ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Σχεδίαση και Υλοποίηση μηχανισμού απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10GbE

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νίκος Δ. Νικολέρης

Επιβλέπων: Νεκτάριος Κοζύρης,
Αν. Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2009

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

Σχεδίαση και Υλοποίηση μηχανισμού απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10GbE

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νίκος Δ. Νικολέρης

Επιβλέπων: Νεκτάριος Κοζύρης
Αν. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 27 Ιουλίου 2009.

.....	.....	.....
Ν. Κοζύρης	Π. Τσανάκας	Δ. Σούντρης
Αν. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

.....
Νίκος Δ. Νικολέρης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στην παρούσα εργασία σχεδιάζεται και υλοποιείται ένα πρωτόκολλο απευθείας απομακρυσμένης πρόσβασης στη μνήμη πάνω από το δίκτυο διασύνδεσης 10G Ethernet. Το περιβάλλον ανάπτυξης είναι ο πυρήνας του Linux τόσο ως λειτουργικό σύστημα στον κόμβο (host) που χρησιμοποιεί το πρωτόκολλο όσο και ως υλικολογισμικό (firmware) στον προσαρμογέα δικτύου (NIC).

Ο σχεδιασμός του πρωτοκόλλου SLURPoE (Simple Lightweight RDMA Protocol over Ethernet) έχει στόχο την πλήρη εκμετάλλευση του εύρους ζώνης (bandwidth) του δικτύου 10G Ethernet και τη διατήρηση χαμηλών χρόνων (latency). Οι επιδόσεις θα πρέπει να συνοδεύονται από μικρή επιβάρυνση για τους σκοπούς της επικοινωνίας στο CPU του κόμβου, αφού στόχος είναι η υπολογιστική τους ισχύ να διατίθεται εξολοκλήρου για τις πραγματικές ανάγκες επεξεργασίας των συστοιχιών. Η απαίτηση για χαμηλό ποσοστό επιβάρυνσης του CPU επιβάλει τη χρήση προγραμματιζόμενων προσαρμογέων δικτύων οι οποίοι μπορούν να επιφορτιστούν με αποδοτικότερο τρόπο του κόστους της επικοινωνίας.

Το μεγαλύτερο κομμάτι του οδηγού υλοποιείται πάνω στον προσαρμογέα δικτύου ενώ στον κόμβο επιλέγεται η πολιτική υλοποίησης όλων των λειτουργιών σε χώρο χρήστη. Στον πυρήνα του κόμβου ανατίθενται μόνο οι λειτουργίες που απαιτούν αυξημένα δικαιώματα και δεν μπορούν να υλοποιηθούν διαφορετικά.

Στο πλαίσιο των αναγκών της επικοινωνίας πάνω από το πρωτόκολλο SLURPoE διαμορφώνονται και οι απαιτήσεις σε υπολογιστική ισχύ καθώς και οι λειτουργικές που πρέπει να ικανοποιεί ο προγραμματιζόμενος προσαρμογέας δικτύου.

Λέξεις-Κλειδιά: Δίκτυα Διασύνδεσης, Υπολογιστικά Συστήματα Υψηλής Απόδοσης, 10G Ethernet, Απευθείας Απομακρυσμένη Πρόσβαση στη Μνήμη, Πρωτόκολλο Δικτύωσης στο Χώρο Χρήστη, InfiniBand, Myrinet, MX, MPI

Abstract

The objective of this study is the design and implementation of a remote direct memory access protocol over 10G Ethernet networks. The development framework is the Linux kernel both as an Operating System, on the host and as a Firmware, on the Network Interface Card.

The design of the SimpLe User–level RDMA Protocol over Ethernet (SLURPoE) targets into achieving low latency along with the line–rate of 10G Ethernet. The performance achieved should impose as low overhead as possible to the host CPU, as its processing power should be devoted to the real computing needs of a compute cluster. This need for low host CPU utilization is achieved using programmable NICs, which can efficiently undertake the overhead of the communication protocol processing.

Most of the driver is implemented on the programmable NIC. On the host computer the driver is implemented as a user–level networking protocol. The kernel is only invoked when the protocol needs to execute higher privileged operations.

Finally both the functional and the performance specifications of the programmable NIC are configured to the needs of the communication over the SLURPoE.

Keywords: Interconnection Networks, High Performance Computing, 10G Ethernet, Remote Direct Memory Access, User Level Networking, InfiniBand, Myrinet, MX, MPI.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου υπό την επίβλεψη του Αναπληρωτή Καθηγητή 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 εκπόνηση της διπλωματικής αυτής αποτελεί έμπνευση των υποψήφιων διδακτόρων Αναστάσιου Νάνου και Ευάγγελου Κούκη. Η βοήθεια τους ήταν ανεκτίμητη τόσο για τις καίριες παρεμβάσεις τους στο σχεδιασμό και την υλοποίηση του πρωτοκόλλου όσο και για τις τεχνικές γνώσεις που μου μετέδωσαν. Χωρίς τη συμβολή τους η διπλωματική εργασία δε θα ήταν εφικτή.

Επίσης θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Κορνήλιο Κούρτη με τη βοήθεια του οποίου κατάφερα να ξεπεράσω αρκετά τεχνικά προβλήματα κατά το στάδιο της υλοποίησης, καθώς και όλα τα μέλη του εργαστηρίου Υπολογιστικών Συστημάτων για τις πολύτιμες συμβουλές τους.

Τέλος θα ήθελα να ευχαριστήσω τη Μάγδα Φετάνη, που με τις επισημάνσεις για ορθογραφικά και συντακτικά λάθη συνέβαλε σημαντικά στη διόρθωση της εργασίας αυτής.

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

Στις μέρες μας, η ανάγκη για διαθέσιμη υπολογιστική ισχύ αυξάνεται ραγδαία. Εφαρμογές που προσομοιώνουν προβλήματα φυσικής, σεισμολογίας ή ιατρικής δεν μπορούν να εκτελεστούν σε συμβατικές υπολογιστικές μηχανές. Οι παράλληλες αρχιτεκτονικές αντιτείνουν στο πρόβλημα της βελτίωσης των επιδόσεων τη χρήση πολλών επεξεργαστών. Ωστόσο, η αποδοτικότητα και η ύπαρξη πολλαπλών επεξεργαστών σε μια αρχιτεκτονική περιορίζεται σημαντικά από το κόστος της διασύνδεσης τους. Το πρόβλημα της διασύνδεσης εμφανίζεται τόσο σε αρχιτεκτονικές συμμετρικής πολυεπεξεργαστασίας (Symetric Multiprocessing Systems – SMP) όσο και σε αρχιτεκτονικές συστοιχιών υπολογιστών.

Σε αντίθεση με τις αρχιτεκτονικές συμμετρικής πολυεπεξεργασίας, οι συστοιχίες υπολογιστικών συστημάτων έχουν πολύ καλύτερες δυνατότητες κλιμάκωσης και καλύτερο λόγο κόστους προς απόδοση. Τα πλεονεκτήματα των συστοιχιών υπολογιστών είναι αυτά που τα κάνουν αρκετά ελκυστικά για τα υπολογιστικά συστήματα υψηλής επίδοσης. Η παρούσα εργασία ασχολείται με ένα από τα πιο σημαντικά προβλήματα της αρχιτεκτονικής αυτής: τη διασύνδεση των κόμβων σε μια συστοιχία μέσω ενός δικτύου διασύνδεσης υψηλής επίδοσης.

Η εμφάνιση των πρώτων συστοιχιών υπολογιστών ακολούθησε την εμφάνιση των πρώτων υποδομών δικτύωσης. Η ανάγκη για υπολογιστική ισχύ πέρα από τα όρια ενός μόνο υπολογιστικού συστήματος οδήγησε στη διασύνδεση πολλών συστημάτων σε συστοιχίες και στην ανάπτυξη παράλληλων μοντέλων προγραμματισμού. Το μέγεθος των αναγκών για υπολογιστική ισχύ από πολλές επιστήμες οδήγησαν στην ανάπτυξη ισχυρότερων συστοιχιών υπολογιστών με εκατοντάδες κόμβους.

Η ύπαρξη πολλών κόμβων σε μια συστοιχία δεν αποτελεί τετριμμένο ζήτημα ακόμα και σήμερα. Καθορίζεται άμεσα από τις δυνατότητες κλιμάκωσης (scale) του δικτύου διασύνδεσης. Επιπλέον η υπολογιστική ισχύς που είναι διαθέσιμη εξαρτάται σε πολύ μεγάλο βαθμό από την αποδοτικότητα του πρωτοκόλλου διασύνδεσης. Στόχος είναι η επικοινωνία να μπορεί να γίνεται σε χρόνους αρκετά μικρούς έτσι, ώστε να μην αδρανεί κανένας κόμβος αναμένοντας κάποιο μήνυμα και το κόστος της επεξεργασίας του πρωτοκόλλου επικοινωνίας να είναι μικρό για το CPU των κόμβων.

Τα τελευταία χρόνια οι σχεδιαστές δικτύων διασύνδεσης έχουν αναπτύξει πρωτόκολλα και υλικό που ικανοποιούν τις βασικές απαιτήσεις των δικτύων διασύνδεσης υψηλών επιδόσεων: εξαιρετικά μικρός χρόνος αρχικής απόκρισης (latency), πολύ μεγάλο εύρος ζώνης και μηδενική επιβάρυνση του CPU του κόμβου. Τα πιο διαδεδομένα από αυτά είναι το InfiniBand το Myrinet και το Ethernet. Το κάθε ένα από αυτά παρουσιάζει πλεονεκτήματα και μειονεκτήματα τα οποία θα αναλυθούν στη συνέχεια, υπό το πρίσμα των αναγκών για την πραγματοποίηση της εργασίας αυτής.

1.1 Σκοπός

Η παρούσα εργασία αφορά στη μελέτη σύγχρονων δικτύων διασύνδεσης, στο σχεδιασμό και την υλοποίηση ενός πρωτοκόλλου επικοινωνίας με μηδενικά αντίγραφα, με στόχο τον καθορισμό των απαιτήσεων σε υλικό για μια αρχιτεκτονική δικτύου διασύνδεσης υψηλών επιδόσεων.

Αρχικά μελετήθηκαν τα υπάρχοντα πρωτόκολλα διασύνδεσης υπολογιστών σε συστοιχίες. Οι παράμετροι και οι ανάγκες στις συστοιχίες των υπολογιστών οριοθέτησαν την προσέγγιση στο πειραματικό πρωτόκολλο που αναπτύχθηκε. Παράλληλα μέσω της βιβλιογραφικής μελέτης τέθηκαν και οι λειτουργικές απαιτήσεις του πρωτοκόλλου.

Ακολουθεί ο σχεδιασμός του πρωτοκόλλου. Οι σχεδιαστικές επιλογές που έγιναν οδήγησαν σε ένα πρωτόκολλο Απομακρυσμένης Απευθείας Πρόσβασης στη Μνήμη (Remote Direct Memory Access) πάνω από 10G Ethernet. Το πρωτόκολλο SLURPoE (Simple Lightweight RDMA Protocol Ethernet) έχει στόχο την πλήρη εκμετάλλευση του εύρους ζώνης (bandwidth) του δικτύου 10G Ethernet και τη διατήρηση χαμηλών χρόνων αρχικής απόκρισης (latency). Παράλληλα η απαίτηση για μικρή επιβάρυνση στο CPU του κόμβου (host) που χρησιμοποιεί το πρωτόκολλο οδήγησε στη σύνθεση ενός προγραμματιζόμενου προσαρμογέα δικτύου υψηλών επιδόσεων για δίκτυα 10G Ethernet.

Το σχεδιασμό ακολούθησε η πραγματοποίηση μιας πρότυπης υλοποίησης στο προγραμματιστικό περιβάλλον του πυρήνα του λειτουργικού συστήματος Linux. Ο οδηγός του πρωτοκόλλου χρησιμοποιεί αρκετά από τα υποσυστήματα του Linux όπως το υποσύστημα διαχείρισης της μνήμης, το μηχανισμό των διακοπών, των ουρών εργασίας και της δικτυακής διεπαφής. Ο προγραμματιζόμενος προσαρμογέας δικτύου υλοποιήθηκε με τη χρήση της πειραματικής κάρτας IQ81342SC, από την Intel και μιας κοινής κάρτας δικτύου 10G Ethernet.

Στο τελευταίο στάδιο η πρότυπη υλοποίηση αξιολογήθηκε με βάση τους στόχους που τέθηκαν. Παράλληλα καθορίστηκαν μελλοντικές επεκτάσεις που θα μπορούσαν να συντελέσουν τόσο στη βελτίωση της αξιοπιστίας αλλά και στην αύξηση των επιδόσεων του πρωτοκόλλου.

1.2 Η δομή του κειμένου

Για την πλήρη περιγραφή του πρωτοκόλλου SLURPoE είναι απαραίτητη η εξέταση ζητημάτων, τα οποία επικαθόρισαν το τελικό αποτέλεσμα. Η περιγραφή ξεκινά από το δεύτερο κεφάλαιο το οποίο περιέχει το απαραίτητο θεωρητικό υπόβαθρο για τη λήψη και την κατανόηση των σχεδιαστικών επιλογών αλλά και της υλοποίησης.

Πιο συγκεκριμένα το πρώτο κομμάτι του κεφαλαίου αφορά στην περιγραφή των πιο διαδεδομένων δικτύων διασύνδεσης που χρησιμοποιούνται στην αγορά των συστημάτων υψηλής απόδοσης. Η παρουσίαση της βιβλιογραφικής μελέτης στο στάδιο αυτό έχει ως στόχο την παρουσίαση των παραμέτρων που οδήγησαν τελικά στο σχεδιασμό του SLURPoE. Στο δεύτερο κομμάτι γίνεται μια αναφορά στις προγραμματιστικές διεπαφές του Linux που χρησιμοποιήθηκαν κατά το στάδιο της υλοποίησης. Στη συνέχεια, γίνεται μια προσπάθεια να κατανοηθούν σε βάθος οι μηχανισμοί ώστε η λειτουργία τους να μπορεί να εκτιμηθεί σε σχέση με τα αποτελέσματα.

Στο τρίτο κεφάλαιο περιγράφεται ο σχεδιασμός του πρωτοκόλλου SLURPoE. Σε κάθε περίπτωση περιγράφονται οι σχεδιαστικές επιλογές καθώς και οι εναλλακτικές τους. Η κάθε μια αιτιολογείται και τίθενται οι στόχοι που θα πρέπει να ικανοποιούν τα αποτελέσματα.

Στο τέταρτο κεφάλαιο γίνεται σύντομη περιγραφή της υλοποίησης. Περιγράφονται τα κομμάτια του οδηγού με βάση τη λειτουργία τους. Πιο συγκεκριμένα αναφέρονται τα πρότυπα των συναρτήσεων καθώς και οι τύποι των δομών, ως μια απεικόνιση των λειτουργιών και των αντικείμενων του πρωτοκόλλου αντίστοιχα.

Τέλος στο πέμπτο κεφάλαιο περιγράφονται τα αποτελέσματα από την υλοποίηση του πρωτοκόλλου, ενώ στο έκτο γίνεται μια αναφορά σε παρόμοιες εργασίες και στις προοπτικές επέκτασης του πρωτοκόλλου.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Το κεφαλαίο αυτό έχει ως σκοπό την εισαγωγή στο σχεδιασμό και την υλοποίηση του πρωτοκόλλου SLURPoE. Χωρίζεται σε δύο μέρη: το πρώτο αφορά στην περιγραφή των Δικτύων Διασύνδεσης και το δεύτερο στην περιγραφή της προγραμματιστικής διεπαφής του Linux.

Στόχος είναι η κατανόηση των απαραίτητων παραμέτρων για την αξιολόγηση των δικτύων διασύνδεσης, αλλά και των πιθανών σχεδιαστικών επιλογών. Η περιγραφή ωστόσο δεν είναι τυπική, καθώς εστιάζει στα πλεονεκτήματα και τα μειονεκτήματα των πιο διαδεδομένων δικτύων διασύνδεσης. Θέτει τα πλαίσια στα οποία το SLURPoE έχει νόημα και τα σημεία στα οποία ξεχωρίζει από εργασίες στο παρελθόν.

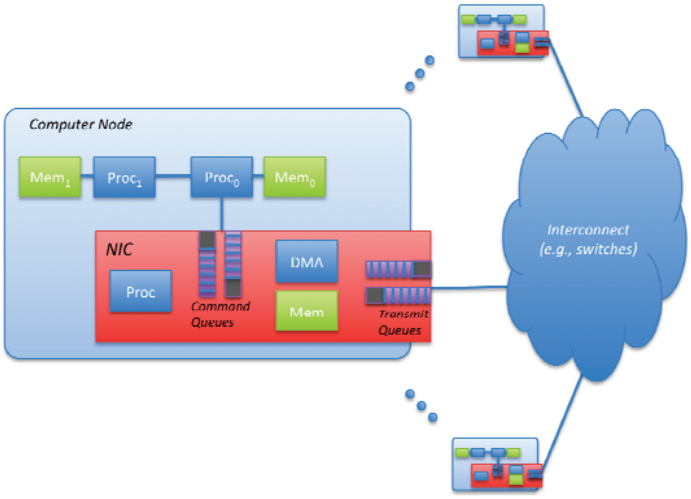
Στο δεύτερο κομμάτι η περιγραφή του Linux αποτελεί απαραίτητη βάση για την κατανόηση της υλοποίησης. Η πολυπλοκότητα του εγχειρήματος απαιτεί εις βάθος κατανόηση του προγραμματιστικού περιβάλλοντος του Linux. Σε πολλά σημεία άλλωστε η υλοποίηση καθορίζει σε μεγάλο βαθμό και τις επιδόσεις των οδηγών.

2.1 Δίκτυα Διασύνδεσης

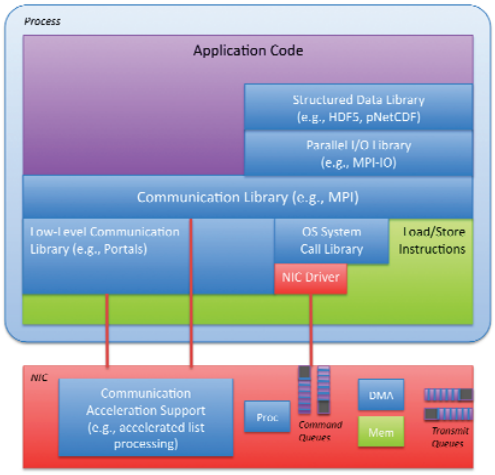
Ένα σύγχρονο σύστημα υψηλής απόδοσης αποτελείται από κόμβους οι οποίοι περιέχουν επεξεργαστές, μνήμη και έναν προσαρμογέα δικτύου. Στις πιο διαδεδομένες αρχιτεκτονικές ο προσαρμογέας δικτύου βρίσκεται στο υποσύστημα Ε/Ε του κόμβου (πχ. το PCI-Express). Μια τυπική διάταξη προσαρμογέα φαίνεται στο σχήμα 2.1. Αυτή περιέχει έναν επεξεργαστή, μια μνήμη, μια μηχανή DMA, ουρές εντολών για την αλληλεπίδραση με το λογισμικό και ουρές για τη μετάδοση από και προς το δίκτυο.

Στο σχήμα 2.2 φαίνεται η αρχιτεκτονική του λογισμικού που τρέχει σε ένα σύστημα υψηλής απόδοσης. Για παράδειγμα, πολλές εφαρμογές δε θα χρησιμοποιήσουν μια βιβλιοθήκη παράλληλης Ε/Ε. Στο σχήμα επίσης φαίνονται και διάφοροι πιθανοί τρόποι για την αλληλεπίδραση με τον προσαρμογέα δικτύου. Σε αυτούς συμπεριλαμβάνονται ο οδηγός του λειτουργικού συστήματος, η βιβλιοθήκη παράκαμψης του λειτουργικού ακόμα και εντολές φόρτωσης/αποθήκευσης σε συστήματα με υποστήριξη

υλικού για Καθολικό Χώρο Διευθύνσεων (Global Address Space). Η απόδοση ενός



Σχήμα 2.1: Μια ενδεικτική αρχιτεκτονική του υλικού των δικτύων διασύνδεσης.



Σχήμα 2.2: Μια ενδεικτική αρχιτεκτονική του λογισμικού των δικτύων διασύνδεσης.

δικτύου διασύνδεσης καθορίζεται από ένα σύνολο παραγόντων εκ των οποίων οι πιο σημαντικοί είναι: το μέγιστο εύρος ζώνης και ο χρόνος αρχικής απόκρισης μικρών μηνυμάτων. Εκτός από την επίδοση των προσαρμογέων δικτύου και των μεταγωγέων στο δίκτυο, οι δύο παράμετροι επηρεάζονται σημαντικά από τη συνολική σχεδίαση του συστήματος. Η επίδοση της μνήμης, η χρησιμοποίηση της κρυφής μνήμης (cache), ακόμα και η αρχιτεκτονική του υλικού μπορεί να έχουν ένα σοβαρό αντίκτυπο στην απόδοση του δικτύου διασύνδεσης.

- Το μέγιστο εύρος ζώνης αυξάνεται με σταθερό ρυθμό αν και δεν μπορεί να ακολουθήσει τους εκθετικούς ρυθμούς ανάπτυξης της επίδοσης των επεξεργαστών, έχει ξεπεράσει κατά πολύ την ανάπτυξη που έχει υποστεί η επίδοση της μνήμης. Αν και το γεγονός αυτό στο μέλλον αναμένεται να αλλάξει, σήμερα υπάρχει μεγάλη αβεβαιότητα στα υλικά, όπως τα οπτικά καλώδια που υιοθετεί η βιομηχανία σε αντίθεση με το απαγορευτικό κόστος και τα φυσικά όρια των αυξανόμενων ρυθμών των σημάτων στα καλώδια χαλκού. Η υιοθέτηση των οπτικών καλωδίων και το αυξανόμενο κόστος της μεταγώγησης στο μέγιστο εύρος ζώνης, έχουν οδηγήσει στην ανάπτυξη ενδιαφέροντος για εναλλακτικές τοπολογίες δικτύων.

- Ο χρόνος αρχικής απόκρισης μειώνεται ασυμπτωτικά με αποτέλεσμα ο χρόνος αρχικής απόκρισης στη διεπαφή MPI να είναι ελαφρώς μεγαλύτερος από το 1usec στα μοντέρνα δίκτυα διασύνδεσης των κορυφαίων συστημάτων υψηλής απόδοσης. Μόνο ένα μικρό μέρος του χρόνου αρχικής απόκρισης οφείλεται στην καθυστέρηση μετάδοσης του μηνύματος πάνω στο καλώδιο. Για την ακρίβεια μεγάλο μέρος του χρόνου αρχικής απόκρισης στη διεπαφή MPI καταναλώνεται στους κόμβους της πηγής και του προορισμού κατά τη μετακίνηση των δεδομένων από το δίκτυο στον επεξεργαστή του κόμβου και στη στοίβα των πρωτοκόλλων επικοινωνίας. Αν και υπάρχει ενδιαφέρον για μείωση του χρόνου μετακίνησης των δεδομένων με την παράκαμψη του PCI-Express, η προσπάθεια να περιοριστεί η επιβάρυνση της επικοινωνίας από το λογισμικό έχει επικεντρωθεί κυρίαρχα στην επιτάχυνση των εργασιών επεξεργασίας μηνυμάτων με την υποστήριξη και από το υλικό.

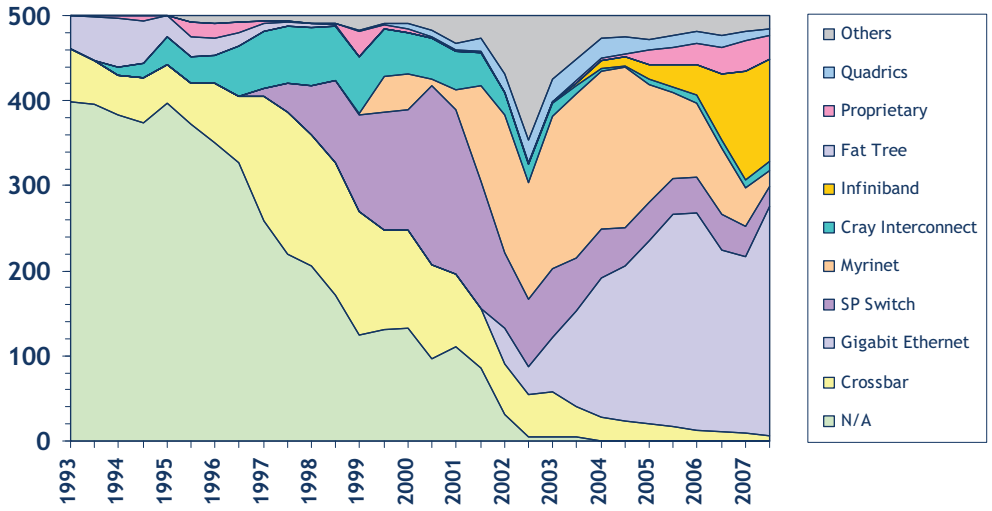
Αυτό που τελικά έχει σημασία για τα σύγχρονα συστήματα υψηλής απόδοσης, είναι η απόλυτη διαθέσιμη φυσική υπολογιστική ισχύς να μπορεί να αποδοθεί στους χρήστες μπορεί με όσο το δυνατόν λιγότερες απώλειες. Τα ερωτήματα ωστόσο για το τι πρέπει να έχει ένα δίκτυο διασύνδεσης είναι πιο πολύπλοκα, ειδικά όταν εμπλέκεται και το κριτήριο του κόστους.

Στο κεφάλαιο αυτό παρουσιάζουμε τα πιο διαδεδομένα δίκτυα διασύνδεσης, τα τελευταία χρόνια, όπως άλλωστε αυτό αποδεικνύεται και από το μερίδιο τους στην αγορά συστημάτων υψηλής απόδοσης[2].

2.1.1 Myrinet

Εισαγωγή

Το Myrinet[3] είναι ένα δίκτυο υψηλής επίδοσης και βασίζεται σε τεχνολογίες που αναπτύχθηκαν για τη διασύνδεση πολλαπλών επεξεργαστών σε αρχιτεκτονικές MPP (Massivily Parallel Processors). Προσφέρει ταχύτητες σύνδεσης μέχρι και 10Gbps, δυνατότητα μετάδοσης δεδομένων και στις δύο κατευθύνσεις (full duplex) και μικρούς χρόνους αρχικής απόκρισης. Το Myrinet βρίσκει εφαρμογή σε συστοιχίες υπολογι-



Σχήμα 2.3: Τα στατιστικά δικτύων διασύνδεσης στη λίστα των 500 πιο ισχυρών υπολογιστικών συστημάτων.

στών και γενικότερα χρησιμοποιείται ως Δίκτυο Περιοχής Συστήματος (System Area Network – SAN).

Για το δίκτυο Myrinet υπάρχουν δύο διαθέσιμες εκδόσεις: το Myri–10G και το Myrinet – 2000. Στο φυσικό επίπεδο το Myrinet–2000 προσφέρει δυνατότητα μετάδοσης δεδομένων προς τις δύο κατευθύνσεις, σε οπτικές συνδέσεις σημείου–προς–σημείο 2+2Gbps. Το Myri–10G βασίζεται στο ίδιο φυσικό επίπεδο (PHY, layer – 1) όπως και το 10G Ethernet, αυξάνοντας έτσι το εύρος διαύλου στα 10Gbps και μπορεί να χρησιμοποιήσει ως επίπεδο συνδέσμου είτε το 10G Ethernet, είτε το Myrinet με δρομολόγηση από την πηγή.

Οι κόμβοι σε ένα δίκτυο Myrinet διασυνδέονται με μεταγωγείς τύπου Crossbar, σε μια τοπολογία Clos. Ένα από τα πιο σημαντικά χαρακτηριστικά του δικτύου είναι η δρομολόγηση από την πηγή: το δίκτυο χαρτογραφείται έτσι ώστε κάθε κόμβος που συμμετέχει να γνωρίζει τη διαδρομή μέχρι κάποιον άλλο, χρησιμοποιώντας δρομολόγηση up/down. Κάθε πακέτο περιέχει όλη την πληροφορία για τη διαδρομή μέχρι τον προορισμό του, ως μια σειρά από πόρτες που πρέπει να περάσει μέσα σε κάθε μεταγωγέα. Για την επικοινωνία των κόμβων χρησιμοποιούνται πακέτα μεταβλητού μήκους.

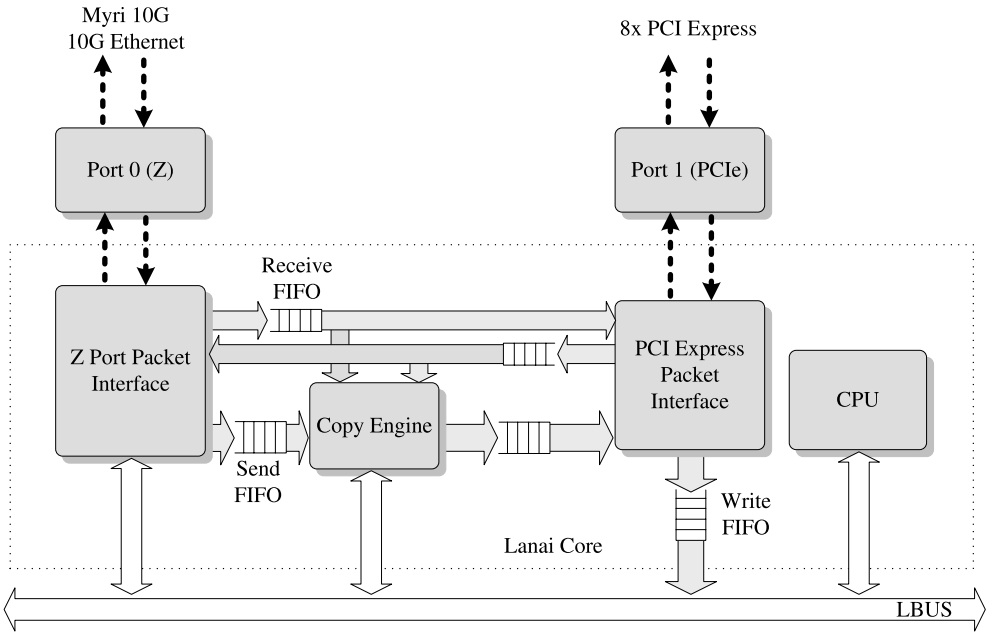
Για να επιτευχθεί η ελάχιστη, κατά το δυνατό, επιβάρυνση του επεξεργαστή από τη χρήση του δικτύου, το Myrinet χρησιμοποιεί τεχνικές υλοποίησης του πρωτοκόλλου στο χώρο χρήστη. Με τον τρόπο αυτό εξασφαλίζεται η απεμπλοκή του πυρήνα από το κρίσιμο μονοπάτι. Σε αυτό το μοντέλο, μια εφαρμογή μπορεί να ελέγχει τη διεπαφή του δικτύου απευθείας. Καθώς ο πυρήνας δεν εμπλέκεται στην επικοινωνία, το ρόλο του αναλαμβάνουν μια βιβλιοθήκη στο χώρο χρήστη, καθώς και το υλικολογισμικό (firmware) στη διεπαφή δικτύου. Η ανταλλαγή δεδομένων μεταξύ της εφαρμογής και του προσαρμογέα δικτύου γίνεται με ένα μηχανισμό που προετοιμάζεται από κώδικα

με αυξημένα δικαιώματα. Ο κώδικας αυτός υλοποιείται μέσα σε ένα module του πυρήνα του λειτουργικού.

Μια εφαρμογή μπορεί να έχει τον έλεγχο του προσαρμογέα δικτύου με την αντιστοίχιση στο χώρο μνήμης της, μέρους της φυσικής του μνήμης. Χρησιμοποιεί εντολές φόρτωσης/αποθήκευσης σε συγκεκριμένες θέσεις μνήμης, πραγματοποιώντας έτσι την επικοινωνία.

Οι προσαρμογείς δικτύου στο Myrinet

Οι προσαρμογείς δικτύου στο Myrinet συνδέονται μέσω του περιφερειακού διαύλου σε έναν κόμβο, που στην περίπτωση του Myri–10G είναι το PCI-Express. Ενσωματώνουν έναν επεξεργαστή RISC που ονομάζεται Lanai, ο οποίος αναλαμβάνει το μεγαλύτερο μέρος της επεξεργασίας του πρωτοκόλλου του δικτύου, καθώς και μια μικρή μνήμη SRAM για τις ανάγκες του Lanai. Ο πυρήνας του Lanai, όπως φαίνεται και στο σχήμα 2.4 αποτελείται από ένα CPU, μια μηχανή αντιγραφής, δύο προσαρμογείς πακέτων, κάθε μια με προσωρινούς χώρους αποστολής και λήψης μέσα στο chip, μια πόρτα τύπου Z (XAUI Myri-10G /10G Ethernet) και μια πόρτα PCI Express. Ο Τοπικός Δίαυλος (Local Bus – LBUS) επιτρέπει την πρόσβαση σε μια πολύ γρήγορη και σύγχρονη στατική μνήμη SRAM. Ο Lanai περιέχει μέσα στο chip του ενδιάμε-

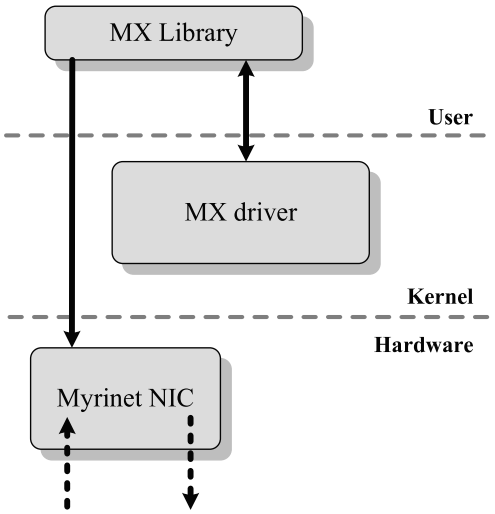


Σχήμα 2.4: Ο Lanai.

σους χώρους αποθήκευσης, οι οποίοι στην πιο αποδοτική λειτουργία του MX, κρατούν την πληροφορία ενός πακέτου. Μόνο οι κεφαλίδες των πακέτων αποθηκεύονται στην SRAM, μειώνοντας έτσι σημαντικά το φόρτο της και αυξάνοντας τους κύκλους μνήμης που είναι διαθέσιμοι για το CPU.

Myrinet Express: Το σύστημα διαβίβασης μηνυμάτων του Myrinet

Το MX [4] (Myrinet Express) είναι ένα λογισμικό διαβίβασης μηνυμάτων χαμηλού επιπέδου προσαρμοσμένο στο Myrinet. Το MX υποστηρίζει μοντέρνες διεπαφές middleware όπως το MPI και το VI, εκμεταλλευόμενο τις ικανότητες επεξεργασίας στον προσαρμογέα δικτύου. Επιτρέπει την εξομοίωση του Ethernet σε ταχύτητες συνδέσμου με μικρή επιβάρυνση και προσφέρει μια απλή προγραμματιστική διεπαφή για την ανάπτυξη εφαρμογών που κάνουν χρήση του Myrinet. Τα κυρίαρχα χαρακτηρι-



Σχήμα 2.5: Η αρχιτεκτονική του MX.

στικά του MX συνοψίζονται στα εξής:

- Προστατευόμενη και ανεξάρτητη πρόσβαση στις δικτυακές λειτουργίες για εφαρμογές στο χώρο χρήστη.
- Διαφανής δήλωση της μνήμης.
- Χαμηλός χρόνος πρώτης απόκρισης για μικρά μηνύματα (της τάξης των 5usec).
- Ασύγχρονες λειτουργίες επικοινωνίας.
- Υποστήριξη αποστολής και λήψης μηνυμάτων από και προς διάσπαρτες θέσεις στη μνήμη.
- Μηχανισμός γενικευμένης ταυτοποίησης μηνυμάτων.
- Επικάλυψη της επικοινωνίας και των υπολογισμών ακόμα και για μεγάλα μηνύματα.
- Αξιόπιστη ταυτοποίηση σειράς μηνυμάτων.

