Έτσι, όταν το μήνυμα επιστρέψει στον πελάτη (στο kernel module), ο πυρήνας ενημερώνεται για την ολοκληρωση της αίτησης με χρήση της `end_request()` δίνοντάς της το `request_handle` σαν όρισμα. Η `end_request()` θα φροντίσει να ξυπνήσουν οι διεργασίες που ζήτησαν τα δεδομένα της αίτησης.

Το κομμάτι του πελάτη που διαχειρίζεται τις εισερχόμενες αιτήσεις είναι ένα νήμα που εκτελείται σε context πυρήνα (kernel thread). Αυτό το νήμα έχει τη μορφή ενός TASK μέσα σε ένα workqueue. Το workqueue ορίζεται στις δηλώσεις του module και η εργασία που εκτελεί (βλέπε σχήμα 5.10) εκκινείται

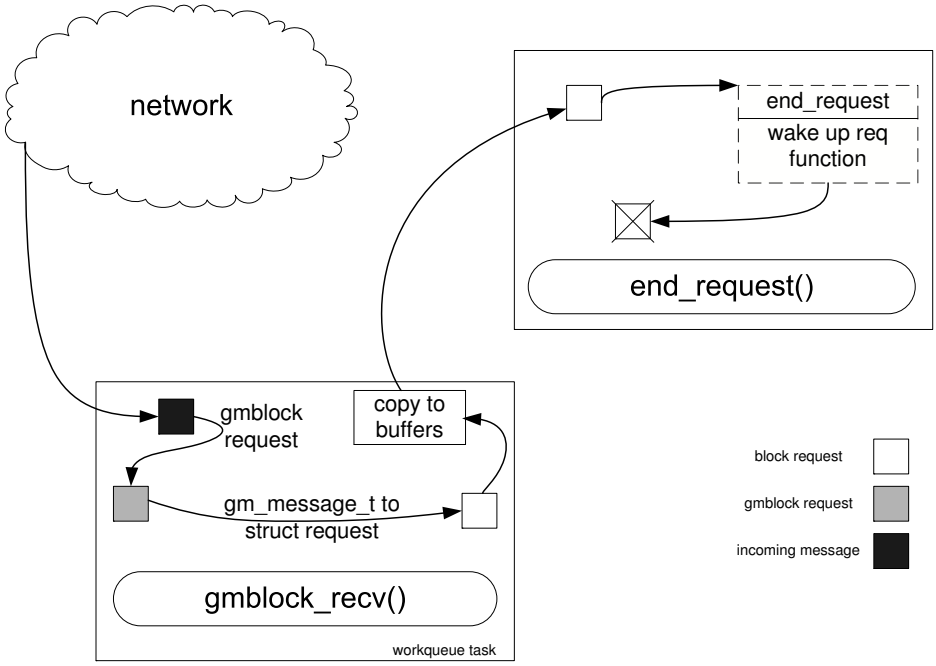


Σχήμα 5.9: Μετάφραση μιας αίτησης block σε gmblock

στη συνάρτηση `init` του module. Με χρήση του workqueue το νήμα μπορεί να χρονοδρομολογηθεί κανονικά (σαν να ήταν σε process context).

Αναλυτικά ο κώδικας εκτέλεσης του task στο workqueue φαίνεται στο σχήμα 5.10. Ουσιαστικά πρόκειται για έναν ατέρμονα βρόχο (`while(1) { ... }`) που λαμβάνει μηνύματα Myrinet, τα μεταφράζει σε εξυπηρετημένες αιτήσεις block, αντιγράφει τα δεδομένα στους buffers που ορίζονται από το λειτουργικό σύστημα και καλεί τη συνάρτηση `end_request()` για τον τερματισμό της αίτησης. Σε περίπτωση που δεν υπάρχει αίτηση για λήψη, το kernel thread μπλοκάρει μέσω της `gm_unknown()` και ξυπνά με ένα interrupt μόνο, όταν λάβει ένα μήνυμα. Με αυτόν τον τρόπο, ο επεξεργαστής (ο πυρήνας) δεν απασχολείται, όταν δεν υπάρχει αίτηση για λήψη.

Στην περίπτωση που ληφθεί κάποιο μήνυμα, ελέγχεται αν πρόκειται για callback απεσταλμένης αίτησης για εξυπηρέτηση ή για μήνυμα εξυπηρετημένης αίτησης. Αν το εισερχόμενο μήνυμα είναι callback (μήνυμα δηλαδή τύπου ″έλαβα την αίτηση επιτυχώς″) αγνοείται, αλλιώς, πρόκειται για αίτηση που έχει εξυπηρετηθεί. Στη δεύτερη περίπτωση, μεταφράζεται σε κανονική μορφή αίτησης block, κλειδώνεται η ουρά, αντιγράφονται τα δεδομένα στους απομονωτές και καλείται η `end_request()` για την ολοκλήρωση της αίτησης.



Σχήμα 5.10: Η συνάρτηση χειρισμού μιας ληφθείσας αίτησης της συσκευής gmblock

Ο buffer που δεσμεύτηκε για τη λήψη του μηνύματος (της αίτησης) επιστρέφεται στο GM και αρχικοποιείται για επόμενη λήψη.

5.3 Ο Εξυπηρετητής

Ο εξυπηρετητής του οδηγού της συσκευής gmblock είναι μια εφαρμογή χώρου χρήστη (userspace application) που δέχεται αιτήσεις του μοντέλου που σχεδιάστηκε, διαβάζει τα δεδομένα που ζητούνται, γεμίζει το μήνυμα της αίτησης με τα δεδομένα και το αποστέλλει πάλι με Myrinet στον πελάτη. Η υλοποίηση του εξυπηρετητή είναι ένας ατέρμονας βρόχος που περιμένει να λάβει μια αίτηση (`gm_receive()`), στη συνέχεια διαβάζει τα δεδομένα από τη συσκευή αποθήκευσης (`read("/dev/hd",&gm_message->buffer,PAGE_SIZE)`) και τέλος αποστέλλει τα δεδομένα μέσω δικτύου Myrinet (`gm_send (port,gm_message, target_node)`).