- Αποδοτική υποστήριξη για μη αναμενόμενα μηνύματα.
- Αποκατάσταση λαθών στο δίκτυο και υψηλή διαθεσιμότητα.
- Βασικός μηχανισμός ταυτοποίησης για κάθε μήνυμα.
- Για κάθε μήνυμα ή για κάθε άκρο συναρτήσεις που κάνουν δοκιμές (polling) ή σταματούν τη ροή ελέγχου μέχρι την ολοκλήρωσή τους (blocking)
- Υποστήριξη για δρομολόγηση διασποράς.
- Ενσωματωμένη υποστήριξη χαρτογράφησης του Myrinet υλικού.
- Υποστήριξη ακύρωσης των αιτήσεων σε αναμονή.
- Βιβλιοθήκες τόσο ενός νήματος αλλά και ασφαλών νημάτων.

Στη συνέχεια περιγράφονται σημαντικές ενέργειες που εκτελούνται στο πλαίσιο της επικοινωνίας με το MX. Η τυπική σειρά την οποία ακολουθεί μια εφαρμογή που κάνει χρήση του λογισμικού είναι: αρχικοποίηση της βιβλιοθήκης, αρχικοποίηση ενός άκρου (endpoint), σύνδεση με τους κόμβους προορισμού, έναρξη αποστολής και λήψης μηνυμάτων που ακολουθούνται από κλήσεις σε συναρτήσεις που παρακολουθούν την εξέλιξη των αιτημάτων, κλείσιμο του άκρου και απελευθέρωση των δομών της βιβλιοθήκης.

Τα άκρα της επικοινωνίας (Endpoints) Ένα άκρο στο MX είναι ένας εικονικός προσαρμογέας δικτύου στο επίπεδο της διεργασίας, ο οποίος παρέχει πρόσβαση στο υλικό διασύνδεσης. Η πρόσβαση αυτή είναι προστατευμένη από άλλες διεργασίες. Το άκρο είναι επίσης ένα στιγμιότυπο της διεπαφής του λογισμικού. Η αναφορά σε αυτό γίνεται με μια μεταβλητή τύπου $\mathrm{mx\_endpoint\_t}$, και χρησιμοποιείται στις περισσότερες από τις λειτουργίες του MX. Όλες οι λειτουργίες σε ένα αρχικοποιημένο άκρο είναι περιορισμένες μέσα σε αυτό. Τα αντικείμενα του MX, όπως είναι ένας χειριστής μιας αίτησης για αποστολή ή λήψη, σχετίζονται με ένα συγκεκριμένο άκρο και δεν έχουν νόημα έξω από αυτό ακόμα και μέσα στην ίδια διεργασία.

Ένα άκρο μπορεί να δημιουργηθεί από την κλήση της $\mathrm{mx\_open\_endpoint()}$, που επιστρέφει ένα δείκτη στη δομή χειρισμού του, που έχει τύπο $\mathrm{mx\_endpoint\_t}$. Αν η $\mathrm{mx\_open\_endpoint}$ δεν επιστρέψει $\mathrm{MX\_SUCCESS}$, τότε η μεταβλητή τύπου $\mathrm{mx\_endpoint\_t}$ που πέρασε ως παράμετρος παραμένει αμετάβλητη.

Διευθυνσιοδότηση των άκρων Για την επικοινωνία με ένα απομακρυσμένο άκρο μια εφαρμογή πρέπει να έχει τη διεύθυνση του, που αναπαρίσταται από μια μεταβλητή τύπου $\mathrm{mx\_endpoint\_addr\_t}$. Η διεύθυνση αυτή κατασκευάζεται από τρεις πληροφορίες: το αναγνωριστικό του προσαρμογέα δικτύου (NIC ID), το αναγνωριστικό του άκρου και μια τιμή φίλτρου. Μια μεταβλητή αυτού του τύπου δημιουργείται από μια κλήση στη συνάρτηση $\mathrm{mx\_connect()}$ και μπορεί να χρησιμοποιηθεί μόνο από το άκρο που πέρασε ως παράμετρος κατά την κλήση της.

Το αναγνωριστικό του προσαρμογέα είναι μεγέθους 64bit και μπορεί να ανακτηθεί μέσα σε έναν κόμβο με την κλήση της συνάρτησης mx_board_number_to_nic¬ _id(). Το αναγνωριστικό του άκρου είναι ένας ακέραιος που υπάρχει για κάθε αρχικοποιημένο άκρο. Μπορεί είτε να οριστεί από την εφαρμογή που το αρχικοποιεί, είτε από τη βιβλιοθήκη του MX. Αυτή είναι μια τιμή ανάμεσα από το 0 και το MX_¬ MAX_ENDPOINTS - 1. Το φίλτρο είναι ένας ακέραιος, που παίρνει τιμή από την εφαρμογή, για να διαχωρίσει ξεχωριστές λήψης της ίδιας εφαρμογής.

Η αντιστοίχιση. Η αντιστοίχιση αφορά στη διαδικασία συσχετισμού ενός εισερχόμενου μηνύματος με μια εκκρεμούσα λήψη. Κάθε διεπαφή διαβίβασης μηνυμάτων ορίζει τους δικούς της κανόνες με βάση στοιχεία που παρέχονται από το άκρο αποστολής και/ή το άκρο λήψης. Ο ισχυρός μηχανισμός αντιστοίχισης είναι απαραίτητος για την εγκατάσταση μιας σύνθετης διεπαφής διαβίβασης μηνυμάτων πάνω από μια χαμηλού επιπέδου διεπαφή ή για την κατευθείαν υλοποίηση εφαρμογών πάνω από αυτή.

Το MX παρέχει μια ευέλικτη αλλά και πολύ ισχυρή διεπαφή αντιστοίχισης. Κάθε μήνυμα στο MX περιέχει 64bits πληροφορίας που χρησιμοποιείται στην αντιστοίχιση. Ο αποστολέας καθορίζει την πληροφορία αυτή, match_send ως μέρος της διαδικασίας αποστολής και ο παραλήπτης παρέχει το match_recv και μια μάσκα match_mask όταν κάνει μια αίτηση λήψης. Ένα εισερχόμενο μήνυμα θα συσχετιστεί με μια εκκρεμούσα λήψη όταν και μόνο όταν, το εισερχόμενο match_send με τη μάσκα match¬ _mask ταιριάζει με την πληροφορία match_recv στην πλευρά της λήψης.

Αιτήσεις Οι αιτήσεις είναι μεταβλητές – χειριστές που ξεχωρίζουν συγκεκριμένα στιγμιότυπα από το σύνολο των ασύγχρονων λειτουργιών που εκκρεμούν. Για όλες τις ασύγχρονες λειτουργίες στο MX, πρέπει να δημιουργηθεί ένα αντικείμενο mx_¬ request_t. Το αντικείμενο αυτό χρησιμοποιείται για να καθορίσει τη λειτουργία που εκκρεμεί σε μια οποιαδήποτε επόμενη λειτουργία. Χειριστές αιτήσεων επιστρέφονται από τις συναρτήσεις mx_isend(), mx_issend() και irecv() και μπορούν να περαστούν ως παράμετροι στις συναρτήσεις mx_test(), mx_wait(), mx_ibuffered() και mx_cancel(). Αν κάποια από τις ασύγχρονες συναρτήσεις δεν επιστρέψει MX¬ _SUCCESS τότε ο χειριστής της άιτησης, τύπου mx_request_t παραμένει αμετάβλητος.

Κάθε αίτηση για λήψη πρέπει να έχει μια επιτυχή αντιστοίχιση σε μια κλήση της mx¬ _test() ή της mx_wait(). Με τον τρόπο αυτό μπορεί να ελευθερώσει και να ανακυκλώσει τους πόρους που σχετίζονταν με την αίτηση, όταν αυτό καταστεί εφικτό.

2.1.2 InfiniBand

Εισαγωγή

Το InfiniBand[5] ορίζει ένα πρότυπο αρχιτεκτονικής Ε/Ε (InfiniBand Architecture – IBA) που χρησιμοποιείται για να διασυνδεθούν εξυπηρετητές, εξοπλισμός υποδομής

επικοινωνιών, συσκευές αποθήκευσης και ενσωματωμένα συστήματα. Ένα σύστημα αρχιτεκτονικής InfiniBand μπορεί να είναι ένας μικρός εξυπηρετητής με έναν επεξεργαστή και μερικές συσκευές Ε/Ε, μέχρι και ένα σύστημα MPP με εκατοντάδες επεξεργαστές και χιλιάδες συσκευές Ε/Ε. Η διασύνδεση στο InfiniBand πραγματοποιείται μέσω μιας υποδομής επικοινωνίας με μεταγωγείς που προσφέρουν μεταφορά δεδομένων με ταχύτητες μέχρι και 120Gbps. Έχει εφαρμογές τόσο μέσα στα πλαίσια ενός κόμβου, όσο και έξω από αυτόν, μέσω εξωτερικών συνδέσεων από χαλκό ή οπτικές ίνες.

Το InfiniBand επιτυγχάνει χαμηλό χρόνο πρώτης απόκρισης, απαιτεί μικρή επιβάρυνση επεξεργασίας και έχει σχεδιαστεί για τη μεταφορά διαφορετικών τύπων κίνησης πάνω από μια μόνο σύνδεση. Πάνω από την ίδια υποδομή διακινούνται με τρόπο αποδοτικό τόσο δεδομένα που αφορούν στη επικοινωνία επεξεργαστών, όσο και αυτά που αφορούν σε συστήματα αποθήκευσης. Το InfiniBand χρησιμοποιείται σε κέντρα δεδομένων (data centers), συστοιχίες υπολογιστών υψηλής απόδοσης (HPC clusters) και ενσωματωμένα συστήματα προσδίδοντας τους δυνατότητες κλιμάκωσης, από δύο μέχρι εκατοντάδες κόμβους.

Η αρχιτεκτονική του InfiniBand προέρχεται από τη συνένωση δυο προτύπων Ε/Ε, το Future I/O (Compaq, IBM και Hewllet – Packard) και το Next Generation I/O (Intel, Microsoft και Sun). Την ανάπτυξη του έχει αναλάβει το InfiniBand Trade Association (IBTA). Η ραγδαία ανάπτυξη των συστοιχιών υπολογιστών και η κοινή προσέγγιση του από τις εταιρίες, καθιστούν το InfiniBand, ένα ανοιχτό και υλοποιημένο από πολλές εταιρίες πρότυπο. Για τη λειτουργία του διατίθεται υποδομή σε ταχύτητες των 10Gbps (SDR – Single Data Rate), των 20Gbps (DDR – Double Data Rate) ενώ το 2008 παρουσιάστηκε από εταιρίες εξοπλισμός για ταχύτητες μέχρι και 40Gbps (QDR – Quad Data Rate).

Έχει σχεδιαστεί από την αρχή για να προσφέρει Αξιοπιστία, Διαθεσιμότητα, και Λειτουργικότητα (Reliability, Availability, Serviceability – RAS). Η αρχιτεκτονική υποστηρίζει το πρωτόκολλο IP για τη σύνδεση με το Διαδίκτυο και τους μακρινούς εξυπηρετητές. Επιπλέον καθορίζει ένα υλικό επικοινωνίας με υποστήριξη μεταγώγεων με υψηλό εύρος ζώνης και χαμηλό χρόνο πρώτης απόκρισης. Το μεγαλύτερο μέρος του υπολογιστικού φόρτου για την επικοινωνία μεταβιβάζεται στο υλικό του InfiniBand.

Τα χαρακτηριστικά της αρχιτεκτονικής InfiniBand

Καλή απόδοση. Το InfiniBand παρέχει συγκριτικά μικρούς χρόνους πρώτης απόκρισης (της τάξης των 5usec) σε σχέση με τα πιο διαδεδομένα δίκτυα διασύνδεσης. Η υποδομή που το υλοποιεί έχει εύρος ζώνης μέχρι και 120Gbps.

Μειωμένη πολυπλοκότητα. Το InfiniBand επιτρέπει τη συγχώνευση πολλαπλών λειτουργιών Ε/Ε σε ένα ενιαίο καλώδιο ή μια εσωτερική διασύνδεση, χαρακτηριστικό κρίσιμο για τους υπολογιστές Blade, τα κέντρα δεδομένων, τα συστήματα αποθήκευσης και τα ενσωματωμένα συστήματα. Το InfiniBand συγχωνεύει τη διαβίβαση σημάτων που αφορούν στις συστοιχίες συστημάτων, στις

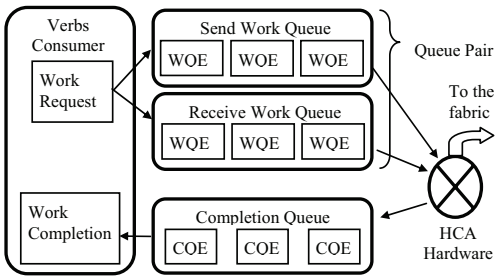
επικοινωνίες, στα συστήματα αποθήκευσης και στα συστήματα διαχείρισης πάνω από μια μόνο σύνδεση. Η συγχώνευση αυτή των Ε/Ε σε ένα ενιαίο υλικό του InfiniBand μειώνει σημαντικά τις ανάγκες σε χώρο, κόστος αλλά και σε ηλεκτρική ενέργεια, στοιχεία πολύ κρίσιμα για τις συστοιχίες εξυπηρετητών. Οι υπόλοιπες τεχνολογίες διασύνδεσης ταιριάζουν λιγότερο σε ενιαία υλικά, γιατί ο αρχικός τους σχεδιασμός δεν έχει γίνει για να υποστηρίζει τη μεταφορά δεδομένων διαφορετικού τύπου.

Υψηλή απόδοση διασύνδεσης. Το InfiniBand αναπτύχθηκε για να παρέχει καλή κλιμάκωση. Για τις ανάγκες της επεξεργασίας κατά την επικοινωνία, παρέχει το κατάλληλο υλικό – αποφορτίζοντας το CPU από το ρόλο αυτό – και επιτρέπει την πλήρη χρησιμοποίηση των πόρων κάθε κόμβου που προστίθεται στη συστοιχία. Επιπλέον υποστηρίζει την Απομακρυσμένη Απευθείας Πρόσβαση στη Μνήμη (RDMA) που αποτελεί ένα βελτιστοποιημένο πρωτόκολλο μεταφοράς δεδομένων και αφήνει τον εξυπηρετητή να επικεντρωθεί στην επεξεργασία των προγραμμάτων που τρέχουν. Το RDMA συμβάλει στην καλή απόδοση τόσο σε εφαρμογές για συστοιχίες εξυπηρετητών, όσο και συστημάτων αποθήκευσης.

Αξιόπιστες και σταθερές συνδέσεις. Το InfiniBand παρέχει αξιόπιστα δίπλα στοιχεία σύνδεσης. Αυτή η ικανότητα υλοποιείται στο υλικό. Επιπλέον, διευκολύνει εικονικές (virtualization) λύσεις διασύνδεσης, οι οποίες επιτρέπουν πολλαπλές εφαρμογές να χρησιμοποιούν απομονωμένα το ίδιο δίκτυο. Κατά συνέπεια, πολλαπλές εφαρμογές τρέχουν ταυτόχρονα σε σταθερές συνδέσεις, ελαχιστοποιώντας το χρόνο που δε λειτουργούν. Τα υλικά κατασκευάζονται με λογικές πλεονασμού σχεδόν σε όλα τα επίπεδα. Στόχος είναι αν μια σύνδεση εμφανίσει κάποια βλάβη, όχι μόνο το σφάλμα να περιορίζεται στη μια αυτή σύνδεση, αλλά και μια πρόσθετη σύνδεση να μπορεί αυτόματα να εξασφαλίσει τη σύνδεση σε όλο το υλικό. Ο πλεονασμός μέσα στο υλικό έχει ως αποτέλεσμα τελικά την υψηλή αξιοπιστία του.

Πολλές λειτουργίες του InfiniBand βασίζονται σε ένα μηχανισμό από δύο ουρές με εντολές προς εκτέλεση από το υλικό. Όπως φαίνεται και στο σχήμα 2.6, υπάρχει μια ουρά για διαδικασίες αποστολής, μια ουρά για διαδικασίες παραλαβής καθώς και μια ουρά ολοκληρωμένων αιτημάτων. Η ουρά αποστολής κρατά εντολές οι οποίες καθορίζουν πώς θα μεταφερθούν τα δεδομένα από τη μνήμη του αποστολέα προς τη μνήμη του παραλήπτη. Η ουρά λήψης κρατά εντολές οι οποίες καθορίζουν πού θα αποθηκευθούν δεδομένα που έχουν παραληφθεί. Όταν μια αίτηση κατατεθεί, η εντολή της τοποθετείται στην κατάλληλη ουρά. Έπειτα ο προσαρμογέας του καναλιού εκτελεί την εντολή όταν έρθει η σειρά της. Μετά το τέλος της εκτέλεσης την τοποθετεί στην ουρά των ολοκληρωμένων αιτημάτων.

Το InfiniBand μπορεί να παρέχει υπηρεσίες τόσο συνδεσιστρεφείς (connection – oriented) όσο και δεδομενογραμμάτων (datagram ή connectionless). Στις συνδεσιστρεφείς υπηρεσίες, υπάρχει μια συσχέτιση μεταξύ ενός προορισμού (ζεύγη ουρών αποστολής και λήψης) στην πλευρά του αποστολέα και του παραλήπτη. Αυτή η συσχέτιση



Σχήμα 2.6: Ο μηχανισμός ουρών στο InfiniBand[1].

είναι απότοκο της αρχιτεκτονικής διαύλου, στην οποία όλη η επικοινωνία μοιράζεται ένα κοινό κανάλι.

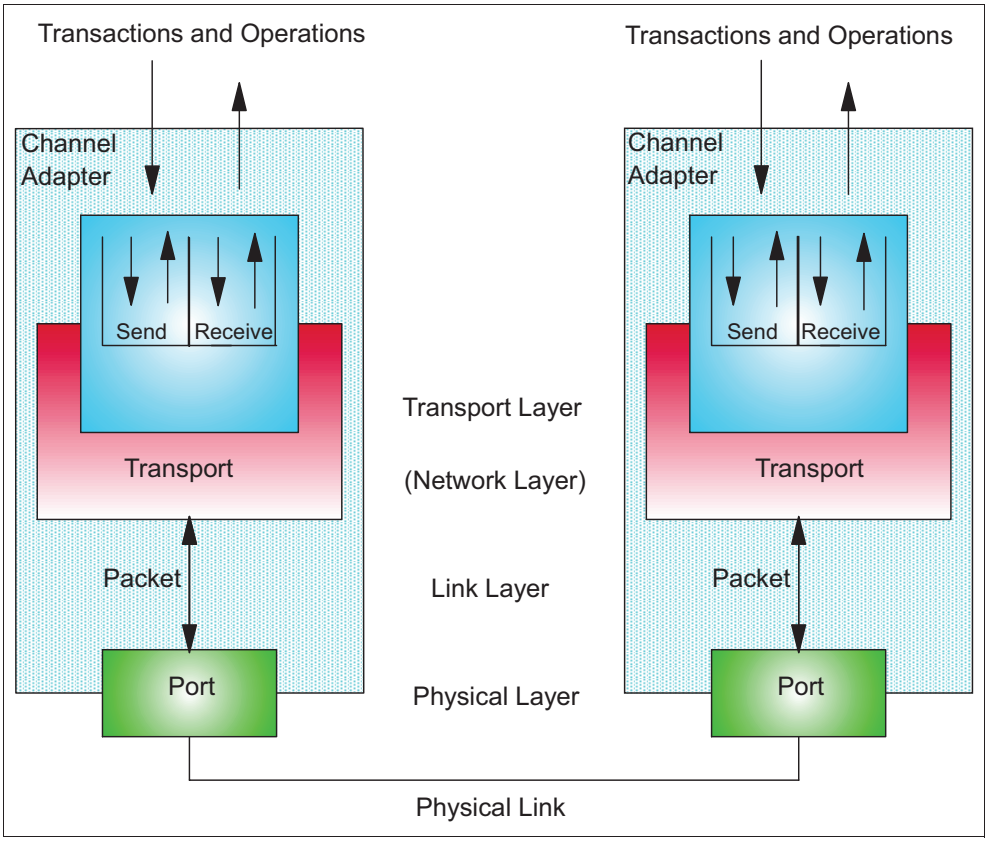
Στη διαδικασία μιας αποστολής, ο προσαρμογέας του καναλιού διερμηνεύει τον τύπο της ενέργειας, δημιουργεί ένα μήνυμα, το τεμαχίζει (αν χρειάζεται) σε πολλαπλά πακέτα, προσθέτει την πληροφορία δρομολόγησης και στέλνει τα πακέτα σε μία πόρτα. Το λογικό κύκλωμα της πόρτας είναι πλέον υπεύθυνο για την αποστολή του πακέτου στο άλλο άκρο. Όταν φθάσουν τα πακέτα στο κύκλωμα της πόρτας του προορισμού, αυτό πιστοποιεί το πακέτο, ο προσαρμογέας του καναλιού το τοποθετεί στην ουρά και έπειτα εκτελεί την αντίστοιχη διαδικασία. Αν έχει ζητηθεί θα δημιουργηθεί και ένα σήμα επιβεβαίωσης το οποίο και θα στείλει πίσω στον αποστολέα του πακέτου.

Τα Επίπεδα του InfiniBand

Η αρχιτεκτονική του InfiniBand[6] χωρίζεται σε πολλαπλά επίπεδα καθένα από τα οποία λειτουργεί ανεξάρτητα. Όπως φαίνεται και στο σχήμα 2.8 το InfiniBand χωρίζεται στα ακόλουθα επίπεδα: Φυσικό, Ζεύξης, Δικτύου, Μεταφοράς και Ανώτερων Επιπέδων.

Το Φυσικό Επίπεδο Το φυσικό επίπεδο καθορίζει τον τρόπο με τον οποίο τα bits αποστέλλονται στο καλώδιο για να μορφοποιούν σύμβολα, οριοθετήσεις και αναμονές. Η αρχιτεκτονική του InfiniBand καθορίζει ηλεκτρικές, οπτικές και μηχανικές απαιτήσεις για αυτό το επίπεδο. Στο πρότυπο περιλαμβάνονται τα καλώδια και βύσματα για τα χάλκινα και οπτικά μέσα, οι υποδοχές για τις συνδέσεις μέσα σε έναν υπολογιστή και οι προδιαγραφές λειτουργίας τους, συμπεριλαμβανομένων και ειδικών περιπτώσεων όπως η εναλλαγή κατά τη λειτουργία (hot-swapped).

Φυσικές Γραμμές Το InfiniBand είναι μία αρχιτεκτονική διασύνδεσης με συνδέσεις σημείο–προς–σημείο, που αναπτύχθηκε με στόχο το μεγάλο εύρος ζώνης και την ικανότητα να κλιμακώνει ανάλογα με τις απαιτήσεις. Κάθε σύνδεση βασίζεται σε μια σύνδεση διπλής αμφίδρομης οπτικής ίνας για την υλοποίηση με οπτικά υλικά και ένα τετραπλό αμφίδρομο καλώδιο για την υλοποίηση με χαλκό, η οποία ονομάζεται φυσική γραμμή. Κάθε γραμμή υποστηρίζει πολλαπλές υπηρεσίες μεταφοράς με

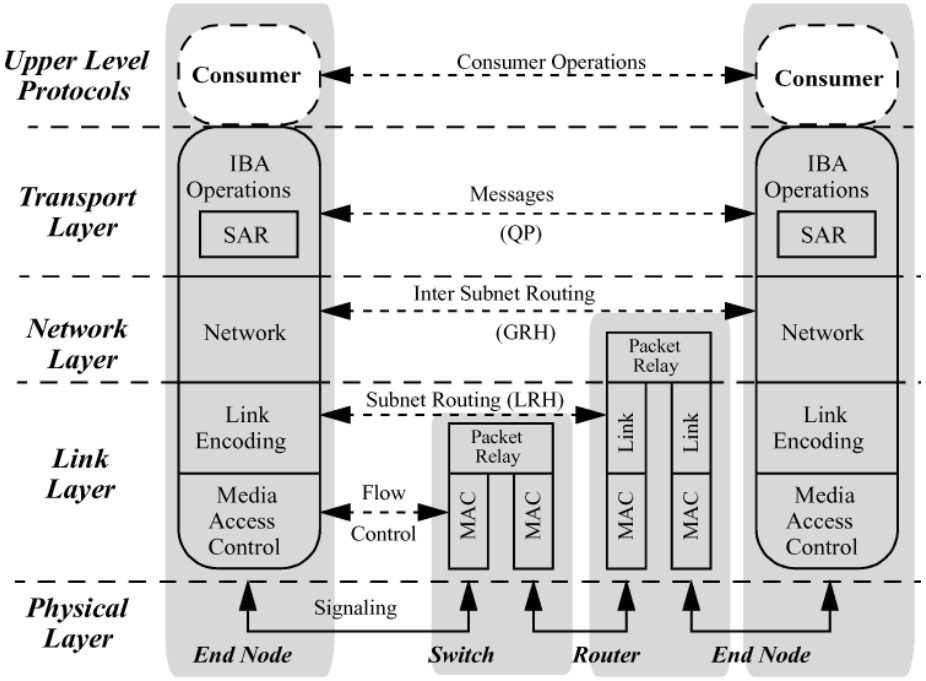


Σχήμα 2.7: Η στοίβα των πρωτοκόλλων του InfiniBand.

αξιοπιστία και πολλαπλά εικονικά κανάλια επικοινωνίας με προτεραιότητες. Οι φυσικές γραμμές ομαδοποιούνται για να υποστηρίξουν τα πρότυπα των εκδόσεων 1X, 4X, 8X και 12X. Τα άκρα στο InfiniBand διαπραγματεύονται κατά τη λειτουργία τους για τη χρήση:

- Απλού Ρυθμού Δεδομένων (Single Data Rate – SDR), που επιτρέπει ταχύτητες μέχρι 2.5Gb/s για κάθε φυσική γραμμή
- Διπλού Ρυθμού Δεδομένων (Double Data Rate – DDR), που επιτρέπει ταχύτητες μέχρι 5.0Gb/s για κάθε φυσική γραμμή
- Τετραπλού Ρυθμού Δεδομένων (Quadruple Data Rate – QDR), που επιτρέπει ταχύτητες μέχρι 10Gb/s για κάθε φυσική γραμμή

Η διαπραγμάτευση της ταχύτητας της σύνδεσης καθορίζει και την ταχύτητα των προσαρμογέων στα δύο άκρα. Τελικά η επιλογή αυτή θα καθορίσει και τη μέγιστη ταχύτητα που μπορεί να επιτευχθεί, με βάση τις ικανότητες του κάθε άκρου και την ποιότητα του σήματος διασύνδεσης.



Σχήμα 2.8: Τα στρώματα του InfiniBand.

Το Επίπεδο Ζεύξης Το επίπεδο ζεύξης (μαζί με το επίπεδο μεταφοράς) είναι από τα πιο σημαντικά στην αρχιτεκτονική του InfiniBand. Το επίπεδο αυτό περιλαμβάνει τη μορφή των πακέτων, τις σημείο–προς–σημείο διαδικασίες και τη μεταγωγή σε ένα τοπικό υποδίκτυο.

Πακέτα Υπάρχουν δύο τύποι πακέτων στο επίπεδο ζεύξης, τα διαχειριστικά και τα πακέτα δεδομένων. Τα διαχειριστικά πακέτα χρησιμοποιούνται για την πραγματοποίηση ρυθμίσεων και ενεργειών συντήρησης. Πληροφορίες για τη συσκευή, όπως η Εικονική Γραμμή (Virtual Line) καθορίζονται με διαχειριστικά πακέτα. Τα πακέτα δεδομένων μπορούν να μεταφέρουν μέχρι και 4KBytes χρήσιμων δεδομένων σε μια ανταλλαγή.

Μεταγωγή Μέσα σε ένα υποδίκτυο, η προώθηση των πακέτων και η μεταγωγή γίνεται από το επίπεδο ζεύξης. Όλες οι συσκευές σε ένα υποδίκτυο έχουν μια Τοπική Ταυτότητα (Local ID – LID) μεγέθους 16bit που δίδεται από το Διαχειριστή του Υποδικτύου (Subnet Manager). Όλα τα πακέτα που αποστέλλονται μέσα σε ένα υποδίκτυο χρησιμοποιούν το LID για τον καθορισμό των διευθύνσεων πηγής και προορισμού. Στο επίπεδο ζεύξης κατά τη μεταγωγή τα πακέτα προωθούνται στη συσκευή που καθορίζεται από το LID το οποίο βρίσκεται μέσα στην Κεφαλίδα Τοπικής Δρομολόγησης (Local Route Header – LRH) του πακέτου. Η LRH υπάρχει πάντα σε όλα τα πακέτα.

Ποιότητα της Υπηρεσίας Η Ποιότητα της Υπηρεσίας (Qos) υποστηρίζεται μέσω των

εικονικών γραμμών. Αυτές είναι χωριστές λογικές ζεύξεις επικοινωνίας που μοιράζονται μια κοινή φυσική ζεύξη. Κάθε ζεύξη μπορεί να υποστηρίξει μέχρι δεκαπέντε εικονικές γραμμές δεδομένων και μια διαχειριστική (VL15). Η τελευταία, VL15, έχει την υψηλότερη προτεραιότητα και η VL0 έχει τη χαμηλότερη. Τα διαχειριστικά πακέτα χρησιμοποιούν αποκλειστικά την VL15. Κάθε συσκευή πρέπει να υποστηρίζει τουλάχιστον τις VL0 και VL15, ενώ οι υπόλοιπες είναι προαιρετικές. Όταν ένα πακέτο μεταφέρεται στο υποδίκτυο, καθορίζεται ένα Επίπεδο Υπηρεσίας (Service Level - SL) το οποίο και εξασφαλίζει την Ποιότητα της Υπηρεσίας. Κάθε ζεύξη μέσα σε ένα μονοπάτι μπορεί να έχει ένα διαφορετικό VL, και το SL παρέχει σε κάθε ζεύξη την επιθυμητή προτεραιότητα. Κάθε μεταγωγέας/δρομολογητής έχει έναν πίνακα αντιστοιχίσεων από SL σε VL, ο οποίος είναι ευθύνη του διαχειριστή του υποδικτύου και κρατά τις προτεραιότητες και τον αριθμό των VLs που υποστηρίζει η κάθε ζεύξη. Με τον τρόπο αυτό η αρχιτεκτονική του InfiniBand μπορεί να εξασφαλίσει Ποιότητα της Υπηρεσίας από άκρο σε άκρο μέσα από μεταγωγείς, δρομολογητές και πάνω από το συνολικό σύστημα.

Έλεγχος Ροής βασισμένος σε Υποδείξεις Ο έλεγχος της ροής χρησιμοποιείται για τη διαχείριση της ροής δεδομένων μεταξύ δύο σημείο–προς–σημείο συνδέσμων. Ο έλεγχος της ροής γίνεται για κάθε VL ξεχωριστά, επιτρέποντας σε ξεχωριστά εικονικά υλικά να διατηρούν την επικοινωνία στο ίδιο φυσικό μέσο. Κάθε άκρο, παραλήπτης σε μια ζεύξη, παρέχει υποδείξεις στη συσκευή αποστολής για να προσδιορίσει πόσα δεδομένα μπορούν να παραλειφθούν χωρίς να χαθεί πληροφορία. Μια ξεχωριστή ζεύξη διαχειρίζεται τη διέλευση αυτών των υποδείξεων μεταξύ των συσκευών έτσι ώστε ο παραλήπτης να ενημερώνει για τον αριθμό των πακέτων που μπορεί να δεχτεί. Τα δεδομένα δεν εκπέμπονται αν ο παραλήπτης δεν υποδείξει ότι έχει διαθέσιμο ενδιάμεσο χώρο για την παραλαβή δεδομένων.

Ακεραιότητα των Δεδομένων Στο επίπεδο ζεύξης υπάρχουν δύο έλεγχοι CRC για κάθε πακέτο, το Μεταβλητό CRC (Variant CRC – VCRC) και το Αμετάβλητο CRC (Invariant CRC – ICRC) που εξασφαλίζουν την ακεραιότητα των δεδομένων. Το αποτέλεσμα του VCRC έχει μέγεθος 16bit, περιλαμβάνει όλα τα πεδία στα πακέτα και υπολογίζεται ξανά σε κάθε κόμβο. Το αποτέλεσμα του ICRC έχει μέγεθος 32bit και καλύπτει μόνο τα πεδία τα οποία δεν αλλάζουν από κόμβο σε κόμβο. Ο VCRC εξασφαλίζει την ακεραιότητα των δεδομένων ανάμεσα σε δύο κόμβους ενώ το ICRC από άκρο σε άκρο. Σε ένα πρωτόκολλο, όπως είναι το Ethernet, που ορίζει μόνο ένα CRC, μπορεί να γίνει ένα λάθος σε μια συσκευή η οποία έπειτα υπολογίζει πάλι το CRC. Ο έλεγχος CRC στον επόμενο κόμβο υπολογίζεται ως έγκυρος αν και τα δεδομένα είναι λάθος. Το InfiniBand συμπεριλαμβάνει το ICRC και η πληροφορία για τέτοιου είδους σφάλμα φτάνουν μέχρι τον προορισμό.

Το Επίπεδο Δικτύου Το επίπεδο δικτύου αναλαμβάνει τη δρομολόγηση των πακέτων από ένα υποδίκτυο σε ένα άλλο. Πακέτα τα οποία αποστέλλονται μεταξύ υπο-

δικτύων περιλαμβάνουν τη Καθολική Κεφαλίδα Δρομολόγησης (Global Route Header – GRH). Η GHR περιλαμβάνει τη διεύθυνση IPv6 μεγέθους 128bit για τον προορισμό και την προέλευση του πακέτου. Τα πακέτα προωθούνται μεταξύ των υποδικτύων μέσα από δρομολογητές με βάση το Καθολικά Μοναδικό Αναγνωριστικό (Globally Unique ID – GUID) κάθε συσκευής που έχει μέγεθος 64bit. Ο δρομολογητής μεταβάλει την LHR με τη σωστή τοπική διεύθυνση μέσα σε κάθε υποδίκτυο. Με τον τρόπο αυτό ο τελευταίος δρομολογητής στο μονοπάτι αντικαθιστά την LID στην LHR με την LID της πόρτας του προορισμού. Το επίπεδο δικτύου δεν είναι απαραίτητο, στα πακέτα του InfiniBand που μετακινούνται μέσα στα όρια ενός μόνο υποδικτύου (σενάριο πολύ πιθανό αφού το InfiniBand πολύ συχνά χρησιμοποιείται σε Δίκτυα Περιοχής Συστήματος)

Το Επίπεδο Μεταφοράς Το επίπεδο μεταφοράς είναι υπεύθυνο για την παραλαβή των πακέτων με τη σειρά, το διαχωρισμό τους, την πολυπλεξία τους στο κανάλι καθώς και για υπηρεσίες που σχετίζονται με τη μεταφορά (αξιόπιστη σύνδεση, αξιόπιστα δεδομενογράμματα, μη αξιόπιστη σύνδεση, μη αξιόπιστα δεδομενογράμματα και ακατέργαστα δεδομενογράμματα). Το επίπεδο μεταφοράς αναλαμβάνει τον τεμαχισμό των δεδομένων κατά την αποστολή και την ανασυγκρότηση τους κατά την παραλαβή. Με βάση τη Μονάδα Μέγιστης Μεταφοράς (Maximum Transfer Unit – MTU) του μονοπατιού, το επίπεδο μεταφοράς χωρίζει τα δεδομένα σε πακέτα κατάλληλου μεγέθους. Ο παραλήπτης ανασυγκροτεί τα πακέτα με βάση την Κεφαλίδα Βασικής Μεταφοράς (Base Transport Header – BTH) που περιέχει το ζευγάρι ουράς του προορισμού και τον αύξοντα αριθμό του πακέτου. Ο παραλήπτης επιβεβαιώνει τη λήψη των πακέτων και ο αποστολέας που λαμβάνει αυτές τις επιβεβαιώσεις, ενημερώνει την ουρά ολοκλήρωσης με την τρέχουσα κατάσταση της διαδικασίας. Η αρχιτεκτονική του InfiniBand προσφέρει μια σημαντική βελτίωση για το επίπεδο μεταφοράς: όλες οι λειτουργίες είναι υλοποιημένες στο υλικό.