Αποτελείται αφενώς από ένα νήμα (pthread) που αναλαμβάνει να ανοίξει το αρχείο συσκευής του μέσου αποθήκευσης, να λάβει τις αιτήσεις και να αποστείλει τα δεδομένα που ζητούνται και αφετέρου τον πατέρα του νήματος

που αρχικοποιεί τη σύνδεση GM και παρακολουθεί την εκτέλεση του νήματος για τυχόν σφάλματα.

Οι δομές που χειρίζεται ο εξυπηρετητής είναι οι ίδιες με αυτές του πελάτη. Για ευκολία διαχείρισης του κώδικα χρησιμοποιήθηκαν μακροεντολές για την αρχικοποίηση του GM και το χειρισμό των μηνυμάτων.

Ο εξυπηρετητής εκκινεί τη σύνδεση (ανοίγει μια πόρτα στην κάρτα Myrinet, ταυτοποείται σαν κόμβος στο χάρτη του Myrinet) και εκκινεί το νήμα εξυπηρέτησης που περιμένει να λάβει αιτήσεις, για να τις εξυπηρετήσει και να στείλει τα δεδομένα στον πελάτη.

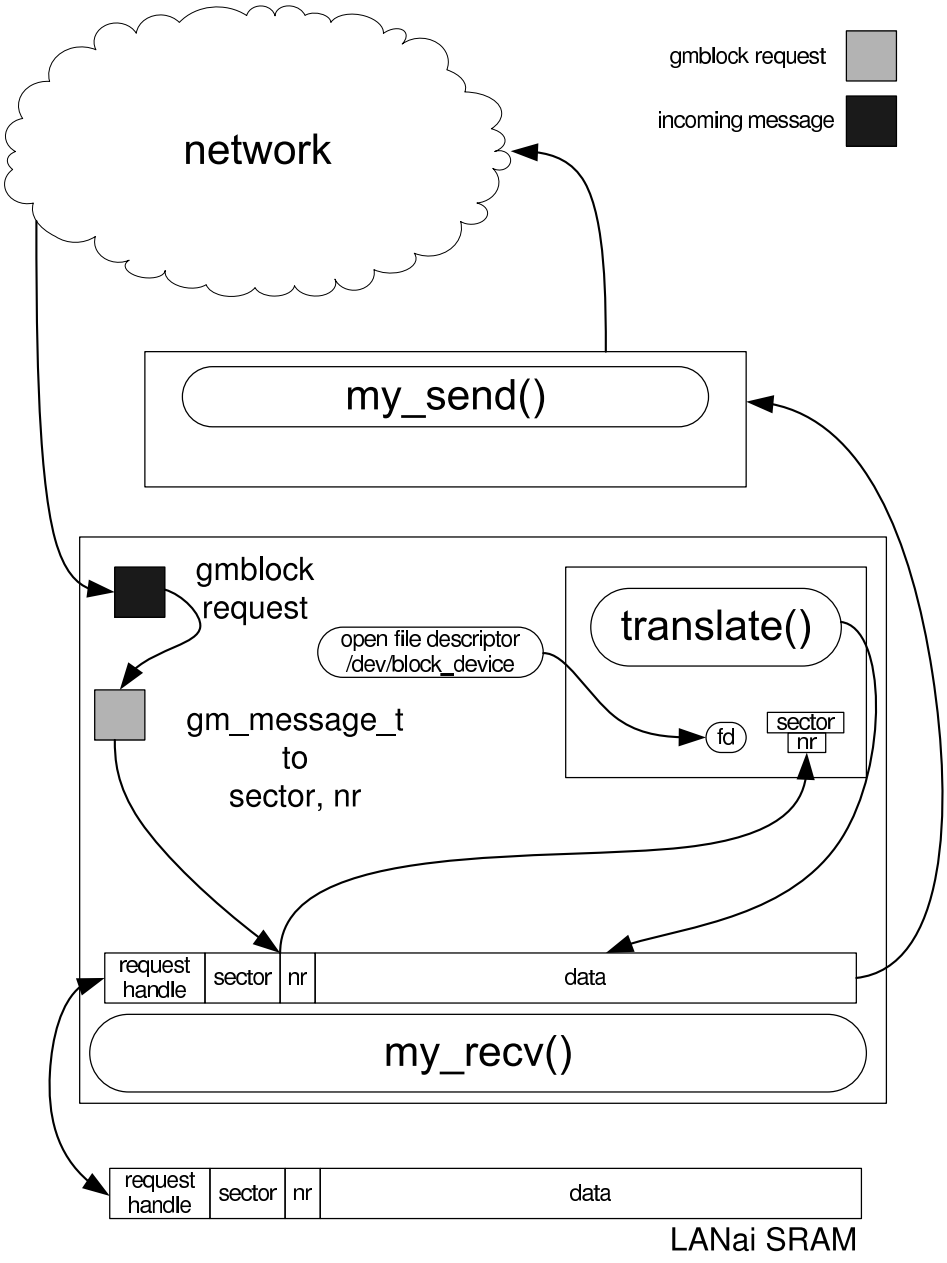
Σε μια συμβατική αποστολή, η αντιγραφή των δεδομένων από το σκληρό δίσκο (μέσο αποθήκευσης) στη στατική μνήμη του LANai (και άρα στο δίκτυο) απαιτεί τη μεταφορά τους πρώτα στην κεντρική μνήμη του συστήματος (στο χώρο χρήστη) και στη συνέχεια στη στατική μνήμη της κάρτας δικτύου. Όπως αναφέρθηκε στο Κεφάλαιο 4 (Σχεδιασμός), είναι δυνατή η παράκαμψη των βημάτων αντιγραφής στην κεντρική μνήμη, αν απεικονιστεί ένα κομμάτι της στατικής μνήμης της κάρτας Myrinet στο χώρο φυσικών διευθύνσεων μνήμης. Έτσι, η μηχανή DMA του σκληρού δίσκου προγραμματίζεται με φυσικές διευθύνσεις μνήμης που αντιστοιχούν όμως σε διευθύνσεις διαύλου PCI, δηλαδή στις φυσικές διευθύνσεις του κομματιού της SRAM.