Το InfiniBand καθορίζει πολλαπλές υπηρεσίες μεταφοράς για την αξιοπιστία των δεδομένων. Ο πίνακας 2.1 περιγράφει κάθε μια από αυτές. Για κάθε ζευγάρι ουράς χρησιμοποιείται ένα επίπεδο μεταφοράς.

Πίνακας 2.1: Οι Υπηρεσίες Υποστήριξης.

Τύπος Υπηρεσίας	Περιγραφή
Αξιόπιστη Σύνδεση	επιβεβαιωμένα – συνδεσιστρεφή
Αξιόπιστα Δεδομενογράμματα	επιβεβαιωμένα – πολυπλεγμένα
Μη Αξιόπιστη Σύνδεση	μη επιβεβαιωμένα – συνδεσιστρεφή
Μη Αξιόπιστα Δεδομενογράμματα	μη επιβεβαιωμένα – ασυνδεσιστρεφή
Ακατέργαστα Δεδομενογράμματα	μη επιβεβαιωμένα – ασυνδεσιστρεφή

2.1.3 Ethernet

Εισαγωγή

Το Ethernet είναι μια οικογένεια τεχνολογιών δικτύωσης Υπολογιστών για Τοπικά Δίκτυα (LANs) βασισμένη σε πακέτα. Το όνομα του προέρχεται από τις φυσικές ιδιότητες του αιθέρα (από την αγγλική λέξη “ether” ή “aether”). Μαζί με το Ethernet ορίζεται και ένας αριθμός από πρότυπα καλωδίωσης και σημάτων για το Φυσικό Επίπεδο (Physical Layer) του δικτυακού μοντέλου OSI, μέσα από τη δικτυακή πρόσβαση στο Επίπεδο Συνδέσμου Μετάδοσης Δεδομένων (Data Link Layer) καθώς και μια κοινή μορφή διευθυνσιοδότησης.

Το Ethernet έγινε το πρότυπο IEEE 802.3. Οι εκδόσεις του, που κάνουν χρήση καλωδίων συνεστραμμένων ζευγών (twisted pair) για τη σύνδεση συστημάτων στο δίκτυο και αυτές που κάνουν χρήση διαφόρων τύπων οπτικών για το δίκτυο κορμού, είναι οι πιο διαδεδομένες τεχνολογίες δικτύωσης για τοπικά δίκτυα. Χρησιμοποιείται από το 1980 έως σήμερα και έχει επικρατήσει ανάμεσα σε άλλα σημαντικά ανταγωνιστικά πρότυπα τοπικών δικτύων όπως το Token Ring, το FDDI και το ARCNET.

Η πλήρης κατανόηση της λειτουργίας του Ethernet συνιστά απαραίτητη προϋπόθεση για τη βέλτιστη σχεδίαση του SLURPoE. Στην ουσία γίνεται μια προσπάθεια να αναγνωριστούν τα πλεονεκτήματα και τα μειονεκτήματα της χρήσης του σε σχέση με τα υπόλοιπα δίκτυα διασύνδεσης. Η αναγνώριση των μειονεκτημάτων άλλωστε είναι απαραίτητη προϋπόθεση έτσι ώστε, όπου αυτό είναι δυνατό, κατα το σχεδιασμό του SLURPoE να επιλυθούν.

Γενική Περιγραφή

Το Ethernet αρχικά βασίστηκε στην ιδέα της επικοινωνίας κόμβων πάνω από ένα μοιραζόμενο ομοαξονικό καλώδιο που λειτουργούσε ως μέσο μεταφοράς εκπεμπόμενων μεταδόσεων. Οι μέθοδοι που χρησιμοποιούνταν εμφανίζουν ομοιότητες με τα ραδιοσυστήματα, αν και υπάρχουν θεμελιώδεις διαφορές όπως το γεγονός ότι είναι πολύ πιο εύκολο να ανιχνευθούν συγκρούσεις πακέτων σε μια εκπομπή στο καλώδιο από ότι σε μια ραδιοεκπομπή. Το κοινό καλώδιο που παρέχει το κανάλι επικοινωνίας παρομοιάζεται με τον αιθέρα.

Από αυτή την πρώιμη και συγκριτικά απλή ιδέα εξελίχθηκε το Ethernet σε μια αρκετά πολύπλοκη τεχνολογία δικτύωσης που σήμερα αποτελεί βάση για τα περισσότερα τοπικά δίκτυα. Το ομοαξονικό καλώδιο αντικαταστάθηκε από σημείο–προς–σημείο συνδέσεις, οι οποίες κατέληγαν σε διανομείς (hubs) και/ή μεταγωγείς (switches) για να μειωθεί το κόστος εγκατάστασης, να αυξηθεί η αξιοπιστία, και να επιτραπεί η καλύτερη διαχείριση και αποσφαλμάτωση του δικτύου. Το StarLan ήταν το πρώτο βήμα στην εξέλιξη του Ethernet από τον ομοαξονικό δίαυλο σε ένα δίκτυο ελεγχόμενο από διανομείς, με καλώδια συνεστραμμένων ζευγών. Η έλευση του καλωδίου συνεστραμμένων ζευγών έριξε δραματικά το κόστος της εγκατάστασης σε σύγκριση με αυτό ανταγωνιστικών τεχνολογιών συμπεριλαμβανομένων και παλαιότερων εκδόσεων του ίδιου του Ethernet.

Πάνω από το φυσικό επίπεδο, οι κόμβοι του Ethernet επικοινωνούν με την ανταλλαγή πακέτων δεδομένων. Όπως και με τα υπόλοιπα IEEE 802 τοπικά δίκτυα, ο κάθε κόμβος του Ethernet έχει μια διεύθυνση MAC των 48bit, που χρησιμοποιείται για να καθοριστεί τόσο ο προορισμός όσο και η πηγή σε κάθε πακέτο δεδομένων. Οι προσαρμογείς και τα ολοκληρωμένα κυκλώματα δικτύου γενικά δεν αποδέχονται πακέτα τα οποία απευθύνονται προς άλλους κόμβους Ethernet. Οι προσαρμογείς προγραμματίζονται με μια καθολικά μοναδική διεύθυνση κατά την κατασκευή τους, οι περισσότεροι από αυτούς ωστόσο μπορούν να υποστηρίξουν τη διαχείριση της διεύθυνσης από το χρήστη.

Παρά τις σημαντικές αλλαγές στο Ethernet, από ένα χοντρό ομοαξονικό καλώδιο, δίαυλο που προσφέρει ταχύτητες μέχρι 10Mbit/s σε σημείο–προς–σημείο συνδέσεις που προσφέρουν ταχύτητες 10Gbit/s και ακόμα μεγαλύτερες, όλες οι εκδόσεις (εκτός από τις πρώτες πειραματικές) μοιράζονται την ίδια διαμόρφωση πακέτου και μπορούν εύκολα να διασυνδεθούν.

Η μορφή του πλαισίου Ethernet Το πρότυπο 802.3 ορίζει μια βασική δομή του πλαισίου που απαιτείται από όλες τις υλοποιήσεις MAC, με αρκετές επιπρόσθετες και προαιρετικές μορφές που χρησιμοποιούνται για να επεκτείνουν τις βασικές δυνατότητες του πρωτοκόλλου. Στη βασική αυτή μορφή περιλαμβάνονται επτά πεδία:

Προοίμιο (Preamble – Pre)
: Αποτελείται από 7bytes. Το Pre είναι μια εναλλασσόμενη αλληλουχία από 1 και 0 που ενημερώνουν τους παραλήπτες για ένα εισερχόμενο πακέτο.

Ένδειξη Έναρξης Πακέτου (Start–of–frame delimiter – SOF)
: Αποτελείται από 1byte. Το SOF είναι μια εναλλασσόμενη αλληλουχία από 1 και 0 που τελειώνει σε 1 και καθορίζει ότι επόμενο bit είναι το πρώτο της διεύθυνσης προορισμού.

Διεύθυνση Προορισμού (Destination Address – DA)
: Αποτελείται από 6bytes. Το πεδίο DA υποδεικνύει ποιοι σταθμοί πρέπει να παραλάβουν το πακέτο. Το πρώτο bit στο πεδίο DA καθορίζει αν η διεύθυνση είναι για ένα μοναδικό προορισμό (καθορίζεται με το 0) ή μια ομάδα προορισμών (καθορίζεται με το 1). Το δεύτερο bit καθορίζει αν το DA διαχειρίζεται καθολικά (καθορίζεται με το 0) ή τοπικά (καθορίζεται με το 1). Τα υπόλοιπα 46bits είναι μια μοναδική τιμή που καθορίζει ένα μοναδικό κόμβο ή μια ορισμένη ομάδα από κόμβους ή όλους τους κόμβους του δικτύου.

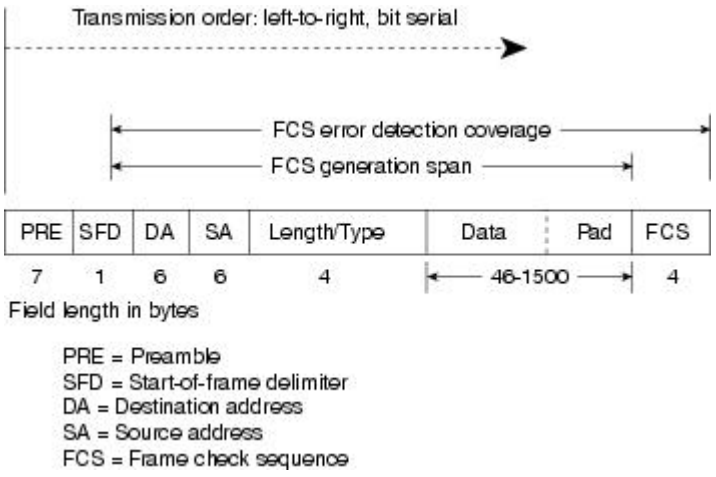
Διεύθυνση Πηγής (Source – SA)
: Αποτελείται από 6bytes. Το πεδίο SA είναι αναγνωριστικό του κόμβου που κάνει την αποστολή. Το SA είναι πάντα διεύθυνση ενός μοναδικού κόμβου και για αυτό το πρώτο του bit είναι πάντα 0.

Μέγεθος/Τύπος (Length/Type)
: Αποτελείται από 2bytes. Αυτό το πεδίο αποτελεί είτε τον αριθμό των bytes που

περιέχονται στο πεδίο των δεδομένων του πακέτου, είτε το αναγνωριστικό του τύπου του πακέτου αν αυτό ήταν φτιαγμένο με βάση μια από τις καθορισμένες προαιρετικές μορφές. Αν το πεδίο Length/Type είναι σε μια τιμή μικρότερη ή ίση με το 1500, ο αριθμός των LLC bytes στο πεδίο Δεδομένων είναι ίσος με το πεδίο Length/Type. Αν το πεδίο Length/Type είναι μεγαλύτερο από 1536 το πακέτο είναι μια προαιρετική μορφή και το πεδίο Length/Type καθορίζει ποια από τις μορφές πακέτου παραλαμβάνεται ή αποστέλλεται.

Δεδομένα (Data)
Είναι μια ακολουθία από n bytes, όπου το n είναι μικρότερο ή ίσο με το 1500. Αν το μέγεθος του πεδίου Δεδομένων είναι μικρότερο από το 46, το πεδίο Δεδομένων πρέπει να επεκταθεί προσθέτοντας κάποια byte ώστε το μήκος του να φτάσει τα 46bytes.

Ακολουθία Ελέγχου Πακέτου (Frame Check Sequence – FCS)
Αποτελείται από 4bytes. Αυτό το πεδίο περιλαμβάνει την τιμή του έλεγχου CRC μεγέθους 32bit, που υπολογίζεται από το Στρώμα Ελέγχου του Μέσου που αποστέλλει και υπολογίζεται ξανά από το αντίστοιχο που παραλαμβάνει. Το FCS δημιουργείται από τα πεδία DA, SA, Length/Type και Data.



Σχήμα 2.9: Η βασική μορφή του πακέτου Ethernet.

Το Ethernet 10G

Το 10Gbit/s Ethernet εγκρίθηκε επίσημα ως μέρος του πρότυπου IEEE 802.3 τον Ιούνιο του 2002. Η τεχνολογία αυτή είναι το επόμενο βήμα στην κλιμάκωση της απόδοσης και της λειτουργικότητας των δικτύων Ethernet, επειδή συνδυάζει το μεγάλο εύρος ζώνης με τη συμβατότητα του Ethernet, με στόχο την επίτευξη κλιμακούμενων, έξυπνων και γρήγορων δικτύων, με συνδέσεις που κυμαίνονται από τα 10Mbps μέχρι

και τα 10Gbps. Αυτή η τεχνολογία είναι πολύ σημαντική όχι μόνο γιατί το Ethernet που πλέον έχει μεγάλο εύρος ζώνης, 10Gbps όχι μόνο γιατί θα εξυπηρετεί τη σύνδεση των τοπικών δικτύων σε μεγάλες ταχύτητες, αλλά πολύ περισσότερο επειδή ευνοεί την περαιτέρω υιοθέτηση του στην αγορά των συστημάτων υψηλής επίδοσης.

Η έλευση του 10GbE φέρνει μαζί της όλα τα πλεονεκτήματα του Ethernet όπως η συμβατότητα και ευκολία στη χρήση, συνδυασμένα με ένα εύρος ζώνης που για τη στιγμή ικανοποιεί ακόμα και την αγορά των συστημάτων υψηλής απόδοσης. Τα επόμενα χρόνια η πτώση της τιμής των υλικών του, αναμένεται να επεκτείνει με μεγαλύτερους ρυθμούς τη διάδοσή του.

Παρά τα πολλά πλεονεκτήματα του 10GbE, που τελικά αναμένεται να είναι ένα φτηνό υψηλής απόδοσης δίκτυο, όπως και οι παλιότερες του εκδόσεις έτσι και αυτό αναδεικνύει το χάσμα ανάμεσα στην υπολογιστική ισχύ που απαιτεί και στην ταχύτητά του. Ένας κανόνας αναφέρει ότι για κάθε 1Mbps δεδομένων στο καλώδιο απαιτούνται από το σύστημα 1MHz επεξεργασίας. Οι ταχύτητες στα 10Gbps θα απαιτούσαν στη λογική αυτή 10GHz επεξεργασίας και για να γίνεται η επικοινωνία προς τις δύο κατευθύνσεις τελικά η απαίτηση ανεβαίνει στα 20GHz. Ακόμα περισσότεροι περιορισμοί ανακύπτουν αν κανείς λάβει υπόψη του τις επιπτώσεις στη χρήση ενός διαύλου PCI-X και στην κεντρική μνήμη σε τέτοιες ταχύτητες.

Η ραγδαία αυτή αύξηση του χάσματος της ταχύτητας μιας συσκευής E/E 10GbE σε σύγκριση με τους πόρους που απαιτούνται για τους σκοπούς της επικοινωνίας από το υπόλοιπο σύστημα, κάνει αναγκαίο τον επαναπροσδιορισμό του τρόπου που οι συσκευές χρησιμοποιούνται σε τέτοιας τάξης ταχύτητες.

2.2 Ο πυρήνας Λ/Σ Linux

Το Linux [7] είναι ένας πυρήνας λειτουργικού συστήματος που μοιάζει με τον πυρήνα του AT&T UNIX. Είναι μία πρωτότυπη υλοποίηση πυρήνα λειτουργικού συστήματος και δε χρησιμοποιεί κώδικα του UNIX. Μπορεί να θεωρηθεί κλώνος του UNIX, αφού διαθέτει τις περισσότερες εντολές του, ενώ η φιλοσοφία της σχεδίασής του πλησιάζει περισσότερο το UNIX από οποιοδήποτε άλλο λειτουργικό σύστημα. Το Linux αναπτύσσεται με βάση το πρότυπο POSIX, το οποίο είναι μία προσπάθεια τυποποίησης όλων των κλώνων του UNIX. Στη συνέχεια η περιγραφή του Linux επικεντρώνεται στα υποσυστήματά του που μελετήθηκαν για το σχεδιασμό και την υλοποίηση της εργασίας.

2.2.1 Οι οδηγοί συσκευών χαρακτήρων

Ο χειρισμός των συσκευών χαρακτήρων (character devices) γίνεται μέσω ονομάτων στο σύστημα αρχείων. Αυτά τα ονόματα καλούνται ειδικά αρχεία ή αρχεία συσκευών και βρίσκονται παραδοσιακά στον κατάλογο /dev.

Οι οδηγοί χαρακτήρων χαρακτηρίζονται από το μείζονα και τον ελάσσονα αριθμό (major, minor number). Οι καινούργιες εκδόσεις του πυρήνα επιτρέπουν περισσότερες από μια συσκευές να μοιράζονται τον ίδιο μείζονα αριθμό, αλλά η συνηθισμένη οργάνωση είναι κάθε οδηγός να έχει το δικό του μείζονα αριθμό.

Ο ελάσσων αριθμός χρησιμοποιείται από τον πυρήνα για να ξεχωρίζει σε ποια συσκευή γίνεται η αναφορά. Συνηθισμένο είναι επίσης ένας οδηγός να κρατά ένα δείκτη στη συσκευή χαρακτήρων που δημιουργεί ή έναν πίνακα συσκευών, όπου ο ελάσσων αριθμός αντιστοιχεί στη θέση της συσκευής μέσα στον πίνακα.

Μέσα στον πυρήνα ορίζεται ο τύπος dev_t που μπορεί να κρατά το μείζονα και τον ελάσσονα αριθμό. Επίσης από τον πυρήνα γίνονται διαθέσιμες και οι συναρτήσεις χειρισμού μιας μεταβλητής αυτού του τύπου. Έτσι από μια μεταβλητή τύπου dev_t μπορούμε να πάρουμε το μείζονα και τον ελάσσονα αριθμό με τις μακροεντολές:

MAJOR(dev_t dev);
MINOR(dev_t dev);

Αντίστοιχα από τους δύο αριθμούς μπορούμε να πάρουμε και την τιμή της μεταβλητής τύπου dev_t:

MKDEV(**int** major, **int** minor);

Για να μπορεί να χρησιμοποιηθεί μια συσκευή χαρακτήρων πρέπει αρχικά να γίνει μια κλήση στη συνάρτηση register_chrdev_region(), που δηλώνεται στο αρχείο κεφαλίδας <linux/fs.h>:

int register_chrdev_region(dev_t first, **unsigned int** count, **char** *name);

Η παράμετρος first είναι το αρχικό ζεύγος από το οποίο θα αρχίσει η δέσμευση των συσκευών χαρακτήρων. Συνήθως ο ελάσσων αριθμός είναι 0, αλλά δεν υπάρχει κάποια απαίτηση για κάτι τέτοιο. Η παράμετρος count είναι ο συνολικός αριθμός των συσκευών που θα χρησιμοποιηθούν. Αν το μέγεθος αυτό είναι μεγάλο, τότε ενδέχεται οι συσκευές που θα δεσμευτούν να εκτείνονται σε συνεχόμενους μείζονες αριθμούς. Τέλος η παράμετρος name είναι το όνομα της συσκευής που σχετίζεται με τους αριθμούς αυτούς.

Τα τελευταία χρόνια η συνάρτηση για τη δέσμευση αριθμών συσκευών χαρακτήρων γίνεται μια προσπάθεια να αντικατασταθεί από τη alloc_chrdev_region(). Με τη συνάρτηση αυτή οι χαρακτηριστικοί αριθμοί μιας συσκευής χαρακτήρων αποδίδονται από τον πυρήνα. Έτσι εξασφαλίζεται μεγαλύτερη φορητότητα στους οδηγούς, αφού είναι στη διαχείριση του πυρήνα να διαθέσει ελεύθερους αριθμούς και όχι στη διορατικότητα του προγραμματιστή.

int alloc_chrdev_region(dev_t *dev, **unsigned int** firstminor, **unsigned int** count, **char** *name);

Στη συνάρτηση αυτή η παράμετρος dev θα πάρει την τιμή που θα αποδώσει ο πυρήνας έπειτα από μια επιτυχή δέσμευση. Η παράμετρος firstminor πρέπει να είναι ο πρώτος

ελάσσων αριθμός που ζητείται και συνήθως έχει την τιμή 0. Τέλος οι παράμετροι count και name έχουν την ίδια χρήση όπως και στη register_chrdev_region().

Ανεξάρτητα από τον τρόπο δέσμευσης η απελευθέρωση των χαρακτηριστικών αριθμών της συσκευής γίνεται με την κλήση της συνάρτησης:

void unregister_chrdev_region(dev_t first, **unsigned int** count);

Οι παραπάνω συναρτήσεις δεσμεύουν αριθμούς για τη χρήση σε έναν οδηγό, αλλά δεν εξασφαλίζουν στον πυρήνα ποιες θα είναι οι λειτουργίες που θα υποστηρίζονται από αυτούς. Πριν μια εφαρμογή στο χώρο χρήστη μπορεί να έχει πρόσβαση σε αυτούς, ο οδηγός θα πρέπει να τους συνδέσει με κάποιες εσωτερικές συναρτήσεις που υλοποιούν τη λειτουργικότητα μιας συσκευής χαρακτήρων.

Οι πιο σημαντικές λειτουργίες σε μια συσκευή χαρακτήρων συμπεριλαμβάνονται από δομές του πυρήνα, κυρίαρχα τη stuct file_operations, την stuct file και την stuct node.

Αν και οι οδηγοί χαρακτήρων είναι αρκετά απλοί, η πλήρης περιγραφή τους εκτείνεται πέρα από τα όρια του σκοπού αυτής της εργασίας. Για το λόγο αυτό στη συνέχεια θα αναφερθούν μόνο εκείνα τα στοιχεία τους που χρησιμοποιήθηκαν κατά το σχεδιασμό και την υλοποίηση του πρωτοκόλλου SLURPoE.

Η δομή struct file_operations

Με τη δομή struct file_operations μια συσκευή χαρακτήρων δημιουργεί τη σύνδεση μεταξύ των αριθμών και των υποστηριζόμενων λειτουργιών της συσκευής. Η δομή αυτή ορίζεται στο αρχείο κεφαλίδας <linux/fs.h> και είναι μια συλλογή από δείκτες σε συναρτήσεις. Κάθε ανοιχτό αρχείο (που αναπαρίσταται εσωτερικά με μια δομή τύπου struct file) σχετίζεται με το δικό του σύνολο συναρτήσεων. Οι λειτουργίες αυτές είναι υπεύθυνες κατά κύριο λόγο για την υλοποίηση κλήσεων συστήματος και για αυτό ονομάζονται open, close, κ.λ.π.. Το αρχείο αυτό μπορεί να θεωρηθεί ως ένα "αντικείμενο" και οι συναρτήσεις που σχετίζονται με αυτό ως οι "μέθοδοί" του, στην ορολογία του αντικειμενοστραφούς προγραμματισμού.

Παραδοσιακά μια μεταβλητή τύπου struct file_operations ή ένας δείκτης σε μια τέτοια μεταβλητή ονομάζεται fops. Κάθε πεδίο της υλοποιεί μια συγκεκριμένη λειτουργία ή μπορεί να έχει τιμή NULL, για λειτουργίες που δεν υποστηρίζονται. Στην ακόλουθη λίστα περιγράφουμε τα πεδία που υποστηρίζουν τις λειτουργίες του οδηγού του πρωτοκόλλου SLURPoE.

- struct module *owner
  Το πρώτο πεδίο της δομής δεν είναι μια λειτουργία, αλλά ένας δείκτης στο module που του ανήκει η συσκευή χαρακτήρων. Η πιο συνηθισμένη τιμή που ανατίθεται στο πεδίο αυτό είναι η μακροεντολή THIS_MODULE που ορίζεται στο αρχείο κεφαλίδας <linux/module.h>.

- ssize_t (*read) (struct file *, char ___user *, size_t, loff_t *)
  Η συνάρτηση αυτή χρησιμοποιείται για την ανάκτηση δεδομένων από τη συσκευή. Μια τιμή NULL στο πεδίο αυτό θα προκαλέσει την επιστροφή της τιμής -EINVAL ("Invalid Argument") σε μια εφαρμογή που επιχειρεί να κάνει την αντίστοιχη κλήση συστήματος read.

- ssize_t (*write) (struct file *, const char ___user *, size_t, loff_t *)
  Η συνάρτηση αυτή χρησιμοποιείται για την αποστολή δεδομένων προς τη συσκευή. Μια τιμή NULL στο πεδίο αυτό θα προκαλέσει αντίστοιχα αποτελέσματα όπως και στη read.

- int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long)
  Η κλήση συστήματος ioctl παρέχει έναν τρόπο να δημιουργούνται με τον έλεγχο του προγραμματιστή εντολές κατάλληλες για κάθε συσκευή. Με την ioctl δηλαδή υλοποιούνται σε μια συσκευή όλες εκείνες οι λειτουργίες που δεν είναι δυνατό με άλλες κλήσεις συστήματος. Υπάρχουν μερικές ioctl εντολές που αναγνωρίζονται από τον πυρήνα ακόμα και όταν η ioctl δεν ορίζεται στη δομή fops. Αν μια συσκευή δεν παρέχει μια κλήση συστήματος ioctl, τότε η τιμή που επιστρέφεται σε μια απόπειρα χρησιμοποίησής της είναι η -ENOTTY ("No such ioctl for device")

- int (*mmap) (struct file *, struct vm_area_struct *)
  Η mmap() χρησιμοποιείται για να δημιουργηθεί από τον πυρήνα μια αντιστοίχιση στο χώρο μνήμης της εφαρμογής σε χώρο μνήμης που απαιτεί αυξημένα δικαιώματα. Παράδειγμα μιας τέτοιας περιοχής είναι η μνήμη μια συσκευής.

- int (*open) (struct inode *, struct file *)
  Η αντίστοιχη κλήση συστήματος που υλοποιείται από τη συνάρτηση αυτή είναι πάντα η πρώτη που χρησιμοποιείται στη συσκευή. Αν η τιμή της είναι NULL, τότε κάθε απόπειρα κλήσης της επιτυγχάνει, χωρίς όμως να λαμβάνεται κάποια ενέργεια από τον οδηγό.

- int (*release) (struct inode *, struct file *)
  Αυτή η συνάρτηση υλοποιεί τη λειτουργία που γίνεται κατά την απελευθέρωση της δομής. Όπως και η open έτσι και η close μπορεί να είναι κενή.

Για λόγους πληρότητας συμπεριλαμβάνονται και τα υπόλοιπα πεδία χωρίς όμως να περιγράφονται:

- loff_t (*llseek) (struct file *, loff_t, int)
- ssize_t (*aio_read) (struct kiocb *, const struct iovec *, unsigned long, loff_t)
- ssize_t (*aio_write) (struct kiocb *, const struct iovec *, unsigned long, loff_t)
- int (*readdir) (struct file *, void *, filldir_t)

- unsigned int (*poll) (struct file *, struct poll_table_struct *)
- int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long)
- long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long)
- long (*compat_ioctl) (struct file *, unsigned int, unsigned long)
- int (*flush) (struct file *, fl_owner_t id)
- int (*fsync) (struct file *, struct dentry *, int datasync)
- int (*aio_fsync) (struct kiocb *, int datasync)
- int (*fasync) (int, struct file *, int)
- int (*lock) (struct file *, int, struct file_lock *)
- ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int)
- unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned long, unsigned long)
- int (*check_flags)(int)
- int (*dir_notify)(struct file *filp, unsigned long arg)
- int (*flock) (struct file *, int, struct file_lock *)
- ssize_t (*splice_write)(struct pipe_inode_info *, struct file *, loff_t *, size_t, unsigned int)
- ssize_t (*splice_read)(struct file *, loff_t *, struct pipe_inode_info *, size_t, unsigned int)
- int (*setlease)(struct file *, long, struct file_lock **)

Η δομή struct file

Η δομή struct file ορίζεται στο αρχείο κεφαλίδας <linux/fs.h>, και είναι η δεύτερη πιο σημαντική δομή σε μια συσκευή χαρακτήρων. Με τη δομή αυτή αναπαρίσταται ένα ανοιχτό αρχείο στο Linux και δεν αποτελεί ειδική δομή των συσκευών χαρακτήρων. Δημιουργείται από τον πυρήνα σε μία κλήση της open και περνά ως όρισμα σε κάθε συνάρτηση που κάνει κάποια λειτουργία στο αρχείο, μέχρι την κλήση της close. Όταν όλα τα στιγμιότυπα του αρχείου κλείσουν τότε ο πυρήνας ελευθερώνει το χώρο της δομής struct file.

Στη συνέχεια περιγράφονται τα πιο σημαντικά πεδία της δομής αυτής:

mode_t f_mode Το πεδίο αυτό καθορίζει το αρχείο σαν αναγνώσιμο ή εγγράψιμο ή και τα δύο με βάση τα bits του FMODE_READ και FMODE_WRITE. Αυτό το πεδίο μπορεί να ελεγχθεί για την υλοποίηση ελεγχομένης πρόσβασης σε μια κλήση open ή ioctl. Αυτόματα ελέγχεται σε μία κλήση της read ή της write.

loff_t f_pos Κρατά την τρέχουσα θέση εγγραφής ή ανάγνωσης στο αρχείο. Ο τύπος loff_t έχει μέγεθος 64bit σε όλες τις αρχιτεκτονικές. Συνήθως η τιμή του πεδίου προορίζεται για ανάγνωση και ενημερώνεται αυτόματα με μια κλήση της read ή της write.

unsigned int flags Αυτές είναι οι σημαίες με τιμές όπως O_RDONLY, O_NONBLOCK και O_SYNC. Ένας οδηγός πρέπει να ελέγξει τη σημαία O_NONBLOCK για να δει αν έχει ζητηθεί κάποια λειτουργία E/E nonblocking. Οι άλλες σημαίες χρησιμοποιούνται πολύ σπάνια.

struct file_operations *f_op Το πεδίο αυτό είναι ένας δείκτης στις λειτουργίες που σχετίζονται με το αρχείο. Ο πυρήνας στην εσωτερική του υλοποίηση, δίνει τιμή σε αυτό το δείκτη με την κλήση της open και διαβάζει από αυτή το κατάλληλο πεδίο για κάθε κλήση συστήματος στο αρχείο. Αυτό επιτρέπει στον οδηγό τη δυνατότητα να αλλάζει την υλοποίηση των λειτουργιών σε χρόνο εκτέλεσης, επιτρέποντας μια σειρά από πολύ χρήσιμες ενέργειες.

void *private_data Το πεδίο αυτό πριν την κλήση της open παίρνει τιμή NULL. Ο προγραμματιστής μπορεί να το χρησιμοποιήσει για τους δικούς του σκοπούς ή να το αγνοήσει. Μπορεί να δεσμεύσει χώρο και να δώσει τη διεύθυνση του χώρου αυτού ως τιμή του δείκτη, λαμβάνοντας όμως υπόψη ότι είναι ευθύνη του και η απελευθέρωση του.

2.2.2 Το υποσύστημα διαχείρισης της μνήμης

Οι διευθύνσεις

Το Linux είναι ένα λειτουργικό σύστημα που χρησιμοποιεί ένα εικονικό σύστημα διαχείρισης της μνήμης, υπό την έννοια ότι οι διευθύνσεις που μπορούν να δουν οι εφαρμογές δεν αντιστοιχούν στις φυσικές διευθύνσεις που χρησιμοποιούνται από το υλικό. Η εικονική μνήμη εισάγει ένα επίπεδο αναφοράς, που επιτρέπει μια σειρά από πλεονεκτήματα. Με την εικονική διαχείριση, προγράμματα τα οποία τρέχουν στο σύστημα μπορούν να δεσμεύσουν και να χρησιμοποιήσουν περισσότερη μνήμη από αυτή που είναι φυσικά διαθέσιμη. Στην πραγματικότητα ακόμα και μια μεμονωμένη εφαρμογή μπορεί να έχει εικονικό χώρο διευθύνσεων μεγαλύτερο από τη φυσική μνήμη του συστήματος. Επίσης η εικονική μνήμη επιτρέπει κάποιους χρήσιμους χειρισμούς με το χώρο διευθύνσεων μια εφαρμογής, όπως είναι η πρόσβαση στη μνήμη μιας συσκευής.

Το Linux χρησιμοποιεί διάφορους τύπους διευθύνσεων, καθένας με διαφορετική σημασιολογική ερμηνεία. Δυστυχώς στον κώδικα του πυρήνα δεν είναι πάντοτε ξεκά-

θαρο ποιος τύπος διεύθυνσης χρησιμοποιείται σε κάθε περίπτωση για αυτό και ο προγραμματιστής πρέπει να έχει τον πλήρη έλεγχο. Η αδυναμία αυτή στο προγραμματιστικό περιβάλλον του Linux οφείλεται στη μη ύπαρξη διαφορετικών τύπων που να υποδεικνύουν σε τι χώρο της μνήμης γίνεται η κάθε αναφορά.

Οι τύποι των διευθύνσεων που ξεχωρίζουν στο Linux είναι οι εξής:

Εικονικές διευθύνσεις του χρήστη (User virtual addresses) Αυτές είναι οι κανονικές διευθύνσεις που χρησιμοποιούνται από εφαρμογές σε χώρο χρήστη. Μπορούν να έχουν μέγεθος 32 ή 64bit, ανάλογα με την αρχιτεκτονική του υλικού που χρησιμοποιείται. Η κάθε εφαρμογή έχει το δικό της εικονικό χώρο διευθύνσεων.

Φυσικές διευθύνσεις (Physical addresses Οι διευθύνεις που χρησιμοποιούνται από τον επεξεργαστή και από τη μνήμη του συστήματος. Οι φυσικές διευθύνσεις είναι και αυτές τιμές των 32 ή 64bit. Ακόμα και στα συστήματα των 32bit μπορεί να χρησιμοποιούνται φυσικές διευθύνσεις των 64bit, όπως για παράδειγμα σε συστήματα που είναι 32bit αλλά οι επεξεργαστές έχουν την ικανότητα που στη βιβλιογραφία απαντάται ως Φυσική Επέκταση Διευθύνσεων (Physical Address Extension – PAE

Διευθύνσεις Διαύλου (Bus addresses) Οι διευθύνσεις αυτές χρησιμοποιούνται μεταξύ των περιφερειακών διαύλων και της μνήμης. Πολύ συχνά ταυτίζονται με τις φυσικές διευθύνσεις που χρησιμοποιούνται από το CPU, αλλά αυτό είναι μια υπόθεση που επιβαρύνει τη φορητότητα του κώδικα. Οι διευθύνσεις διαύλου εξαρτώνται πολύ από την αρχιτεκτονική.

Λογικές διευθύνσεις του πυρήνα (Kernel logical addresses) Αυτός είναι ο κανονικός χώρος διευθύνσεων του πυρήνα. Αυτές οι διευθύνσεις αντιστοιχούν στο μεγαλύτερο μέρος αν όχι σε όλη την κεντρική μνήμη και συχνά αντιμετωπίζονται σαν να ήταν φυσικές διευθύνσεις. Στις περισσότερες αρχιτεκτονικές, οι λογικές διευθύνσεις και οι αντίστοιχες φυσικές διαφέρουν μόνο κατά μια σταθερά. Οι λογικές διευθύνσεις έχουν το μέγεθος του δείκτη του υλικού και για αυτό δεν μπορούν να διευθυνσιοδοτήσουν όλη τη φυσική μνήμη στα συστήματα 32bit. Ο χώρος που μπορεί να δεσμευτεί με την κλήση της συνάρτησης kmalloc είναι στο λογικό χώρο διευθύνσεων του πυρήνα.