Η παράκαμψη λοιπόν της ιεραρχίας μνήμης συνίσταται στην απευθείας μεταφορά που εκτελεί η μηχανή DMA του σκληρού δίσκου από το μέσο στην SRAM της κάρτας δικτύου Myrinet με τη βοήθεια της συνάρτησης `gm_get_sram_buf (port,&ptr)`.

Το νήμα που εκτελείται φαίνεται στο σχήμα 5.11. Η συνάρτηση `gm_get_sram_buf (port,&ptr)` δίνει στον ptr την εικονική διεύθυνση του κομματιού της στατικής μνήμης της κάρτας δικτύου Myrinet που έχει απεικονιστεί στο χώρο εικονικών διευθύνσεων της διεργασίας. Εκεί γράφονται τα δεδομένα που ζητούνται μέσω της συνάρτησης `my_recv()` που αναλαμβάνει να λάβει την αίτηση, να διαβάσει τα δεδομένα με τη βοήθεια της `translate()`, να τα αντιγράψει με μια μεταφορά DMA και να τα στείλει στον πελάτη με την `my_send()`.

Η επιλογή της υλοποίησης του επεξεργαστή σαν μια εφαρμογή χώρου χρήστη δεν ήταν τυχαία. Μια userspace εφαρμογή προσδίδει σταθερότητα στην υλοποίηση και προστασία από προβλήματα λόγω σφαλμάτων στον κώδικα αφού αν δημιουργηθεί οποιοδήποτε πρόβλημα, το σύστημα που φιλοξενεί την εκτέλεση της εφαρμογής δεν καταρρέει. Επίσης, η εκσφαλμάτωση της εφαρμογής μπορεί να γίνει με συμβατικά προγράμματα (ένα από τα πιο ισχυρά για το Linux είναι το gdb). Η ανάγνωση από το μέσο αποθήκευσης μπορεί να γίνει συμβατικά, με χρήση κλήσεων συστήματος που επιτρέπουν τη φορητότητα της υλοποίησης και δεν περιορίζουν το μέσο αποθήκευσης συγκεκριμένα σε σκληρό δίσκο ή σε συστοιχία σκληρών δίσκων. Δεδομένης της userspace

εφαρμογής, θα μπορούσαν πολλά στιγμιότυπα να αναφέρονται σε διαφορετικά κομμάτια ενός κοινού μέσου αποθήκευσης ή και διαφορετικών δυνατοτήτων πρόσβασης σε χρήστες. Τέλος, η χρήση βιβλιοθηκών για την κρυπτογράφηση των δεδομένων είναι πλεονέκτημα των εφαρμογών του χώρου χρήστη.



Σχήμα 5.11: Ο εξυπηρετητής του οδηγού συσκευής gmblock

5.4 Υλοποίηση του βελτιστοποιημένου μονοπατιού

Μέχρι τώρα περιγράψαμε το μοντέλο επικοινωνίας μεταξύ των εφαρμογών (του πελάτη στο χώρο πυρήνα και του εξυπηρετητή στο χώρο χρήστη), καθώς και τη διαδικασία έκδοσης μιας αίτησης από τον πελάτη και ολοκλήρωσης και αποστολής της από τον εξυπηρετητή. Στη συνέχεια, για την πλευρά του εξυπηρετητή αναλύονται οι αλλαγές που χρειάζονται στην πλευρά του εξυπηρετητή για την υλοποίηση του βελτιστοποιημένου μονοπατιού των δεδομένων, απευθείας από το σκληρό δίσκο στη στατική μνήμη του LANai (σχήματα 4.8 και 4.9).

Οι βασικές αλλαγές που χρειάζονται για την παράκαμψη της κεντρικής μνήμης τους συστήματος για τη μεταφορά δεδομένων από το σκληρό δίσκο στην κάρτα δικτύου είναι:

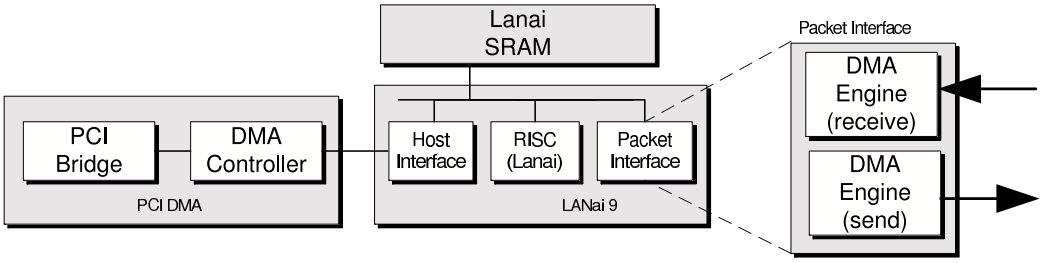
- αλλαγές στον οδηγό της κάρτας δικτύου Myrinet (GM βιβλιοθήκη σε χώρο χρήστη, kernel module driver και firmware)

- αλλαγές στον πυρήνα του Linux για τη μεταφορά δεδομένων με `O_DIRECT` σε κομμάτι της μνήμης του LANai που βρίσκεται σε χώρο διευθύνσεων του δίαυλου PCI. Ένα κομμάτι της στατικής μνήμης της κάρτας, απεικονίζεται στο χώρο φυσικών διευθύνσεων και χαρακτηρίζεται από το λειτουργικό σύστημα ως μνήμη στην οποία μπορεί να γίνει Ε/Ε με `O_DIRECT`.

5.4.1 Αλλαγές στο GM

Σύμφωνα με το μοντέλο που περιγράψαμε στο Κεφάλαιο 2, η αποστολή μηνυμάτων μέσω Myrinet προϋποθέτει αφενώς την αποθήκευση των δεδομένων προς αποστολή στο χώρο χρήστη (στην κεντρική μνήμη του συστήματος, σε κομμάτι που μπορεί να αποτελέσει πηγή μεταφοράς DMA), και αφετέρου κλήση της συνάρτησης `gm_send()` που αντιγράφει τα δεδομένα με τη μηχανή DMA της κάρτας Myrinet από την κεντρική μνήμη (στο χώρο χρήστη) στη στατική μνήμη του LANai (DMA controller και Host Interface σχήμα 5.12) και από εκεί με DMA (DMA send engine σχήμα 5.12) στο δίκτυο.