Οι εικονικές διευθύνσεις του πυρήνα (Kernel virtual addresses) Αυτές οι διευθύνσεις διαφέρουν από τις λογικές στο ότι δεν έχουν απαραίτητα μια απευθείας αντιστοίχιση σε φυσικές διευθύνσεις και μπορεί να εμφανίζουν ως συνεχόμενο ένα χώρο διάσπαρτο στη φυσική μνήμη. Όλες οι λογικές διευθύνσεις είναι και εικονικές. Η κλήση της συνάρτησης ioremap που αντιστοιχεί μνήμη στις συσκευές, καθώς και η συνάρτηση vmalloc, που δεσμεύει χώρο στην κεντρική μνήμη, επιστρέφουν εικονικές διευθύνσεις του πυρήνα.

Οι πίνακες σελιδοποίησης

Όταν ένα πρόγραμμα αναφέρεται σε μια εικονική διεύθυνση, το CPU πρέπει να τη μετατρέψει σε φυσική για να πραγματοποιήσει την πρόσβαση στη φυσική μνήμη. Η διαδικασία αυτή συνήθως γίνεται με το διαχωρισμό της διεύθυνσης σε πεδία. Το κάθε πεδίο λειτουργεί ως ένας δείκτης πίνακα, που καλείται πίνακας σελίδων και περιέχει είτε τη διεύθυνση του επόμενου πίνακα είτε τη φυσική σελίδα που αντιστοιχεί στην εικονική διεύθυνση που γίνεται η αναφορά.

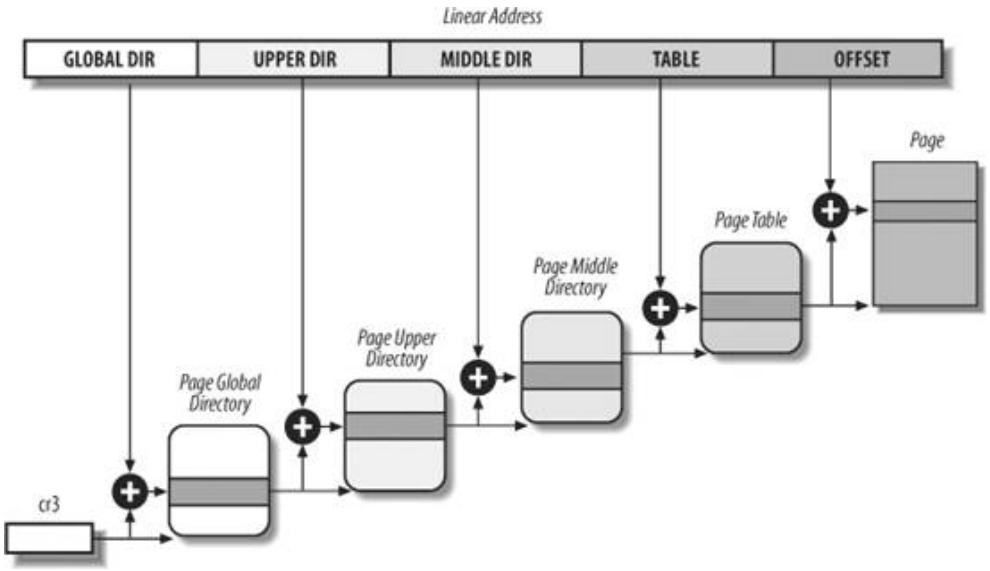
Ο πυρήνας Linux διαχειρίζεται ένα σύστημα από πίνακες σελιδοποίησης τεσσάρων επιπέδων για να αντιστοιχίσει μια εικονική σε μια φυσική διεύθυνση. Τα πολλαπλά επίπεδα οργάνωσης κάνουν εφικτή τη διασπορά των φυσικών διευθύνσεων που συμπεριλαμβάνονται μέσα στο εικονικά διαθέσιμο εύρος διευθύνσεων που φαίνεται συνεχές.

Ακόμα και σε υλικό το οποίο υποστηρίζει δύο επίπεδα οργάνωσης στους πίνακες σελιδοποίησης, είτε σε υλικό το οποίο χρησιμοποιεί ένα διαφορετικό τρόπο να κάνει την αντιστοίχιση, το Linux χρησιμοποιεί τον ίδιο τρόπο οργάνωσης τεσσάρων επιπέδων. Η χρήση της οργάνωσης αυτής που υλοποιείται με έναν ανεξάρτητο τρόπο από την αρχιτεκτονική του CPU επιτρέπει τη χρήση του Linux σε κάθε CPU και τον προγραμματισμό των διάφορων μερών του με μια ενιαία αντιμετώπιση του υποσυστήματος της μνήμης. Το σημαντικό στοιχείο που χαρακτηρίζει την επιλογή αυτή είναι ότι δε συνοδεύεται από επιπλέον επιβάρυνση στα συστήματα που υποστηρίζουν λιγότερα επίπεδα οργάνωσης. Αυτό το πολύ σημαντικό χαρακτηριστικό του σχεδιασμού εξασφαλίζεται από το μεταγλωττιστή που απομακρύνει το ένα ή τα δύο επίπεδα που δεν χρειάζονται μέσω βελτιστοποιήσεων.

Το Linux υιοθετεί ένα κοινό μοντέλο σελιδοποίησης για τις αρχιτεκτονικές των 32bits και των 64bits. Για ένα σύστημα 32bit η οργάνωση δύο επιπέδων είναι αρκετή, για ένα σύστημα όμως 64bit είναι απαραίτητη η οργάνωση περισσότερων επιπέδων. Τα τέσσερα είδη των πινάκων σελιδοποίησης είναι τα εξής:

- Καθολικός Κατάλογος Σελιδοποίησης (Page Global Directory)
- Ανώτερος Κατάλογος Σελιδοποίησης (Page Upper Directory)
- Μεσαίος Κατάλογος Σελιδοποίησης (Page Middle Directory)
- Πίνακας Σελίδων (Page Table)

Ο Καθολικός Κατάλογος Σελιδοποίησης περιέχει τις διευθύνσεις μερικών Ανώτερων Καταλόγων Σελιδοποίησης, οι οποίοι με τη σειρά τους περιέχουν τις διευθύνσεις μερικών Μεσαίων Καταλόγων Σελιδοποίησης και αυτοί με τη σειρά τους τις διευθύνσεις μερικών Πινάκων Σελίδων. Κάθε τιμή του Πίνακα Σελίδων είναι η διεύθυνση μιας φυσικής σελίδας. Έτσι η εικονική διεύθυνση μπορεί να χωριστεί σε πέντε μέρη. Στο σχήμα 2.10 φαίνεται αυτός ο διαχωρισμός μιας διεύθυνσης χωρίς όμως να φαίνεται το μέγεθος κάθε πεδίου αφού αυτό εξαρτάται από την αρχιτεκτονική του υπολογιστή.



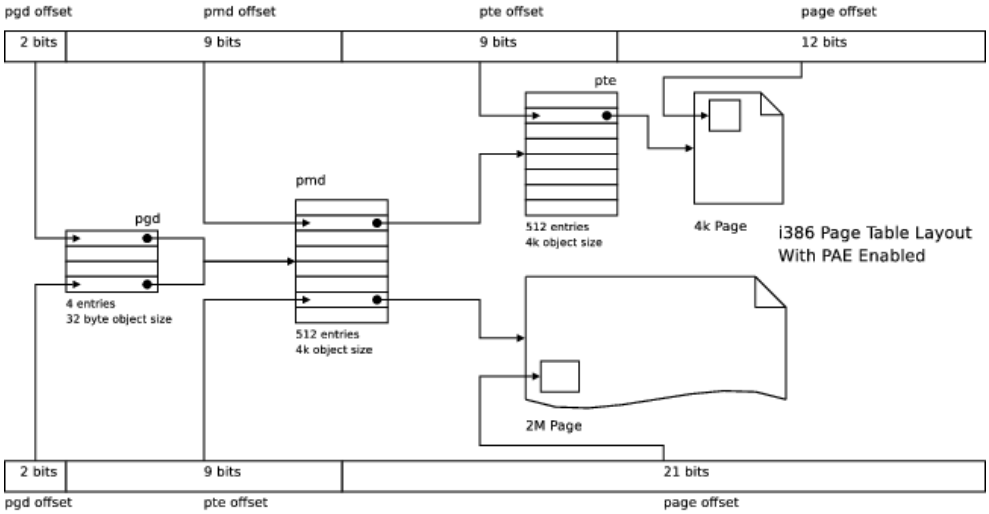
Σχήμα 2.10: Το μοντέλο σελιδοποίησης του Linux.

Για αρχιτεκτονικές 32bit με τη Φυσική Επέκταση Σελίδων ενεργοποιημένη χρησιμοποιούνται τρία επίπεδα. Ο Καθολικός Κατάλογος Σελιδοποίησης του Linux αντιστοιχεί στον Πίνακα Δεικτών Καταλόγου Σελίδων (Page Directory Pointer Table) του 80x86, ο Ανώτερος Πίνακας Σελιδοποίησης εξαλείφεται, ο Μεσαίος Πίνακας Σελιδοποίησης αντιστοιχεί στον Κατάλογο Σελίδων (Page Directory) του 80x86 και ο Πίνακας Σελίδων του Linux αντιστοιχεί στον Πίνακα Σελίδων (Page Table) του 80x86. Για αρχιτεκτονικές 64bit χρησιμοποιούνται τρία ή τέσσερα επίπεδα σελιδοποίησης με βάση το πώς χωρίζεται η εικονική διεύθυνση από το υλικό.

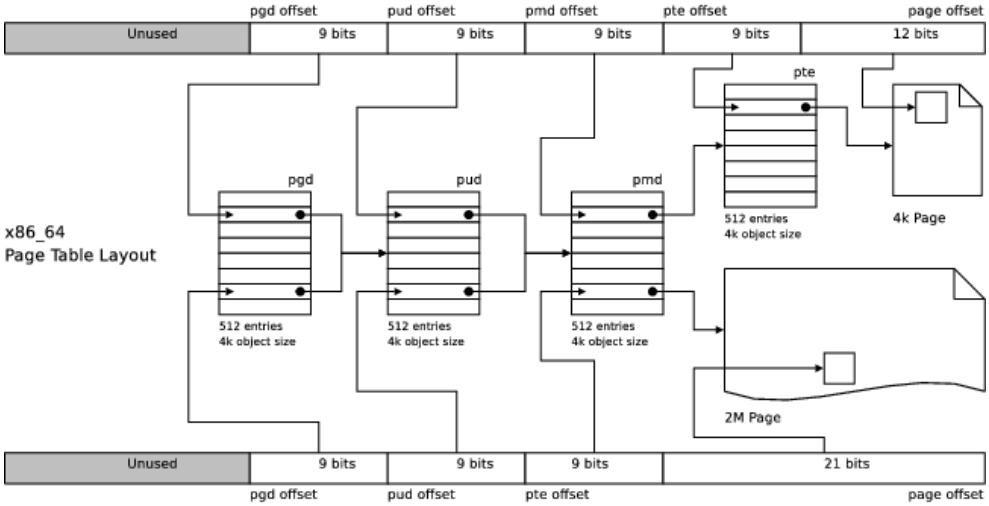
Η διαχείριση των εφαρμογών από το Linux βασίζεται πολύ στη σελιδοποίηση. Για την ακρίβεια, η αυτόματη μετάφραση των διευθύνσεων σε φυσικές σελίδες επιτυγχάνει τα εξής χαρακτηριστικά:

- Αποδίδονται διαφορετικοί φυσικοί χώροι διευθύνσεων σε κάθε εφαρμογή, γεγονός που εξασφαλίζει έναν αποδοτικό τρόπο προστασίας έναντι σε σφάλματα διευθύνσεων.
- Διαφοροποιούνται οι εικονικές σελίδες από τις φυσικές. Ο διαχωρισμός αυτός επιτρέπει σε μια εικονική σελίδα να αποθηκευτεί σε μια φυσική σελίδα, έπειτα όταν αυτό καταστεί αναγκαίο να αποθηκευτεί στο δίσκο και μετά να μεταφερθεί πάλι στη μνήμη σε μια διαφορετική φυσική σελίδα. Αυτό είναι και το βασικό χαρακτηριστικό ενός εικονικού συστήματος διαχείρισης της μνήμης.

Τα σχήματα 2.11, 2.12, 2.13 και 2.14 αποτελούν παραδείγματα του συστήματος σελιδοποίησης που χρησιμοποιεί το Linux για κάποιες ευρέως διαδεδομένες αρχιτεκτονικές



Σχήμα 2.11: Το μοντέλο σελιδοποίησης του Linux στην αρχιτεκτονική i386 με PAE.

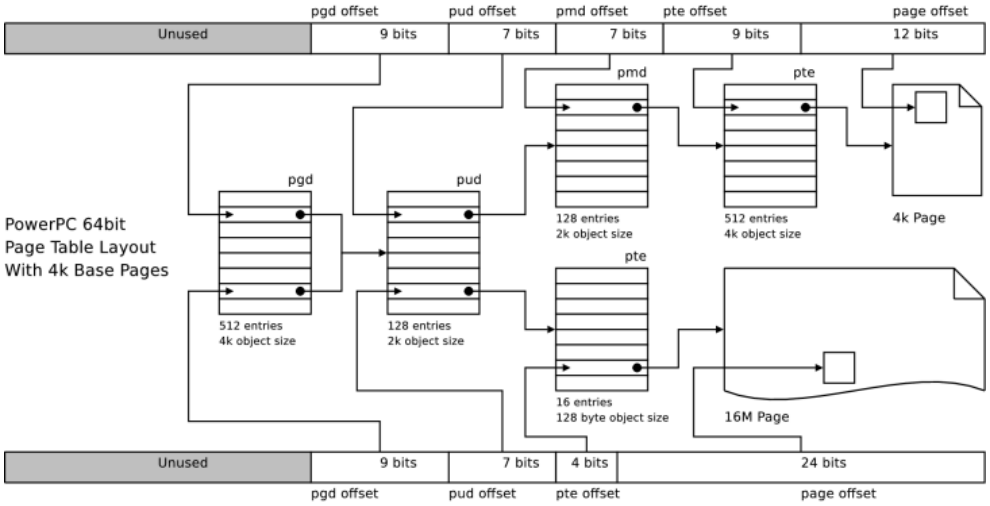


Σχήμα 2.12: Το μοντέλο σελιδοποίησης του Linux στην αρχιτεκτονική x86_64.

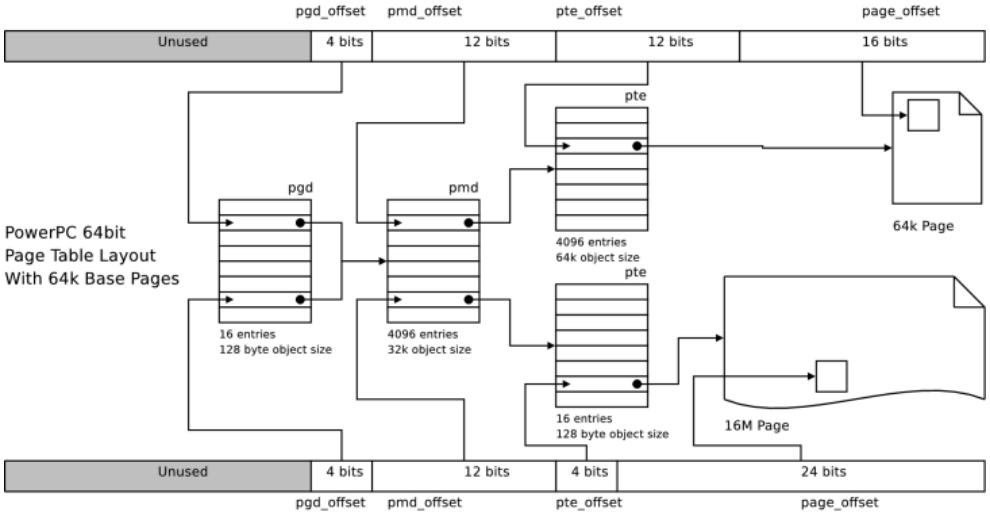
2.2.3 Ο μηχανισμός των διακοπών

Όπως το όνομα τους συνιστά, οι διακοπές παρέχουν έναν τρόπο για να εκτρέπουν τον επεξεργαστή σε κώδικα έξω από την κανονική ροή ελέγχου. Όταν ένα σήμα διακοπής φτάνει, το CPU πρέπει να σταματήσει την τρέχουσα επεξεργασία και να τρέξει μία καινούργια ενέργεια. Αυτό πραγματοποιείται σώζοντας την τρέχουσα τιμή του μετρητή προγράμματος (program counter) στη στοίβα του πυρήνα και θέτοντάς του μια νέα τιμή που εξαρτάται από τον τύπο της διακοπής.

Υπάρχει μια ειδοποιός διαφορά μεταξύ του χειρισμού μιας διακοπής και της αλλαγής



Σχήμα 2.13: Το μοντέλο σελιδοποίησης του Linux στην αρχιτεκτονική PowerPC με μέγεθος σελίδας 4kB.



Σχήμα 2.14: Το μοντέλο σελιδοποίησης του Linux στην αρχιτεκτονική PowerPC με μέγεθος σελίδας 64kB.

ελέγχου ανάμεσα σε διεργασίες: ο κώδικας που εκτελείται από μια συνάρτηση χειρισμού μιας διακοπής (Interrupt Service Routine – ISR) δεν είναι μια διεργασία. Αντίθετα είναι ένα μονοπάτι ελέγχου μέσα στον πυρήνα, που τρέχει με κόστος της ίδιας διεργασίας που έτρεχε όταν εμφανίστηκε η διακοπή. Σαν ένα μονοπάτι ελέγχου του πυρήνα, η ISR είναι πολύ ελαφρύτερη από μια διεργασία (έχει λιγότερες εντολές ενώ και η ροή ελέγχου χρειάζεται μικρότερο χρόνο για να εισέλθει καθώς και να εξέλθει από αυτή).

Ο χειρισμός μιας διακοπής εξαρτάται από τον τύπο της. Για τους σκοπούς της περιγραφής μας διαχωρίζουμε τις διακοπές σε τρεις τύπους:

Διακοπές Ε/Ε Αυτές ενεργοποιούνται όταν μια συσκευή Ε/Ε απαιτεί έναν ειδικό χειρισμό από τον πυρήνα. Η αντίστοιχη ISR πρέπει να την εξυπηρετήσει με έναν ειδικό τρόπο σύμφωνα με τις απαιτήσεις της συσκευής. Στην παρούσα εργασία αυτές είναι αποκλειστικά οι διακοπές που χρησιμοποιούνται και για το λόγο αυτό περιγράφονται διεξοδικά.

Διακοπές Ρολογιού Ένα ρολόι, είτε το τοπικό APIC ρολόι είτε ένα εξωτερικό, έχει ενεργοποιήσει μια διακοπή. Αυτός ο τύπος της διακοπής ενημερώνει τον πυρήνα ότι έχει παρέλθει ένα συγκεκριμένο χρονικό διάστημα.

Διακοπές μεταξύ επεξεργαστών Ένα CPU έχει ενεργοποιήσει μια διακοπή προς ένα άλλο CPU σε ένα σύστημα πολλών επεξεργαστών.

Σε γενικές γραμμές μια συνάρτηση χειρισμού μιας διακοπής Ε/Ε πρέπει να είναι αρκετά ευέλικτη για να αντιμετωπίσει πολλές συσκευές. Για παράδειγμα πολλές συσκευές μπορεί να μοιράζονται την ίδια γραμμή IRQ. Ως αποτέλεσμα το διάνυσμα διακοπών από μόνο του δεν περιέχει όλη την απαραίτητη πληροφορία για την ακριβή προέλευση της διακοπής.

Η ευελιξία ενός χειριστή διακοπών επιτυγχάνεται με δύο διαφορετικούς τρόπους:

Μοιραζόμενο IRQ Η ISR εκτελεί διάφορες συναρτήσεις χειρισμού διακοπής. Κάθε ISR είναι μια συνάρτηση που σχετίζεται με μια συσκευή που μοιράζεται την ίδια γραμμή IRQ. Επειδή δεν είναι γνωστό από πριν ποια συσκευή ενεργοποίησε τη διακοπή, εκτελούνται όλες οι ISR για να εξακριβωθεί ποια συσκευή χρειάζεται τον ειδικό χειρισμό.

Δυναμική δέσμευση IRQ Μια γραμμή IRQ σχετίζεται με έναν οδηγό συσκευής το αργότερο δυνατό. Για παράδειγμα η IRQ γραμμή ενός οδηγού ενός προσαρμογέα δικτύου δεσμεύεται μόνο όταν η συσκευή χρησιμοποιείται. Σε αυτήν την περίπτωση φυσικά οι συσκευές που μοιράζονται την ίδια γραμμή IRQ δεν μπορούν να χρησιμοποιηθούν ταυτόχρονα.

Όταν συμβεί μια διακοπή δεν είναι όλες οι ενέργειες, που λαμβάνονται ίδιας προτεραιότητας. Για την ακρίβεια, η ISR δεν είναι το πιο κατάλληλο μέρος για κάθε ενέργεια. Μεγάλες σε διάρκεια και όχι εξαιρετικά κρίσιμες ενέργειες δεν πρέπει να γίνονται γιατί κατά τη διάρκεια που μια ISR εκτελείται, τα σήματα στην αντίστοιχη γραμμή IRQ προσωρινά αγνοούνται. Πολύ σημαντικό επίσης είναι το γεγονός, ότι η διεργασία για λογαριασμό της οποίας εκτελείται μια ISR, πρέπει να παραμένει σε ενεργή κατάσταση (TASK_RUNNING), διαφορετικά το σύστημα μπορεί να "κολλήσει". Για το λόγο αυτό οι ISR δεν μπορούν να εκτελούν blocking διεργασίες.

Η δυναμική εγκατάσταση ενός χειριστή διακοπών

Οι γραμμές IRQ [7] όπως προαναφέρθηκε είναι ένας πολύτιμος και συχνά πολύ περιορισμένος πόρος, ειδικά όταν υπάρχουν μόνο 15 ή και 16 από αυτές. Ο πυρήνας κράτα αρχείο για το ποιες χρησιμοποιούνται και ποιες όχι. Ένα module πρέπει να ζητήσει μια γραμμή IRQ πριν τη χρησιμοποιήσει και να την απελευθερώσει αμέσως μετά. Σε πολλές περιπτώσεις οι οδηγοί συσκευών πρέπει να μπορούν να μοιράζονται τις γραμμές διακοπών με άλλους. Οι ακόλουθες συναρτήσεις δηλώνονται στο αρχείο κεφαλίδας <linux/interrupt.h> και υλοποιούν τη διεπαφή δέσμευσης διακοπών:

int request_irq(**unsigned int** irq, irqreturn_t (*handler)(**int**, **void** *), unsinged **long** flags, **const char** *dev_name, **void** *dev_id);
void free_irq(**unsigned int** irq, **void** *dev_id);

Η τιμή που επιστρέφει η request_irq() στη συνάρτηση που την καλεί είναι 0 αν επιτύχει διαφορετικά ένας αρνητικός κωδικός που επισημαίνει το λάθος. Δεν είναι πολύ ασυνήθιστο για τη συνάρτηση να επιστρέψει την τιμή -EBUSY, στην περίπτωση που ένας άλλος οδηγός ήδη χρησιμοποιεί τη ζητούμενη γραμμή διακοπής. Τα ορίσματα της είναι τα εξής:

unsinged int irq Ο αριθμός της γραμμής που ζητείται.

irqreturn_t (*handler)(int, void *) Ο δείκτης στη συνάρτηση χειρισμού της διακοπής.

unsigned long flags Μια μάσκα δυαδικών τιμών που σχετίζεται με τη διαχείριση της διακοπής.

const char *name Το όνομα που θα εμφανίζεται στο αρχείο /proc/interrupts για να δείχνει τον ιδιοκτήτη της διακοπής.

void *dev_id Ένας δείκτης που χρησιμοποιείται για μοιραζόμενες διακοπές. Είναι ένα μοναδικό αναγνωριστικό που χρησιμοποιείται όταν η γραμμή διακοπής ελευθερώνεται και μπορεί επίσης να χρησιμοποιηθεί από τον οδηγό για να δείχνει στον ιδιωτικό της χώρο. Αν η διακοπή δεν είναι μοιραζόμενη τότε η τιμή της μπορεί να είναι το NULL.

Οι πιο συχνά χρησιμοποιούμενες τιμές που μπορούν να είναι μέρος της μάσκας που περνά ως παράμετρος flags στη request_irq() είναι:

IRQF_DISABLED Δηλώνει ένα γρήγορο χειριστή διακοπής. Όταν είναι στη μάσκα των flags τότε οι διακοπές απενεργοποιούνται συνολικά στον επεξεργαστή.

IRQF_SAMPLE_RANDOM Η διακοπή αυτή χρησιμοποιείται για είσοδο στην τυχαία γεννήτρια.

IRQF_SHARED Η διακοπή αυτή θα μοιράζεται με άλλες συσκευές.

IRQF_PROBE_SHARED Η διακοπή αυτή χρησιμοποιείται όταν κάποιες συσκευές ενδέχεται να μη δέχονται το μοίρασμά της.

IRQF_TIMER Η διακοπή αυτή προέρχεται από ρολόι.

IRQF_PERCPU Η διακοπή αυτή προέρχεται από κάποιο CPU.

IRQF_IRQPOLL Η διακοπή αυτή χρησιμοποιείται για polling (μόνο η πρώτη διακοπή σε μια μοιραζόμενη μπορεί να πάρει το χαρακτήρα αυτό για λόγους απόδοσης).

2.2.4 Οι ουρές εργασιών στο Linux (Workqueues)

Οι ουρές εργασιών (Work Queues – WQ) εμφανίστηκαν στον πυρήνα του Linux στις εκδόσεις 2.6. Επιτρέπουν σε συναρτήσεις του πυρήνα να ενεργοποιούνται και αργότερα να εκτελούνται από ειδικά νήματα του πυρήνα που καλούνται νήματα εργάτες.

Η κύρια δομή των ουρών εργασιών είναι η struct work_queuestruct, που ορίζεται στο αρχείο κεφαλίδας <linux/workqueue.h>. Η δημιουργία μιας τέτοιας δομής μπορεί να γίνει με τη χρήση δύο συναρτήσεων:

```
struct workqueue_struct *create_worqueue(const char *name);
struct workqueue_struct *create_singlethreaded_worqueue(
    const char *name);
```

Κάθε ουρά εργασιών έχει μια ή περισσότερες διεργασίες (νήματα πυρήνα), που τρέχουν όταν υποβάλλονται συναρτήσεις στην ουρά. Η χρήση της συνάρτησης create¬ _workqueue() έχει ως αποτέλεσμα μια ουρά η οποία διαθέτει ένα νήμα σε κάθε διαθέσιμο επεξεργαστή στο σύστημα. Στα νήματα αυτά εκτελούνται οι εργασίες που εισέρχονται στην ουρά. Αν ένα μόνο νήμα είναι αρκετό για τις ανάγκες ενός οδηγού, τότε η ουρά εργασιών μπορεί να δημιουργηθεί με την κλήση της συνάρτησης create¬ _singlethreaded_workqueue().

Για να αποδοθεί μια εργασία σε μια ουρά πρέπει να εισαχθεί σε μια δομή struct work¬ _struct. Αυτό μπορεί να γίνει σε χρόνο μεταγλώττισης με τη χρήση της μακροεντολής:

```
DECLARE_WORK(name, void (*function)(void *), void *data);
```

Όπου name είναι το όνομα της μεταβλητής τύπου struct struct work_struct που δηλώνεται, function είναι ο δείκτης στη συνάρτηση που καλείται από την ουρά εργασίας και data είναι ο δείκτης σε μια μεταβλητή που περνά ως όρισμα στη συνάρτηση. Διαφορετικά κάτι τέτοιο μπορεί να γίνει και σε χρόνο εκτέλεσης από τις διαθέσιμες μακροεντολές:

```
INIT_WORK(struct work_struct *work, void (*function)(void *), void *data);
PREPARE_WORK(struct work_struct *work, void (*function)(void *),
    void *data);
```

Η INIT_WORK κάνει μια πιο ενδελεχή εργασία κατά την αρχικοποίηση της δομής και για αυτό χρησιμοποιείται την πρώτη φορά που δημιουργείται η δομή. Η PREPARE_WORK έχει το ίδιο σχεδόν αποτέλεσμα, αλλά δεν αρχικοποιεί του δείκτες που χρησιμοποιούνται για να συνδεθεί η δομή stuct work_struct με την ουρά εργασιών. Αν μια δομή struct work_struct έχει αποδοθεί σε μια ουρά εργασιών και χρειάζεται να εισέλθει σε κάποια άλλη ουρά τότε πρέπει να χρησιμοποιηθεί η PREPARE_WORK και όχι η INIT_WORK.

Υπάρχουν δύο συναρτήσεις για να αποδώσουμε μια εργασία σε μια ουρά:

int queue_work(**struct** workqueue_struct *queue, **struct** work_struct *work);
int queue_delayed_work(**struct** workqueue_struct *queue, **struct** work_struct *work, **unsigned long** delay);

Και οι δύο προσθέτουν εργασίες σε μια ουρά. Αν όμως χρησιμοποιηθεί η queue_¬ delayed_work(), η εργασία δεν προστίθεται παρά μόνο μετά την παρέλευση χρόνου που υποδεικνύεται από την παράμετρο delay. Όπως συμβαίνει συνήθως και εδώ η τιμή που επιστρέφεται είναι 0 σε περίπτωση επιτυχίας και σε περίπτωση αποτυχίας αρνητική με μια τιμή που δηλώνει το σφάλμα.

Σε κάποια στιγμή, η συνάρτηση της εργασίας θα κληθεί με την κατάλληλη παράμετρο. Η συνάρτηση θα τρέχει στο πλαίσιο του νήματος εργάτη και αν χρειαστεί μπορεί να "κοιμηθεί". Αυτό που η συνάρτηση δεν μπορεί να κάνει είναι να έχει πρόσβαση στο χώρο χρήστη. Αφού η εκτέλεση της είναι μέσα σε ένα νήμα του πυρήνα δεν υπάρχει άλλωστε και κάποιος συνδεδεμένος χώρος χρήστη για να έχει πρόσβαση.

Αν χρειαστεί να ακυρωθεί η είσοδος σε μια ουρά εργασίας χρησιμοποιείται η συνάρτηση:

int cancel_delayed_work(**struct** work_struct *work);

Η τιμή που επιστρέφει είναι μη μηδενική αν η εργασία ακυρώθηκε πριν ξεκινήσει η εκτέλεσή της. Ο πυρήνας εγγυάται ότι η εκτέλεση της συγκεκριμένης εργασίας δεν θα ξεκινήσει μετά από την κλήση της συνάρτησης αυτής. Αν όμως η επιστροφή από τη cancel_delayed_work() είναι 0 τότε μπορεί ήδη η εργασία να τρέχει σε κάποιον άλλο επεξεργαστή και να συνεχίσει να τρέχει και μετά την κλήση της cancel_delayed_work(). Για να είναι σίγουρο πως η εργασία δε θα τρέχει πουθενά στο σύστημα μετά την κλήση της συνάρτησης cancel_delayed_work() που επιστρέφει 0 πρέπει να γίνει κλήση στη συνάρτηση:

void flush_workqueue(**struct** workqueue_struct *queue);

Μετά την κλήση της συνάρτησης flush_workqueue() καμία εργασία που είχε εισαχθεί στην ουρά δεν τρέχει στο σύστημα.

Μετά το τέλος της χρήσης μιας ουράς εργασίας, μπορούμε να ελευθερώσουμε τους πόρους που έχουν διατεθεί για αυτή με μια κλήση στη συνάρτηση:

void destroy_workqueue(**struct** workqueue_struct *queue);

2.2.5 Οι οδηγοί προσαρμογέων δικτύου

Οι οδηγοί των προσαρμογέων δικτύου [8] είναι διαφορετικοί από τις υπόλοιπες κλάσεις οδηγών στο Linux γιατί δε χρησιμοποιούν κάποιο αρχείο στους καταλόγους /dev ή /sys για να επικοινωνήσουν με το λειτουργικό σύστημα. Αντίθετα οι εφαρμογές αλληλεπιδρούν με έναν οδηγό προσαρμογέα δικτύου διαμέσου μιας διεπαφής δικτύου. Ένα παράδειγμα είναι ή eth0, για την πρώτη διεπαφή Ethernet, η οποία αποτελεί μια αφαίρεση σε μια στοίβα πρωτοκόλλων που βρίσκεται από πίσω.

Ένας οδηγός για έναν προσαρμογέα δικτύου χρησιμοποιεί τις παρακάτω δομές:

1. Δομές που διαμορφώνουν τα δομικά στοιχεία της στοίβας των πρωτοκόλλων δικτύου. Η βασική δομή είναι η struct sk_buff που ορίζεται στο αρχείο κεφαλίδας <include/linux/sk_buff.h> και χρησιμοποιείται στην υλοποίηση του πυρήνα για τη στοίβα TCP/IP.

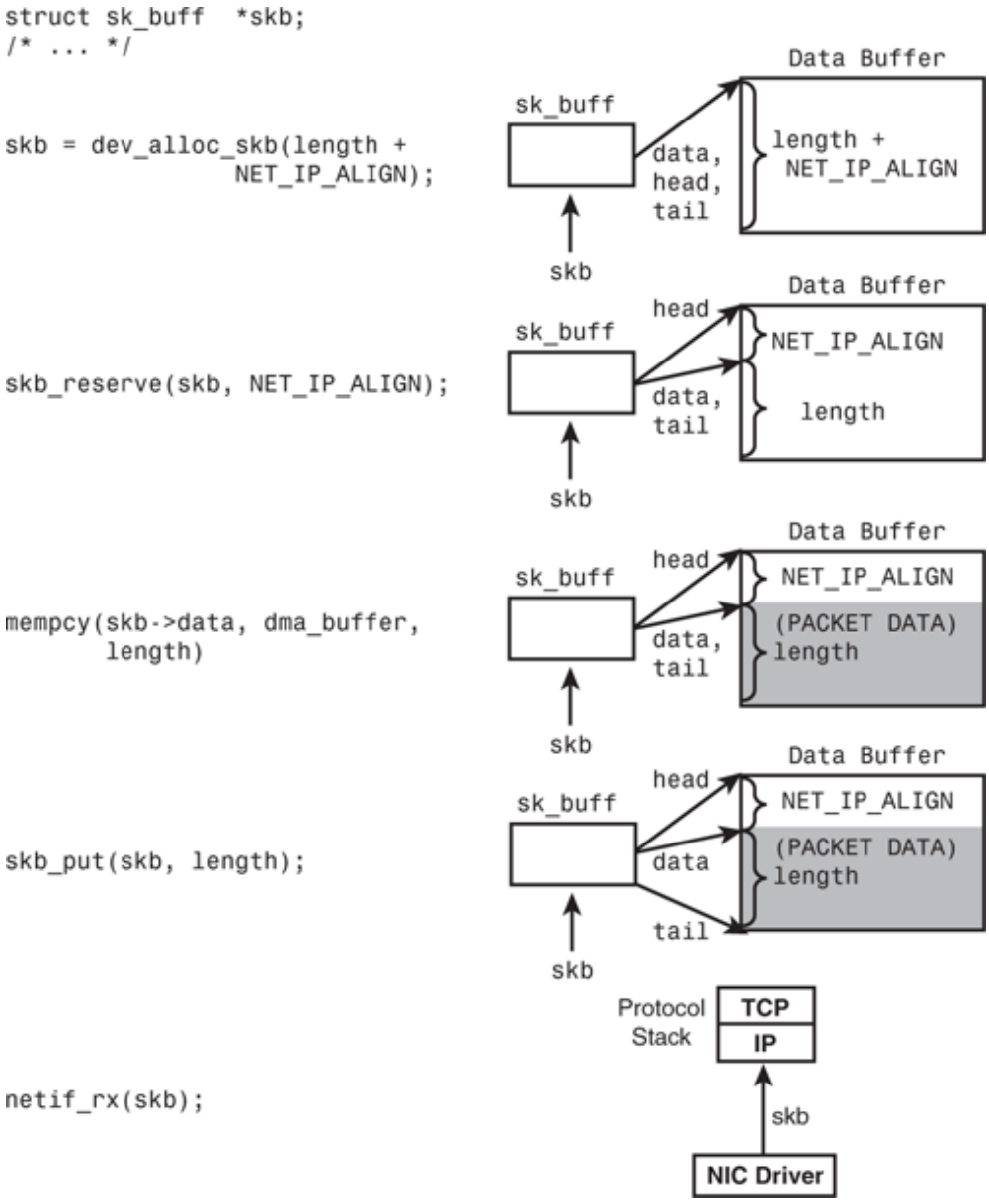
2. Δομές που ορίζουν μια διεπαφή μεταξύ του οδηγού του προσαρμογέα δικτύου και τη στοίβα πρωτοκόλλων. Η struct net_device που ορίζεται στην κεφαλίδα <include/linux/netdevice.h> είναι η βασική δομή που υλοποιεί αυτή τη διεπαφή.

3. Δομές που αναφέρονται στο δίαυλο Ε/Ε. Ο δίαυλος PCI και τα παράγωγά του είναι οι πιο συχνοί δίαυλοι που χρησιμοποιούνται από τους σύγχρονους προσαρμογείς δικτύου.

Η δομή struct sk_buff

Με τις δομές αυτές οι οδηγοί του Linux υλοποιούν αποδοτικούς μηχανισμούς χειρισμού και ελέγχου της ροής στα δικτυακά επίπεδα. Μια δομή struct sk_buff εμπεριέχει πληροφορίες ελέγχου που περιγράφουν προσωρινούς χώρους μνήμης για τη χρήση στην προετοιμασία πακέτων δικτύου. Είναι στην ουσία πολύ μεγάλες δομές που περιέχουν ένα μεγάλο αριθμό στοιχείων, η περιγραφή τους όμως θα περιοριστεί στο απαραίτητο υπόβαθρο για την κατανόηση της υπόλοιπης εργασίας. Η struct sk¬ _buff συνδέεται με τον αντίστοιχο προσωρινό χώρο μνήμης των πακέτων χρησιμοποιώντας, κυρίαρχα, πέντε πεδία:

- head, που δείχνει στην αρχή του πακέτου.
- data, που δείχνει στην αρχή των δεδομένων που μεταφέρει το πακέτο.
- tail , που δείχνει στο τέλος των δεδομένων που μεταφέρει το πακέτο.
- end, που δείχνει στο τέλος του πακέτου.
- len, που είναι το μέγεθος των δεδομένων που μεταφέρονται από το πακέτο.



Σχήμα 2.15: Οι λειτουργίες σε μια δομή struct sk_buff.

Αν skb είναι ένας δείκτης σε μία δομή struct sk_buff, τα πεδία που περιέχει η δομή skb→head, skb→data, skb→tail και skb→end, όσο αυτό περνάει από επίπεδο σε επίπεδο στη στοίβα πρωτοκόλλων του δικτύου μετακινούνται για να δείχνουν σε κατάλληλες θέσεις μέσα στον προσωρινό χώρο αποθήκευσης του πακέτου. Το skb→data για παράδειγμα δείχνει στην κεφαλίδα του τρέχοντος πρωτοκόλλου που επεξεργάζεται το πακέτο. Όταν το πακέτο φτάσει στο επίπεδο του πρωτοκόλλου IP στην πλευρά του παραλήπτη το skb→data δείχνει στην κεφαλίδα του IP. Όταν το πακέτο περνά

στο επίπεδο του TCP, το skb→data μετακινείται στην αρχή της κεφαλίδας του TCP. Όσο το πακέτο συνεχίζει να διατρέχει τα διάφορα επίπεδα μιας στοίβας πρωτοκόλλων που προσθέτουν ή αφαιρούν κεφαλίδες, το skb→len αλλάζει επίσης. Η δομή struct sk_buff περιέχει και άλλους δείκτες εκτός από αυτούς που προαναφέρθηκαν. Ένα παράδειγμα είναι το skb→nh, που αποθηκεύει τη θέση του κατωτέρου στη στοίβα πρωτοκόλλου του δικτύου ανεξάρτητα από την τρέχουσα θέση του skb→data.

Στο σχήμα 2.15 φαίνονται οι αλλαγές στο μονοπάτι παραλαβής ενός πακέτου. Για λόγους διευκόλυνσης της απεικόνισης, έχει γίνει η υπόθεση ότι οι ενέργειες εκτελούνται σειριακά. Στην πραγματικότητα, για λόγους απόδοσης, τα πρώτα δύο βήματα (dev_¬ alloc_skb() και skb_reserve()) γίνονται κατά την αρχικοποίηση ενός οδηγού όπου δεσμεύονται ένα σύνολο προσωρινών χώρων αποθήκευσης για πακέτα που παραλαμβάνονται. Το τρίτο βήμα γίνεται απευθείας από το υλικό του προσαρμογέα δικτύου με τη DMA μεταφορά πακέτων που παραλαμβάνονται σε ένα ήδη δεσμευμένο struct sk_buff. Τα δύο τελευταία βήματα (skb_put() και net_if_rx()) εκτελούνται από τη συνάρτηση χειρισμού της διακοπής που ενεργοποιείται από μια παραλαβή.

Για να δημιουργηθεί ένα struct sk_buff που αποθηκεύει ένα πακέτο που παραλαμβάνεται, στο σχήμα χρησιμοποιείται η dev_alloc_skb(). Αυτή είναι μια ασφαλής απέναντι σε διακοπές ρουτίνα που δεσμεύει χώρο για ένα struct sk_buff και το συσχετίζει με έναν προσωρινό χώρο αποθήκευσης των δεδομένων του πακέτου. Η dev_¬ skb_kfree() πραγματοποιεί την αντίστροφη λειτουργία. Στο σχήμα 2.15 στο δεύτερο βήμα καλείται η skb_reserve() για να προσθέσει ένα παραγέμισμα μεγέθους 2 byte μεταξύ της αρχής του προσωρινού χώρου του πακέτου και του χώρου των δεδομένων. Με τον τρόπο αυτό η κεφαλίδα του IP ξεκινά σε μια θέση με την οποία επιτυγχάνεται η αποδοτικότερη επεξεργασίας της. Δεδομένου ότι η κεφαλίδα του Ethernet είναι 14bytes με την προσθήκη 2bytes, εξασφαλίζεται η ευθυγράμμιση της κεφαλίδας στα 16bytes. Οι υπόλοιπες λειτουργίες στο σχήμα γεμίζουν τον προσωρινό χώρο με τα δεδομένα του πακέτου και αλλάζουν κατάλληλα τα πεδία skb→data, skb→tail και skb→len.

Στην προγραμματιστική διεπαφή του Linux υπάρχουν πολλές ακόμα σχετικές συναρτήσεις. Για παράδειγμα η skb_clone(), δημιουργεί ένα αντίγραφο ενός struct sk_¬ buff χωρίς να αντιγράφει τα δεδομένα του σχετιζόμενου προσωρινού χώρου αποθήκευσης του πακέτου. Μέσα στο αρχείο <net/core/skbuff.c> μπορούν να βρεθούν οι περισσότερες από τις συναρτήσεις χειρισμού των struct sk_buff.

Η δομή struct net_device

Οι οδηγοί των καρτών δικτύων χρησιμοποιούν μια κοινή διεπαφή με την οποία αλληλεπιδρούν με τη στοίβα TCP/IP. Η δομή struct net_device, η οποία είναι ακόμα μεγαλύτερη από την struct sk_buff, ορίζει τη διεπαφή επικοινωνίας. Παρόμοια με τα βήματα στο σχήμα 2.15, κατά την αρχικοποίηση ενός οδηγού ενός προσαρμογέα δικτύου ακολουθούνται τα εξής βήματα:

- Ο οδηγός δεσμεύει χώρο για ένα struct net_device με τη συνάρτηση alloc_¬ netdev(). Πιο συχνά χρησιμοποιείται μια πιο κατάλληλη συνάρτηση που κάνει

με τη σειρά της κλήση της alloc_netdev(). Ένας οδηγός για έναν προσαρμογέα Ethernet, για παράδειγμα, χρησιμοποιεί την alloc_etherdev(). Ένας οδηγός για έναν προσαρμογέα ασύρματου δικτύου χρησιμοποιεί την alloc_ieee¬ 80211(). Όλες αυτές οι συναρτήσεις δέχονται ως όρισμα το μέγεθος ενός ιδιωτικού χώρου και δεσμεύουν το χώρο αυτό μαζί με το χώρο για τη δομή struct net_device:

```
struct net_device *netdev;
struct priv_struct *mycard_priv;
netdev = alloc_etherdev(sizeof(struct priv_struct));
/* The private space allocated by alloc_etherdev() */
mycard_priv = netdev→priv;
```

- Ο οδηγός υπολογίζει διάφορα πεδία στο struct net_device που δεσμεύτηκε και το καταχωρεί στο επίπεδο δικτύου με την κλήση της register_netdev(¬ netdev).
- Ο οδηγός διαβάζει τη διεύθυνση MAC από τη μνήμη EEPROM του προσαρμογέα δικτύου και ρυθμίζει το Wake–On–Lan (WOL) αν αυτό χρειάζεται. Οι προσαρμογείς Ethernet συνήθως έχουν μια μη μεταβλητή EEPROM για να κρατά πληροφορίες όπως είναι η MAC διεύθυνση και το πρότυπο του WOL. Η διεύθυνση MAC όπως έχει προαναφερθεί είναι μια τιμή μεγέθους 48bit που είναι καθολικά μοναδική. Το πρότυπο του WOL είναι μια "μαγική" ακολουθία, η οποία αν αυτή βρεθεί μέσα σε ένα πακέτο τότε ο προσαρμογέας δικτύου ενεργοποιείται, αν βρισκόταν σε ανενεργή κατάσταση.
- Αν ο προσαρμογέας χρειάζεται λογισμικό για να λειτουργήσει, ο οδηγός το μεταφορτώνει χρησιμοποιώντας τη συνάρτηση request_firmware().

Κεφάλαιο 3

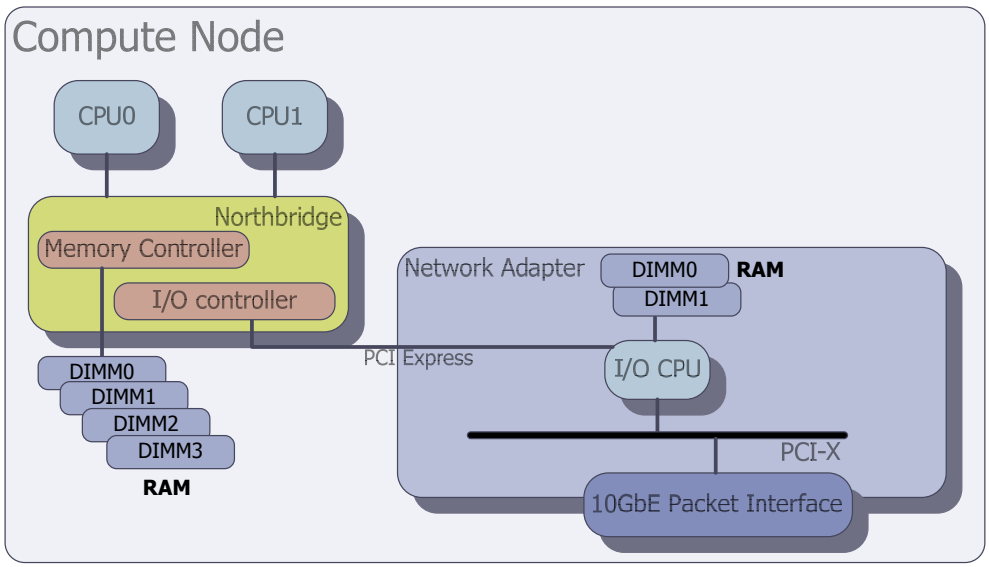
Σχεδιασμός του πρωτοκόλλου SLURPoE

Στο κεφάλαιο αυτό περιγράφεται η δομή και οι αρχές λειτουργίας του πρωτοκόλλου που σχεδιάστηκε στο πλαίσιο της εργασίας αυτής. Στο εξής θα αναφερόμαστε σε αυτό με τον όρο SLURPoE (SimpLe User – level RDMA Protocol over Ethernet). Ουσιαστικά πρόκειται για ένα μηχανισμό μεταφοράς δεδομένων από κόμβο σε κόμβο χωρίς τη δημιουργία αντιγράφων και τη μεσολάβηση του πυρήνα του λειτουργικού συστήματος πάνω από ένα διαδεδομένο δίκτυο διασύνδεσης, το Ethernet.

Ένας από τους λόγους για τους οποίους χρησιμοποιείται το Ethernet είναι γιατί το κόστος του είναι μικρό και το υλικό του ευρέως διαδεδομένο. Έτσι, ο προσαρμογέας που χρησιμοποιείται είναι απλός και έχει χαμηλή τιμή (commodity).

Η απαίτηση για τη μη δημιουργία αντιγράφων ικανοποιείται με τη σχεδιαστική επιλογή της χρήσης ενός πρωτοκόλλου απομακρυσμένης απευθείας πρόσβασης στη μνήμη (Remote Direct Memory Access – RDMA). Η απαίτηση για την παράκαμψη του πυρήνα του λειτουργικού συστήματος ικανοποιείται από τη σχεδίαση του πρωτοκόλλου σε χώρο χρήστη. Τέλος, η υψηλή επίδοση του πρωτοκόλλου απαιτεί τη χρήση προσαρμογέων με ταχύτητες μετάδοσης 10G. Επομένως, για την υλοποίηση του πρωτοκόλλου πρέπει να υπάρχει υποστήριξη από το υλικό. Συνεπώς, θα πρέπει να παρέχει υποδομή για επεξεργασία και προσωρινή αποθήκευση πακέτων δικτύου καθώς και μηχανισμούς μεταφοράς από και προς το σύστημα στο οποίο φιλοξενείται. Για το λόγο αυτό, είναι αναγκαίο να χρησιμοποιηθεί ένας προγραμματιζόμενος προσαρμογέας δικτύου 10G Ethernet που αναλαμβάνει τις υπολογιστικές ανάγκες της επικοινωνίας του πρωτοκόλλου.

Αρχικά σχεδιάζεται ένα δίκτυο διασύνδεσης που κάνει χρήση αυτού του προγραμματιζόμενου προσαρμογέα ώστε να παρατηρηθεί πώς η μεταβολή στις δυνατότητές του επηρεάζει την απόδοση του πρωτοκόλλου. Στόχος είναι, να διαμορφωθεί μια αρχιτεκτονική πρόταση για το πρόβλημα καθορισμού των υπολογιστικών και λειτουργικών δυνατοτήτων που πρέπει να έχει ένας προσαρμογέας 10GbE για συστήματα υψηλής επίδοσης.



Σχήμα 3.1: Το φυσικό διάγραμμα του προσαρμογέα δικτύου.

Στην συνέχεια θα χρησιμοποιηθούν οι όροι: **κόμβος**, **προσαρμογέας δικτύου** ή απλά **προσαρμογέας** και **προσαρμογέας πακέτων**. Για να αποφευχθεί οποιαδήποτε σύγχυση κατά την περιγραφή θεωρούμε την εξής σύμβαση: με τον όρο **κόμβο** θα αναφερόμαστε στο σύστημα του οποίου οι εφαρμογές μπορούν να αποτελούν άκρα του πρωτοκόλλου. Με τον όρο **προσαρμογέας** ορίζουμε τον έξυπνο προσαρμογέα δικτύου ο οποίος συνδέεται απευθείας στον κόμβο, περιλαμβάνει CPU, μνήμη και μηχανές DMA και αναλαμβάνει το μεγαλύτερο μέρος της υπολογιστικής ισχύς που απαιτεί το πρωτόκολλο. Με τον όρο **προσαρμογέας πακέτων** αναφερόμαστε στο υλικό που αναλαμβάνει τη μετατροπή των πακέτων Ethernet σε ηλεκτρικά σήματα στο καλώδιο (σχήμα 3.1).

Η διάταξη καθορίστηκε με τον τρόπο αυτό, για να μπορεί:

- Να ικανοποιήσει αρχικά τις λειτουργικές ανάγκες του πρωτοκόλλου.
- Να προσαρμοστούν οι δυνατότητες του προσαρμογέα έτσι, ώστε να υπάρχει ευχέρεια στη παραμετροποίηση της πειραματικής διάταξης.

3.1 Γενικά

Στην παρούσα εργασία υλοποιείται ένα πειραματικό δίκτυο διασύνδεσης το οποίο βασίζεται σε ένα πρωτόκολλο RDMA που έχει ως φυσικό επίπεδο το 10G Ethernet.

Η προσέγγιση που ακολουθούμε αφορά σε ένα μοντέλο διαχωρισμένου οδηγού (Split Driver Model) και αποτελείται από δύο μέρη: το πρώτο εκτελείται στον κόμβο και το άλλο στον προσαρμογέα.

Το μεγαλύτερο μέρος του οδηγού του κόμβου εκτελείται σε χώρο χρήστη (user – space) και μόνο τα σημεία που χρειάζονται περισσότερα δικαιώματα εκτελούνται σε χώρο πυρήνα (kernel – space). Η σχεδιαστική αυτή επιλογή απαντάται στη βιβλιογραφία ως δικτύωση στο επίπεδο χρήστη (User – Level Networking).

3.2 Επιλογές της σχεδίασης

3.2.1 Απευθείας Απομακρυσμένη Πρόσβαση στη Μνήμη

Η Απομακρυσμένη Απευθείας Πρόσβαση στη Μνήμη (Remote Direct Memory Access – RDMA) επιτρέπει τη μεταφορά δεδομένων από τη μνήμη ενός κόμβου στη μνήμη ενός άλλου με μονομερείς ενέργειες. Με την τεχνική αυτή επιτυγχάνεται δικτύωση υψηλής ταχύτητας και χαμηλού χρόνου αρχικής απόκρισης, χαρακτηριστικά ιδιαίτερα χρήσιμα σε μεγάλες συστοιχίες υπολογιστών. Το RDMA βασίζεται στην ουσία σε μια επέκταση της λογικής της Απευθείας Πρόσβασης στη Μνήμη (DMA).

Με τη χρήση του RDMA επιτυγχάνεται το κυρίαρχο ζητούμενο που είναι η επικοινωνία χωρίς τη δημιουργία αντιγράφων. Ο προσαρμογέας μπορεί να μεταφέρει δεδομένα απευθείας από και προς τη μνήμη της εφαρμογής, εξαλείφοντας την ανάγκη για αντίγραφα μεταξύ της μνήμης της εφαρμογής και των ενδιάμεσων χώρων αποθήκευσης του λειτουργικού συστήματος. Τέτοιες μεταφορές δεν απαιτούν κάποια ενέργεια από το CPU, τη χρήση της κρυφής μνήμης (cache) ή διακοπές ανά πακέτο στο CPU και οι μεταφορές μπορούν να συνεχίζουν παράλληλα με άλλες λειτουργίες του συστήματος. Όταν μια εφαρμογή κάνει ένα αίτημα για RDMA εγγραφή ή ανάγνωση, τα δεδομένα της εφαρμογής παραδίδονται απευθείας από και προς το δίκτυο, μειώνοντας το χρόνο αρχικής απόκρισης και κάνουν εφικτές τις μεταφορές μηνυμάτων σε μεγάλες ταχύτητες.

Το RDMA παρουσιάζει προβλήματα που αφορούν στο γεγονός ότι ο κόμβος στόχος δεν ενημερώνεται με την ολοκλήρωση του αιτήματος (μονομερής επικοινωνία). Ένας απλός τρόπος για την επίλυση του προβλήματος αυτού είναι η εγγραφή ενός byte στη μνήμη όταν ολοκληρωθεί η μεταφορά των δεδομένων. Το γεγονός αυτό περιορίζει τη χρήση του RDMA σε υπολογιστικά συστήματα υψηλής επίδοσης για δύο σημαντικούς λόγους:

- Ο στόχος διεξάγει συνεχώς δοκιμές, για να εντοπίσει την αλλαγή του συγκεκριμένου byte με αποτέλεσμα να καταναλώνει κύκλους από το CPU.
- Το απαιτούμενο byte – αποτύπωμα στη μνήμη αυξάνει γραμμικά το χρόνο αρχικής απόκρισης ανάλογα με τον αριθμό των κόμβων που στέλνουν ταυτόχρονα.

Σε αντίθετη περίπτωση, το CPU του κόμβου μπορεί να ενημερώνεται στο τέλος του μηνύματος, με την εμφάνιση μιας διακοπής. Στη σχεδίαση του SLURPoE το πρόβλημα αυτό αντιμετωπίζεται από τους χειρισμούς της εφαρμογής και του Πρωτοκόλλου στο Ανώτερο Επίπεδο (Upper Level Protocol – ULP). Το αποτέλεσμα είναι το πρωτόκολλο

να παρέχει όλους τους πιθανούς μηχανισμούς και το ULP να αντιμετωπίζει ανάλογα με τις ανάγκες του το συγκεκριμένο πρόβλημα.

Τα δίκτυα διασύνδεσης Αποστολής/Λήψης με μηδενικά αντίγραφα (zero–copy send/receive interconnects), όπως το Myrinet και το Quadrics εμφανίζουν διαφορετικού τύπου προβλήματα, όπως η υλοποίηση ασύγχρονων λειτουργιών. Η προγραμματιστική τους διεπαφή είναι παρόμοια με αυτή του MPI και στόχο έχει την ανταλλαγή μηνυμάτων.

Κάτι που δεν μπορούμε να αγνοήσουμε όσον αφορά στο πρωτόκολλο είναι η επιβάρυνση από την ανάγκη για τη δήλωση μνήμης. Τα πρωτόκολλα μηδενικής αντιγραφής στην πραγματικότητα πρέπει να εξασφαλίζουν ότι η εικονική περιοχή της μνήμης που εμπλέκεται στην επικοινωνία θα παραμείνει (α) στη φυσική μνήμη, (β) στην ίδια διεύθυνση, (γ) για τουλάχιστον το χρόνο που χρειάζεται για να γίνει η μεταφορά στο προσαρμογέα δικτύου. Για παράδειγμα, πρέπει να εξασφαλίσουμε ότι η περιοχή της εικονικής μνήμης δεν θα μεταφερθεί στο δίσκο (swap–out). Διαφορετικά, η μηχανή DMA μπορεί να χρησιμοποιήσει μη ενημερωμένα δεδομένα με αποτέλεσμα να αυξάνονται οι πιθανότητες για μεταφορά μη έγκυρων δεδομένων. Ο πιο συνηθισμένος τρόπος είναι να καθηλωθεί η περιοχή αυτή έτσι, ώστε να παραμείνει στη φυσική μνήμη. Αυτό ωστόσο αποτελεί μια παράπλευρη επιβάρυνση αφού η δήλωση αυτή της μνήμης έχει αρκετό κόστος. Αυτή η επιβάρυνση, με τη σειρά της, αυξάνει το χρόνο αρχικής απόκρισης ανάλογα με το μέγεθος των δεδομένων. Παρακάτω παρουσιάζονται λύσεις με τις οποίες το πρόβλημα μπορεί να εξαλειφθεί [10], [11]:

- Μεταφορά της δήλωσης της μνήμης *εκτός* του κρίσιμου μονοπατιού. Με αυτό τον τρόπο "κρύβεται" η αύξηση του χρόνου πρώτης απόκρισης.
- Με τεχνικές επαναχρησιμοποίησης ήδη δηλωμένων χώρων μνήμης. Με το τρόπο αυτό μειώνεται η επιβάρυνση για την εφαρμογή που κάνει την επικοινωνία και χρησιμοποιεί την ίδια περιοχή της μνήμης αρκετές φορές.
- Κάνοντας χρήση τεχνικών σωλήνωσης (pipeline) και επικάλυψης (overlapping) για τη δήλωση της μνήμης και τη μεταφορά δεδομένων όπως γίνεται στο Infiniband και το Myrinet.
- Με αλλαγές στο υποσύστημα διαχείρισης του λειτουργικού συστήματος. Η λύση αυτή, που προτείνεται και στο Quadrics, είναι λίγο πιο πολύπλοκη: προϋποθέτει μια προσθήκη στον πυρήνα, που ειδοποιεί το προσαρμογέα δικτύου, όταν πρόκειται να μετακινήσει τον εικονικό χώρο μνήμης, που έχει δεσμευτεί από την εφαρμογή. Έτσι όταν η μηχανή DMA του προσαρμογέα θέλει να μεταφέρει δεδομένα ζητά από τον πυρήνα, αν αυτό είναι αναγκαίο, να φτιάξει καινούργιες αντιστοιχίσεις φυσικών – εικονικών διευθύνσεων και να ενημερώσει τον πίνακα διευθύνσεων του προσαρμογέα για τις θέσεις μνήμης στις οποίες βρίσκονται τα δεδομένα που πρέπει να μεταφέρει.

Στην υλοποίηση που εξετάζουμε επιλέγουμε έναν συνδυασμό του πρώτου και του δεύτερου τρόπου. Η υλοποίηση είναι απλή ενώ έχει μικρές απαιτήσεις από τα χαρακτηριστικά που διαθέτει ο προσαρμογέας δικτύου. Ενδιαφέρον παρουσιάζει και ο τρόπος

που προτείνει το Myrinet και το Infiniband με τη χρονική επικάλυψη της δήλωσης της μνήμης με την αποστολή. Αξίζει όμως να σημειωθεί ότι ο τρόπος που χρησιμοποιούμε είναι αρκετά αποδοτικός στη περίπτωση που γίνεται επαναχρησιμοποίηση των χώρων που δηλώνονται κατά την αρχικοποίηση. Ο τέταρτος τρόπος, που προτείνει το Quadrics, παρουσιάζει την καλύτερη απόδοση, ωστόσο απαιτεί εξειδικευμένα κυκλώματα πάνω στον προσαρμογέα δικτύου καθώς και προσθήκες σε ένα πολύ βασικό κομμάτι του λειτουργικού συστήματος: στο υποσύστημα διαχείρισης της μνήμης.

Η αποδοχή του RDMA σε γενικές γραμμές περιορίζεται από την ανάγκη για την εγκατάσταση μιας εξειδικευμένης δικτυακής υποδομής. Σκοπός της παρούσας εργασίας είναι η χρήση ενός διαδεδομένου δικτύου διασύνδεσης προσδίδοντας του χαρακτηριστικά που θα το κάνουν αποδεκτό για χρήση σε συστήματα υψηλής απόδοσης, με εμφανή πλεονεκτήματα τόσο στην ομοιογένεια των υποδομών δικτύου όσο και στο κόστος.

3.2.2 10G Ethernet

Το Ethernet είναι η πιο διαδεδομένη τεχνολογία για τα τοπικά δίκτυα γενικής χρήσης. Το Gigabit Ethernet δεν κατάφερε να χρησιμοποιηθεί ως δίκτυο διασύνδεσης σε συστήματα υψηλής επίδοσης. Αυτό οφείλεται κυρίως στη χαμηλή απόδοση και στο χαμηλό εύρος ζώνης σε σύγκριση με το Infiniband και τις υπόλοιπες τεχνολογίες διασύνδεσης υψηλής επίδοσης. Στις περισσότερες περιπτώσεις επιδεικνύει υψηλότερο χρόνο πρώτης απόκρισης καθώς και υψηλό ποσοστό χρησιμοποίησης του CPU.

Η διάθεση μεταγωγέων 10GbE χαμηλού χρόνου αρχικής απόκρισης (low – latency 10GbE switches) καθώς και "έξυπνων" (προγραμματιζόμενων) προσαρμογέων δικτύου 10GbE δίνει τη δυνατότητα στο Ethernet να ενταχθεί στα δίκτυα διασύνδεση υψηλών επιδόσεων. Οι προσαρμογείς 10GbE επιφορτίζονται με το κόστος της επεξεργασίας των πρωτοκόλλων που έχουν να κάνουν με τις συστοιχίες υπολογιστών καθώς και των συστημάτων αποθήκευσης, αποφορτίζοντας το CPU των επιμέρους κόμβων. Αυτές οι επιπρόσθετες βελτιώσεις επιτρέπουν την πλήρη αξιοποίηση των ταχυτήτων που προσφέρουν οι γραμμές του 10GbE από τους εξυπηρετητές, ενώ παράλληλα μειώνουν το χρόνο αρχικής απόκρισης από άκρο σε άκρο κάτω από τα 10usec και τη χρησιμοποίηση του CPU σε λιγότερο από 10% για μεταφορές που αξιοποιούν το πλήρες εύρος ζώνης της γραμμής (line–rate bandwidth).

Σαν αποτέλεσμα η απόδοση του 10GbE από άκρο σε άκρο μπορεί να συγκριθεί με αυτή πιο ειδικευμένων δικτύων διασύνδεσης για συστοιχίες υπολογιστών, περιορίζοντας το μειονέκτημα της απόδοσης που υπήρχε στην υιοθέτηση του. Η μεταφορά του κόστους επεξεργασίας των πρωτοκόλλων επικοινωνίας των συστοιχιών υπολογιστών από το CPU σε έξυπνους προσαρμογείς δικτύου 10GbE μπορεί επίσης να συμβάλει σημαντικά στη συνολική ενεργειακή απόδοση των συστημάτων. Οι επεξεργαστές που φιλοξενούνται σε αυτούς τους προσαρμογείς είναι αποδοτικότεροι στη διαχείριση του φόρτου των πρωτοκόλλων.

Στην παρούσα εργασία, με βάση τα διαθέσιμα πρωτόκολλα, επιλέγεται η χρήση του 10Gb Ethernet για την υποδομή του δικτύου διασύνδεσης που θέλουμε να υλοποι-

ήσουμε και να αξιολογήσουμε. Αποφεύγουμε να χρησιμοποιήσουμε πρωτόκολλα σε ανώτερα επίπεδα στη στοίβα του TCP/IP για δύο λόγους:

1. Οι υπάρχουσες υλοποιήσεις των επιπέδων αυτών στον πυρήνα του λειτουργικού συστήματος απαιτούν πολλαπλά αντίγραφα τα οποία καθιστούν το χρόνο αρχικής απόκρισης απαγορευτικό για συστήματα υψηλής επίδοσης. Επίσης το εύρος διαύλου στο 10GbE υπερπληρώνει (saturates) το διαθέσιμο εύρος διαύλου της μνήμης αλλά και το ποσοστό χρησιμοποίησης του CPU.

2. Η χρήση ενός δικτύου διασύνδεσης αφορά σε τοπικά και μόνο δίκτυα. Το Ethernet μπορεί να καλύψει τις ανάγκες επικοινωνίας, οι οποίες δεν περιλαμβάνουν ούτε δρομολόγηση πακέτων, που θα έκανε απαραίτητο το επίπεδο δικτύου, ούτε συνδέσεις κακής ποιότητας από άκρο σε άκρο, που θα μπορούσαν να κάνουν απαραίτητη τη χρήση ενός επιπέδου μεταφοράς.

3.2.3 Πρωτόκολλο στο Επίπεδο Χρήστη (User Level Protocol – ULP)

Ο οδηγός του πρωτοκόλλου SLURPoE στην πλευρά του κόμβου σχεδιάζεται, κυρίως, στο χώρο χρήστη. Αυτή η σχεδιαστική επιλογή βασίζεται στην απαίτηση για πολύ μικρό χρόνο αρχικής απόκρισης. Λαμβάνοντας υπόψη όλες τις βιβλιογραφικές αναφορές κατά τη σχεδίαση, φαίνεται πως σε ταχύτητες της τάξης των 10G, το κόστος σε χρόνο για την είσοδο στον πυρήνα, επιβαρύνει το χρόνο αρχικής απόκρισης, που οφείλεται στο λογισμικό επικοινωνίας. Η μόνη επιβεβαίωση σε αυτή την υπόθεση δεν μπορεί παρά να είναι πειραματική. Η μεταφορά του κώδικα από τη βιβλιοθήκη υποστήριξης που είναι στο επίπεδο χρήστη μέσα στον οδηγό του πυρήνα δεν είναι δύσκολη. Μάλιστα θα μπορούσε να αποτελέσει και ένα επιπλέον χαρακτηριστικό του πρωτοκόλλου να μπορεί να δουλεύει και στο επίπεδο χρήστη και στο επίπεδο του πυρήνα σε μια παρόμοια λογική όπως και το uDAPL[12] και το kDAPL [13]. Οι δύο αυτές εργασίες αποτελούν δύο εναλλακτικές εκδοχές για το πρωτόκολλο DAPL (Direct Access Protocol Library) που χρησιμοποιείται στο Infiniband. Το uDAPL είναι στο χώρο χρήστη και το kDAPL στο χώρο του πυρήνα.

Η παράκαμψη του πυρήνα του λειτουργικού συστήματος είναι πολύ σημαντική για το κρίσιμο μονοπάτι. Οι λειτουργίες ελέγχου πρέπει να περνούν από τον πυρήνα ενώ οι λειτουργίες μεταφοράς δεδομένων γίνονται απευθείας από το χώρο χρήστη. Στη σχεδίαση του πρωτοκόλλου για μια RDMA μεταφορά υπάρχουν, κατά την αρχικοποίηση, λειτουργίες που εκτελούν κλήσεις συστήματος, όπως π.χ. είναι το άνοιγμα ενός άκρου και η δήλωση της μνήμης (memory registration). Στη συνέχεια οι εντολές για την πραγματοποίηση μιας RDMA μεταφοράς απευθύνονται στον προσαρμογέα, με αποτέλεσμα να παρακάμπτεται ο πυρήνας.

3.3 Υποστήριξη στην πλευρά του κόμβου

3.3.1 Οδηγός του πυρήνα του λειτουργικού συστήματος

Με δεδομένη την επιλογή ενός πρωτοκόλλου στο χώρο χρήστη, η συμμετοχή του πυρήνα στο σχεδιασμό περιορίζεται στο ελάχιστο δυνατό, χωρίς όμως να είναι εφικτή η πλήρης απεμπλοκή του. Υπάρχουν κυρίαρχα δυο απαιτήσεις που καθιστούν αναγκαίες τις κλήσεις προς το σύστημα και άρα την εμπλοκή του πυρήνα: οι αποφάσεις που αφορούν στην ασφάλεια του συστήματος καθώς και οι περιορισμοί από το σύστημα εικονικής μνήμης του λειτουργικού συστήματος.

Στόχος του σχεδιασμού είναι η υλοποίηση ενός συστήματος επικοινωνίας μεταξύ μιας εφαρμογής και του οδηγού στην πλευρά του προσαρμογέα. Το σύστημα αυτό θα πρέπει να είναι απομονωμένο και να μην επιτρέπει την παρεμβολή από άλλες εφαρμογές. Επίσης ο πυρήνας θα πρέπει να εκχωρεί στην εφαρμογή όλα εκείνα τα δικαιώματα που είναι απαραίτητα για τη λειτουργία του πρωτοκόλλου χωρίς την περαιτέρω εμπλοκή του και χωρίς να θυσιάζει την ασφάλεια του συστήματος. Σημαντικό αποτέλεσμα του σχεδιασμού με τον τρόπο αυτό είναι ότι η εμπλοκή του πυρήνα περιορίζεται στην αρχικοποίηση του πρωτοκόλλου, δηλαδή εκτός του κρίσιμου μονοπατιού σε μία RDMA εγγραφή/ανάγνωση. Στις επόμενες παραγράφους περιγράφουμε τις απαιτήσεις και τη σχεδίαση ενός module του λειτουργικού συστήματος του κόμβου.