Οι αλλαγές που πρέπει να γίνουν στο GM είναι:



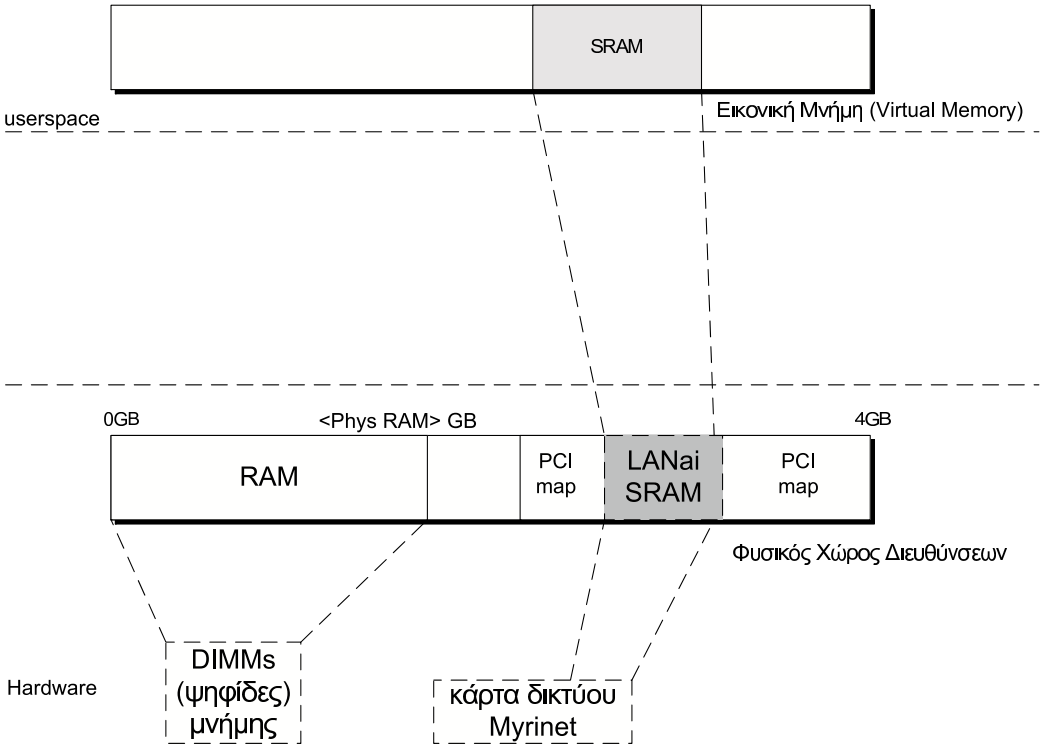
Σχήμα 5.12: Η κάρτα δικτύου Myrinet (με τα DMA engines)

- *δέσμευση του απομονωτή στη στατική μνήμη του LANai.*
  Η παράκαμψη της αποθήκευσης στην κεντρική μνήμη στο χώρο χρήστη υλοποιείται με την προσθήκη ενός δείκτη σε ένα κομμάτι μνήμης που δεσμεύεται στη στατική μνήμη του LANai μέσα στη δομή `lanai_globals_t` η οποία μοιράζεται σε όλο το firmware, στον driver και στη βιβλιοθήκη χώρου χρήστη του GM.

- *απεικόνιση του* `sram_buf` *στο χώρο εικονικών διευθύνσεων του συστήματος.*
  Αυτό το κομμάτι μνήμης που μπορεί να προσπελαστεί με φυσικές διευθύνσεις διαύλου απεικονίζεται στο χώρο εικονικών διευθύνσεων της διεργασίας, και έτσι εγγραφή σε αυτές τις διευθύνσεις σημαίνει εγγραφή απευθείας στον απομονωτή στη στατική μνήμη του Lanai.

- *αποστολή στο δίκτυο μέσω του απομονωτή στην SRAM της κάρτας δικτύου Myrinet.*
  Όπως αναφέρθηκε προηγουμένως, η αποστολή των δεδομένων μέσω της `gm_send()` συνίσταται στην αποθήκευσή τους σε ένα κομμάτι της κεντρικής μνήμης από το οποίο μπορούν να μεταφερθούν με DMA στη στατική μνήμη και να σταλούν στο δίκτυο. Η συνάρτηση `gm_send()` στη βιβλιοθήκη του οδηγού συσκευής (`libgm`) χρησιμοποιεί σαν όρισμα την εικονική διεύθυνση στην οποία είναι αποθηκευμένα τα δεδομένα προς αποστολή, για να την προωθήσει στο firmware, να μεταφραστεί σε φυσική διεύθυνση και να προγραμματιστεί η μηχανή DMA, η οποία θα εκτελέσει τη μεταφορά στη στατική μνήμη της κάρτας δικτύου. Όταν η αποστολή γίνεται από την ίδια τη στατική μνήμη του Lanai (από τον απομονωτή `sram_buf`), αυτή η μετάφραση (από εικονική σε φυσική) δεν υπάρχει στους πίνακες αντιστοιχίας εικονικών φυσικών διευθύνσεων, με αποτέλεσμα η αποστολή να αποτυγχάνει. Για αυτό το λόγο πρέπει η εικονική διεύθυνση να μεταφραστεί σε ένα offset μέσα στον απομονωτή, δηλαδή από τη εικονική διεύθυνση που βρίσκονται τα δεδομένα, να αφαιρεθεί η διεύθυνση που έχει απεικονιστεί ο απομονωτής στο χώρο

εικονικών διευθύνσεων, ώστε να προκύψει το σημείο που βρίσκονται τα δεδομένα μέσα στον απομονωτή (σχήμα 5.13).