Αρχικοποίηση δομών του πυρήνα και δέσμευση διακοπών

Στο πρώτο στάδιο ο οδηγός αρχικοποιεί δομές που είναι απαραίτητες για τη λειτουργία του. Οι δομές αυτές περιέχουν πληροφορίες για κάθε ανοιχτό άκρο που έχει ζητήσει πρόσβαση στον προσαρμογέα, καθώς και πεδία που αφορούν στους χώρους μνήμης που έχουν δεσμεύσει.

Δημιουργείται επίσης μια διεπαφή επικοινωνίας, την οποία χρησιμοποιούν οι εφαρμογές κατά τις κλήσεις συστήματος προς τον οδηγό του SLURPoE. Επίσης, μέσω του πυρήνα εγκαθίσταται η υποστήριξη για διακοπές. Οι διακοπές αυτές είναι χρήσιμες στο σύστημα και αποτελούν μια ακόμα επιλογή σε πολλές λειτουργίες έτσι, ώστε να αποφευχθεί η άσκοπη χρησιμοποίηση του CPU, που επιβάλλει η εναλλακτική των δοκιμών (polling). Παράδειγμα χρήσης των διακοπών στην πλευρά του συστήματος είναι η ενημέρωση για καινούργιο μήνυμα που έχει προέλευση μια εφαρμογή.

Εκχώρηση δικαιωμάτων πρόσβασης στο χώρο μνήμης του προσαρμογέα δικτύου από την εφαρμογή στο χώρο χρήστη

Η επικοινωνία της εφαρμογής που κάνει χρήση του πρωτοκόλλου με τον προσαρμογέα δικτύου γίνεται με την εγγραφή και την ανάγνωση από το χώρο μνήμης του προσαρμογέα. Στο εικονικό σύστημα μνήμης του λειτουργικού, για λόγους ασφαλείας, μια εφαρμογή δεν μπορεί να έχει πρόσβαση στη μνήμη μιας περιφερειακής συσκευής.

Μια εφαρμογή μπορεί να απόκτηση δικαιώματα πρόσβασης στη μνήμη του προσαρμογέα αν το αίτημα αυτό επιτραπεί από τον πυρήνα.

Ο πυρήνας κάνει διαθέσιμη μια κλήση συστήματος μέσω της διεπαφής επικοινωνίας με την εφαρμογή, με την οποία εκχωρεί δικαιώματα χρήσης προκαθορισμένων περιοχών της μνήμης του προσαρμογέα στην εφαρμογή που την καλεί. Έτσι, δημιουργούνται οι αντιστοιχίσεις μεταξύ φυσικών διευθύνσεων της μνήμης του προσαρμογέα και μιας περιοχής του εικονικού χώρου μνήμης της εφαρμογής. Η εφαρμογή έπειτα από αυτή την κλήση συστήματος αντιμετωπίζει τη μνήμη στο προσαρμογέα ακριβώς με τον ίδιο τρόπο που κάνει εγγραφές και αναγνώσεις στον υπόλοιπο εικονικό χώρο διευθύνσεων που της αποδίδεται.

Δήλωση της μνήμης και ενημέρωση του προσαρμογέα δικτύου

Η απαίτηση του σχεδιασμού για επικοινωνία χωρίς αντίγραφα σε συνδυασμό με τη χρήση μεταφορών με μηχανές DMA, απαιτεί ειδικό χειρισμό από τον πυρήνα του λειτουργικού συστήματος. Αυτό που εξασφαλίζεται είναι ότι ο χώρος αποστολής ή λήψης δεδομένων δε θα υποστεί καμία από τις παρενέργειες που προκαλούνται από το υποσύστημα διαχείρισης μνήμης του πυρήνα (π.χ. swapping). Οι χώροι της μνήμης, που η εφαρμογή επιλέγει να αποθηκεύσει δεδομένα προς μεταφορά, είναι αντικείμενα ειδικής αντιμετώπισης, αφού προορίζονται για DMA μεταφορές. Οι σελίδες που τους συνθέτουν πρέπει να καθηλωθούν.

Η εφαρμογή ενημερώνει τον πυρήνα μέσω μια κλήσης συστήματος, για τα στοιχεία της προσωρινής μνήμης που θα χρησιμοποιήσει, την εικονική διεύθυνση και το μέγεθος της. Στη συνέχεια εξασφαλίζεται από τον πυρήνα ότι η DMA λειτουργία δε θα επηρεάσει την εύρυθμη λειτουργία του συστήματος.

Παράλληλα όμως, δημιουργείται και μια δεύτερη απαίτηση από τον πυρήνα, που οφείλεται στο γεγονός ότι οι λειτουργίες DMA γίνονται από τον οδηγό του προσαρμογέα: ο οδηγός αυτός δεν είναι σε θέση να κάνει τη μετάφραση από εικονικές σε φυσικές διευθύνσεις για σελίδες του κόμβου. Η ανάγκη αντιστοίχισης εικονικών σε φυσικές σελίδες καλύπτεται με την παράλληλη αντιγραφή της αντιστοίχισης αυτής από το σύστημα σελιδοποίησης του λειτουργικού συστήματος σε μια ειδική δομή στη μνήμη του προσαρμογέα.

3.3.2 Βιβλιοθήκη υποστήριξης

Η βιβλιοθήκη υποστήριξης αποτελείται από συναρτήσεις και ορισμούς τύπου δομών, που έχουν το ρόλο μιας προγραμματιστικής διεπαφής. Η διεπαφή αυτή χρησιμοποιείται από εφαρμογές που επικοινωνούν με το πρωτόκολλο SLURPoE. Με αυτό τον τρόπο οι εφαρμογές μπορούν να χρησιμοποιούν το πρωτόκολλο SLURPoE, χωρίς να γνωρίζουν τις λεπτομέρειες υλοποίησης, ενώ επιτυγχάνεται η επαναχρησιμοποίηση κώδικα. Κατά κανόνα ο κώδικας αυτός είναι και πιο ασφαλής.

Ο τρόπος επικοινωνίας μιας εφαρμογής στο χώρο χρήστη με τον πυρήνα γίνεται μέσω της διεπαφής επικοινωνίας εφαρμογής–πυρήνα, η οποία δημιουργείται κατά την αρχικοποίηση του οδηγού. Η διεπαφή αυτή χρησιμοποιείται και από τη σχεδίαση της βιβλιοθήκης υποστήριξης για την υλοποίηση των βασικών λειτουργιών που απαιτούν την αλληλεπίδραση με τον πυρήνα.

Άνοιγμα/κλείσιμο ενός άκρου

void slurpoe_open_endpoint(**struct** slurpoe_endpoint *endpoint);
void slurpoe_close_endpoint(**struct** slurpoe_endpoint *endpoint);

Για να χρησιμοποιήσει μια εφαρμογή οποιαδήποτε λειτουργία του πρωτοκόλλου SLURPoE είναι απαραίτητο να ανοίξει ένα άκρο. Με την ενέργεια αυτή δημιουργούνται τόσο στο module του πυρήνα όσο και στον οδηγό του προσαρμογέα δομές που είναι απαραίτητες για τη χρήση του SLURPoE.

Οι δομές αυτές χαρακτηρίζονται από το pid της εφαρμογής, το οποίο συχνά ελέγχεται με στόχο να διασφαλιστεί ότι καμία άλλη εφαρμογή δεν μπορεί να χρησιμοποιήσει δομές άκρου που δεν της ανήκουν τόσο στον πυρήνα όσο και στον οδηγό του προσαρμογέα. Με τον τρόπο αυτό εξασφαλίζουμε και την απαίτηση για απομονωμένη επικοινωνία για κάθε ξεχωριστή εφαρμογή και αποτρέπουμε ανεπιθύμητες παραβιάσεις.

Το άκρο χαρακτηρίζεται επίσης από ένα μοναδικό αναγνωριστικό που μπορεί να ζητείται από την εφαρμογή είτε να αποδίδεται αυτόματα από τον πυρήνα. Η λειτουργικότητα που συνοδεύει την απόδοση ενός τέτοιου αναγνωριστικού έχει αρκετές ομοιότητες με τις πόρτες του πρωτοκόλλου στο επίπεδο μεταφοράς TCP. Με βάση αυτό το αναγνωριστικό δηλαδή, μπορούν οι διάφορες αιτήσεις από τις εφαρμογές να διαχωρίζονται και να έχουν συγκεκριμένη αναφορά κατά τη λήψη τους από τον προσαρμογέα δικτύου. Έτσι τα μηνύματα του πρωτοκόλλου έχουν διευθύνσεις προορισμού και προέλευσης, που χαρακτηρίζονται από τη διεύθυνση MAC και το αναγνωριστικό του αντίστοιχου άκρου.

Το άνοιγμα ενός άκρου συνοδεύεται και από την εκχώρηση δικαιωμάτων πρόσβασης σε όλους τους απαραίτητους χώρους της μνήμης του προσαρμογέα που υλοποιούν το μηχανισμό μηνυμάτων ελέγχου.

Στο τέλος και αφού ολοκληρωθούν όλες οι λειτουργίες μια εφαρμογής που σχετίζονται με το πρωτόκολλο SLURPoE, το άκρο πρέπει να κλείσει. Οι δομές στο module του κόμβου και του προσαρμογέα ελευθερώνονται και το άκρο μένει διαθέσιμο προς χρήση από κάποια άλλη εφαρμογή. Μετά το κλείσιμο ενός άκρου δεν μπορούν να αποσταλούν μηνύματα με αυτό το αναγνωριστικό του άκρου προέλευσης, ενώ και τα μηνύματα που είχαν προορισμό το άκρο αυτό στον ίδιο κόμβο απορρίπτονται.

Δήλωση/διαγραφή δεσμευμένων χώρων στη μνήμη

int slurpoe_register_user_region(**struct** slurpoe_endpoint *endpoint, **void** *
    buf, **size_t** size);

int slurpoe_deregister_user_region(**struct** slurpoe_endpoint *endpoint, **void** * buf, **size_t** size);

Μια εφαρμογή πρέπει να εγγράψει ένα χώρο μνήμης για να τον χρησιμοποιήσει σε μια RDMA εγγραφή/ανάγνωση. Ο χώρος που δηλώνεται από μια κλήση της συνάρτησης αυτής έχει προηγουμένως δεσμευτεί από την εφαρμογή, μέσω του λειτουργικού συστήματος. Η συνάρτηση δήλωσης εκτελεί αυτή ακριβώς τη λειτουργία για την εφαρμογή. Οι χώροι που δηλώνονται συνδέονται με το άκρο στο οποίο ανήκουν και δεν μπορούν να πάρουν μέρος σε μια αποστολή/λήψη ενός άλλου άκρου. Μόνο μετά τη δήλωση ενός χώρου που έχει δεσμευτεί από την εφαρμογή μπορεί να κληθεί η συνάρτηση αποστολής/λήψης ενός μηνύματος. Σε αντίθετη περίπτωση και η λειτουργία θα αποτύχει.

Μετά το τέλος της επικοινωνίας, ο χώρος που είχε δηλωθεί μπορεί να αποδεσμευτεί. Τα δεδομένα που έχει παραμένουν ανέπαφα προς χρήση από την εφαρμογή μέχρι ο χώρος αυτός να απελευθερωθεί, με χρήση των συναρτήσεων δυναμικής διαχείρισης μνήμης του λειτουργικού.

Συνάρτηση RDMA ανάγνωσης/εγγραφής

void slurpoe_rdma_read(**struct** slurpoe_endpoint *endpoint, **void** *buf, **size_t** size, **unsigned** *src_endpoint_id, **char** src_addr[6], **void** *src_buf);
void slurpoe_rdma_write(**struct** slurpoe_endpoint *endpoint, **void** *buf, **size_t** size, **unsigned int** dst_endpoint_id, **char** dst_addr[6], **void** *dst_buf);

Αφού γίνει η δήλωση ενός χώρου μνήμης μπορεί πλέον μια εφαρμογή να εκτελέσει μια εντολή RDMA εγγραφής/ανάγνωσης. Οι συναρτήσεις της βιβλιοθήκης με τα κατάλληλα ορίσματα προσδίδουν τη λειτουργικότητα αυτή στην εφαρμογή. Σημειώνεται στο σημείο αυτό ότι οι λειτουργίες μιας RDMA εγγραφής/ανάγνωσης είναι μονομερείς. Και στα δυο άκρα γίνεται η απαραίτητη δήλωσητης μνήμης και από εκεί και πέρα το κάθε άκρο μπορεί να γράφει και να διαβάζει προς και από αυτά.

3.4 Υποστήριξη στην πλευρά του προσαρμογέα δικτύου

Ο προσαρμογέας δικτύου αναλαμβάνει το μεγαλύτερο μέρος των υπολογιστικών απαιτήσεων του πρωτοκόλλου. Σκοπός είναι η μείωση στο ελάχιστο του ποσοστού χρησιμοποίησής του CPU του συστήματος. Οι λειτουργίες που υλοποιεί το λογισμικό του προσαρμογέα αφορούν σε δύο τύπους, αυτές ανάμεσα στον προσαρμογέα και τον κόμβο και αυτές ανάμεσα στον προσαρμογέα και τον προσαρμογέα πακέτων.

3.4.1 Μηχανισμός μηνυμάτων ελέγχου

Ο μηχανισμός μηνυμάτων υλοποιείται πάνω στη μνήμη του προσαρμογέα. Κατά την αρχικοποίηση του οδηγού του λειτουργικού συστήματος δημιουργούνται: ένας χώ-

ρος που θα γράφονται και θα διαβάζονται μηνύματα και μια ειδική δομή για τη διαχείρισή τους. Παράλληλα, τόσο στον κόμβο όσο και στο προσαρμογέα δεσμεύουμε μια διακοπή με την οποία η κάθε πλευρά θα ενημερώνει την άλλη για ένα νέο μήνυμα.

Ο χώρος των μηνυμάτων είναι μια συνεχόμενη περιοχή στη φυσική μνήμη του προσαρμογέα με προκαθορισμένο μέγεθος. Το μέγεθος αυτό καθορίζεται από τα διαθέσιμα μηνύματα που μπορούν να υπάρχουν προς εγγραφή και ανάγνωση ταυτόχρονα. Τη διαχείριση των μηνυμάτων την αναλαμβάνει ένας μηχανισμός ουράς. Η κατεύθυνση των μηνυμάτων έχει αναφορά τον προσαρμογέα, έτσι εισερχόμενα είναι τα μηνύματα που αποστέλλονται από τον κόμβο προς τον προσαρμογέα ενώ εξερχόμενα είναι τα μηνύματα που ο προσαρμογέας στέλνει προς τον κόμβο.

Τη διαχείριση του χώρου αυτού αναλαμβάνει ένας μηχανισμός ουράς. Αυτός ο μηχανισμός στη ουσία περιλαμβάνει τέσσερις ουρές:

- των εισερχόμενων ελεύθερων μηνυμάτων (incoming free),
- των εισερχομένων απεσταλμένων μηνυμάτων (incoming post),
- των εξερχομένων απεσταλμένων μηνυμάτων (outgoing post),
- των εξερχομένων ελεύθερων μηνυμάτων (outgoing free).

Ο μηχανισμός διαχείρισης των μηνυμάτων χρησιμοποιεί τις ουρές αυτές για να υλοποιήσει την επικοινωνία ελέγχου μεταξύ του προσαρμογέα και του κόμβου. Στη συνέχεια περιγράφουμε τις ενέργειες που απαιτούνται από τη σχεδίαση του.

Όταν μια εφαρμογή θέλει να αποστείλει ένα μήνυμα, διαβάζει από την κορυφή της ουράς εισερχόμενων ελεύθερων μηνυμάτων μια διαθέσιμη θέση μνήμης από το χώρο των μηνυμάτων και η διεύθυνση αυτή αφαιρείται από την ουρά. Έπειτα στη διεύθυνση αυτή γράφει το μήνυμα που θέλει να στείλει και γράφει τη διεύθυνση στην κορυφή των εισερχόμενων απεσταλμένων μηνυμάτων. Η διακοπή που ενημερώνει το προσαρμογέα για ένα καινούργιο μήνυμα ενεργοποιείται και ο προσαρμογέας διαβάζει τη διεύθυνση του απεσταλμένου μηνύματος από την κορυφή της ουράς εισερχομένων απεσταλμένων μηνυμάτων. Έπειτα αφαιρεί από την ουρά τη διεύθυνση αυτή και διαβάζει το αντίστοιχο μήνυμα. Τέλος αφού διαβάσει το μήνυμα, τοποθετεί τη διεύθυνση του μηνύματος στο τέλος της ουράς των εισερχόμενων ελεύθερων μηνυμάτων.

Αντίστοιχα όταν ο προσαρμογέας θέλει να αποστείλει ένα μήνυμα, διαβάζει από την κορυφή της ουράς εξερχόμενων ελεύθερων μηνυμάτων μια διαθέσιμη θέση μνήμης από το χώρο των μηνυμάτων και η διεύθυνση αυτή αφαιρείται από την ουρά. Έπειτα στη διεύθυνση αυτή γράφει το μήνυμα που θέλει να στείλει και γράφει τη διεύθυνση στην κορυφή των εξερχόμενων απεσταλμένων μηνυμάτων. Η διακοπή που ενημερώνει τον κόμβο για ένα καινούργιο μήνυμα ενεργοποιείται και η εφαρμογή διαβάζει τη διεύθυνση του απεσταλμένου μηνύματος από την κορυφή της ουράς εξερχομένων απεσταλμένων μηνυμάτων. Έπειτα αφαιρείται από την ουρά η διεύθυνση αυτή και η εφαρμογή διαβάζει το αντίστοιχο μήνυμα. Τέλος αφού διαβάσει το μήνυμα, τοποθετείται η διεύθυνση του μηνύματος αυτού πίσω στο τέλος της ουράς των εξερχόμενων ελεύθερων μηνυμάτων.

3.4.2 Μηχανισμός αντιστοίχισης των εικονικών διευθύνσεων του λειτουργικού συστήματος του κόμβου

Ο μηχανισμός αυτός είναι πολύ σημαντικός για το πρωτόκολλο λόγω της σχεδιαστικής επιλογής για ένα πρωτόκολλο στο χώρο χρήστη. Η απεμπλοκή του πυρήνα από το κρίσιμο μονοπάτι απαιτεί από τον προσαρμογέα να έχει τη δυνατότητα να αναγνωρίζει εικονικές διευθύνσεις τις οποίες δέχεται με μηνύματα από την εφαρμογή. Για παράδειγμα κατά την RDMA εγγραφή, μια εφαρμογή θα αποστείλει ένα μήνυμα που θα περιγράφει το χώρο που θα πρέπει να αποστείλει ο προσαρμογέας. Η περιγραφή αυτή θα αφορά σε εικονικές διευθύνσεις, εφόσον δε μεσολαβεί ο πυρήνας στην επικοινωνία με το προσαρμογέα και γίνεται απευθείας από την εφαρμογή. Η αντιστοίχιση όμως σε φυσικές διευθύνσεις είναι απαραίτητη για τον προγραμματισμό της DMA μηχανής.

Στο σχεδιασμό το πρόβλημα αυτό λύνεται με τη διατήρηση ενός πίνακα αντιστοίχισης των προς αποστολή και λήψη χώρων μνήμης του κόμβου. Αυτός ο πίνακας που δημιουργείται και χρησιμοποιείται ξεχωριστά για κάθε άκρο κατά την αρχικοποίηση του, πρέπει να αποτελεί και στοιχείο για να μπορεί ο προσαρμογέας να ελέγξει την εγκυρότητα του χώρου που ζητείται να εγγραφεί ή να διαβαστεί και έπειτα να μπορεί να δίνει την αντιστοίχιση σε φυσική μνήμη. Ο λόγος που δε διατηρούμε έναν κεντρικό πίνακα αντιστοίχισης αλλά ένα για κάθε άκρο, είναι επειδή οι εικονικές διευθύνσεις είναι μοναδικές μόνο στα πλαίσια μιας εφαρμογής και όχι καθολικά στο σύστημα. Με τον τρόπο αυτό εξασφαλίζουμε την απαίτηση της απομονωμένης και ασφαλούς επικοινωνίας στο επίπεδο αυτό.

3.4.3 Προγραμματισμός της μηχανής DMA

Μια πολύ σημαντική παράμετρος της σχεδίασης του πρωτοκόλλου είναι η όσο το δυνατόν μικρότερη χρησιμοποίηση του CPU. Η απαίτηση αυτή κάνει απαραίτητη τη χρήση μηχανών DMA που θα εκτελούν τις μεταφορές από τη μνήμη του κόμβου προς τη μνήμη του προσαρμογέα και αντίστροφα. Η χρήση των μεθόδων που περιγράφηκαν στην παράγραφο 3.3.1 για την καθήλωση της μνήμης εξασφαλίζει την εγκυρότητα των DMA μεταφορών.

Με βάση το σχεδιασμό ο προσαρμογέας διαθέτει τουλάχιστον μια μηχανή DMA. Σε οποιαδήποτε λειτουργία του πρωτοκόλλου που περιλαμβάνει μεταφορές από και προς τη μνήμη του κόμβου ο προσαρμογέας χρησιμοποιεί μια μηχανή DMA. Σε ειδικούς καταχωρητές που προγραμματίζουν την μηχανή DMA εγγράφονται:

- η φυσική διεύθυνση του χώρου στον κόμβο που συμμετέχει στη μεταφορά και προκύπτει ως αποτέλεσμα του μηχανισμού αντιστοίχισης των εικονικών διευθύνσεων,
- η φυσική διεύθυνση ενός προσωρινού χώρου του προσαρμογέα που δεσμεύεται για την ενδιάμεση αποθήκευση,

- το μέγεθος των δεδομένων που θα μεταφερθούν.

Τέλος η μηχανή θα ενεργοποιηθεί για να εκτελεστεί η μεταφορά με την εγγραφή ενός ειδικού bit σε κατάλληλο καταχωρητή. Το τέλος της μεταφοράς σηματοδοτείται με χρήση ενός μηχανισμού δοκιμών (polling) στον προσαρμογέα.

3.4.4 Προγραμματισμός του προσαρμογέα πακέτων

Ο προσαρμογέας πακέτων είναι η συσκευή που μεταφέρει τα πακέτα Ethernet στο καλώδιο. Ο προσαρμογέας έχει αποκλειστικό έλεγχο της συγκεκριμένης συσκευής την οποία προγραμματίζει έτσι, ώστε να μεταφέρει από και προς το καλώδιο πακέτα που έχουν προορισμό ή προέλευση άλλες εφαρμογές που κάνουν χρήση του πρωτοκόλλου.

Οι απαιτήσεις προδιαγραφών του σχεδιασμού απαιτούν από τον οδηγό του προσαρμογέα τέτοιο χειρισμό που δε δημιουργεί αντίγραφα στην μνήμη του. Έτσι, τα δεδομένα, από τη μνήμη του κόμβου, μεταφέρονται στη μνήμη του προσαρμογέα με τη DMA μηχανή του τελευταίου. Από εκεί, η μηχανή DMA του προσαρμογέα πακέτων μεταφέρει τα δεδομένα στο καλώδιο. Η ίδια απαίτηση υπάρχει και για πακέτα που έρχονται από το δίκτυο και έχουν προορισμό τη μνήμη του κόμβου. Παρόμοιοι χειρισμοί γίνονται και σε αυτή τη περίπτωση.

3.5 Μηνύματα ελέγχου μεταξύ του κόμβου και του προσαρμογέα δικτύου

Χειραψία (MSG_ID_HANDSHAKE).
: Το μήνυμα αυτό αρχικά αποστέλεται από τον πυρήνα του συστήματος κατά τη φόρτωση του. Ο προσαρμογέας στη λήψη του αρχικοποιεί όλες τις απαραίτητες δομές που χρησιμοποιεί και στέλνει ένα μήνυμα ίδιου τύπου πίσω στο σύστημα με πληροφορίες για την έκδοση του λογισμικού της. Με τον τρόπο αυτό γίνεται και ένας στοιχειώδης έλεγχος συμβατότητας.

Άνοιγμα ενός άκρου (MSG_ID_ENDPOINT_OPEN).
: Με το μήνυμα αυτό το σύστημα ενημερώνει τον προσαρμογέα για το άνοιγμα ενός καινούργιου άκρου. Ο προσαρμογέας δεσμεύει χώρο για τη δομή που θα διατηρεί πληροφορίες για το ανοιχτό άκρο. Σε αυτή συμπεριλαμβάνεται ο πίνακας σελίδων που αφορά στις αντιστοιχίσεις εικονικών–φυσικών διευθύνσεων του κόμβου.

Επιβεβαίωση ανοίγματος άκρου (MSG_ID_ENDPOINT_READY).
: Με το μήνυμα αυτό ο προσαρμογέας επιβεβαιώνει ότι το ζητούμενο άκρο είναι διαθέσιμο και η δέσμευση και η αρχικοποίηση των δομών που το συνοδεύουν στον προσαρμογέα έχει ολοκληρωθεί. Το μήνυμα αυτό, το οποίο έχει προορισμό τον κόμβο, συνοδεύει η διεύθυνση του πίνακα αντιστοίχισης για αυτό το άκρο.

Αποτυχία ανοίγματος άκρου (MSG_ID_ENDPOINT_OPEN_FAIL).
Ήδη ανοιχτό άκρο (MSG_ID_ENDPOINT_ALREADY_OPEN).
Με το μήνυμα αυτό η κάρτα ενημερώνει τον πυρήνα του συστήματος είτε ότι το ζητούμενο άκρο δεν είναι διαθέσιμο είτε κάποιο μη αναμενόμενο σφάλμα έχει συμβεί με αποτέλεσμα την αποτυχία ανοίγματός του.

Κλείσιμο ενός άκρου (MSG_ID_ENDPOINT_CLOSE)
Αντίστοιχα με το άνοιγμα, το μήνυμα αυτό κλείνει ένα άκρο, δηλαδή ελευθερώνει το χώρο που έχει δεσμευτεί και κάνει διαθέσιμες τις δομές σε ένα καινούργιο άνοιγμα του άκρου με το ίδιο αναγνωριστικό.

RDMA Εγγραφή (MSG_ID_RDMA_WRITE).
RDMA Ανάγνωση (MSG_ID_RDMA_READ).
Το μήνυμα αυτό ενημερώνει τον προσαρμογέα για μια RDMA Εγγραφή/Ανάγνωση. Μαζί με αυτό συμπεριλαμβάνονται και στοιχεία που είναι απαραίτητα στη μεταφορά:

- η διεύθυνση MAC του καταναλωτή/της πηγής δεδομένων,
- το αναγνωριστικό του άκρου του καταναλωτή/της πηγής δεδομένων,
- η εικονική διεύθυνση του χώρου του καταναλωτή/της πηγής δεδομένων που θα γίνει εγγραφή/ανάγνωση
- το αναγνωριστικό του άκρου της πηγής/του καταναλωτή δεδομένων,
- η εικονική διεύθυνση του χώρου των δεδομένων στην πηγή/στον καταναλωτή που θα γίνει αποστολή/λήψη
- το μέγεθος των δεδομένων που θα αποσταλούν/ληφθούν.

3.6 Μηνύματα του πρωτοκόλλου στο δίκτυο διασύνδεσης

Στο καλώδιο τα μηνύματα μεταφέρονται μέσα σε πακέτα που κατακερματίζονται σύμφωνα με το μέγιστο μέγεθος που ορίζεται από το MTU του δικτύου. Στο μέγεθος αυτό περιλαμβάνεται όλο το πακέτο Ethernet μαζί με την κεφαλίδα του. Για τη λειτουργία του πρωτοκόλλου εκμεταλλευόμαστε την κεφαλίδα του Ethernet για να ορίσουμε τον κόμβο – προορισμό και τον κόμβο – προέλευση με βάση τις διευθύνσεις MAC. Στο σώμα του πακέτου Ethernet ενθυλακώνονται τα πακέτα του πρωτοκόλλου SLURPoE.

Τα πακέτα του πρωτοκόλλου SLURPoE διαθέτουν και αυτά κεφαλίδες που μεταφέρουν απαραίτητες παραμέτρους για το χειρισμό των δεδομένων. Ο τύπος ενός πακέτου καθορίζεται από τα πρώτα 8bits. Στη σχεδίαση του πρωτοκόλλου που έχει ως στόχο τη μελέτη μόνο του RDMA μονοπατιού, υπάρχουν δύο ήδη πακέτων. Το πρώτο πεδίο θα πρέπει να έχει μέγεθος 8bits και να φέρει ένα από τα αναγνωριστικά που θα υποδεικνύουν ποιο είναι το είδος του.

Το πρώτο από τα δύο είδη φαίνεται στο σχήμα 3.2 και αφορά στην πραγματοποίηση μιας RDMA εγγραφής. Στην κεφαλίδα αυτή το πρώτο πεδίο είναι αυτό που προσδίδει στο πακέτο τον τύπο της RDMA εγγραφής (SLURPOE_PKT_RDMA_¬ WRITE). Αμέσως μετά βρίσκεται ένα πεδίο 8bits του αναγνωριστικού του άκρου του προορισμού και ένα ίδιο του αντίστοιχου προέλευσης. Ακολουθεί ένα πεδίο 8bits με τον αύξοντα αριθμό του πακέτου που υποδεικνύει τη σειρά του σε ένα κατακερματισμένο RDMA μήνυμα, ένα πεδίο 16bits με το μέγεθος του πακέτου και ένα πεδίο 64bits με την εικονική διεύθυνση προορισμού. Το πακέτο τελειώνει με τα δεδομένα του μηνύματος, τα οποία πρέπει να έχουν κατακερματιστεί, ώστε να χωράνε στο προβλεπόμενο μέγεθος πακέτου το οποίο περιορίζεται από το MTU του Ethernet. Το δεύτερο φαίνεται στο σχήμα 3.3 και αφορά στην πραγματοποίηση μιας RDMA

pkt_type (0-7)	dst_endpoint (8-15)	src_endpoint (16-23)	seqnum (24-31)
data size (32-63)			
destination virtual address (64-95)			
destination virtual address (96-127)			
payload (40-9000 byte)			

Σχήμα 3.2: Η μορφή ενός πακέτου RDMA εγγραφής.

ανάγνωσης. Στην κεφαλίδα αυτή το πρώτο πεδίο είναι αυτό που προσδίδει στο πακέτο τον τύπο της RDMA εγγραφής (SLURPOE_PKT_RDMA_READ). Αμέσως μετά βρίσκεται ένα πεδίο 8bits του αναγνωριστικού του άκρου του προορισμού και στη συνέχεια το αντίστοιχο του άκρου της προέλευσης. Ακολουθεί ένα πεδίο 8bits που μένει αχρησιμοποίητο για τη δυνατότητα επέκτασης, ένα πεδίο 32bits με το μέγεθος του μηνύματος που ζητείται, ένα πεδίο 64bits με την εικονική διεύθυνση του μηνύματος που ζητείται και ένα ακόμα πεδίο 64bits με την εικονική διεύθυνση του χώρου που θα δηλωθεί. Ο προσαρμογέας του προορισμού, που γίνεται αποδέκτης ενός τέτοιου μηνύματος απαντά με το αντίστοιχο μήνυμα RDMA εγγραφής.

pkt_type (0-7)	dst_endpoint (8-15)	src_endpoint (16-23)	pad
data size (32-63)			
destination virtual address (64-95)			
destination virtual address (96-127)			
source virtual address (128-159)			
source virtual address (160-191)			

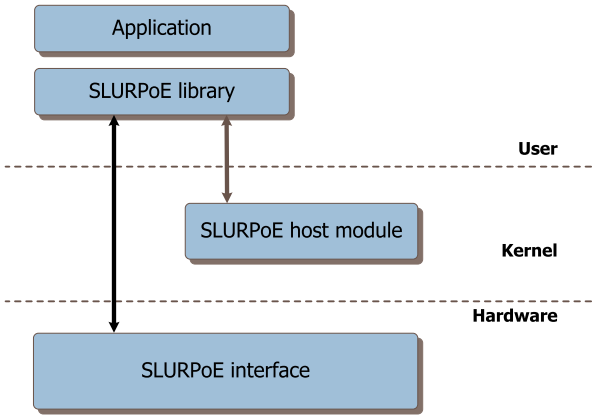
Σχήμα 3.3: Η μορφή ενός πακέτου RDMA ανάγνωσης.

Κεφάλαιο 4

Υλοποίηση του πρωτοκόλλου SLURPoE

Στο κεφάλαιο αυτό περιγράφεται η υλοποίηση του πρωτοκόλλου SLURPoE. Η υλοποίηση έχει ως στόχο την πειραματική αξιολόγηση της σχεδίασης, μέσα από τη διεξαγωγή μετρήσεων που αφορούν στα σημαντικά μεγέθη που χαρακτηρίζουν την απόδοση ενός δικτύου διασύνδεσης σε συστήματα υψηλής επίδοσης.

Το περιβάλλον ανάπτυξης είναι το λειτουργικό σύστημα Linux. Το Linux χρησιμοποιείται τόσο στον κόμβο ως λειτουργικό σύστημα όσο και στο προσαρμογέα ως υλικολογισμικό. Ο οδηγός του πρωτοκόλλου υλοποιείται σύμφωνα με το μοντέλο χωρισμένου οδηγού που έχει προδιαγραφεί.

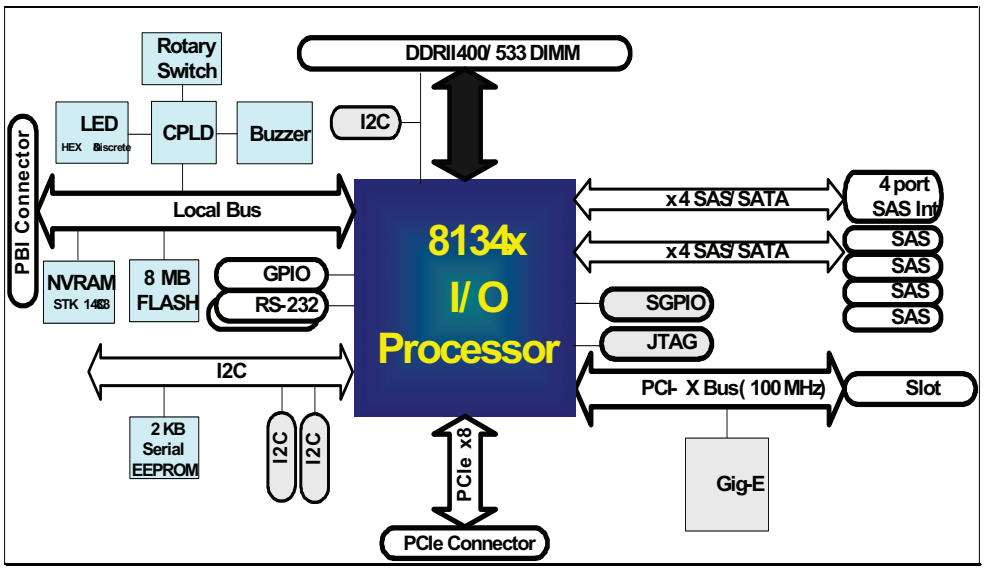


Σχήμα 4.1: Η αρχιτεκτονική του SLURPoE

4.1 Οι απαιτήσεις του προσαρμογέα δικτύου

Η υλοποίηση του προσαρμογέα δικτύου για το πρωτόκολλο SLURPoE έγινε με χρήση ενός προγραμματιζόμενου προσαρμογέα E/E Intel IQ81340SC[14]. Όπως φαίνεται και από τα σχήματα 4.2, 4.3 η κάρτα αυτή φιλοξενεί έναν επεξεργαστή E/E Intel 81342 (IOP342), μνήμη τύπου SDRAM μεγέθους 256MB στα 533MHz και τρεις μηχανές DMA. Επειδή ο προσαρμογέας αυτός δεν διαθέτει προσαρμογέα πακέτων 10G Ethernet, για τους σκοπούς τις εργασίας αυτής, έχει τοποθετηθεί ένας προσαρμογέας δικτύου Intel 82597EX πάνω στη θύρα του δίαυλου PCI-X.