Σχήμα 5.13: Η απεικόνιση της στατικής μνήμης του Lanai

5.4.2 Αλλαγές στον πυρήνα

Όπως αναφέρθηκε στο κεφάλαιο 3, η εκτέλεση λειτουργιών εισόδου / εξόδου για συσκευές αποθήκευσης (σκληρός δίσκος) γίνεται με ενδιάμεση αποθήκευση των δεδομένων σε απομονωτές στην `page_cache` με αποτέλεσμα την επιβάρυνση του επεξεργαστή και της κεντρικής μνήμης. Βέβαια, αναφέρθηκε επίσης ότι υπάρχει η δυνατότητα ανάγνωσης απευθείας από το σκληρο δίσκο (με παράκαμψη της `page_cache`) με τη μέθοδο Direct I/O.

Αφού υλοποιήθηκαν οι εφαρμογές (το module και ο εξυπηρετητής) και έγιναν οι παραπάνω αλλαγές (στο firmware στον οδηγό συσκευής Myrinet και στη βιβλιοθήκη), παρατηρήθηκε ότι η μεταφορά των δεδομένων με `O_DIRECT` από το σκληρό δίσκο στη μνήμη του Lanai δεν μπορούσε να εκτελεστεί[1]. Ο λό-

[1]Η επιστροφή της συνάρτησης read() ήταν EFAULT

γος είναι ότι με το O_DIRECT μεταφέρονται δεδομένα σε φυσικές διευθύνσεις μνήμης. Στην περίπτωσή μας, ο απομονωτής στον οποίο πρέπει να εγγραφούν τα δεδομένα προς αποστολή είναι πάνω στη στατική μνήμη της κάρτας δικτύου, δηλαδή σε χώρο διευθύνσεων διαύλου PCI. Ο απομονωτής αυτός ενώ απεικονίζεται στο χώρο εικονικών διευθύνσεων, οι φυσικές διευθύνσεις που του αντιστοιχούν δεν είναι διευθύνσεις κεντρικής μνήμης αλλά διαύλου, με αποτέλεσμα το λειτουργικό σύστημα να μην μπορεί να αντιμετωπίσει το συγκεκριμένο υλικό ως μνήμη. Αυτό οφείλεται στο γεγονός ότι για κάθε κομμάτι μνήμης υπάρχει μια δομή struct page που περιέχει δεδομένα που χαρακτηρίζουν το συγκεκριμένο κομμάτι.

Επομένως για να λειτουργήσει το μοντέλο της συσκευής gmblock, ήταν αναγκαίο να μπορεί το λειτουργικό σύστημα να αντιμετωπίσει τις διευθύνσεις διαύλου PCI σαν φυσικές διευθύνσεις μνήμης. Αυτό έγινε με την προσθήκη μίας ακόμη ζώνης στο διαχωρισμό της μνήμης, της Memory-mapped PCI. Η συγκεκριμένη ζώνη ορίζεται ως μνήμη που είναι κατειλημμένη, για να μην μπορεί το λειτουργικό να την παραχωρήσει σε διεργασίες που ζητούν μνήμη. Οι συγκεκριμένες διευθύνσεις πλέον, αντιστοιχούνται σε δομές struct page και μπορούν να αντιμετωπιστούν ως διευθύνσεις μνήμης, στις οποίες μπορεί να γίνει DMA. Παρακάτω φαίνονται αναλυτικά οι αλλαγές που χρειάστηκαν στον πυρήνα για την επίτευξη του στόχου αυτού.

Στην αρχή καλό είναι να περιγραφεί ο τρόπος με τον οποίο μεταφράζεται μια κλήση συστήματος ανάγνωσης από κάποιο μέσο αποθήκευσης (στο χώρο χρήστη, read()) σε κλήσεις συναρτήσεων στον πυρήνα που πραγματοποιούν την αντιγραφή των δεδομένων από το μέσο αποθήκευσης στη μνήμη που δεσμεύεται από την εφαρμογή του χώρου χρήστη.

Η read() που καλεί ο εξυπηρετητής, είναι ουσιαστικά η sys_read() που καλείται μέσα στον πυρήνα. Στο παρακάτω σχήμα φαίνεται με γραφικό τρόπο η ακολουθία κλήσεων που γίνεται για την πρόσβαση στα δεδομένα και την προσθήκη ενός bio (Κεφάλαιο 3 / block device drivers) στην ουρά αιτήσεων του οδηγού του μέσου αποθήκευσης.

Όπως φαίνεται στα σχήματα 5.14 και 5.15, η κλήση συστήματος read() σε χώρο χρήστη υλοποιείται από την generic_file_read() που αρχικοποιεί τους descriptors iovec και kiocb και καλεί την __generic_file_aio_read(). Η τελευταία συνάρτηση βεβαιώνεται ότι ο απομονωτής στο χώρο χρήστη που περιγράφεται από τον iovec είναι έγκυρος, ελέγχει και τις υπόλοιπες παραμέτρους, για να περαστούν ως ορίσματα στην generic_file_direct_IO(). Με τη χρήση της μεθόδου Direct I/O, που ισοδυναμεί με μια κλήση στην

```
if (filp->f_flags & O_DIRECT) {
  if (count == 0 ||
      *ppos > filp->f_mapping->host->i_size)
         return 0;
  retval =
      generic_file_direct_IO(READ, iocb, iov, *ppos, 1);
  if (retval > 0)
      *ppos += retval;
      file_accessed(filp);
      return retval;
}
```