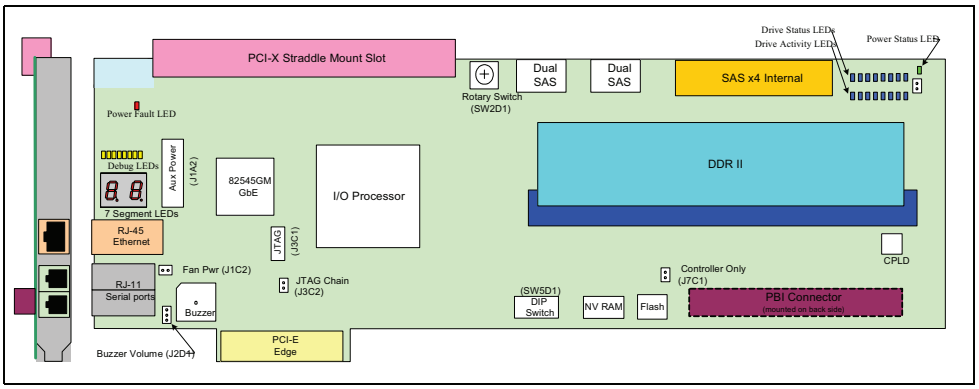
Με τη διάταξη αυτή προσεγγίζεται σε πολύ μεγάλο βαθμό ο στόχος που τέθηκε από το σχεδιασμό. Έχουμε ένα προσαρμογέα δικτύου ο οποίος διαθέτει υπολογιστική ισχύ, μηχανές που του προσδίδουν λειτουργικές ικανότητες και ένα προσαρμογέα πακέτων για τη μεταφορά των δεδομένων στο δίκτυο 10G Ethernet. Ο Intel 81342, σχήμα



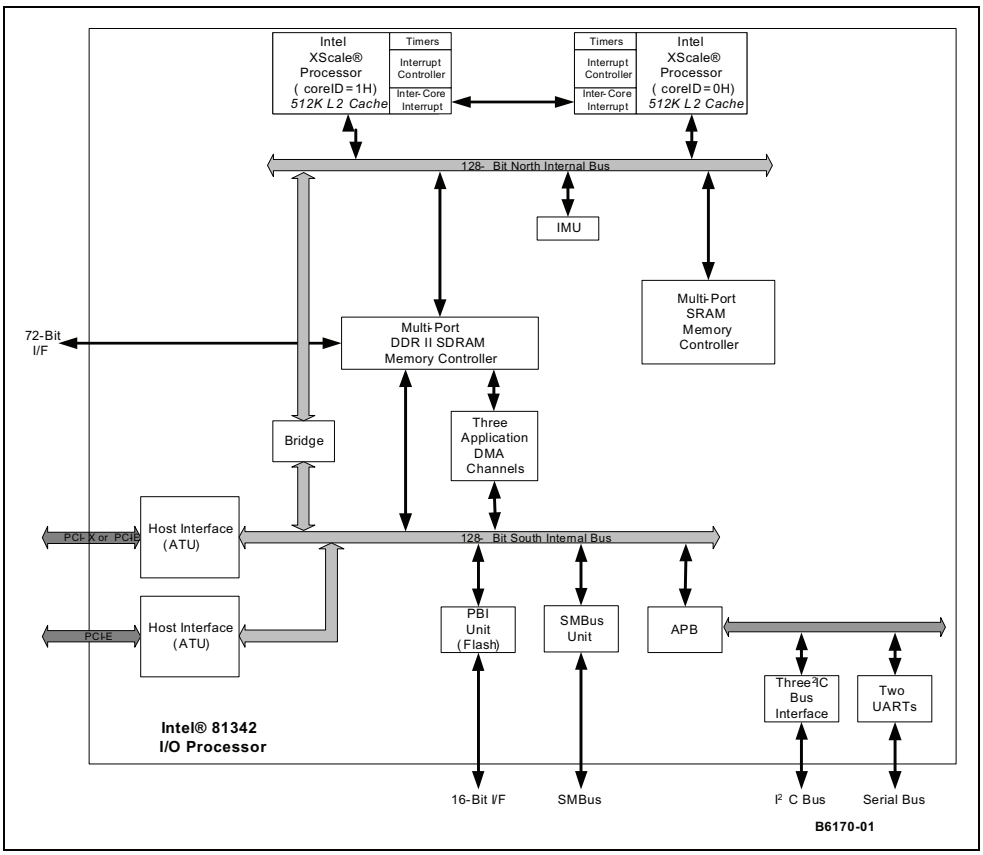
Σχήμα 4.2: Το φυσικό διάγραμμα της προσαρμογέα E/E Intel IQ81340.

4.4, είναι ένας διπύρηνος επεξεργαστής που λειτουργεί σε συχνότητα 1.5GHz, διαθέτει κρυφή μνήμη εντολών μεγέθους 32KB και είναι 32–way set associative, κρυφή μνήμη δεδομένων με τα ίδια χαρακτηριστικά και μια επιπλέον μικρή κρυφή μνήμη δεδομένων μεγέθους 2KB που είναι 2–way set associative. Οι δυο πυρήνες μοιράζονται μια κρυφή μνήμη στο δεύτερο επίπεδο μεγέθους 512KB η οποία είναι 8–way set associative.

Ο Intel 81342 ως ένας επεξεργαστής E/E διαθέτει μηχανισμούς που διευκολύνουν λειτουργίες E/E. Για το σκοπό της εργασίας αυτής χρησιμοποιούνται οι DMA μηχανές και η μονάδα διαβίβασης μηνυμάτων (Messaging Unit – MU).



Σχήμα 4.3: Το φυσικό διάγραμμα της προσαρμογέα E/E Intel IQ81340SC.



Σχήμα 4.4: Το σχεδιάγραμμα του επεξεργαστή E/E Intel 81342.

4.2 Οι μηχανισμοί που συνθέτουν το SLURPoE

Το SLURPoE μπορεί να αναλυθεί σε υποσυστήματα τα οποία έχουν καλώς ορισμένη προγραμματιστική διεπαφή, αναφέρονται ως επί το πλείστον σε διαφορετικά δεδομένα και προσδίδουν διαφορετική λειτουργικότητα στον οδηγό του πρωτοκόλλου. Οι λειτουργίες των υποσυστημάτων μπορεί να χωρίζονται σε ενέργειες που πρέπει να γίνουν τόσο από το χώρο χρήστη του κόμβου όσο και από το module του πυρήνα του κόμβου και του προσαρμογέα.

Αυτή η αντικειμενοστρεφής προσέγγιση της υλοποίησης θα χρησιμοποιηθεί και για τη περιγραφή στο κεφάλαιο αυτό. Αρχικά θα αναλυθούν δύο βασικές δομές που είναι απαραίτητες για το σύνολό των μηχανισμών. Στη συνέχεια περιγράφεται ο μηχανισμός διαβίβασης μηνυμάτων ελέγχου, ο μηχανισμός διαχείρισης της μνήμης, ο μηχανισμός των DMA μεταφορών και τέλος ο μηχανισμός διαχείρισης των πακέτων του 10G Ethernet.

Οι κλήσεις συστήματος από το χώρο χρήστη προς το module του κόμβου γίνονται μέσω μιας συσκευής χαρακτήρα. Η συσκευή χαρακτήρα υλοποιεί ουσιαστικά τη διεπαφή ανάμεσα στο χώρο χρήστη και το χώρο πυρήνα και παρέχει στο χώρο χρήστη όλες τις λειτουργίες που χρειάζεται και πρέπει να εκτελεστούν από σε χώρο με αυξημένα διακαιώματα. Μέσω της συσκευής χαρακτήρα γίνεται η αντιστοίχιση χώρων πάνω στη μνήμη του προσαρμογέα με αντίστοιχους της εφαρμογής καθώς και η διαχείριση της μνήμης.

4.2.1 Δομή του άκρου

Μια πολύ σημαντική δομή για το πρωτόκολλο SLURPoE είναι αυτή του άκρου. Η δομή αυτή, περιγράφεται με διαφορετικό τρόπο στο χώρο χρήστη, στο module του πυρήνα του κόμβου και στο module του πυρήνα του προσαρμογέα. Ο λόγος για τον οποίο διαφοροποιείται σημαντικά είναι οι διαφορετικές ανάγκες που υπάρχουν για το χειρισμό του άκρου σε κάθε περίπτωση. Στο χώρο χρήστη λειτουργεί ως διεπαφή επικοινωνίας τόσο με τον πυρήνα όσο και με τον προσαρμογέα. Στον πυρήνα περιγράφει πόρους που έχουν διατεθεί για αυτό το άκρο ενώ στον προσαρμογέα η δομή περιέχει τον πίνακα αντιστοίχισης φυσικών – εικονικών διευθύνσεων του κόμβου.

Η υλοποίησή του για τη βιβλιοθήκη στο χώρο χρήστη και για το module του πυρήνα του κόμβου είναι:

```
enum hpsi_endpoint_status{
  /* endpoint is free and may be open */
  HPSI_ENDPOINT_STATUS_FREE,
  /* endpoint is already being open by somebody else */
  HPSI_ENDPOINT_STATUS_INITIALIZING,
  /* endpoint is ready to be used */
  HPSI_ENDPOINT_STATUS_OK,
  /* endpoint is being closed by somebody else */
```

```
    HPSI_ENDPOINT_STATUS_CLOSING,
    /* endpoint in error condition */
    HPSI_ENDPOINT_STATUS_ERROR,
};

struct slurpoe_endpoint {
    uint8_t id;
#ifdef __KERNEL__
    pid_t opener_pid;
    char opener_comm[TASK_COMM_LEN];

    struct kref refcount;
    volatile enum slurpoe_endpoint_status status;
    spinlock_t status_lock;
    int closing;

    struct slurpoe_dev *dev;
    struct slurpoe_region *region[SLURPOE_REGIONS_MAX];
    struct slurpoe_host_pte *pte;
    int regions_count;
#else
    int fd;
    struct sluroe_dev_mu_control *mu_ctrl;
    void *ctrl_messages_ptr;
    uint32_t ctrl_messages_offset;

    struct slurpoe_region_hastbl regions[
        SLURPOE_REGION_HASHTBL_SIZE];
    int regions_count;
#endif
  }
```

Στην πλευρά του προσαρμογέα η δομή του άκρου είναι:

```
struct slurpoe_endpoint {
    uint8_t id;

    volatile enum slurpoe_endpoint_status status;
    spinlock_t status_lock;
    struct kref refcount;

    struct slurpoe_board_pte *pte;
}
```

Η δομή του άκρου χρησιμοποιείται σε κάθε λειτουργία του πρωτοκόλλου και σε κάθε υποσύστημα παρέχοντας πληροφορίες για τις εντολές αλλά και για τη διασφάλιση

της απομονωμένης πρόσβασης, απαίτηση που περιγράφηκε αναλυτικά στη σχεδίαση του SLURPoE.

4.2.2 Μηχανισμός διαβίβασης μηνυμάτων ελέγχου

Ο μηχανισμός διαβίβασης μηνυμάτων ελέγχου ικανοποιεί την ανάγκη, να μπορεί ο κόμβος τόσο σε χώρο χρήστη όσο και στο πυρήνα να ελέγχει τον προσαρμογέα δικτύου. Ανάμεσα στο προσαρμογέα και τον κόμβο δημιουργείται ένα κανάλι επικοινωνίας που αφορά σε μηνύματα ελέγχου. Τα μηνύματα αυτά έχουν μέγεθος 32bytes και η γενική δομή που τα περιγράφει είναι:

```
struct slurpoe_mu_msg {
  unsigned char func; /* 1 byte */
  uint8_t pad[3]; /* 3 bytes */
  uint32_t param0, param1, param2, param3, param4, param5, param6; /* 28
      bytes */
};
```

Η δομή αυτή έχει παραλλαγές και εξειδικεύεται ανάλογα με τον τύπο του μηνύματος. Κάθε παραλλαγή πρέπει στο πρώτο byte να υποδηλώνει τον τύπο του μηνύματος και να έχει μέγεθος 32bytes.

Χειρισμός των μηνυμάτων ελέγχου στον προσαρμογέα δικτύου

Η μονάδα διαβίβασης μηνυμάτων του επεξεργαστή 81342 παρέχει για τους σκοπούς του πρωτοκόλλου τον απαραίτητο μηχανισμό για την επικοινωνία μεταξύ του κόμβου και του προσαρμογέα. Η μονάδα αυτή υλοποιεί τέσσερις διαφορετικούς τρόπους επικοινωνίας: τους καταχωρητές μηνυμάτων, τις κυκλικές ουρές, τους καταχωρητές doorbell, τους καταχωρητές δεικτών και την επισήμανση διακοπών με μηνύματα που είναι και λειτουργία που περιγράφεται στο πρότυπο του PCI (Message–Signaled Interrupts – MSI). Για κάθε περίπτωση παρέχει τη δυνατότητα χρήσης διακοπών σε κάθε νέο μήνυμα.

Στην παρούσα υλοποίηση του SLURPoE οι κυκλικές ουρές είναι ο μηχανισμός που χρησιμοποιείται για την υλοποίηση του μηχανισμού διαβίβασης μηνυμάτων ελέγχου. Παρέχει όλες τις λειτουργίες που έχουν προδιαγραφεί στο σχεδιασμό και εξασφαλίζει τη σωστή διαβίβαση μηνυμάτων ελέγχου μεταξύ του κόμβου και του προσαρμογέα.

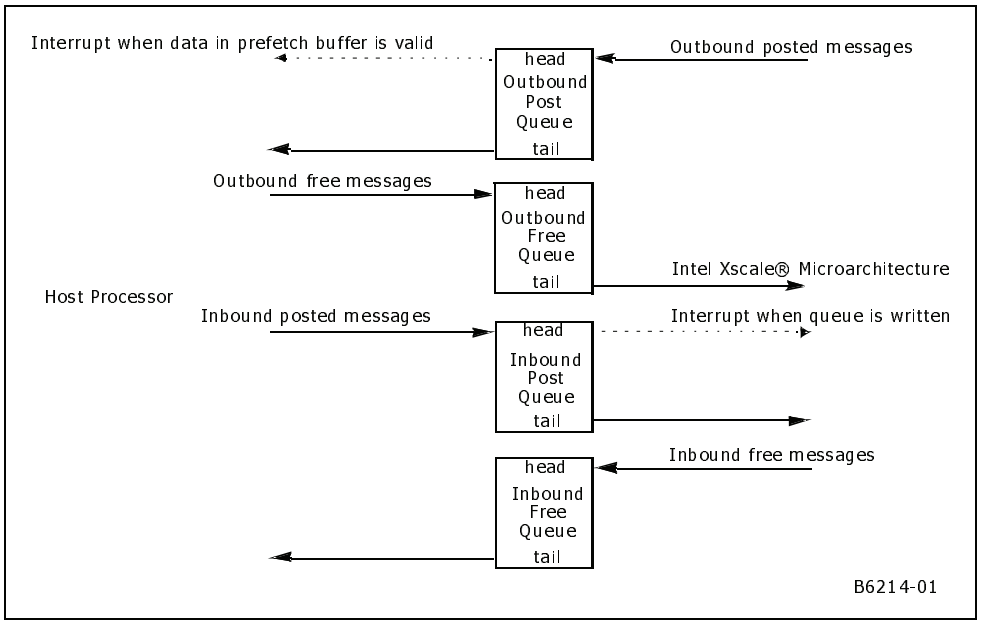
Όπως και στο σχεδιασμό εισερχόμενο είναι ένα μήνυμα που έχει αποστολέα τον κόμβο και παραλήπτη τον προσαρμογέα και εξερχόμενο αυτό που έχει αντίθετη κατεύθυνση. Στον προσαρμογέα έχουν υλοποιηθεί οι τέσσερις ουρές που απαιτούνται: **ουρά εισερχομένων απεσταλμένων μηνυμάτων**, **ουρά εισερχομένων ελεύθερων μηνυμάτων**, **ουρά εξερχομένων ελεύθερων μηνυμάτων** και **ουρά εξερχομένων απεσταλμένων μηνυμάτων**.

Οι ουρές υλοποιούνται σε χώρο που δεσμεύεται από το module του προσαρμογέα. Στο χώρο που υλοποιούνται οι ουρές δείχνει ο δείκτης queue_baseaddr. Συνολικά

δεσμεύονται 16KB που μπορούν να αποθηκεύσουν 4000 τιμές και για τις τέσσερις, ίδιου μεγέθους ουρές. Κατά την αρχικοποίηση προγραμματίζονται τέσσερα ζευγάρια καταχωρητών ειδικού σκοπού, με την κεφαλή και το τέλος της κάθε ουράς μέσα στο συνεχόμενο χώρο:

- inbound_freeheaderptr, inbound_freetailptr
- inbound_postheaderptr, inbound_tailptr
- outbound_freeheaderptr, outbound_freetailptr
- outbound_postheaderptr outbound_tailptr

Καταχωρητές ειδικού σκοπού προγραμματίζονται με το μέγεθος του χώρου που δεσμεύτηκε queue_size καθώς και με ειδικές τιμές – μάσκες που ενεργοποιούν τις διακοπές στην πλευρά του προσαρμογέα mu_inbound_interrupts και στη πλευρά του κόμβου mu_outbound_interrupts. Κάθε τιμή μέσα στις κυκλικές ουρές έχει μέγε-



Σχήμα 4.5: Ο μηχανισμός των κυκλικών ουρών μηνυμάτων του επεξεργαστή E/E Intel 81342.

θος 32bit, γεγονός που δεν επιτρέπει τη μεταφορά των μηνυμάτων ελέγχου μεγέθους 32byte. Τα 32bit ωστόσο είναι αρκετά για την αποθήκευση μιας διεύθυνσης. Ως εκ τούτου, το πρόβλημα αυτό αντιμετωπίζεται με τη δέσμευση ενός χώρου που θα μπορεί να αποθηκεύσει ένα προκαθορισμένο αριθμό μηνυμάτων ελέγχου και οι ουρές θα περιέχουν τιμές διευθύνσεων προς το χώρο αυτό.

Ο χώρος των μηνυμάτων, mu_msgs_ptr δεσμεύεται δυναμικά από το module του προσαρμογέα. Είναι συνεχόμενος και έχει μέγεθος κατάλληλο για να μπορεί να αποθηκεύει 16 μηνύματα ελέγχου.

Στο τέλος της αρχικοποίησης, η μονάδα μηνυμάτων ενεργοποιείται [15] με την αλλαγή ενός bit σε ένα ειδικό καταχωρητή. Οι διευθύνσεις των μηνυμάτων από τον χώρο που δεσμεύτηκε τοποθετούνται στην ουρά εισερχόμενων ελεύθερων μηνυμάτων για να μπορεί να τα χρησιμοποιήσει ο κόμβος. Η διαδικασία αυτή υλοποιείται από τη συνάρτηση:

int slurpoe_mu_init(**struct** slurpoe_mu_control *mu_ctrl);

Η συνάρτηση αυτή χρησιμοποιεί δύο βοηθητικές συναρτήσεις που δημιουργούν το χώρο για τις ουρές και το χώρο μηνυμάτων. Επίσης προγραμματίζουν τους κατάλληλους καταχωρητές.

Ιδιαίτερη προσοχή χρειάζεται στο χειρισμό του χώρου των μηνυμάτων ελέγχου. Ο χώρος αυτός θα πρέπει να μπορεί να διαβάζεται και να εγγράφεται τόσο από τον κόμβο όσο και από τον προσαρμογέα. Η χρήση της κρυφής μνήμης από το προσαρμογέα μπορεί να δημιουργήσει ανεπιθύμητες ενέργειες. Για την αποφυγή των παρενεργειών που οφείλονται στη κρυφή μνήμη ο χώρος των μηνυμάτων ελέγχου δεσμεύεται με τη χρήση της συνάρτησης dma_alloc_coherent() που αποτρέπει τη χρήση της κρυφής μνήμης, όταν ο επεξεργαστής αναφέρεται σε αυτόν.

int slurpoe_alloc_queues(**struct** slurpoe_mu_control *mu_ctrl, **int** size);
int slurpoe_alloc_messages(**struct** slurpoe_mu_control *mu_ctrl, **int** msg_count, **int** msg_size);

Για το χειρισμό των τεσσάρων ουρών έχουν δημιουργηθεί στην πλευρά του module του προσαρμογέα οι παρακάτω συναρτήσεις:

static struct slurpoe_mu_msg *
slurpoe_mu_read_inb_post_queue(**struct** slurpoe_mu_control *mu_ctrl);

static void
slurpoe_mu_write_inb_free_queue(**struct** slurpoe_mu_control *mu_ctrl, **struct** slurpoe_mu_msg *inb_msg);

static struct slurpoe_mu_msg *
slurpoe_mu_read_outb_free_queue(**struct** slurpoe_mu_control *mu_ctrl);

static void
slurpoe_mu_write_outb_post_queue(**struct** slurpoe_mu_control *mu_ctrl, **struct** slurpoe_mu_msg *outb_msg);

Με τις συναρτήσεις αυτές επιτυγχάνεται η ανάγνωση ενός μηνύματος από τις ουρές εξερχόμενων ελεύθερων μηνυμάτων και εισερχόμενων απεσταλμένων μηνυμάτων και η εγγραφή κάποιου στις ουρές εξερχόμενων απεσταλμένων μηνυμάτων και εισερχόμενων ελεύθερων μηνυμάτων αντίστοιχα.

Με την ενεργοποίηση των διακοπών στη μονάδα ελέγχου, τόσο στον κόμβο όσο και στο προσαρμογέα ενεργοποιείται η κατάλληλη διακοπή στη πλευρά που πρέπει να ενημερωθεί για ένα νέο μήνυμα. Μια συνάρτηση χειρισμού δηλώνεται στην κατάλληλη γραμμή διακοπής με μια κλήση στη συνάρτηση request_irq(). Η συνάρτηση χειρισμού από τη πλευρά του προσαρμογέα είναι η:

static irqreturn_t slurpoe_mu_board_isr(**int** irq, **void** *data);

Επειδή η δήλωση γίνεται με ενεργοποιημένη τη σημαία IRQF_DISABLED ο χειριστής δεν μπορεί παρά να κάνει σύντομες ενέργειες.

Ο χειρισμός νέων μηνυμάτων ελέγχου γίνεται σε μια ουρά εργασιών slurpoe_msg¬ _workq_handler. Η ουρά αυτή αρχικοποιείται με κλήσεις στις συναρτήσεις INIT¬ _WORK και create_singlethread_workqueue.

Με βάση την υλοποίηση αυτή κάθε φορά που έρχεται ένα νέο μήνυμα στην ουρά εισερχομένων απεσταλμένων μηνυμάτων, ο προσαρμογέας ενημερώνεται με μια διακοπή. Η συνάρτηση χειρισμού της διακοπής προσθέτει στην ουρά εργασιών την περιγραφή της εργασίας που πρέπει να γίνει, στο πλαίσιο της οποίας τελικά εκτελείται. Δηλαδή, ο χειρισμός των μηνυμάτων δε γίνεται σε χρόνο εξυπηρέτησης της διακοπής, χρόνο κατά τον οποίο οι διακοπές είναι απενεργοποιημένες.

Χειρισμός των μηνυμάτων ελέγχου στον κόμβο

Ο κόμβος χειρίζεται τον προσαρμογέα μέσω του καναλιού επικοινωνίας που αναλύθηκε παραπάνω. Η επικοινωνία αυτή διευκολύνεται από το ίδιο το υλικό του προσαρμογέα και συγκεκριμένα από τον επεξεργαστή E/E Intel 81342. Ο προσαρμογέας κάνει διαθέσιμους δύο ειδικούς καταχωρητές μέσω του χώρου ρυθμίσεων (PCI Configuration Registers) στην πόρτα PCI–Express:

- Ο καταχωρητής Inbound Queue Port, που έχει αντιστοιχηθεί στον οδηγό στο δείκτη inbound_queueport, χρησιμοποιείται για ανάγνωση της διεύθυνσης ενός μηνύματος από την ουρά εισερχόμενων ελεύθερων μηνυμάτων και για εγγραφή της διεύθυνσης ενός μηνύματος στην ουρά εισερχόμενων απεσταλμένων μηνυμάτων.
- Ο καταχωρητής Outbound Queue Port, που έχει αντιστοιχηθεί στο δείκτη outbound_queueport, προορίζεται για την ανάγνωση της διεύθυνσης ενός μηνύματος από την ουρά εξερχόμενων απεσταλμένων μηνυμάτων και την εγγραφή της διεύθυνσης ενός μηνύματος στη ουρά εξερχόμενων ελεύθερων μηνυμάτων.

Οι τιμές που διαβάζονται και εγγράφονται από τους δυο καταχωρητές ειδικού σκοπού είναι διευθύνσεις μέσα στο χώρο διαθέσιμων μηνυμάτων που παρέχεται από το module του προσαρμογέα. Στη διεύθυνση που υποδεικνύεται από τους καταχωρητές αυτούς, εγγράφονται και διαβάζονται τα μηνύματα ελέγχου, που πρέπει να ανταλλάσσονται μεταξύ του προσαρμογέα και του κόμβου.

Απαραίτητη προϋπόθεση για τη λειτουργία του μηχανισμού είναι να γίνει η αντιστοίχηση των καταχωρητών του χώρου ρυθμίσεων στη πόρτα PCI–Express στο χώρο διευθύνσεων τόσο της εφαρμογής όσο και του πυρήνα. Η αντιστοίχιση αυτή για το χώρο χρήστη δημιουργείται από το πυρήνα με μια κλήση στη συνάρτηση ioremap().

Το module του κόμβου ενημερώνεται για νέα μηνύματα μέσω μιας διακοπής. Η αντίστοιχη συνάρτηση χειρισμού ξεχωρίζει τα μηνύματα ελέγχου και εκτελεί την κατάλληλη λειτουργία σε κάθε περίπτωση.

static irqreturn_t slurpoe_mu_host_isr(**int** irq, **void** *data);

Λαμβάνοντας υπόψη ότι στην περίπτωση του κόμβου οι διακοπές χρησιμοποιούνται από πολύ περισσότερες συσκευές, η εγγραφή της διακοπής γίνεται με ενεργοποιημένη τη σημαία IRQF_SHARED, που δηλώνει ότι ενδεχομένως η γραμμή να μοιράζεται με άλλες συσκευές.

4.2.3 Μηχανισμός διαχείρισης της μνήμης

Στο πρωτόκολλο SLURPoE, όπως φαίνεται και στο σχήμα 4.6, η μεταφορά των δεδομένων γίνεται μέσα από ένα μονοπάτι το οποίο δεν περιλαμβάνει την εμπλοκή τόσο του πυρήνα όσο και του επεξεργαστή του κόμβου. Η απαίτηση αυτή κάνει αναγκαίο τον ειδικό χειρισμό του χώρου της μνήμης που συμμετέχει σε τέτοιες μεταφορές. Στόχος είναι να μη συμμετέχει στο μηχανισμό αντικατάστασης του υποσυστήματος μνήμης του Linux και αυτό καθίσταται εφικτό με τη καθήλωση του αντίστοιχου χώρου.

Στη βιβλιοθήκη χρήστη υπάρχουν δύο συναρτήσεις που εξασφαλίζουν τη σωστή δέσμευση και αποδέσμευση της μνήμης.

void *slurpoe_malloc(**struct** slurpoe_endpoint *endpoint, size *t);
void slurpoe_free(**struct** slurpoe_endpoint *endpoint, **void** *buf);

Η συνάρτηση slurpoe_malloc παίρνει ως ορίσματα το χειριστή ενός έγκυρου και αρχικοποιημένου άκρου και το μέγεθος της μνήμης που πρέπει να δεσμευθεί. Επιστρέφει ένα δείκτη στο χώρο της μνήμης που δεσμεύτηκε ο οποίος πλέον μπορεί, με ασφάλεια, να λάβει μέρος σε μια RDMA εγγραφή/ανάγνωση. Μη έγκυρη αρχικοποίηση της μνήμης έχει ως αποτέλεσμα την επιστροφή της τιμής NULL.

Η υλοποίησή της συνάρτησης αυτής περιλαμβάνει αρχικά μια κλήση στη συνάρτηση malloc. Με το τρόπο αυτό δεσμεύει μνήμη κατάλληλου μεγέθους στο χώρο της εφαρμογής. Αν η δέσμευση επιτύχει γίνεται μία κλήση ioctl στη συσκευή χαρακτήρα του module του κόμβου. Οι παράμετροι της κλήσης αυτής είναι η διεύθυνση του χώρου και το μέγεθός της. Ο πυρήνας ελέγχει αν το άκρο που ζητάει τη δηλωση είναι έγκυρο και ανήκει στην ίδια εφαρμογή που κάνει την κλήση ioctl. Ο έλεγχος αυτός γίνεται συγκρίνοντας το pid της εφαρμογής που κάνει τη κλήση ioctl με το pid που χαρακτηρίζει το άκρο από την αρχικοποίηση του.

Η συνάρτηση χειρισμού της κλήσης ioctl για δήλωση της μνήμης διατρέχει σελίδα προς σελίδα το χώρο που πήρε ως παράμετρο και τον καθηλώνει. Η καθήλωση γίνεται με την κλήση της συνάρτησης dma_map_sg(), η οποία παίρνει ως παραμέτρους

μια δομή struct scatterlist που περιγράφει το χώρο στη μνήμη, μια δομή που περιγράφει την συσκευή η οποία θα μπορεί να έχει πρόσβαση με ασφάλεια καθώς και την κατεύθυνση της μεταφοράς. Σε κάθε σελίδα του χώρου αυξάνεται το πεδίο count, γεγονός που της εξασφαλίζει την παραμονή στη φυσική μνήμη.

Οι πληροφορίες της δομής τύπου struct scatterlist αντιγράφονται σε κατάλληλες θέσεις του πίνακα αντιστοιχίσεων που περιέχεται στη δομή του άκρου του προσαρμογέα. Ο προσαρμογέας έχει τις φυσικές διευθύνσεις και το μέγεθος των συνεχόμενων κομματιών για κάθε χώρο μνήμης που έχει δεσμεύσει μια εφαρμογή. Με βάση τον πίνακα αυτό, μπορεί να κάνει την αντιστοίχιση εικονικών διευθύνσεων σε φυσικές με τη συνδρομή μόνο του υλικού του προσαρμογέα. Κατά συνέπεια, μπορεί να βρει τις φυσικές σελίδες που θα μεταφέρει με τις DMA μηχανές, όταν η εφαρμογή θέλει να κάνει μια RDMA μεταφορά από μια εγγεγραμμένη θέση μνήμης.

Η συνάρτηση slurpoe_free παίρνει ως ορίσματα το χειριστή ενός έγκυρου και αρχικοποιημένου άκρου και ένα δείκτη σε ένα χώρο, που δεσμεύτηκε με κλήση της συνάρτησης slurpoe_malloc. Μετά την κλήση της συνάρτησης αυτής οποιαδήποτε απόπειρα αναφοράς σε αυτόν το δείκτη οδηγεί σε σφάλμα.

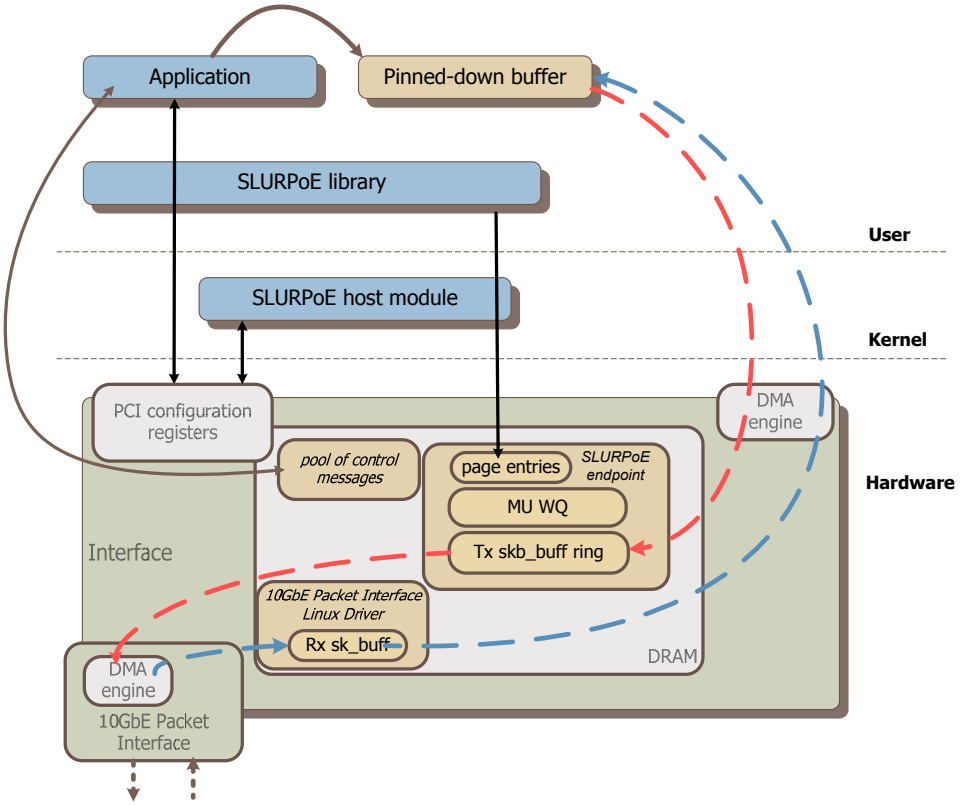
Η συμμετοχή του πυρήνα είναι αναπόφευκτη και στην αποδέσμευση του χώρου μνήμης. Από τη πλευρά του κόμβου αίρεται ο περιορισμός που είχε τεθεί για τις φυσικές σελίδες που συνθέτουν το χώρο. Μια κλήση στη συνάρτηση put_page μειώνει το πεδίο count της σελίδας με αποτέλεσμα να επανέρχεται στην κανονική της κατάσταση. Επίσης από τον πυρήνα διαγράφονται οι εγγραφές που είχαν γίνει προηγούμενα στον πίνακα αντιστοιχίσεων της δομής του συγκεκριμένου άκρου. Τέλος μια κλήση της συνάρτησης free ελευθερώνει το χώρο.

4.2.4 Μηχανισμός των DMA μεταφορών

Ο μηχανισμός των DMA μεταφορών αποτελεί ένα από τα πιο σημαντικά στοιχεία στην υλοποίηση των RDMA εγγραφών/αναγνώσεων. Σε κάθε περίπτωση RDMA μεταφοράς το πρώτο και το τελευταίο βήμα είναι η μεταφορά DMA από τον εικονικό χώρο της εφαρμγής προς το προσαρμογέα ή αντίστροφα. Σε μεταφορές DMA μπορούν να συμμετέχουν μόνο χώροι της μνήμης που έχουν δηλωθεί. Ο έλεγχος αυτός εξασφαλίζεται από το προσαρμογέα, μέσω του πίνακα αντιστοιχίσεων που διατηρεί για κάθε άκρο. Ο πίνακας αντιστοιχίσεων άλλωστε, είναι αυτός που επιτρέπει στις εφαρμογές να μπορούν να ζητήσουν τη μεταφορά χώρων μνήμης, από και προς το προσαρμογέα, οι οποίοι περιγράφονται από την εικονική τους διεύθυνση.

Στον οδηγό του προσαρμογέα που είναι ένα module του Linux, χρησιμοποιείται το DMA API του πυρήνα. Στην τρέχουσα έκδοσή του μάλιστα (2.6.30) το API αυτό έχει υλοποιηθεί και για συστήματα που βασίζονται στον επεξεργαστή E/E Intel 81342.