Σχήμα 5.14: Ανάγνωση με O DIRECT

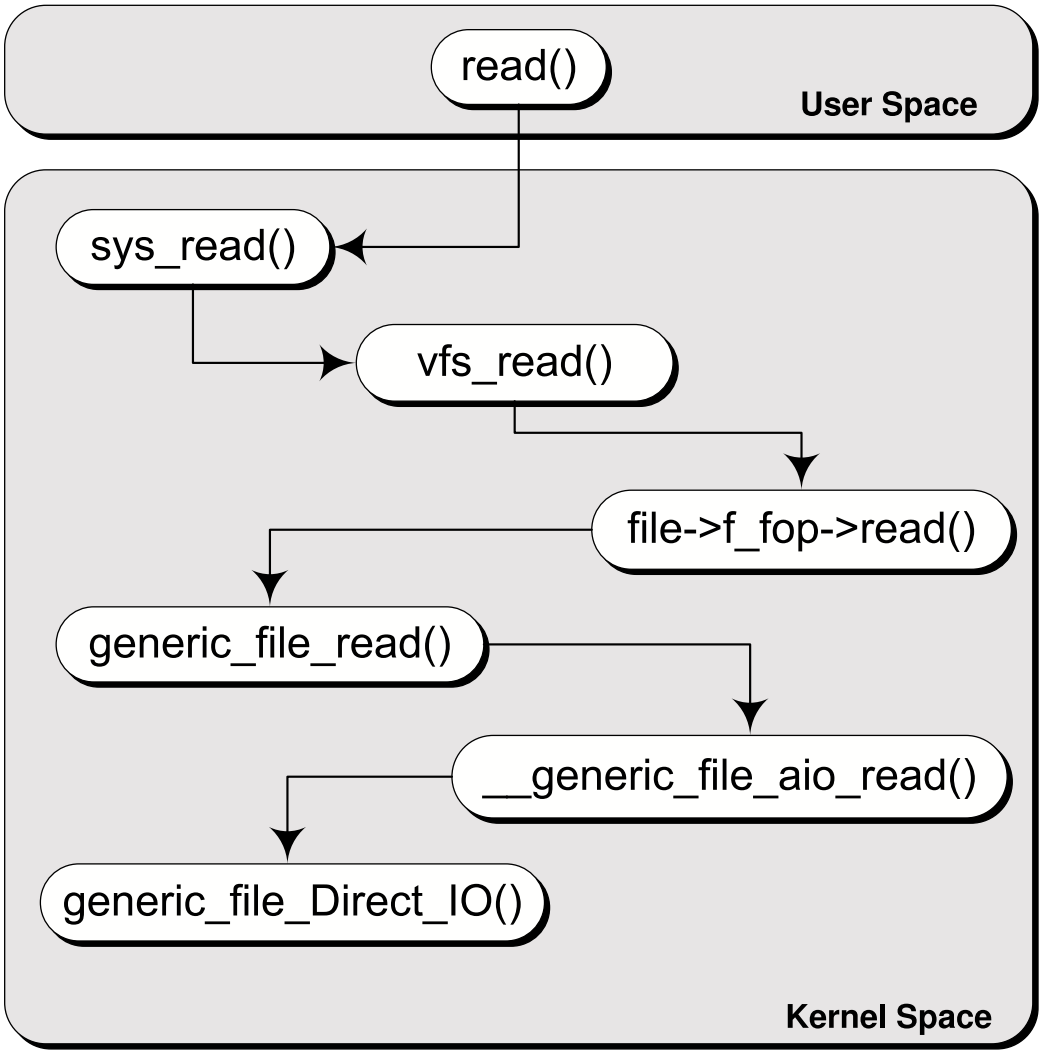
`__blockdev_direct_IO()`[2] χωρίζονται τα δεδομένα προς ανάγνωση σε blocks, εντοπίζονται στο μέσο αποθήκευσης και ενημερώνονται οι bio descriptors που περιγράφουν τις λειτουργίες που πρέπει να επιτελεστούν. Όταν η συνάρτηση `__blockdev_direct_IO()` επιστρέψει, τα δεδομένα είναι ασφαλώς γραμμένα στους απομονωτές χώρου χρήστη (στη LANai SRAM δηλαδή), η `__generic_file_aio_read()` με της σειρά της ενημερώνει τις δομές που αναφέρονται σε πληροφορίες για τα δεδομένα που προσπελάστηκαν και επιστρέφει. Περισσότερες πληροφορίες για την απευθείας προσπέλαση δεδομένων από μέσα αποθήκευσης (Direct I/O) υπάρχουν στο Κεφάλαιο 3.

Η προσθήκη της επιπλέον ζώνης της μνήμης συνίσταται στα εξής βήματα:

- ορισμός των συνόρων της ζώνης.
  Η ζώνη Memory-mapped PCI τοποθετείται μεταξύ του τέλους της φυσικής μνήμης (των ψηφίδων μνήμης RAM) και του ορίου των 4GB. Έτσι, ο πυρήνας του Linux γνωρίζει ότι το σύστημα έχει 4GB μνήμη και φτιάχνει `struct pages` και για τα 4GB. Αντικαθιστώντας τη μεταβλητή `max_pfn`[3] με μια προσωρινή μεταβλητή `max_realmem_pfn`, μπορούμε να διαχωρίσουμε την πραγματική μνήμη από την εικονική (τις διευθύνσεις φυσικής μνήμης από αυτές της memory-mapped pci μνήμης). Επομένως αναγκάζοντας τον πυρήνα να θεωρεί ότι η `max_pfn` είναι 4GB, έχουμε δομές `struct pages` ακόμα και για τις διευθύνσεις διαύλου PCI. Έτσι, η μηχανή DMA του σκληρού δίσκου μπορεί να προγραμματιστεί, για να γράψει τα ζητούμενα δεδομένα με `O_DIRECT` σε

[2] Η ανάλυσή της αφορά παραμέτρους που δεν κρίνεται χρήσιμο να αναφερθούν στην παρούσα εργασία

[3] Η max_pfn ορίζει το μέγιστο αριθμό μνήμης που έχει το τρέχον σύστημα



Σχήμα 5.15: Οι κλήσεις συστήματος για ανάγνωση με O DIRECT

φυσικές διευθύνσεις διαύλου PCI, στη στατική μνήμη του LANai, πάνω στην κάρτα δικτύου Myrinet.

- ένταξη της ζώνης στο σύνολο `ZONE_DMA` `ZONE_NORMAL` `ZONE_HIGHMEM` Στις παραπάνω ζώνες, προστίθεται η `ZONE_PCI` και το αποτέλεσμα εμφανίζεται στο `/proc/zoneinfo`.