Το πρόβλημα που αντιμετωπίστηκε κατά την υλοποίηση του module του προσαρμογέα είναι ότι το DMA API του πυρήνα του Linux ορίζει συναρτήσεις μεταφορών DMA που έχουν τόσο ως προέλευση όσο και ως προορισμό τη φυσική μνήμη του προσαρμογέα. Στα πλαίσια της υλοποίησης της SLURPoE οι μεταφορές DMA πρέπει να



Σχήμα 4.6: Τα μονοπάτια ελέγχου και δεδομένων στο SLURPoE

είναι δυνατό να γίνονται και για χώρους που δεν βρίσκονται στο φυσική μνήμη του προσαρμογέα.

Για να γίνει εφικτό δημιουργήθηκε μια επέκταση του DMA API του Linux και ορίστηκαν δύο επιπλέον συναρτήσεις:

```
dma_cookie_t
dma_async_memcpy_host_to_buf(struct dma_chan *chan, void *dest,
    u64 src, size_t len);
dma_cookie_t
dma_async_memcpy_buf_to_host(struct dma_chan *chan, u64 dest,
    void *src, size_t len);
```

Οι συναρτήσεις αυτές μπορούν να χρησιμοποιηθούν για να προγραμματιστεί μια από τις DMA μηχανές του Intel 81342 και να μεταφερθούν δεδομένα από τη φυσική μνήμη του κόμβου προς τη φυσική μνήμη του προσαρμογέα και αντίστροφα.

Αρχικοποίηση

Πριν γίνει οποιαδήποτε DMA μεταφορά πρέπει στον οδηγό του προσαρμογέα να εγγραφούν και να αρχικοποιηθούν τα DMA κανάλια. Κάθε DMA κανάλι είναι η διεπαφή του πυρήνα του Linux για μια DMA μηχανή που είναι διαθέσιμη από το υλικό.

```
static enum dma_state_client slurpoe_dma_add_channel(struct dma_chan *
    chan);
static enum dma_state_client slurpoe_dma_remote_channel(struct
    dma_chan *chan);

static enum dma_state_client;
slurpoe_dma_event_callback(struct dma_client *client, struct dma_chan *
    chan,
                           enum dma_state state)
{
  enum dma_state_client ack = DMA_NAK;

  switch (state) {
    case DMA_RESOURCE_AVAILABLE:
      ack = slurpoe_dma_add_channel(chan);
      break;
    case DMA_RESOURCE_REMOVED:
      ack = slurpoe_dma_remove_channel(chan);
      break;
    default:
      printk_warn("Unhandled␣event␣%u␣(%s)\n", state,
      dev_name(&chan→dev));
      break;
  }

  return ack;
}

static struct dma_client slurpoe_dma_client = {
  .event_callback = slurpoe_dma_event_callback,
};

dma_cap_set(DMA_MEMCPY_TO_HOST, iop_dma_client.cap_mask);
dma_cap_set(DMA_MEMCPY_FROM_HOST,
    iop_dma_client.cap_mask);
dma_async_client_register(&slurpoe_dma_client);
dma_async_client_chan_request(&sluroe_dma_client);
```

Με την κλήση στη συνάρτηση dma_cap_set δίνονται τα δικαιώματα που πρέπει να έχει μια DMA μηχανή για να μπορεί να κάνει μεταφορές από και προς τον κόμβο. Η συνάρτηση slurpoe_dma_add_channel() δεσμεύει το κανάλι για χρήση από τον οδηγό και αρχικοποιεί δομές και μεταβλητές απαραίτητες για τη λειτουργία του. Αντίστοιχα η συνάρτηση slurpoe_dma_free_channel() αποδεσμεύει το κανάλι και ελευθερώνει τους πόρους που είχαν δεσμευτεί.

Μεταφορές DMA σε μια εντολή RDMA Εγγραφής

Όπως περιγράφεται και στο σχεδιασμό του πρωτοκόλλου η βιβλιοθήκη υποστήριξης στο χώρο χρήστη για την εκτέλεση μιας RDMA εγγραφής παρέχει τη συνάρτηση:

void slurpoe_rdma_write(**struct** slurpoe_endpoint *endpoint, **void** *buf,
    size_t size, **uint8_t** dst_endpoint_id, **char** dst_addr[6], **void** *dst_buf);

Στην υλοποίηση της εντολής αυτής δε συμμετέχει καθόλου ο πυρήνας του κόμβου. Όλες οι απαραίτητες λειτουργίες γίνονται αφενός σε χώρο χρήστη και αφετέρου από τον προσαρμογέα. Η υλοποίηση αυτή απεμπλέκει πλήρως τον πυρήνα του λειτουργικού από το κρίσιμο μονοπάτι μιας RDMA εγγραφής.

Το πρώτο βήμα είναι (α) η προετοιμασία του μηνύματος ελέγχου που θα αποσταλεί στο προσαρμογέα και (β) η αποστολή του μέσω του καναλιού που έχει δημιουργηθεί. Ο τύπος του μηνύματος ελέγχου που ενημερώνει το προσαρμογέα για μια RDMA εγγραφή, αποτελεί ειδίκευση του γενικού τύπου που έχει ήδη παρουσιαστεί:

```
struct slurpoe_mu_msg_rdma_write {
  unsigned char func; /* 1 byte */
  char sa_data[6]; /* 6 bytes */
  char pad1; /* 1 byte */
  uint32_t src_lo_addr; /* 4 bytes */
  uint32_t src_hi_addr; /* 4 bytes */
  uint64_t dst_addr; /* 8 bytes */
  uint32_t size; /* 4 bytes */
  uint8_t src_endpoint_id; /* 1 byte */
  uint8_t dst_endpoint_id; /* 1 byte */
  char pad2[2]; /* 2 bytes */
};
```

Με το τέλος της αποστολής του μηνύματος αυτού, ο έλεγχος της μεταφοράς περνά στο προσαρμογέα. Με χρήση του πίνακα αντιστοιχίσεων για κάθε σελίδα που αφορά στο χώρο που έχει δηλωθεί προγραμματίζεται η μηχανή DMA για τη μεταφορά.

Η DMA μεταφορά γίνεται με μια κλήση στη συνάρτηση dma_async_memcpy_¬host_to_buf(), η οποία επιστρέφει ένα cookie. Το cookie αυτό χρησιμοποιείται για να ελεγχθεί αν η αίτηση για μεταφορά έχει επιτύχει αλλά και για να ενημερωθεί ο οδηγός για την εξέλιξη της μεταφοράς. Ένας τέτοιος έλεγχος είναι απαραίτητος αφού

μια DMA μεταφορά γίνεται χωρίς την εμπλοκή του επεξεργαστή και άρα χωρίς να επηρεάζει καθόλου τη ροή ελέγχου του οδηγού.

Το μήνυμα αυτό, όταν φτάσει στον προσαρμογέα του κόμβου που έχει καθοριστεί με βάση την MAC διεύθυνσή του, πρέπει να μεταφερθεί με μια DMA μεταφορά προς το χώρο μνήμης του άκρου προορισμού. Στο προσαρμογέα προορισμού ελέγχεται τόσο αν το άκρο είναι έγκυρο όσο και αν ο χώρος της μνήμης που ζητείται είναι εγγεγραμμένος. Αν οι έλεγχοι αυτοί επιτύχουν τότε η DMA μηχανή με μια κλήση στη συνάρτηση dma_async_memcpy_buf_to_host() μεταφέρει τα δεδομένα με τον ίδιο ακριβώς τρόπο.

Μεταφορές DMA σε μια εντολή RDMA Ανάγνωσης

Όπως περιγράφεται και στο σχεδιασμό του πρωτοκόλλου η βιβλιοθήκη υποστήριξης στο χώρο χρήστη παρέχει για την εκτέλεση μιας RDMA ανάγνωσης τη συνάρτηση:

```
void slurpoe_rdma_read(struct slurpoe_endpoint *endpoint, void *buf,
    size_t size, uint8_t src_endpoint_id, char src_addr[6], void *src_buf);
```

Στην υλοποίηση και αυτής της εντολής αυτής δε συμμετέχει καθόλου ο πυρήνας του κόμβου. Όλες οι απαραίτητες λειτουργίες γίνονται αφενός σε χώρο χρήστη και αφετέρου από τον προσαρμογέα. Η υλοποίηση αυτή απεμπλέκει το πυρήνα του λειτουργικού πλήρως από το κρίσιμο μονοπάτι μιας RDMA ανάγνωσης.

Το πρώτο βήμα και εδώ είναι η αποστολή του αντίστοιχου μηνύματος ελέγχου στο προσαρμογέα. Ο τύπος του μηνύματος ελέγχου που ενημερώνει το προσαρμογέα για μια RDMA ανάγνωση αποτελεί ειδίκευση του γενικού τύπου που έχει ήδη παρουσιαστεί:

```
struct slurpoe_mu_msg_rdma_read {
  unsigned char func;
  char sa_data[6];
  char pad1;
  uint32_t dst_lo_addr;
  uint32_t dst_hi_addr;
  uint64_t src_addr;
  uint32_t size;
  uint8_t src_endpoint_id;
  uint8_t dst_endpoint_id;
  char pad2[2];
};
```

Με το τέλος της αποστολής του μηνύματος αυτού ο έλεγχος της μεταφοράς περνά στον προσαρμογέα. Στο σημείο αυτό ο προσαρμογέας αποστέλλει στο προσαρμογέα προορισμού ένα πακέτο που έχει τα στοιχεία της αντίστοιχης RDMA εγγραφής που πρέπει να εκτελέσει. Στην ουσία η λειτουργία αυτή δεν διαφέρει σε τίποτα από την

RDMA εγγραφή που περιγράφηκε με τη διαφορά ότι εκτελείται αντίστροφα και εκκινείται από το άκρο του προορισμού και όχι της προέλευσης. Κατά συνέπεια και οι DMA μεταφορές που απαιτούνται για μια RDMA ανάγνωση γίνονται με τον ίδιο ακριβώς τρόπο όπως και στην RDMA εγγραφή.

4.2.5 Μηχανισμός διαχείρισης των πακέτων στο δίκτυο Ethernet

Οι βασικές δομές που χρησιμοποιούνται από το υποσύστημα έλεγχου του προσαρμογέα πακέτων είναι αυτές που χρησιμοποιούνται για τη δημιουργία των πακέτων του πρωτοκόλλου και πιο συγκεκριμένα των κεφαλίδων τους. Για τα δύο μηνύματα που υλοποιούνται έχουμε τις εξής δομές:

```
enum slurpoe_pkt_type {
  SLURPOE_PKT_TYPE_RDMA_WRITE=0,
  SLURPOE_PKT_TYPE_RDMA_READ,
};

struct slurpoe_pkt_rdma_write {
  enum slurpoe_pkt_type pkt_type:1;
  uint8_t dst_endpoint_id;
  uint8_t src_endpoint_id;
  uint8_t seqnum;
  uint32_t length;
  uint64_t dst_vaddr;
};

struct slurpoe_pkt_rdma_read {
  enum slurpoe_pkt_type pkt_type:1;
  uint8_t dst_endpoint_id;
  uint8_t src_endpoint_id;
  uint8_t pad;
  uint32_t length;
  uint64_t dst_vaddr;
  uint64_t src_vaddr;
};

struct slurpoe_pkt_hdr {
  struct ethhdr eth;
  union {
    struct slurpoe_pkt_rdma_write write;
    struct slurpoe_pkt_rdma_read read;
  } body;
};
```

Αρχικοποίηση Αρχικά ο διαθέσιμος προσαρμογέας πακέτων δεσμεύεται για να χρησιμοποιηθεί από τον οδηγό του προσαρμογέα. Αυτό γίνεται μέσω της διεπαφής του στο Linux και μια κλήση στη συνάρτηση:

ifp = dev_get_by_name(&init_net, ethernet_dev);

Η κλήση της συνάρτησης επιστρέφει το δείκτη ifp με τον οποίο είναι δυνατός ο χειρισμός του προσαρμογέα πακέτων.

Κατά την αρχικοποίηση του μηχανισμού διαχείρισης των πακέτων Ethernet γίνεται η δέσμευση ενός πινάκα με δομές struct sk_buff. Με το τρόπο αυτό και με τη σωστή διαχείριση και επαναχρησιμοποίηση των διαθέσιμων struct sk_buff μπορεί να αποφευχθεί η δέσμευση τους στο κρίσιμο μονοπάτι μιας RDMA μεταφοράς.

Ο χειρισμός των λήψεων είναι αρκετά διαφορετικός. Η υλοποίηση βασίζεται στη λογική ότι για κάθε πακέτο που έρχεται στον προσαρμογέα πακέτων θα ελέγχεται ο τύπος του. Αν πρόκειται για πακέτο του SLURPoE τότε το αντίστοιχο stuct sk_buff θα περνά ως όρισμα στη συνάρτηση χειρισμού που έχει δηλωθεί.

Αρχικά δημιουργείται μια δομή struct packet_type. Στο πεδίο της type δίνεται η τιμή ___constant_htons(ETH_P_SLURPOE) και στο πεδίο func το όνομα της συνάρτησης χειρισμού τέτοιων πακέτων slurpoe_recv(). Με όρισμα τη δομή αυτή καλείται η συνάρτηση dev_add_pack().

int slurpoe_recv(**struct** sk_buff *skb, **struct** net_device *ifp,
    struct packet_type *pt, **struct** net_device *orig_ifp);

Η συνάρτηση αυτή με την παράμετρο skb παίρνει εισερχόμενα πακέτα που αφορούν το πρωτόκολλο και αφού τα ξεχωρίσει διαβάζοντας τη κεφαλίδα τους εκτελεί τη κατάλληλη συνάρτηση. Αξίζει να σημειωθεί ότι για το χειρισμό των πακέτων έχει δημιουργηθεί ένας πίνακας δεικτών σε συναρτήσεις, όπου το πρώτο κελί έχει πάρει την τιμή της συνάρτησης χειρισμού μιας RDMA εγγραφής και το δεύτερο έχει πάρει την τιμή της συνάρτησης χειρισμού μιας RDMA ανάγνωσης.

Με τον τρόπο αυτό και λαμβάνοντας υπόψη ότι ο τύπος του πακέτου είναι μια απαρίθμηση στη C, αφού αναγνωριστεί ποιος είναι ο τύπος του εισερχόμενου πακέτου, αρκεί να καλέσουμε τη συνάρτηση που είναι στο κελί με αριθμό αυτόν του τύπου. Έτσι σε ένα κρίσιμο σημείο αποφεύγουμε τη χρήση μιας εντολής switch που θα επιβάρυνε τον κώδικα σε χρόνο εκτέλεσης.

static int slurpoe_rcv_rdma_write(**struct** sk_buff *skb,
    struct net_device *ifp, **struct** slurpoe_pkt_hdr *hdr);
static int slurpoe_rcv_rdma_read(**struct** sk_buff *skb,
    struct net_device *ifp, **struct** slurpoe_pkt_hdr *hdr);

static int (*slurpoe_pkt_type_handler[SLURPOE_PKT_TYPE_MAX+1])
    (**struct** sk_buff *skb, **struct** net_device *ifp, **struct** slurpoe_pkt_hdr *hdr);
slurpoe_pkt_type_handler[SLURPOE_PKT_TYPE_RDMA_WRITE] =
    slurpoe_rcv_rdma_write;

```
slurpoe_pkt_type_handler[SLURPOE_PKT_TYPE_RDMA_READ] =
    slurpoe_rcv_rdma_read;
```

Η συνάρτηση slurpoe_rcv_rdma_write() χειρίζεται ένα εισερχόμενο πακέτο που έχει τύπο RDMA εγγραφής. Με βάση την κεφαλίδα του πακέτου θα πρέπει να μεταφέρει με μια μηχανή DMA τα δεδομένα στη μνήμη του κόμβου. Αυτό όμως θα γίνει αφού κάνει τους απαραίτητους ελέγχους που αφορούν στο άκρο.

Η συνάρτηση slurpoe_rcv_rdma_read θα χειριστεί ένα εισερχόμενο πακέτο που έχει τύπο RDMA ανάγνωσης. Στην ουσία κάνει όλες εκείνες τις ενέργειες που κάνει και ο χειριστής των μηνυμάτων ελέγχου όταν λάβει ένα μήνυμα RDMA εγγραφής. Η ειδοποιός διαφορά είναι ότι σε αυτή την περίπτωση τα στοιχεία που προσδιορίζουν τις παραμέτρους της RDMA εγγραφής προέρχονται από την κεφαλίδα ενός πακέτου και όχι από μήνυμα του υποσυστήματος ελέγχου του προσαρμογέα.

Αποστολή πακέτων

Η αποστολή των δεδομένων προς το δίκτυο πρέπει να γίνεται, με τρόπο έξυπνο ώστε να αποφεύγονται αντίγραφα στη μνήμη του προσαρμογέα. Ο χειρισμός αυτός ξεκινά πριν τη μεταφορά δεδομένων από τη μνήμη του κόμβου. Όταν ένα μήνυμα μεταφοράς δεδομένων φτάσει στο προσαρμογέα που επιβάλει τη μεταφορά δεδομένων προς το δίκτυο, ο οδηγός του SLURPoE δεν πρέπει να κάνει τη μεταφορά σε έναν οποιοδήποτε ενδιάμεσο χώρο στη μνήμη του προσαρμογέα. Με ένα τέτοιο χειρισμό η αποστολή των δεδομένων στο δίκτυο και η ενθυλάκωση τους σε πακέτα θα απαιτούσε μια ακόμα αντιγραφή.

Για να αποφευχθεί το αντίγραφο αυτό, η μεταφορά DMA γίνεται απευθείας στο επόμενο ελεύθερο struct sk_buff από τον πίνακα που έχει δεσμευτεί κατά την αρχικοποίηση. Παράλληλα, με βάση τη πληροφορία του μηνύματος ελέγχου γράφονται τα κατάλληλα δεδομένα στην κεφαλίδα του πρωτοκόλλου. Με το τέλος της μεταφοράς DMA τα πακέτα είναι έτοιμα να αποσταλούν και αρκεί μια κλήση στη συνάρτηση dev_queue_xmit() για να τα μεταφέρει στο καλώδιο.

Λήψη πακέτων

Η σωστή λήψη των πακέτων εξασφαλίζεται από τη σωστή αρχικοποίηση. Οι συναρτήσεις χειρισμού που έχουν οριστεί λαμβάνουν τις κατάλληλες ενέργειες από τις πληροφορίες των πακέτων.

Κεφάλαιο 5

Αξιολόγηση

5.1 Σύνοψη

Στην παρούσα εργασία περιγράφουμε ένα πρωτόκολλο μεταφοράς δεδομένων με απευθείας απομακρυσμένη πρόσβαση στη μνήμη (RDMA). Στόχος της εργασίας είναι να καθοριστούν οι απαιτήσεις ενός προγραμματιζόμενου προσαρμογέα δικτύου που θα μπορεί να εγκαταστήσει ένα μονοπάτι μεταφοράς δεδομένων από κόμβο–σε–κόμβο μέσα σε μια συστοιχία υπολογιστών. Το μονοπάτι αυτό δεν περιέχει αντίγραφα δεδομένων και γι' αυτό το λόγο μπορεί να επιτύχει μικρούς χρόνους αρχικής απόκρισης, γεγονός που καθιστά το δίκτυο διασύνδεσης ιδανικό για συστήματα υψηλής επίδοσης.

Αρχικά δίνεται στον αναγνώστη το θεωρητικό υπόβαθρο για να κατανοήσει το σχεδιασμό και την υλοποίηση του πρωτοκόλλου. Αναλύονται τα βασικότερα χαρακτηριστικά των πιο διαδεδομένων δικτύων διασύνδεσης υψηλής επίδοσης καθώς και οι βασικές παράμετροι του σχεδιασμού τους. Η μελέτη των επιλογών των παραπάνω δικτύων διασύνδεσης, με έμφαση στα πλεονεκτήματα και τα μειονεκτήματα που διαθέτουν, οδήγησε στις βασικές σχεδιαστικές επιλογές του πρωτοκόλλου SLURPoE (SimpLe User–level RDMA Protocol over Ethernet).

Επιπρόσθετα, παρουσιάζονται τα υποσυστήματα ενός λειτουργικού συστήματος που θα μπορούσε να φιλοξενήσει ένα τέτοιο πρωτόκολλο. Επιλέξαμε το Linux λόγω του ανοικτού κώδικα και της δυνατότητας μελέτης των λειτουργικών χαρακτηριστικών σε βάθος. Το γεγονός αυτό μας έδωσε τη δυνατότητα να χρησιμοποιήσουμε τους μηχανισμούς του στα υποσυστήματα του οδηγού του πρωτοκόλλου. Σε πολύ μεγάλο βαθμό, η προσέγγιση που ακολουθήσαμε κατά την υλοποίηση σέβεται σημασιολογικά την προγραμματιστική διεπαφή που προσφέρει το Linux.

Στη συνέχεια παρουσιάζουμε το σχεδιασμό του πρωτοκόλλου SLURPoE που διατηρεί τα βασικά χαρακτηριστικά των δικτύων που περιγράφηκαν προηγουμένως όπως οι μεταφορές δεδομένων μηδενικών αντιγράφων (zero–copy data transfers) καθώς και η δικτύωση σε χώρο χρήστη (user–level networking). Παράλληλα, καθορίστηκαν και

οι λειτουργικές απαιτήσεις που πρέπει να ικανοποιεί ένας προσαρμογέας δικτύου έτσι, ώστε να μπορεί να υποστηρίξει το SLURPoE.

Τέλος περιγράφουμε την υλοποίηση του SLURPoE στο Linux. Οι βασικές δομές καθώς και τα πρωτότυπα συναρτήσεων δίνονται για την πλήρη κατανόηση των αντικειμένων και των λειτουργιών του πρωτοκόλλου. Δεν κρίθηκε αναγκαίο να δημοσιευτεί ο κώδικας καθώς ο σκοπός της εργασίας δεν αποτελεί την υλοποίηση ενός ακόμα RDMA πρωτοκόλλου αλλά είναι ο καθορισμός των απαιτήσεων ενός "έξυπνου" προγραμματιζόμενου προσαρμογέα δικτύου για δίκτυα διασύνδεσης υψηλής επίδοσης.

5.2 Συμπεράσματα

Η μελέτη των υποσυστημάτων του Linux, των δικτύων διασύνδεσης υψηλών επιδόσεων καθώς και των χαρακτηριστικών που διαθέτουν τα παράλληλα προγραμματιστικά μοντέλα μας οδήγησε στα εξής συμπεράσματα:

- Το κρίσιμο μονοπάτι ενός πρωτοκόλλου επικοινωνίας πρέπει να περιέχει μόνο τα υποσυστήματα που απαιτούνται για τη μεταφορά δεδομένων. Είναι εξαιρετικά σημαντικό να παραλείπονται λειτουργίες που προκαλούν υψηλές καθυστερήσεις και μειώνουν αισθητά τη ρυθμαπόδοση του συστήματος. Το SLURPoE εξαιρεί από το κρίσιμο μονοπάτι τη δήλωση των χώρων μνήμης με την επαναχρησιμοποίηση των ήδη εγγεγραμμένων χώρων από την αρχικοποίηση της επικοινωνίας.

- Η δημιουργία αντιγράφων στο μονοπάτι μεταφοράς δεδομένων αποτελεί σημείο επιπλέον επιβάρυνση στο χρόνο πρώτης απόκρισης ενός πρωτοκόλλου επικοινωνίας υψηλής επίδοσης. Το SLURPoE λειτουργεί από άκρο σε άκρο με μηδενικά αντίγραφα, χωρίς να θυσιάζει την ασφάλεια του συστήματος και χωρίς αυτό να επηρεάζει τις λειτουργικές του δυνατότητες. Αυτό επιτυγχάνεται με χρήση των μηχανών DMA και των μηχανισμών μηνυμάτων του προσαρμογέα.

- Το 10G Ethernet με το εύρος ζώνης που διαθέτει έχει σημαντικές απαιτήσεις σε επεξεργαστική ισχύ του κόμβου. Οι σύγχρονοι προσαρμογείς δικτύου πρέπει να διαθέτουν λειτουργικά χαρακτηριστικά τέτοια, ώστε να αναλαμβάνουν το κόστος της επικοινωνίας διατηρώντας ταυτόχρονα την ταχύτητα μεταφοράς στα όρια της γραμμής στο φυσικό επίπεδο. Με βάση τα παραπάνω, το σχεδιασμό και την υλοποίηση του SLURPoE, ένας προσαρμογέας δικτύου διασύνδεσης υψηλών επιδόσεων πρέπει να διαθέτει:

  - μηχανές DMA που μπορούν να αναλάβουν το κόστος μεταφοράς δεδομένων από/προς τη μνήμη του κόμβου χωρίς να διακόπτουν τον επεξεργαστή του τελευταίου.

- δυνατότητα σύνδεσης σε θύρες κοντά στο σύστημα μνήμης του κόμβου έτσι, ώστε η μεταφορά των δεδομένων να μην επιβαρύνει τους υπόλοιπους διαύλους Ε/Ε του συστήματος καθώς και να επιτυγχάνεται η θεωρητική ταχύτητα που επιτρέπει το φυσικό επίπεδο.
- επεξεργαστή που να διασφαλίζει την εύρυθμη και αποκομμένη επικοινωνία του προσαρμογέα με τον κόμβο και με το δίκτυο.
- γρήγορο δίαυλο μνήμης–επεξεργαστή έτσι, ώστε αν χρειαστεί, ο επεξεργαστής να εξετάζει τα δεδομένα που διέρχονται από τον προσαρμογέα (π.χ. σε περίπτωση κρυπτογράφησης σε πραγματικό χρόνο).

Η πειραματική αξιολόγηση του πρωτοκόλλου SLURPoE αποτελεί βασικό στόχο επέκτασης της συγκεκριμένης εργασίας. Ο εκτεταμένος χρόνος που αφιερώθηκε στη μελέτη της βιβλιογραφίας, στο σχεδιασμό, στην υλοποίηση και στην εκσφαλμάτωση του κώδικα τόσο σε χώρο–χρήστη όσο και σε χώρο–πυρήνα δε μας έδωσε τη δυνατότητα να διεξάγουμε μετρήσεις και να παρουσιάσουμε μια εκτίμηση της απόδοσης του SLURPoE.

Κεφάλαιο 6

Επίλογος

6.1 Παρεμφερείς Εργασίες

Η τρέχουσα τάση για εκμετάλλευση του 10G Ethernet στα υπολογιστικά συστήματα υψηλής επίδοσης έχει αποτελέσει την αφορμή για την ανάπτυξη αρκετών εργασιών που έχουν ως στόχο τη βέλτιστη χρήση του.

Μια αρκετά ενδιαφέρουσα προσέγγιση παρουσιάζεται στο Open-MX [16]. Η εργασία αυτή αφορά στη χρήση του πρωτοκόλλου για πέρασμα μηνυμάτων MX πάνω από κοινές κάρτες Ethernet. Ο στόχος είναι η επίτευξή καλύτερων επιδόσεων από παράλληλες εφαρμογές που κάνουν χρήση της προγραμματιστικής διεπαφής MPI σε κοινούς προσαρμογείς δικτύου. Το αποτέλεσμα είναι πολύ καλό τόσο σε χρόνο αρχικής απόκρισης όσο και σε αξιοποίηση του εύρους ζώνης του 10G Ethernet. Η επιβάρυνση ωστόσο στο CPU του κόμβου είναι αρκετά σημαντική καθώς αναλαμβάνει το σύνολο των υπολογιστών απαιτήσεων της επικοινωνίας.

Προσπάθεια για εκμετάλλευσή της λογικής του RDMA πάνω από συσκευές Ethernet γίνεται και από το Internet Engineering Task Force (IETF). Το iWARP[17] αποτελεί την τελευταία πρόταση για μεταφορές μηδενικών αντιγράφων πάνω από το επίπεδο TCP της στοίβας πρωτοκόλλων TCP/IP. Οι προδιαγραφές του είναι ανοιχτές και διαθέσιμες προς υλοποίηση από όλους τους κατασκευαστές προσαρμογέων δικτύου. Απαιτεί από τις συσκευές υπολογιστική ισχύ ώστε να μπορούν να αναλάβουν το σύνολο των απαιτήσεων της επεξεργασίας του πρωτοκόλλου TCP.

6.2 Επεκτάσεις

Η υλοποίηση του SLURPoE επιδέχεται βελτιώσεις, οι οποίες θα μπορούσαν να έχουν αντίκτυπο στην τελική απόδοση αλλά και στην αξιοπιστία. Ο κώδικας έχει δομηθεί με βάση τη λειτουργικότητά του και σε γενικές γραμμές έχει χωριστεί σε υποσυστήματα, τα οποία παρέχουν συγκεκριμένες υπηρεσίες και επενεργούν σε συγκεκριμένες

δομές δεδομένων. Παράδειγμα αποτελεί το υποσύστημα μηνυμάτων ελέγχου, το υποσύστημα διαχείρισης της μνήμης και το υποσύστημα διαχείρισης του δικτύου.

Η λειτουργία του πρωτοκόλλου είναι δυνατόν να εμφανίσει σε πραγματικές συνθήκες απώλειες πακέτων γεγονός το οποίο οφείλεται στην έλλειψη μηχανισμού ελέγχου της ροής. Η υλοποίηση ενός τέτοιου μηχανισμού, θα μπορέσει να αντιμετωπίσει τα προβλήματα από την απώλεια πακέτων πάνω σε κάποιο από τα κυκλώματα ή το καλώδιο του 10G Ethernet.

Επίσης μια υλοποίηση του πρωτοκόλλου σε χώρο πυρήνα όπως αναφέρθηκε και στο σχεδιασμό, με σχετικά μικρή προσπάθεια, θα μπορούσε να αποτελέσει ένα πολύ καλό πείραμα για τη σύγκριση των δύο τεχνικών: υλοποίηση του δικτυακού πρωτοκόλλου σε χώρο πυρήνα και σε χώρο χρήστη. Θα μπορούσε επίσης να επιβεβαιώσει ή και να διαψεύσει τα επιχειρήματα για τη σχεδιαστική επιλογή του SLURPoE.

Από τη σχεδίαση έχει καθοριστεί ότι το SLURPoE μπορεί να χρησιμοποιηθεί από οποιαδήποτε υλικό συνδέεται σε μια θύρα E/E ενός κόμβου και διαθέτει κατ' ελάχιστο την υπολογιστική ισχύ και τα κυκλώματα που μπορούν να στηρίξουν τις στοιχειώδεις λειτουργίες μιας RDMA μεταφοράς. Το SLURPoE μπορεί πολύ εύκολα να επεκταθεί για να υποστηρίξει οποιαδήποτε τέτοια συσκευή που μπορεί να αποτελέσει τη διεπαφή του με το 10G Ethernet. Ο μηχανισμός μάλιστα των μηνυμάτων θα μπορούσε να υλοποιηθεί με υποστήριξη για το πρότυπο MSI, το οποίο υποστηρίζεται γενικά από το δίαυλο PCI και τις επεκτάσεις του έτσι, ώστε ο μηχανισμός αυτός χωρίς αλλαγές να μπορεί να λειτουργεί σε κάθε πιθανή συσκευή.

Τέλος μια από τις πιο σημαντικές σχεδιαστικές επιλογές που έγινε για να μπορέσει να υλοποιηθεί ένα τέτοιου μεγέθους εγχείρημα στο χρονικό διάστημα διεξαγωγής μιας διπλωματικής εργασίας, αφορά στη χρήση του Linux ως υλικολογισμικό πάνω στον προσαρμογέα του πρωτοκόλλου. Το Linux αποτελεί ένα πλήρες λειτουργικό σύστημα η χρήση του ως υλικολογισμικό είναι πιθανό να δημιουργεί επιπλέον επιβάρυνση.

Ένα από τα ζητήματα που θα μπορούσε να εξεταστεί αφορά στη σύγκριση του Linux με ένα λειτουργικό σύστημα πραγματικού χρόνου (Real Time Operating System – RTOS) όσον αφορά τη καταλληλότητα της χρήσης του ως υλικολογισμικό. Η χρονοδρομολόγηση με το τρόπο που γίνεται από το Linux είναι πιθανό ότι αποτελεί μια άσκοπη επιβάρυνση σε ένα σύστημα όπως ο προσαρμογέας, ο οποίος έχει από πριν προβλεπόμενες ενέργειες που πρέπει να εκτελεστούν στον επεξεργαστή. Έτσι θα μπορούσε αντί για χρονοδρομολόγηση να χρησιμοποιηθεί μια μηχανή καταστάσεων που θα διεκπεραίωνε λειτουργίες με συγκεκριμένη βαρύτητα.

Επιπλέον το Linux εμπλέκει στην υλοποίηση του ως ένα λειτουργικό σύστημα γενικής χρήσης πολλά υποσυστήματα που σαν υλικολογισμικό δεν είναι απαραίτητα. Η πολυπλοκότητα τους είναι πιθανό να εισάγουν υπολογιστική επιβάρυνση σε ένα προσαρμογέα δικτύου με συγκεκριμένες απαιτήσεις. Η υλοποίηση ωστόσο ενός τέτοιου ειδικού υλικολογισμικού για τον προσαρμογέα του SLURPoE ξεφεύγει κατά πολύ από το σκοπό και τα χρονικά περιθώρια της εργασίας αυτής.

Βιβλιογραφία

[1] L. Liss, Y. Birk, and A. Schuster, "Efficient exploitation of kernel access to infiniband: A software dsm example," *High-Performance Interconnects, Symposium on*, vol. 0, p. 130, 2003.

[2] H. Meuer, *TOP500 Supercomputer Sites; electronic version*. TOP500.Org, 2002.

[3] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. king Su, "Myrinet - A Gigabit-per-Second Local-Area Network," *IEEE Micro*, vol. 15, pp. 29–36, 1995.

[4] Myricom Inc., "Myrinet EXpress (MX): A High Performance, Low-level, Message-Passing Interface for Myrinet," tech. rep., 2006.

[5] B. White, D. Jorna, J. McDonough, I. Neville, and C. Zass, *Getting Started with InfiniBand on System z10 and System z9*. IBM Redbooks, 2009.

[6] S. Toor, "Introduction to infiniband white paper," 2003.

[7] J. Corbet, A. Rubini, and G. K. Hartman, *Linux Device Drivers, 3rd Edition*. O'Reilly Media, Inc., 3 ed., February 2005.

[8] S. Venkateswaran, *Essential Linux Device Drivers*. Prentice Hall Open Source Software Development Series, Boston, MA: Pearson, 2008.

[9] J. Pinkerton, "The Case for RDMA," 2002.

[10] H. Tezuka, F. O'Carroll, A. Hori, and Y. Ishikawa, "Pin-down cache: A virtual memory management technique for zero-copy communication," 1998.

[11] I. Schoinas and M. D. Hill, "Address translation mechanisms in network interfaces," in *In Proceedings of the Fourth International Symposium on High-Performance Computer Architecture (HPCA-4*, pp. 219–230, IEEE Computer Society, 1998.

[12] DAT Collaborative, "uDAPL: User Direct Access Programming Library Version 2.0," 2007.

[13] DAT Collaborative, "kDAPL: Kernel Direct Access Programming Library Version 2.0," 2004.

[14] Intel, *Intel 81341 and 81342 I/O Processors Developer's Manual*. Intel, Inc, December 2007.

[15] Intel, *Intel 81341 and 81342 I/O Processors Specification Update*. Intel, Inc, September 2008.

[16] B. Goglin, "Design and Implementation of Open-MX: High-Performance Message Passing over generic Ethernet hardware," in *CAC 2008: Workshop on Communication Architecture for Clusters, held in conjunction with IPDPS 2008*, (Miami, FL), IEEE Computer Society Press, Apr. 2008.

[17] R. Recio, P. Culley, D. Garcia, and J. Hilland, "An RDMA Protocol Specification," tech. rep., IETF, 2002.