Κεφάλαιο **6**

Επίλογος

6.1 Σύνοψη

Στην παρούσα εργασία αναλύθηκε, όσο το επιτρέπει ο σκοπός της, ένα μοντέλο μεταφοράς δεδομένων από συσκευές αποθήκευσης σε δίκτυο Myrinet αφού παρακάμφθηκε η ιεραρχία μνήμης. Κρίθηκε αναγκαίο να αναφερθούν βασικά στοιχεία των συσκευών αποθήκευσης, τρόποι ανάκτησης δεδομένων από αυτά καθώς και διαφορετικές προσεγγίσεις δικτυακών μέσων αποθήκευσης. Επίσης, περιγράφηκαν συνοπτικά τα δίκτυα διασύνδεσης που κυριαρχούν στις μέρες μας με έμφαση στο Myrinet, το δίκτυο που χρησιμοποιήθηκε στην παρούσα εργασία. Αναλύθηκαν τα βασικά σημεία της επικοινωνίας σε δίκτυο Myrinet, οι εφαρμογές που χρησιμοποιεί καθώς και τα εργαλεία / βιβλιοθήκες που υλοποιούν τη δικτυακή επικοινωνία. Δόθηκε έμφαση στο σενάριο αποστολής που αποτελεί ένα σημαντικό σκέλος της εργασίας. Αναφέρθηκαν επίσης στοιχεία από το λειτουργικό σύστημα Linux, με βάση το οποίο έγινε ο σχεδιασμός και η συγκεκριμένη υλοποίηση του μοντέλου. Αναλύθηκε εκτενώς ο τρόπος με τον οποίο το λειτουργικό σύστημα χειρίζεται τις συσκευές εισόδου / εξόδου, όσον αφορά τόσο στα μέσα αποθήκευσης όσο και στις συσκευές δικτύου. Κρίθηκε αναγκαίο να αναλυθούν επίσης τα στρώματα του πυρήνα που χειρίζονται συσκευές αποθήκευσης και δικτυακές συσκευές.

Στη συνέχεια, αφού παρουσιάστηκαν τα σημεία θεωρητικής προσέγγισης τα οποία αποτελούν τη βάση για το σχεδιασμό του μοντέλου, αναλύεται διεξοδικά ο τρόπος με τον οποίο επιτυγχάνεται μια δικτυακή μεταφορά δεδομένων για συσκευές αποθήκευσης σε ευρέως διαδεδομένα δίκτυα, όπως το TCP/IP καθώς και το Myrinet. Ακολούθησε ανάλυση του μοντέλου επικοινωνίας που υλοποιήθηκε με έμφαση στα σημεία παράκαμψης της ιεραρχίας μνήμης. Το μονοπάτι που ακολουθούν τα δεδομένα στο μοντέλο της συσκευής gmblock, υλοποιείται χωρίς τη μεσολάβηση του επεξεργαστή και της κεντρικής μνήμης του συστήματος εξυπηρέτησης με αποτέλεσμα το latency αντιγραφής των δεδομένων από το μέσο αποθήκευσης στο δίκτυο να μειώνεται δραματικά. Ο

επεξεργαστής του συστήματος καθώς και η κύρια μνήμη δεν επηρεάζονται από την αντιγραφή των δεδομένων στο δίκτυο, αφού αρκεί μία μεταφορά DMA μέσω του διαύλου PCI.

Τέλος παρουσιάζεται η υλοποίηση του συγκεκριμένου μοντέλου δικτυακής συσκευής αποθήκευσης (ανάγνωσης μόνο) με στόχο την περιγραφή του βελτιστοποιημένου μονοπατιού παρά την περιγραφή μιας εφαρμογής έτοιμης προς χρήση. Στόχος της παρούσας εργασίας είναι να αποδείξει ότι το μονοπάτι των δεδομένων προς αντιγραφή, παρακάμπτοντας τον επεξεργαστή και την κεντρική μνήμη, υλοποιείται με μικρές αλλαγές στον πυρήνα του λειτουργικού συστήματος Linux και στον οδηγό της κάρτας δικτύου Myrinet.

Η στατική μνήμη της κάρτας δικτύου Myrinet απεικονίζεται στο χώρο εικονικών διευθύνσεων στο σύστημα του εξυπηρετητή. Με πρόσθεση μίας επιπλέον ζώνης στη μνήμη, που αντιστοιχεί στο χώρο διευθύνσεων διαύλου PCI, η στατική (μοιραζόμενη) μνήμη του LANai εντάσσεται στο σύστημα διαχείρισης μνήμης του πυρήνα με αποτέλεσμα η μηχανή DMA του μέσου αποθήκευσης (σκληρός δίσκος, RAID array, κλπ) να μπορεί να αντιγράψει απευθείας (με Direct I/O) τα δεδομένα που ζητούνται από τον πελάτη μέσω του διαύλου PCI.

6.2 Επεκτάσεις

Η εφαρμογή του εξυπηρετητή που υλοποιήθηκε στην παρούσα εργασία δεν αποτελεί μια εφαρμογή τελικής χρήσης. Σ' αυτό συντέλεσαν πολλές παράμετροι, η κυριότερη των οποίων είναι τα πλαίσια του διαθέσιμου χρόνου για τη μελέτη, το σχεδιασμό και την υλοποίηση του μοντέλου. Μελλοντικές επεκτάσεις του μοντέλου μπορούν να άρουν επαναστατικά τα όρια για τη δικτυακή αποθήκευση πάνω από γρήγορα δίκτυα με έμφαση στο χαμηλό latency μεταφοράς δεδομένων μέσα στο υπολογιστικό σύστημα. Η σταθερότητα, η φορητότητα και η ευκολότερη εκσφαλμάτωση της εφαρμογής χώρου χρήστη, δεν επιτρέπουν τη μετάβαση σε διαφορετικό μοντέλο. Παρόλα αυτά μικρές αλλαγές στο μοντέλο μπορούν να επεκτείνουν τη λειτουργία του. Μερικές από τις πιθανές μελλοντικές επεκτάσεις παρουσιάζονται παρακάτω:

- πολλές αιτήσεις σε εξέλιξη.
  Με στόχο την απόδειξη της προσπελασιμότητας του βελτιστοποιημένου μονοπατιού (παράκαμψης του επεξεργαστή και της κεντρικής μνήμης του συστήματος του επεξεργαστή), η υλοποίηση της παρούσας εργασίας περιορίστηκε στο χειρισμό μίας αίτησης σε εξέλιξη τόσο στον πελάτη, όσο και στον εξυπηρετητή. Παρόλα αυτά υπάρχει πρόβλεψη για περισσότερες αιτήσεις σε εξέλιξη αφού χρησιμοποιείται το ψευδομοναδικό

αναγνωριστικό της αίτησης (`request_handle`) για την αντιστοίχιση των αιτήσεων gmblock στις αντίστοιχες αιτήσεις block του πελάτη.

- ανάγνωση-εγγραφή.
  Θα μπορούσε αρχικά να επεκταθεί η συγκεκριμένη εικονική συσκευή αποθήκευσης από ανάγνωσης-μόνο σε ανάγνωσης-εγγραφής. Δεδομένου πως η συγκεκριμένη υλοποίηση επιτυγχάνεται με χρήση της μεθόδου O_DIRECT για πρόσβαση στο αποθηκευτικό μέσο, ο πυρήνας γνωρίζει ανά πάσα στιγμή αν η cache περιέχει δεδομένα του δίσκου είτε σε εγγραφή είτε σε ανάγνωση. Επομένως, η προσθήκη της παραμέτρου "εγγραφής" μπορεί να επιτευχθεί αν υλοποιηθεί το βελτιστοποιημένο μονοπάτι για μηνύματα που λαμβάνονται από την κάρτα δικτύου Myrinet, χωρίς να λαμβάνουμε υπόψη τα προβλήματα που μπορούν να δημιουργηθούν λόγω της cache.

- parallel file system.
  Επέκταση της παρούσας συσκευής δικτυακής ανάκτησης δεδομένων θα μπορούσε να είναι η διερεύνηση ώστε να διαπιστωθεί αν μπορεί να υποστηρίξει ένα παράλληλο σύστημα αρχείων. Με αυτόν τον τρόπο, η μεταφορά των δεδομένων θα είναι αμεσότερη και θα μειωθεί σημαντικά η επίδραση που θα έχει μια μεταφορά σε ένα υπάρχον pfs στο εύρος ζώνης της μνήμης.

Βιβλιογραφία

[BCF+95] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29--36, Feb 1995.

[BRB98] Raoul A. F. Bhoedjang, Tim Rühl, and Henri E. Bal. User-level network interface protocols. *Computer*, 31(11):53--60, 1998.

[CFM+01] S. Coll, Jose Flich, Manuel P. Malumbres, Pedro Lopez, Jose Duato, and F. J. Mora. A first implementation of in-transit buffers on myrinet gm software. In *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, page 162, Washington, DC, USA, 2001. IEEE Computer Society.

[Cis] Cisco. http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ip.htm.

[CRU03] Olivier Cozette, Cyril Randriamaro, and Gil Utard. READ^2: Put Disks at Network Level. In *CCGRID'03, Workshop on Parallel I/O*, Tokyo (Japan), May 2003.

[DPB05] Marco Cesati Daniel P. Bovet. *Understanding the Linux Kernel, 3rd Edition*. O' Reilly, 2005.

[GC03] Teddy Surya Gunawan and Wentong Cai. Performance analysis of a myrinet-based cluster. *Cluster Computing*, 6(4):299--313, 2003.

[Geo02] Patrick Geoffray. OPIOM: Off-Processor I/O with Myrinet. *Future Gener. Comput. Syst.*, 18(4):491--499, 2002.

[GPP+01] Patrick Geoffray, CongDuc Pham, Louc Prylli, Bernard Tourancheau, and Roland Westrelin. Protocols and software for exploiting myrinet clusters. In *ICCS '01: Proceedings of the International Conference on Computational Sciences-Part I*, pages 233--242, London, UK, 2001. Springer-Verlag.

[HP03] John Hennessy and David Patterson. *Computer Architecture A Quantitative Approach*. Morgan Kaufmann, 2003.

[JC05] Greg Kroah-Hartman Jonathan Corbet, Alessandro Rubini. *Linux Device Drivers*. O' Reilly, 2005.

[JOY+99] Dongming Jiang, Brian O'Kelley, Xiang Yu, Sanjeev Kumar, Angelos Bilas, and Jaswinder Pal Singh. Application scaling under shared virtual memory on a cluster of smps. In *ICS '99: Proceedings of the 13th international conference on Supercomputing*, pages 165--174, New York, NY, USA, 1999. ACM Press.

[KKJ02] Kangho Kim, Jin-Soo Kim, and Sung-In Jung. GNBD/VIA: A Network Block Device over Virtual Interface Architecture on Linux. In *Proceedings of the 14th International Parallel and Distributed Processing Symposium (IPDPS)*, 2002.

[Kou] Kornilios-Antonios B. Kourtis. Σχεδίαση και Υλοποίηση μιας δικτυακής Συσκευής block στο λειτουργικό σύστημα linux. Διπλωματική Εργασία τμήματος ΗΜ-ΜΥ,ΕΜΠ.

[MAFS03] Kostas Magoutis, Salimah Addetia, Alexandra Fedorova, and Margo I. Seltzer. Making the Most Out of Direct-Access Network Attached Storage. In *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 189--202, Berkeley, CA, USA, 2003. USENIX Association.

[Myr03] Myricom. GM: A Message-Passing System for Myrinet Networks, 2003. http://www.myri.com/scs/GM-2/doc/html/.

[Pav] Pavel. Network block device. http://nbd.sourceforge.net/.

[sam] http://bsd7.cs.sunysb.edu/~samson/.

[SH02] Frank Schmuck and Roger Haskin. GPFS: A Shared-Disk File System for Large Computing Clusters. In *Proc. of the First Conference on File and Storage Technologies (FAST)*, pages 231--244, January 2002.

[SRO96] Steven R. Soltis, Thomas M. Ruwart, and Matthew T. O'Keefe. The Global File System. In *Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems*, pages 319--342, College Park, MD, 1996. IEEE Computer Society Press.

[Tan02] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall PTR., 2002.